République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abdelhamid Ibn Badis de Mostaganem
Département de Mathématiques

T H È S E

Pour obtenir le diplôme de

DOCTORAT EN SCIENCES

Spécialité : Mathématiques

Option : Optimisation et Contrôle Optimal

Présentée et soutenue par

Rachid Belgacem

Approche Sous-gradient pour le problème du TSP
" Travelling Salesman Problem "

---

soutenue publiquement le 26 04 2018

Devant le jury

Président :	M. Bahri Sidi Mohamed	(Pr UMAB Mostaganem)
Examinateur :	M. Elosmani Mohamed	(MCA ENP d'Oran)
Examinateur :	M. Benahmed Boubakeur	(Pr ENP d'Oran)
Directeur de thèse :	M. Amir Abdessamad	(Pr UMAB Mostaganem)

Laboratoire de **M**athématiques **P**ures et **A**ppliquées (LMPA).
Faculté des **S**ciences **E**xactes et de l'**I**nformatiques (FSEI).
Site 2 - Ines, 27000 Mostaganem, Algérie.

T H E S E

Remerciements

Je tiens à remercier mon directeur de thèse, le Professeur **Amir Abdessamad**, pour m'avoir accueilli dans son groupe de recherche, et pour m'avoir permis de mener à bien ce travail. Merci pour son soutien permanent et la confiance qu'il m'a accordée, et surtout pour tout le temps qu'il a consacré pour ce travail.

Je remercie également **Mr Bahri Sidi mohammed**, Professeur à l'UABM qui me fait l'honneur de présider ce jury.

J'exprime ma sincère reconnaissance à **Mr Benahmed Boubakeur**, Professeur à l'ENP d'Oran, et à **Mr Mohamed Elosmani** Professeur à l'ENP d'Oran, qui me font l'honneur d'être les rapporteurs scientifiques de ce travail.

Je tiens également à remercier tous mes amis pour le soutien qu'ils n'ont cessé de m'apporter.

J'exprime ma gratitude à **Mr ADNANE Yassine**, Professeur à l'université du Havre, France pour m'avoir guidé, aidé et conseillé durant la période de mon stage au laboratoire LMAH , France.

J'adresse mes remerciements à tous les membres du Laboratoire des Mathématiques Pures et Appliquées, Université de Mostaganem. Une pensée particulière pour Ahmed Boukhari.

Et enfin, un profond merci à ma famille.

Résumé : Un nouvel algorithme de sous-gradient dévié est présenté pour calculer une borne inférieure du problème dual. Ces bornes peuvent être utiles, dans l'évaluation des nœuds dans un l'algorithme 'Branch and Bound', pour trouver la solution optimale des problèmes de programmation linéaire en nombre entier à grande taille. La direction de recherche déviée utilisée dans cette thèse est une combinaison convexe de la technique de gradient modifié et de la stratégie de direction moyenne. Dans ce contexte, nous identifions le paramètre de combinaison convexe optimal permettant à la direction du vecteur sous-gradient dévié de former un angle plus aigu avec la meilleure direction vers une solution optimale. L'algorithme modifié donne des résultats encourageants pour des instances de problèmes de voyageur de commerce symétrique (TSPs) sélectionnés depuis la base de données TSPLIB.

Abstract : A new deflected subgradient algorithm is presented for computing a tighter lower bound of the dual problem. These bounds may be useful in nodes evaluation in a Branch and Bound algorithm to find the optimal solution of large-scale integer linear programming problems. The deflected direction search used in this thesis is a convex combination of the Modified Gradient Technique and the Average Direction Strategy. In this context, we identify the optimal convex combination parameter allowing the deflected subgradient vector direction to form a more acute angle with the best direction towards an optimal solution. The modified algorithm gives encouraging results for a selected symmetric travelling salesman problem (TSPs) instances taken from TSPLIB library.

ملخص: يتم تقديم هنا خوارزمية فرعية متدرجة جديدة من أجل حساب أحسن قيمة الحد الأدنى للمشكلة المزدوجة. قد تكون هذه الحدود مفيدة في خوارزمية الفروع والضمنية للعثور على الحل الأمثل لمشكلات البرمجة الخطية ذات المتغيرات الصحيحة الموجبة. إن البحث عن الاتجاه المنحرف المستخدم في هذه الأطروحة هو مزيج محدب من تقنية التدرج المعدل وتقنية متوسط الاتجاه. وفي هذا السياق، نحدد معلمة الجمع المُحسنة المثلى التي تسمح لاتجاه الفرعي المنحرف لتشكيل زاوية أكثر حدة مع أفضل اتجاه نحو الحل الأمثل. تعطي الخوارزمية المعدلة نتائج مشجعة لبعض حالات مشكلة المسافر المتجول **(TSP)** تم اختيارها من قاعدة بيانات**TSPLIB** .

Table des matières

Table des figures

Liste des tableaux

Quelques notations :

Les notations suivantes apparaissent fréquemment dans cette thèse :

$\mathbb{Z}_+$: Ensemble des entiers relatifs positifs.

$\mathbb{Z}^n$: Espace des vecteurs à n composantes entière.

$\mathbb{R}$: Ensemble des nombres réels.

$\mathbb{R}^+$: Ensemble des nombres réels non-négatifs.

$\mathbb{R}^m$: Espace des vecteurs à m composantes réelles.

$\mathbb{R}^m_*$: Espace des vecteurs non nuls à m composantes réelles.

$\mathbb{R}^{m\times n}$: Espace des matrices réelles de dimension $m \times n$.

A^T : Transposée de la matrice A.

A^{-1} : Inverse de la matrice A.

B^{adj} : Adjoint de la matrice B.

I_k : Matrice identité d'ordre k.

0_n : Matrice nulle d'ordre n.

$V = \{1, ..., n\}$: Ensemble des sommets.

$E = \{e = (i, j) : i, j \in V, i \prec j\}$: Ensemble des arêtes.

c_e : Coût de l'arête e.

$E(S) = \{(i, j) \in E : i, j \in S\}$: Ensemble de toutes les arêtes de E avec les deux extrémités dans S.

$\delta(S) = \{(i, j) \in E : i \in S,\ j \notin S\ ou\ i \notin S,\ j \in S\}$: Ensemble de toutes les arêtes avec une extrémité dans S et l'autre en $V \backslash S$.

$\delta(i) = \delta(\{i\}), i \in S$: Ensemble des arêtes incidentes au nœud i.

$\cap, \cup$: Intersection (respectivement réunion) d'ensembles.

$S \subset V$: S est strictement contenu dans V.

$|S|$: Cardinale de S.

$c^T x$: Produit scalaire des vecteur c et x.

$\|.\|$: Norme euclidienne.

P^+ : Opérateur de projection.

$\emptyset$: Ensemble vide.

det : Déterminant d'une matrice.

LB : Borne inférieure.

UB : Borne supérieure.

Abréviations

(LP) : Linear Programming.

(ILP) : Integer Linear Programming.

($BILP$) : Binary Integer Linear Programming.

($MILP$) : Mixed Integer Linear Programming.

(COP) : Combinatorial Optimization Problem.

(TSP) : Travelling Salesman Problem.

(AP) : Assignment Problem.

$B\&B$: Branch and Bound.

LB : Lower Bound.

UB : Upper Bound.

MGT : Modified Gradient Technique.

ADS : Average Direction Strategy

Introduction

0.1 Motivation

En pratique, il arrive fréquemment que dans un problème d'optimisation linéaire certaines variables soient astreintes à être entières ou même binaires $x_j = 0$ ou 1, on parle alors de programme linéaire en nombre entiers (ILP) ou programme linéaire binaire ($BILP$).

Un des problèmes les plus étudiés dans la classe des (ILP) est le problème du voyageur de commerce ("The Travelling Salesman Problem " noté " TSP"). Dans ce problème, un voyageur de commerce doit visiter plusieurs villes (ou clients) en passant une et une seule fois par chacune d'entre elles, en minimisant la distance totale parcourue. Pour une étude complète des méthodes de résolution, des applications et des problèmes connexes nous renvoyons le lecteur aux références [E.L. Lawler (1985)] et [G. Laporte (1992)]).

Il est bien connu que le problème TSP est *NP-difficile* [G. Laporte (1992)], [D.S. Jonhson(1997)] et [S. Arora (1998)]. En effet, dans sa version symétrique [1], le nombre totale de solutions possibles est $\frac{(n-1)!}{2}$, où n est le nombre de villes. Avec une telle complexité factorielle, une résolution efficace de TSP nécessite donc le recours à des méthodes d'optimisation très performantes.

Les méthodes de résolution du TSP peuvent être réparties en deux groupes de nature différente :

- Le premier groupe comprend les méthodes exactes qui garantissent la complétude de résolution : c'est le cas par exemple de la méthode exacte "Séparation et Évaluation" ("Branch and Bound" noté "$B\&B$") ; le temps de calcul nécessaire d'une telle méthode augmente en général exponentiellement avec la taille du problème à résoudre. Citons aussi les méthodes qui s'appuient sur des techniques de relaxation, telle que la relaxation Lagrangienne.

1. c-àd dans le cas où le graphe associé n'est pas orienté.

Le second groupe comprend les méthodes approchées dont le but est de trouver une solution de bonne qualité en un temps de calcul raisonnable sans garantir l'optimalité de la solution obtenue.

La relaxation Lagrangienne est une technique bien connue en optimisation [M.L. Fisher (1985), A.M. Geoffrion (1974)]. Elle a été utilisée dans le cadre du TSP à partir des années 1970. Les travaux de [M.H. Held(1970), M.H. Held(1971)] basés sur la méthode de sous-gradient ont permis de résoudre des instances TSP de grandes tailles. Des algorithmes un peu plus performants ont établit par [E. Allen (1987), K.H. Helbig-Hansen(1974), G. Reinelt (1994), T. Volgenant (1982)] et [C.L. Valenzuela (1995)].

Bien que simple à implémenter, la convergence de la méthode sous-gradient est trop sensible à certain paramètres initiaux. Beaucoup d'efforts ont été établit [M.S. Bazaraa (1981), J.L. Goffin (1977)], afin d'assurer la convergence et accélerer le temps de calcul.

La direction de recherche dans l'algorithme de sous-gradient affecte énormément les performances de calcul. En effet, la méthode sous gradient pure, engendre souvent le phénomène de zig-zag. Aussi, l'angle entre la direction du sous-gradient peut former un angle obtus avec la direction précédent, ce qui peut aussi ralentir la convergence de la procédure.

Le but principal de notre travail est de développer un outil pour surmonter cette difficulté, une nouvelle procédure de sous-gradient dévié modifiée est utilisée dans cette procédure, la direction déviée est calculée en combinant le sous-gradient proposé par P.M. Camerini et al [P.M. Camerini(1975)] et la direction proposée par H.D. Sherali et al [H.D. Sherali(1989)].

0.2 Organisation de la thèse et principales contributions

Le premier chapitre de cette thèse est une introduction aux problèmes de programmation linéaire en nombre entier (ILP), dont le but est de mieux mesurer le degré de difficulté du problème en introduisant des variables discrètes dans un modèle de programmation linéaire. L'accent sera mis sur le problème du voyageur de commerce (TSP) comme exemple de problèmes en variables binaires. Cependant et pour des raisons de commodités, nous introduisons aussi le problème d'affectation ("Assignement Problem" noté "AP"). En suite, les principes de base d'un algorithme "séparation-évaluation" ont été introduit pour résoudre les problèmes (ILP) et particulièrement pour le problème (TSP).

Dans le second chapitre, nous nous somme intéressé aux méthodes de relaxation Lagrangienne. Ces techniques sont très utilisées pour la résolution de problèmes de programmation linéaire en nombres entiers de grands dimension. Nous discutons également des propriétés du problème dual Lagrangien, des conditions d'optimalité et de la structure de la fonction objectif duale. Nous présentons à la fin de ce chapitre une application de cette technique sur le problème TSP. En effet, en injectent les contraintes de degré dans la fonction objectif du problème TSP, la relaxation Lagrangienne sera exprimée comme une relaxation 1- arbre.

Le troisième chapitre est dédié à l'étude des méthodes d'optimisation de type sous-gradient qui peuvent être fréquemment utilisées pour résoudre des problèmes d'optimisation non-différentiables, plus particulièrement pour maximiser efficacement la fonction duale Lagrangienne, d'un problème de la programmation linéaire en nombre entier, où la fonction objectif n'est pas différentiable en tout point. Il existe différentes procédures de sous-gradient qui diffèrent dans leurs performances, a savoir, méthode de sous-gradient pur et méthode de sous-gradient dévié. Ces procédures seront discutées en détail par la suite.

Dans le quatrième chapitre, nous présenterons une nouvelle approche de sous-gradient dévié qui repose sur une combinaisons convexe de le direction du gradient modifié d^k_{MGT} [P.M. Camerini(1975)] et la direction moyenne d^k_{ADS} [H.D. Sherali(1989)]. Notre approche aborde les deux volets : théorique et appliqué. Le principal résultat théorique est l'identification du paramètre de combinaison convexe qui force l'algorithme pour une meilleure recherche de déviation que ceux données par rapport le sous-gradient pur, MGT et ADS. Pour une comparaison numérique entre notre approche et les deux techniques MGT et ADS, nous avons opté le problème TSP dont l'importance réside dans la diversité de ses applications [M. Diaby (2016)] et [N.A. El-Sherbeny(2010)].

Enfin, nous terminerons cette thèse par une conclusion.

CHAPITRE 1

Introduction à la programmation linéaire en variables entières

Sommaire

1.1 Problème de programmation linéaire

Un problème de programmation linéaire, en anglais Linear Programming, abrégé (LP) est un problème d'optimisation, dans lequel nous cherchons à trouver un ensemble de valeurs pour les variables continues $(x_1, x_2, ..., x_n)$ qui maximise ou minimise une fonction objectif linéaire z, tout en satisfaisant un ensemble de contraintes linéaires (un système d'équations

et/ou inéquations). Les programmes linéaires peuvent être utilisés pour modéliser un grand nombre de problèmes qui se posent dans la pratique. Mathématiquement, un (LP) dans sa forme standard est exprimé comme suit :

$$(LP)\begin{cases} z^* = \max\ c^\top x \\ \text{s.t. } Ax \leq b \\ x \geq 0, \end{cases} \tag{1.1}$$

où les matrices $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^{n\times 1}$ et $b \in \mathbb{R}^{m\times 1}$ sont les données du problème.

Un concept fondamental de la programmation linéaire est la dualité. Le programme linéaire dual du problème donné par le programme primal (1.1) est défini comme suit :

$$(DLP)\begin{cases} w^* = \min\ u^\top b \\ \text{s.t. } u^\top A \geq c \\ u \geq 0. \end{cases} \tag{1.2}$$

Il est facile de vérifier que le dual du problème dual est le problème primal. Un aspect important du concept de dualité est indiqué dans le théorème suivant [J.C. Culioli (1994)].

Théorème 1.1 *Soient (LP) et (DLP) deux programmes linéaires définis ci-dessus par (1.1) et (1.2). Supposons qu'il existe des vecteurs x^* et u^* satisfaisant les contraintes du (LP) et (DLP) resp. Alors, nous avons*

1. *La valeur de la fonction objectif z en x^* (dans problème (LP)) est inférieure ou égale à la valeur de la fonction objectif w en u^* (dans problème (DLP)).i.e., $z^* \leq w^*$.*
2. *Si (LP) ou (DLP) admet une solution optimale finie, alors il en est de même pour l'autre et leurs valeurs optimales associées sont égales.i.e., $z^* = w^*$.*
3. *Si (LP) ou (DLP) admet une valeur optimale infinie, alors l'autre n'admet pas de solution.*

Ce théorème est de grande portée théorique et pratique. D'autres propriétés et théorèmes liés à la dualité sont largement utilisés en programmation linéaire. Nous renvoyons le lecture aux ouvrages spécialisés [J.C. Culioli (1994), D. Goldfarb (1989)] et [J. Teghem (2003)], pour une présentation complète.

La dualité présente de nouvelles relations entre le problème primal et le problème dual. Notons en particulier, que le problème dual peut être utilisé pour donner des bornes pour la valeur optimale du problème primal (et vice versa).

1.1.1 Résolution des programmes linéaires

L'algorithme du simplexe est une des méthodes les plus communément utilisée pour résoudre les problèmes de programmation linéaire [G.B. Dantzig (1951)]. La méthode du simplexe est une méthode exacte et itérative. En effet, d'un point de vue géométrique, étant donné qu'une solution d'un programme linéaire se trouve toujours sur un point extrême du polyèdre des contraintes, le simplexe consiste à se déplacer d'un point extrême à l'autre, le long des arêtes du polyèdre, jusqu'à trouver le point associé à la solution optimale. Algébriquement, le simplexe s'interprète comme la détermination d'une suite de bases adjacentes réalisables telles que les valeurs associées de la fonction objectif soient croissantes.

Cependant, s'il se révèle très efficace en pratique, Klee et Minty [V. Klee (1972)] ont trouvé des exemples qui montraient pour la première fois que la méthode du simplexe pouvait prendre un nombre exponentiel d'itérations. Une question demeurait toutefois ouverte : exister-il un algorithme de complexité polynomiale pour résoudre un LP ?. La recherche d'un algorithme polynomial[1] pour la programmation linéaire était lancée.

Le premier algorithme de ce type, appelé méthode ellipsoïde, a été développé en 1979 par Khachian [L.G. Khachiyan (1979)], montrant alors que les problèmes de programmation linéaire sont polynomiaux. L'idée est d'utiliser une suite d'ellipsoïdes de volume décroissant mais qui contiennent à chaque itération la solution optimale du programme linéaire à résoudre. On peut trouver une présentation détaillée de cet algorithme dans [D. Goldfarb (1989)]. Bien que plus rapide que le simplexe sur les problèmes de Klee et Minty, il reste plus lent sur les problèmes réels.

Karmakar a développé un deuxième algorithme polynomial en 1984 [N. Karmarkar (1984)]. Il s'agit d'une méthode de points intérieurs qui se base sur des principes de géométrie projective et de programmation non linéaire. L'idée est de chercher des points, à l'intérieur du polyèdre des contraintes, permettant de se diriger rapidement vers le sommet optimal. Des dérivés de cet algorithme, en particulier la méthode dite de barrière, commencent à concurrencer le simplexe sur certains problèmes de grande taille. Pour plus de détails sur cet algorithme, on peut se référer à des ouvrages spécialisés tels que [D. Goldfarb (1989)].

Le simplexe reste cependant la méthode privilégiée de résolution des programmes linéaires formulés dans la pratique. Ceci est dû à ses performances et également à sa capacité

1. Un algorithme est polynomial si on peut borner le temps maximal de son exécution par une fonction polynomiale de la taille du problème considéré.

à fournir des solutions de base, très importantes dans les approches de décomposition ou encore dans des procédures de ré-optimisation itératives [M. Lebbar (2000)].

L'objectif ici est d'étudier les extensions de la programmation linéaires quand nous accordons des conditions d'intégralité supplémentaires sur toutes ou certaines variables de décisions.

1.2 Particularité des programmes linéaires en variables entières

1.2.1 Formulation du problème

Dans le problème général de la programmation linéaire, toutes les inconnues peuvent varier de façon continue. Si, en revanche, on impose à toutes les variables du problème d'avoir des valeurs entières, on se trouve devant un problème de programmation linéaire à variables entières. Si une partie seulement des variables est soumise à cette contrainte supplémentaire d'intégralité, il s'agit alors d'un problème de programmation linéaire en variables mixtes.

Définition 1.1 *Un programme linéaire à variables entières mixtes noté* ***(MILP)*** *("Mixed Integer Linear Programming") est donnée par les matrices $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^{n\times 1}$, $b \in \mathbb{R}^{m\times 1}$ et un nombre $p \in \mathbb{N}^*$. L'objectif du problème est de trouver un vecteur x solution du problème d'optimisation suivant :*

$$(MILP)\begin{cases} z^* = \max\ c^\top x \\ s.t. \quad Ax \leq b \\ x \geq 0 \\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{cases} \tag{1.3}$$

Si $p = 0$, alors il n'y a pas de contraintes d'intégralité, donc nous obtenons le programme linéaire (1.1). D'autre part, si $p = n$, toutes les variables doivent être entières. Dans ce cas, nous parlons d'un programme linéaire en nombres entiers noté ***(ILP)*** *("Integer Linear Programming") :*

$$(ILP)\begin{cases} z^* = \max\ c^\top x \\ s.t. \quad Ax \leq b \\ x \geq 0 \\ x \in \mathbb{Z}^n_+, \end{cases} \tag{1.4}$$

Si dans un (ILP) *toutes les variables sont restreintes aux valeurs de l'ensemble* $B = \{0, 1\}$*, nous avons un programme linéaire binaire noté* ***(BILP)*** *("Binary integer program") :*

$$(BILP)\begin{cases} z^* = \max\ c^\top x \\ s.t. \quad Ax \leq b \\ x \in B^n, \end{cases} \tag{1.5}$$

La plupart des problèmes réels comportent des variables qui doivent, par nature, prendre nécessairement une valeur entière. Par exemple, si une variable représente l'existence ou l'absence d'une usine, ou un nombre minimal de navires destinés au transport de certaines marchandises, la solution d'un problème de programmation linéaire qui donnerait 0,44 usine ou 3,55 navires n'est pas satisfaisante ; si on essaie alors d'arrondir à une valeur entière voisine, il se peut que la solution ainsi trouvée soit éloignée de la solution optimale, ou même ne soit pas réalisable. Dans de telles situations, il est donc souhaitable de pouvoir utiliser une méthode plus adaptée.

1.2.2 Complexité

Les problèmes généraux de programmation linéaire en variables entières (ILP), $(MILP)$ ou $(BILP)$ sont beaucoup plus difficiles à résoudre que le problème de programmation linéaire en variables continues (LP).

L'étude initiale du problème (LP) nous a montré immédiatement que :

- L'obtention de la solution optimale ne nécessite pas de s'intéresser qu'à un nombre fini de solutions, à savoir les seuls sommets du polyèdre $P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$.
- Ces sommets sont facilement caractérisables et ils peuvent donc être aisément déterminés.

Par contre, dans les situations (ILP), $(MILP)$ ou $(BILP)$, la solution optimale n'est, en général, pas un sommet de P et donc un point intérieur ou point situé n'importe où sur la frontière. On perd ainsi toute caractérisation particulière de la solution optimale qui est, pour cette raison, bien difficile à déterminer.

Cette approche intuitive est évidement confirmée par l'approche rigoureuse de la complexité des algorithmes. Le problème de décision :

$$\text{« } \textit{Existe-t-il une solution } x \in S = P \bigcap \mathbb{Z}^n \textit{ ?} \text{ »}$$

est un problème *NP-complet*(voir [A. Schrijver (1986)]), il s'ensuit que le problèmes (ILP), $(MILP)$ ou $(BILP)$ sont *NP-difficiles*. Il existe certaines classe de problèmes *NP-difficile*

dont des réalisations de grands dimension peuvent être résolues rapidement. Par ailleurs, même dans le cas où la solution optimale n'est pas obtenue en un temps raisonnable, certaines techniques de résolution telle celle du " Branch and Bound " décrite au paragraphe (1.6.1), permettent souvent d'obtenir une solution optimale.

1.3 Relaxation

Dans l'absence d'une caractérisation de la solution, comment prouver qu'une solution x^* est optimale ? On a donc besoin de trouver une borne inférieure $z^{LB} \leq z^*$ et une borne supérieure $z^* \leq z^{UB}$ telle que $z^* = z^{LB} = z^{UB}$.

Sur le plan pratique, cela signifie qu'un algorithme donné doit trouver une série décroissante de bornes supérieures $z_1^{UB} > \cdots > z^*$ et une série croissante de bornes inférieures $z_1^{LB} < \cdots < z^*$ et arrêter l'algorithme quand $|z_s^{LB} - z_t^{UB}| \leq \varepsilon$ où ε représente une précision numérique préalablement déterminée.

Bornes primales

Toutes les solutions réalisables $x \in X = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ pour le problème de maximisation donnent une borne inférieure $z^{LB} = c^T x \leq z^*$. C'est le seul moyen connu pour obtenir des bornes inférieures. Pour certains problèmes en nombres entiers, trouver des solutions réalisables n'est pas une tâche difficile. C'est le cas par exemple du problème du voyageur de commerce.

Bornes duales

Trouver des bornes supérieures pour un problème de maximisation (ou de bornes inférieures pour un problème de minimisation) est un véritable défi. Ces bornes sont appelées duales par oppositions avec les bornes primales. Il s'agit de remplacer un programme (ILP) "difficile" par un problème d'optimisation simple.

Définition 1.2 *Un problème d'optimisation* $\min f(x) : x \in T \subseteq \mathbb{R}^n$ *est une relaxation d'un problème* $\min c^T x : x \in X \subseteq \mathbb{R}^n$ *si* $X \subseteq T$ *et* $f(x) \leq c^T x$ *pour tout* $x \in X$.

Il existe trois grandes classe de relaxations.

1.3.1 Relaxation des contraintes

Une technique simple de relaxation consiste à ignorer certaines contraintes du problème. On obtient alors un problème dont la solution optimale est plus facile à calculer. Par exemple soit un polyèdre P défini par deux ensembles de contraintes :

$$P = \{x \in \mathbb{R}^n : A_1 x \leq b_1, A_2 x \leq b_2\}.$$

Il peut s'avérer que la suppression d'un des deux ensembles de contraintes - par exemple le second - permet d'obtenir un polyèdre convexe

$$P^{'} = \{x \in \mathbb{R}^n : A_1 x \leq b_1\},$$

tel que le problème relaxé $(RILP)$ définie par

$$(RILP) \begin{cases} z^* = \max \ c^\top x \\ \text{s.t.} \quad x \in R^{(i)} = P^{'(i)} \bigcap \mathbb{R}^n_+ \end{cases}$$

est un problème classique, évident ou simple à résoudre. Le problème relaxé $(RILP)$ est alors appelé sous problème du problème (ILP). Une bonne illustration de ce type de relaxation est donnée par le problème du voyageur de commerce (*NP-difficiles*) dont un sous problème est le problème d'affectation (de classe **P**) analysé au paragraphe (1.5.1).

1.3.2 Relaxation linéaire continue

Définition 1.3 *Le problème (LP)(donné par (1.1)) est appelé la relaxation linéaire du problème (ILP)(donné par (1.4)). Il s'agit d'un relâchement des contraintes $Ax \leq b$ et $x \geq 0$ en $x \in \mathbb{R}^n_+$.*

Remarque 1.1

1. *Pour un problème de minimisation, la solution optimale du (LP) donne une borne inférieure de la valeur de la solution optimale du (ILP).*
2. *La relaxation linéaire ne donne pas seulement une borne supérieure mais par fois la solution optimale si celle ci est entière.*

Une méthode immédiate pour résoudre efficacement un (ILP) consiste à résoudre le problème relaxé (LP), puis à prendre comme solution la solution entière la plus proche.

Exemple 1.1 *Le programme de la relaxation linéaire du problème (ILP) suivant :*

$$\begin{cases} z \ = \max 7x_1 + 4x_2 + 5x_3 + 2x_4 \\ \quad 3x_1 + 3x_2 + 4x_3 + 2x_4 \leq 6 \\ \quad x_i \in \{0, 1\} \qquad i = 1, \ldots, 4 \end{cases}$$

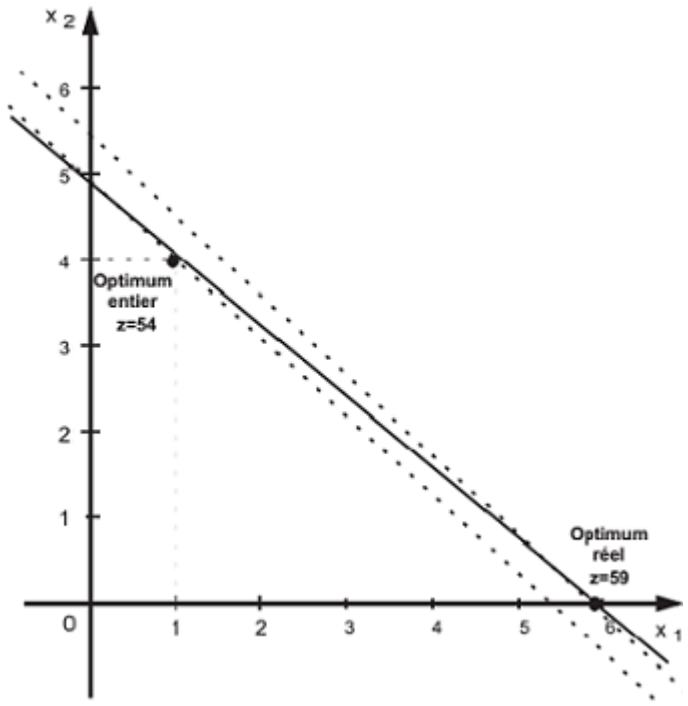


FIGURE 1.1 – Problème du sac à dos

a une solution optimale $x^* = (1, 1, 0, 0)^T$. Comme les x_i sont des entiers, alors le problème (ILP) est associé à la mème solution.

L'exemple suivant indique que la solution du $(RILP)$ n'a parfois rien à voir avec la solution (ILP).

Exemple 1.2 *Considérons le programme linéaire en nombres entiers suivant :*

$$\begin{cases} \min z = & -10x_1 - & 11x_2 \\ & 10x_1 + 12x_2 & \leq 59 \\ & x_1, x_2 & entières \end{cases}$$

En variables continues, la solution optimale est : $x_1 = 5.9, x_2 = 0$ qui n'est pas entière (comme on le voit Figure 1.1). Un arrondi de cette solution serait $x_1 = 6$ et $x_2 = 0$. Ce n'est en aucun cas une solution du (ILP) pour deux raisons :

- *Premièrement, cette solution ne satisfait pas les contraintes.*
- *Deuxièmement, ce n'est pas une solution entière .*

Sachant que la solution entière est $x_1 = 4$ et $x_2 = 1\,(z = -54)$, on remarque qu'elle est même très éloignée de la solution optimale continue.

1.3.3 Relaxation Lagrangienne

La relaxation Lagrangienne s'articule sur l'idée de relâcher les contraintes difficiles, non pas en les supprimant totalement, mais en les prenant en compte dans la fonction objectif de sorte qu'elles pénalisent les valeurs des solutions qui les violent (voire le chapitre (2)).

1.4 Le cas particulier des matrices totalement unimodulaires

Certains problèmes à coefficients entiers ont naturellement des solutions entières. Nous donnons d'abord les définitions suivantes :

Définition 1.4 *Une matrice carrée B à coefficients entiers est dite* ***unimodulaire*** **(UM)** *si son déterminant est égal à $+1$ ou -1(i.e. $det(B) = \mp 1$).*

Définition 1.5 *Une matrice $A(m \times n)$ à coefficients entiers est dite* ***totalement unimodulaire*** **(TUM)** *si tout sous-déterminant de A vaut $0, +1$ ou -1.*

En particulier, tout coefficient d'une matrice totalement unimodulaire est égal à $0, +1$ ou -1.

Si B est formée à partir d'un sous-ensemble de m colonnes linéairement indépendantes de A, elle détermine la solution de base (voire [J. Teghem (2003)])

$$x = B^{-1}b = \frac{B^{adj}b}{\det(B)},$$

où B^{adj} est l'adjoint de B, et si B est **UM** et b est entier, on déduit que le vecteur x est entier.

Considérons le problème (LP) (donné par (1.1)) donné sous la forme standard, i.e

$$x \in D = \{x : Ax = b, \quad x \geq 0\},$$

Nous avons le théorème suivant

Théorème 1.2 *Si A est* **TUM**, *alors tous les sommets de D sont entiers pour tout vecteur entier b.*

Donc, la résolution par simplexe d'un (ILP) standard relaxé, dont la matrice des contraintes est **TUM** donne la solution entière exacte de (ILP).

Dans le cas de contraintes inégalité (i.e $x \in P = \{x : Ax \leq b, \quad x \geq 0\}$,), nous avons le théorème suivant :

Théorème 1.3 *Si A est **TUM**, alors tous les sommets de P sont entiers pour tout vecteur entier b.*

Preuve

Il revient donc à montrer que si A est **TUM**, alors $(A|I)$ l'est. Nous ajoutons des variables d'écart et on applique le théorème 1. Soit C une sous-matrice carré inversible de $(A|I)$. Les lignes de C peuvent être permutés afin de pouvoir écrire

$$C = \left(\begin{array}{c|c} B & 0 \\ \hline D & I_k \end{array}\right)$$

où I_k est une matrice identité de taille k et B est une sous-matrice carrée de A. On a $det(C) = det(b) = \pm 1$ parce que A est **TUM** et C est inversible. ■

Remarque 1.2 *La condition d'unimodularité est suffisante mais non nécessaire pour l'existence d'une solution optimale entière du problème (LP). En pratique, de nombreux problèmes d'optimisation à données entières ont une solution entière bien que la matrice A ne soit pas unimodulaire, mais l'intégrité de la solution risque d'être perdue si des paramètres changent.*

1.5 Problèmes d'optimisation combinatoires

Définition 1.6 *Un problème d'optimisation combinatoire noté **(COP)** ("Combinatorial Optimization Problem") est donné par un ensemble fini N, une fonction de poids $c : N \to \mathbb{R}$ et une famille $F \subset 2^{|N|}$ de sous-ensembles réalisable de N. L'objectif est de résoudre*

$$(COP) \qquad optimiser \left\{ \sum_{j \in S} c_j : S \subseteq F \right\}.$$

Le terme "optimisation combinatoire" recouvre l'ensemble des problèmes d'optimisation en variables binaires et les techniques appropriées pour le résoudre. Des exemples bien connus de (COP) sont le problème de l'affectation classique et le problème du voyageur de commerce (" Travelling Salesman Problem' :TSP,) [E.L. Lawler (1985)]. Ainsi il existe de nombreux problèmes classiques pouvant se modéliser à l'aide de variables discrètes, en

particulier de type binaire : le problème de chargement ou du sac à dos ("Knapsack problem" :KP [S. Martello(1990)]), les problèmes de localisation dans un graphe ('Location Problem" [P. Merchandani(1990)]) et les problèmes de tournées de véhicules ("Vehicule Routing" [B.L. Golden(1988)]).

La programmation linéaire en variables binaires est très utilisée comme langage de modélisation, car la plupart des problèmes réels comportent des variables qui doivent, par nature, prendre nécessairement une valeur binaire. Nous présentons rapidement ici deux problèmes classiques d'optimisation combinatoire : Le problème d'affectation et le problème du voyageur de commerce.

1.5.1 Problème d'affectation

Le problème d'affectation (*"Assignment problem", AP*) est un problème *(BILP)*, il consiste à établir des liens entre les éléments de deux ensembles distincts, de façon à minimiser un coût et en respectant des contraintes d'unicité de lien pour chaque élément.

On considère n tâches et n agents. Pour tout couple $(i, j)(i = 1...n, j = 1...n)$, l'affectation de la tâche i à j entraîne un coût de réalisation noté $c_{ij}(c_{ij} \geq 0)$. Le problème consiste à affecter au mieux n tâches à n agents (voire Figure 1.2). Chaque agent peut réaliser une unique tâche pour un coût donné et chaque tâche doit être réalisé par un unique agent. Les affectations (c'est à dire les couples agent-tâche) ont toutes un coût défini. il s'agit donc de définir une bijection de $\{1, ..., n\}$ sur $\{1, ..., n\}$ ou encore une permutation de n objets.

Le but étant de minimiser le coût total des affectations afin de réaliser toutes les tâches. Le problème peut être modéliser de la manière suivante :

Désignons par $i = 1, 2, ..., n$ les agents, $j = 1, 2, ..., n$ les tâches et introduisons les n^2 variables binaires :

$$x_{ij} = \begin{cases} 1 & \text{si l'agent } i \text{ réalise la tâche } \ j \\ 0 & \text{sinon.} \end{cases}$$

Les contraintes du problème d'affectation s'écrivent donc simplement :

Le nombre d'agents réalisant la tâche i (Chaque agent i fait une seule tâche j) est donné par :

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{pour } i = 1, \ldots, n.$$

Le nombre de tâches réalisées par l'agent j(Chaque tâche j est réalisée par un seul agent i) est donné par :

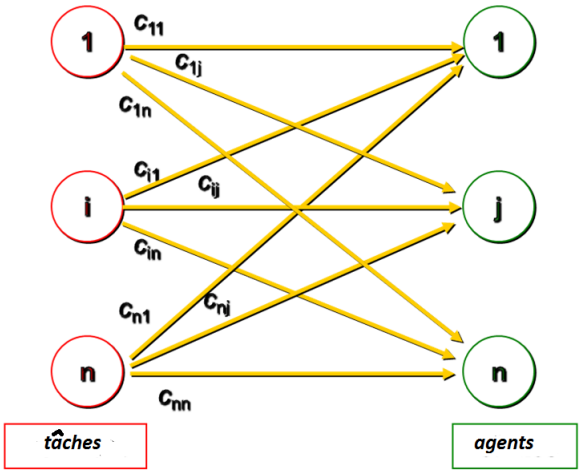


FIGURE 1.2 – Problème d'affectation

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{pour } j = 1, \ldots, n,$$

Définition de la fonction objectif : Nous traiterons ici le problème linéaire d'affectation. Le coût total de réalisation des tâches s'exprime alors par la somme :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

On peut donc modéliser le problème linéaire d'affectation sous la forme :

$$(AP) \begin{cases} \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \\ \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \\ x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, n, \qquad j = 1, \ldots, n \end{cases}$$

1.5.2 Problème de voyageur de commerce

Le problème du voyageur de commerce " *Traveling Salesman Problem, TSP*", est l'un des problèmes le plus étudié dans l'optimisation combinatoires qui consiste à la recherche d'un trajet minimal, permettant à un voyageur de visiter n villes séparées par des distances données en passant par chaque ville exactement une fois (voire Figure 1.3). Il commence par une ville quelconque et termine en retournant à la ville de départ. Quel chemin faut-il choisir afin de minimiser la distance parcourue ?

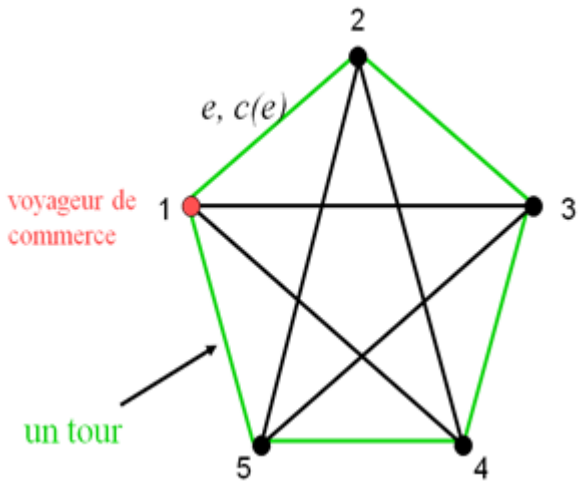


FIGURE 1.3 – Problème de voyageur de commerce

La notion de distance peut-être remplacée par d'autre notions, comme le temps qu'il met ou l'argent qu'il dépense. En générale, on cherche à minimiser le coût. Théoriquement, le *TSP* peut se représenter sous forme d'un graphe $G = (V, E)$ où V est l'ensemble de n sommets (les villes) $V = \{1, ..., n\}$ et E l'ensemble des arêtes ou arcs $E = \{(i, j), i \neq j, i, j \in V\}$, chaque arête a un poids c_{ij} (le coût) avec $c_{ii} = \infty, i = 1, ..., n$ (Afin d'éviter les sous-tours d'ordre 1 "$x_{ii} = 1$"). Le problème est de trouver un circuit qui passe par tous les sommets une et une seule fois avec un coût minimal. On distingue deux types de *TSP* ; symétrique ($c_{ij} = c_{ji}$) et asymétrique ($c_{ij} \neq c_{ji}$), ici, uniquement le cas symétrique est considéré.

Les domaines d'applications immédiats du TSP sont nombreux : problèmes de logistique, de transport, d'ordonnancement. Il présente de nombreuses applications dans des domaines plus éloignés comme la génétique (en remplaçant les villes par des gènes et la distance par la similarité).

Le TSP est difficile à résoudre et on ne connaît pas de méthode de résolution permettant d'obtenir des solutions optimales en un temps polynomial. il est un problème **NP-complet**.

Le *TSP* peut se modéliser comme *(BILB)*. En effet, étant donné n villes, notons $C = (c_{ij})$ la matrice des coûts et x_{ij} les variables de décision définies par

$$x_{ij} = \begin{cases} 1 & \text{si le voyageur va immédiatement de la ville i vers la ville j,} \\ 0 & \text{sinon.} \end{cases}$$

La formulation linéaire classique du problème est la suivante :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1.6}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{pour } i = 1, \ldots, n \tag{1.7}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{pour } j = 1, \ldots, n \tag{1.8}$$

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \geq 1, \quad \forall S \subset V, S \neq \emptyset \tag{1.9}$$

$$x_{ij} \in \{0, 1\} \text{ pour } i, j = 1, \ldots, n \qquad i \neq j. \tag{1.10}$$

Dans cette formulation, la relation (1.6) décrit la fonction objectif. Les contraintes (1.7) et (1.8) s'appellent les contraintes de degré qui assurent qu'une ville est visitée qu'une seule fois : on y arrive une et une seule fois (1.7), on en part une et une seule fois (1.8), ces contraintes ne sont pas suffisantes pour décrire les tours, d'où la nécessité d'introduire les contraintes (1.9) appelées contraintes d'élimination des sous-tours, avec S un sous ensemble de V et $\overline{S}$ son complémentaire dans V, $|S|$ est le cardinal de S. Enfin, les contraintes (1.10) sont les contraintes d'intégrité de variables.

Une autre formulation des contraintes d'élimination de sous-tours(1.9) peut être donnée par la relation suivante :

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -1, \qquad S \subset V, \qquad 2 \leq \mid S \mid \leq n - 2 \tag{1.11}$$

Cette formulation de *TSP* contient $n(n-1)$ variables binaires, $2n$ contraintes de degré et $2^n - 2n - 2$ contraintes d'élimination de sous tours.

1.6 Méthodes de résolution

Pour les problèmes classiques de l'optimisation combinatoire il existe trois grandes catégories de méthodes de résolution : les méthodes exactes, les méthodes heuristiques et les méthodes métaheuristiques.

Les méthodes exactes permettent d'obtenir une solution optimale à chaque fois, mais leurs durées de calcul tendent à augmenter exponentiellement avec la taille du problème.

A l'inverse, les méthodes heuristiques sont des méthodes spécifiques, permettant d'obtenir rapidement une solution approchée de bonne qualité, mais qui n'est pas nécessairement

optimale. Les plus utilisées sont celle du " plus proche voisin" et les " méthodes d'insertion". Les méthodes métaheuristiques [1] sont des algorithmes d'optimisation généralement de type stochastique combinant plusieurs approches heuristiques. Les métaheuristiques sont souvent inspirées des systèmes naturels, qu'ils soient pris en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en écologie (cas des algorithmes de colonies de fourmis). Leur principe est basé sur une recherche aléatoire guidée au sein d'un voisinage de la solution courante, afin d'éviter les optimaux locaux, ces méthodes acceptent dans certaines situations une dégradation provisoire de la fonction économique. Les deux types de méthodes heuristiques et métaheuristiques ne seront pas abordées dans cette thèse.

Des algorithmes de programmation linéaire en nombres entiers ont été développés pour résoudre de façon exacte le problème TSP. En particulier, la procédure par séparation et évaluation. Nous donnons dans la section suivante les grandes lignes de cette méthode. De plus, on dispose des techniques d'optimisation de calculs de bornes inférieures, basés par exemple sur une relaxation Lagrangienne du TSP, permettant de fournir un bon encadrement de l'optimum.

1.6.1 Méthode Branch and Bound

Ce paragraphe est consacré à la méthode par Séparation et Évaluation, appelée en anglais, Branch and Bound (B &B). Cette méthode est le moyen générique le plus utilisé pour la résolution exacte des problèmes d'optimisation combinatoire, et en général pour la résolution des (ILP). Ces méthodes sont basées sur une énumération " intelligente " des solutions admissibles d'un problème d'optimisation combinatoire. L'idée, est de prouver l'optimalité d'une solution en partitionnant l'espace des solutions. Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées. L'algorithme est volontairement présenté dans un cadre relativement simple :

Soit S un ensemble fini mais de " grande " cardinalité qu'on appelle ensemble (ou espace) des solutions réalisables. On dispose d'une fonction f qui, pour toute solution réalisable x de S, renvoie un coût $f(x)$. Le but du problème est de trouver la solution réalisable x de coût minimal. D'un point de vue purement existentiel, le problème est trivial : une telle solution x existe bien car l'ensemble S est fini. En revanche, l'approche effective du problème se confronte à deux difficultés. La première est qu'il n'existe pas forcément un algorithme

1. https ://fr.wikipedia.org/wiki/Métaheuristique

simple pour énumérer les éléments de S. La seconde est que le nombre de solutions réalisables est très grand, ce qui signifie que le temps d'énumération de toutes les solutions est prohibitif (la complexité algorithmique est en général exponentielle).

Dans les méthodes par séparation et évaluation, la séparation permet d'obtenir une méthode générique pour énumérer toutes les solutions tandis que l'évaluation évite l'énumération systématique de toutes les solutions.

- **Séparation** : séparer (brancher) de manière récursive le problème en sous-problèmes de cardinalité inférieure tels que l'union de leurs espaces de solutions forme l'espace des solutions du problème-père, le nœud séparé en priorité est celui qui produit la borne inférieure la plus faible. Un sous-ensemble qui ne peut être séparé est appelé ensemble sondé.
- **Évaluation** : L'évaluation d'un nœud de l'arbre de recherche a pour but de déterminer l'optimum de l'ensemble des solutions réalisables associé au nœud en question ou, au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème (typiquement, qu'il n'y a pas de solution optimale). Lorsqu'un tel nœud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions.

À un nœud donné, l'optimum du sous-problème peut être déterminé lorsque le sous-problème devient " suffisamment simple ". Par exemple, lorsque l'ensemble des solutions réalisable devient un singleton, le problème est effectivement simple : l'optimum est l'unique élément de l'ensemble. Dans d'autres cas, il arrive que par le jeu des séparations, on arrive à un sous-problème dans lequel les décisions " difficiles " ont été prises et qui peut ainsi être résolu en temps polynomial.

Cette recherche par décomposition de l'ensemble des solutions peut être représentée graphiquement par un arbre, où chaque nœud correspond à un sous-problème (voir la Figure 1.4).

La stratégie de cette méthode favorise l'exploration des sous-problèmes possédant la plus petite borne inférieur. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une évaluation supérieure à la valeur optimale.

Les bonnes performances d'un (ILP) reposent essentiellement sur sa capacité à fournir des bornes intéressantes (borne inférieure en problème de minimisation et borne supérieur en problème de maximisation). Les techniques les plus classiques pour le calcul de bornes sont basées sur l'idée de relaxation de certaines contraintes. Un cadre particulièrement intéressant pour la relaxation est celui donné par la dualité Lagrangienne. Son principe consiste à ajouter

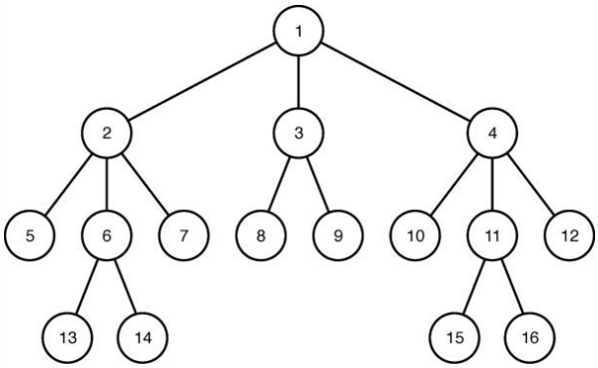


FIGURE 1.4 – Arbre de Branch and Bound

des termes d'une inégalité dans la fonction objectif en ajoutant un multiplicateur visant à pénaliser la fonction objectif si la solution trouvée ne vérifie pas cette inégalité. Cette relaxation fournit fréquemment de très bonne valeur de relaxation, bien souvent meilleure que la relaxation continue (nous allons voir cela en détail dans le prochain chapitre).

CHAPITRE 2

Relaxation et dualité Lagrangienne

Sommaire

L'utilisation d'une procédure de type Branch-and-Bound, nécessite la connaissance d'une borne sur la valeur du problème de programmation linéaire en nombre entiers (ILP). Dans ce chapitre, nous examinons une technique qui donne cette borne par la relaxation Lagrangienne. Nous discutons également des propriétés du problème dual lagrangien, des conditions d'optimalité et de la structure de la fonction objectif dual. La seconde section est une illustration du principe de la relaxation lagrangienne à l'exemple célèbre du voyageur de commerce. En injectant les contraintes de degré dans la fonction objectif du problème TSP, la relaxation lagrangienne serait exprimée comme une relaxation 1- arbre.

2.1 Relaxation Lagrangienne

La relaxation Lagrangienne est une manipulation classique en optimisation sous contraintes. En particulier, elle permet d'obtenir des bornes de la valeur optimale de certains problèmes d'optimisation combinatoire. L'idée consiste à relaxer une partie des contraintes qui rendent le problème compliqué, ces contraintes sont introduites dans la fonction objectif sous la forme d'une pénalité qui combine linéairement les contraintes relaxées. Les

coefficients de cette combinaison linéaire sont appelées les variables duales associées à la relaxation Lagrangienne.

2.1.1 Définitions et résultats généraux

Un exemple très fréquent en pratique se situe dans le cadre des (ILP) où certaines contraintes sont jugées "difficiles" à traiter. On peut alors écrire ici un problème de programmation linéaire en nombres entiers (ILP) de la façon suivante :

$$(ILP)\begin{cases} z^* = \min\ c^\top x \\ \text{s.t.} \quad A_1 x \leq b_1 \\ x \in X = \left\{x \in \mathbb{Z}_+^n : A_2 x \leq b_2\right\}, \end{cases} \tag{2.1}$$

où $x \in \mathbb{R}^{n\times 1}$, $c \in \mathbb{R}^{n\times 1}$, $b_1 \in \mathbb{R}^{m\times 1}$, $b_2 \in \mathbb{R}^{k\times 1}$, $A_1 \in \mathbb{R}^{m\times n}$ et $A_2 \in \mathbb{R}^{k\times n}$ sont des matrices, $\mathbb{Z}_+^n$ est l'ensemble des nombres entiers et X est un ensemble de points discrets dans un polyédrique supposé être non vide et borné.

Le problème (ILP) sera appelé problème primal et sa solution une solution primale. Ici, les contraintes $A_1 x \leq b_1$ sont considérées comme les contraintes complicantes, dans le sens où on suppose que l'on dispose d'un algorithme "efficace" pour minimiser la fonction z sur l'ensemble X. Observez qu'aucune hypothèse particulière n'a été formulée sur le problème primal. les contraintes relâchées sont réinjectées dans la fonction objectif, pondérées par les coefficients $\lambda = (\lambda_1, ..., \lambda_m)^\top \in \mathbb{R}_+^m$.

Définition 2.1 *(Fonction Lagrangienne) On appelle fonction Lagrangienne ou Lagrangien associé au problème (ILP) la fonction $L : X \times \mathbb{R}_+^m \mapsto \mathbb{R}$ définit par*

$$L(x, \lambda) = c^T x + \lambda^T (A_1 x - b_1), \tag{2.2}$$

Les coefficients positifs $\lambda \in \mathbb{R}_+^m$ sont appelés les multiplicateurs de Lagrange.

Remarque 2.1 *Lorsque les m contraintes qui sont dualisée sont des contraintes d'égalité de la forme "$A_1 x = b_1$", les multiplicateurs de Lagrange correspondant sont de signe quelconque $\lambda \in \mathbb{R}^m$.*

Définition 2.2 *(Fonction duale) On appelle fonction duale associée au problème (ILP) $w : \mathbb{R}_+^m \mapsto \mathbb{R}$ la fonction définie par :*

$$(L_\lambda) \qquad \begin{cases} w(\lambda) = \min L(x, \lambda) \\ x \in X. \end{cases} \tag{2.3}$$

Les paramètres $\lambda \in \mathbb{R}_+^m$ sont appelés variables duales.

Définition 2.3 *(Problème dual) On appelle problème dual (D) du problème (ILP) le problème qui consiste à résoudre*

$$(D)\begin{cases} w^* = \max w(\lambda) \\ \lambda \in \mathbb{R}^m_+. \end{cases}$$

Dans ce contexte, le problème de départ (ILP) s'appelle problème primal.

2.1.2 Propriétés du problème et fonction dual

Dans cette section, nous examinons certaines propriétés de la fonction duale liée à la concavité et à la sous-différentiabilité. Nous résumons également les principales propriétés du problème dual Lagrangien (D).

La fonction duale fournit une borne inférieure pour le problème (ILP). La preuve du prochain théorème résulte directement du fait que (L_λ), pour tout $\lambda \in \mathbb{R}^m_+$, est une relaxation de (ILP) et en plus ce théorème montre que toute solution possible du problème dual (D), ainsi que sa valeur maximale w^* est une borne inférieure à la valeur optimale z^* du problème primal (ILP).

Théorème 2.1 *[L.A. Wolsey (1998)]* ***(Théorème de la dualité faible) :*** *Soient les problèmes (ILP), (L_λ) et (D) définis par (2.1),(2.3) et (2.3) respectivement et x une solution réalisable de (ILP). Alors pour tout $\lambda \geq 0$,*

$$w(\lambda) \leq w^* \leq z^*$$

Preuve Soient x une solution réalisable de (ILP) et $\lambda \geq 0$. Par définition de la fonction duale, on a

$$w(\lambda) \leq c^T x + \lambda^T (A_1 x - b_1).$$

Comme $\lambda^T (A_1 x - b_1) \leq 0$, $w^* \leq c^T x$.

En particulier, par définition de w^* et z^* on obtient l'inégalité annoncée,

$$w(\lambda) \leq w^* \leq z^*.$$

■

Théorème 2.2 ***(Concavité de la fonction duale) :*** *La fonction duale $\lambda \to w(\lambda)$ est une fonction concave affine par morceaux.*

Preuve Soient λ^1 et λ^2 deux vecteurs réels positifs et soit $t \in [0,1]$. Alors

$$w(\lambda^1) = \min L(x, \lambda^1)$$
$$x \in X.$$

et,

$$w(\lambda^2) = \min L(x, \lambda^2)$$
$$x \in X.$$

Posons alors $\lambda = (1-t)\lambda_1 + t\lambda_2$. Pour ce λ donné, on sait par hypothèse qu'il existe un $\overline{x} \in X$ tel que

$$w(\lambda) = c^T\overline{x} + \lambda^T(A_1\overline{x} - b_1).$$

Par définition de w appliquée à λ^1 et λ^2, on a

$$w(\lambda^1) \leq c^T\overline{x} + (\lambda^1)^T(A_1\overline{x} - b_1) \qquad et \qquad w(\lambda^2) \leq c^T\overline{x} + (\lambda^2)^T(A_1\overline{x} - b_1).$$

Par combinaison convexe des deux inégalités, on obtient donc

$$(1-t)w(\lambda^1) + tw(\lambda^2) \leq (1-t)(c^T\overline{x} + (\lambda^1)^T(A_1\overline{x} - b_1)) + t(c^T\overline{x} + (\lambda^2)^T(A_1\overline{x} - b_1))$$

Par conséquent,

$$(1-t)w(\lambda^1) + tw(\lambda^2) \leq c^T\overline{x} + ((1-t)\lambda^1 + t\lambda^2)^T(A_1\overline{x} - b_1) = w(\lambda).$$

puisque X est fini, w est le minimum d'un nombre fini de fonctions affines de λ. Donc, $w(\lambda)$ est fonction concave affine par morceaux et cela vient compléter la preuve (voire Figure (2.1). ■

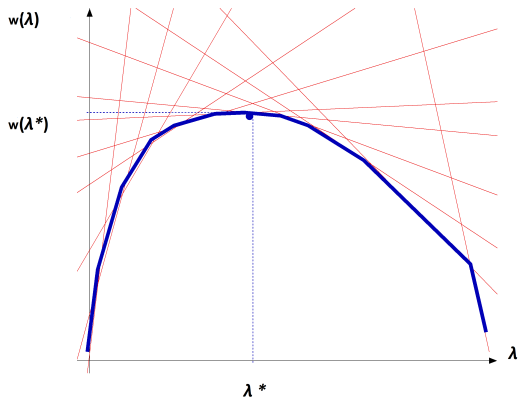


FIGURE 2.1 – Forme de la fonction duale

Ainsi la fonction w est concave, mais pas différentiable en tout point mais la concavité de w permet cependant de montrer que tout optimum local de w est global : ce qui fait

que le problème dual est en général plus facile à résoudre que le problème primal. L'une des questions qui se pose est comment on peut résoudre le problème dual Lagrangien. Les méthodes de sous-gradient qui seront décrites au chapitre (3) sont le plus simple pour résoudre le problème dual Lagrangien et en général pour résoudre les problèmes d'optimisation non différentiable. La méthode est conçue pour résoudre le problème de maximisation (minimisation) d'une fonction concave (convexe) non-différentiable.

Le théorème suivant fournit les conditions pour lesquelles le problème dual optimal donne une solution à (ILP).

Théorème 2.3 ***(Théorème de la dualité forte) :*** *Soient les problèmes* (ILP), (L_λ) *et* (D) *définis par (2.1),(2.3) et (2.3) respectivement. Si*

i) $x_{\overline{\lambda}}$ *est une solution optimale du problème* $(L_{\overline{\lambda}})$ *pour certains* $\overline{\lambda} \in \mathbb{R}^m_+$, *et en plus*

ii)

$$A_1 x_{\overline{\lambda}} \leq b_1 \qquad \textit{(admissibilité)} \tag{2.4}$$

$$\overline{\lambda}^T (A_1 x_{\overline{\lambda}} - b_1) = 0 \qquad \textit{(complémentarité)} \tag{2.5}$$

alors $x_{\overline{\lambda}}$ *est une solution optimale du problème primal* (ILP) *et* $\overline{\lambda}$ *est une solution optimale du problème dual* (D).

Preuve (i) $w^* \geq w(\overline{\lambda}) = c^T x_{\overline{\lambda}} + \overline{\lambda}^T (A_1 x_{\overline{\lambda}} - b_1)$,
par la relation (2.5) $c^T x_{\overline{\lambda}} + \overline{\lambda}^T (A_1 x_{\overline{\lambda}} - b_1) = c^T x_{\overline{\lambda}}$,
par la relation (2.4) $x_{\overline{\lambda}}$ est réalisable pour (ILP) et par conséquent $c^T x_{\overline{\lambda}} \geq z^\star$.
D'après le théorème de dualité faible (2.1) on aura

$$z^\star \geq w^\star \geq w(\overline{\lambda}) = c^T x_{\overline{\lambda}} + \overline{\lambda}^T (A_1 x_{\overline{\lambda}} - b_1) = c^T x_{\overline{\lambda}} \geq z^\star.$$

Alors, $w^\star = z^\star = c^T x_{\overline{\lambda}}$ et $x_{\overline{\lambda}}$ est optimal pour (ILP). De plus, en utilisant ce résultat et le fait que $c^T x_{\overline{\lambda}}$ est une borne supérieure de $w^\star$, on obtient

$$w(\overline{\lambda}) \leq w^\star \leq c^T x_{\overline{\lambda}} = w(\overline{\lambda}),$$

ce qui signifie $w^\star = w(\overline{\lambda})$ et $\overline{\lambda}$ est une solution optimale pour (D). ■

Le théorème suivant caractérise le problème dual Lagrangien. Rappelons que l'enveloppe convexe d'un ensemble X notée $conv(X)$ est l'ensemble de toutes les combinaisons convexes des points de X, i.e.

$$conv(X) = \{\sum_i^k \alpha_i x^i : k \in \mathbb{N}, \sum_i \alpha_i^k = 1, \alpha_i \geq 0, x^i \in X, \forall i = 1, ..., k\}.$$

Théorème 2.4 *(Geoffrion, 1974) : Soient les problèmes (ILP), (L_λ) et (D) tels que définis ci-dessus par (2.1),(2.3) et (2.3) respectivement. Alors la valeur optimale w^* du dual Lagrangien (D) est égale à la valeur optimale du problème suivant :*

$$(P') \begin{cases} \min \ c^\top x \\ s.t. \qquad A_1 x \leq b_1 \\ \qquad\qquad x \in conv(X), \end{cases} \tag{2.6}$$

Preuve Puisque $X \subset \mathbb{Z}_+^n$ contient un nombre fini de points que l'on note $\mathbb{X} = \{x^1, x^2, ..., x^k\}$. Dans ce cas, la fonction de Lagrange est alors

$$w(\lambda) = \min c^T x^i + \lambda^T (A_1 x^i - b_1), i = 1, ..., k.$$

On a

$$w^* = \max_{\lambda \geq 0} \{w(\lambda)\} = \max_{\lambda \geq 0} \{\min c^T x^i + \lambda^T (A_1 x^i - b_1), i = 1, ..., k\}.$$

On peut voir que le dual Lagrangien (D) peut se réécrire comme un problème de programmation linéaire (P'') (généralement de grande taille) avec des variables $(\eta, \lambda) \in \mathbb{R} \times \mathbb{R}_+^m$

$$(P'') \begin{cases} \max \eta \\ \text{s.t.} \quad \eta \leq c^T x^i + \lambda^T (A_1 x^i - b_1), i = 1, ..., k \\ \qquad\qquad (\eta, \lambda) \in \mathbb{R} \times \mathbb{R}_+^m, \end{cases}$$

où la variable η est une borne inférieure de $\{c^T x^i + \lambda^T (A_1 x^i - b_1) : i = 1, ..., k\}$. En utilisant le théorème de dualité en programmation linéaire [J. Teghem (2003)], on obtient le programme dual (D') de (P'')

$$(D') \begin{cases} w^* = \min c^T (\sum_{i=1}^k \alpha_i x^i) \\ \text{s.t.} \sum_{i=1}^k \alpha_i (A_1 x^i - b_1) \geq 0 \\ \sum_{i=1}^k \alpha_i = 1 \\ \alpha_i \in \mathbb{R}_+, \forall i \in \{1, ..., k\} \end{cases}$$

Considérons maintenant un point quelconque $x \in conv(X)$, par définition $x = \sum_i \alpha_i x^i$ avec $\sum_i \alpha_i = 1, \alpha_i \geq 0$ pour tout $i = 1, ..., k$. Par conséquent (D') est exactement équivalent au (P') le programme donné dans l'énoncé du Théorème (2.4). ■

Remarque 2.2 *Dans le cas où $conv(X) \subsetneq \{x \in \mathbb{R}^n : A_1 x \leq b_1\}$, ce qui est le plus fréquent en (ILP), la relaxation Lagrangienne donne donc, une meilleure évaluation que la relaxation linéaire.*

La qualité de la borne obtenue par la relaxation Lagrangienne dépend de la structure du problème original. Nous rencontrerons au paragraphe (2.2) suivant une application pour le problème de voyageur de commerce TSP pour laquelle la relaxation Lagrangienne est très efficace.

2.2 Application de la relaxation lagrangienne au problème du voyageur de commerce

Soit $G = (V, E)$ un graphe pour lequel on recherche un cycle hamiltonien de plus petite longueur où c_e ; $e \in E$, est la longueur associée à une arête. Le problème est donc de trouver un tour à poids minimal

$$z^{TSP} = \min_{T \subseteq E} \left\{ \sum_{e \in T} c_e : T \text{ est un tour} \right\}$$

Une relaxation intéressante est obtenue en remarquant :

a) Chaque tour est constitué de deux arêtes adjacentes à 1 sommet et un chemin reliant les sommets $\{2, ..., n\}$.

b) Un chemin est un arbre spécial.

Définition 2.4 *Un 1-arbre est un sous graphe de deux arêtes adjacentes au sommet* 1*, additionnées aux arêtes d'un arbre de sommets* $\{2, ..., n\}$.

Remarque 2.3 *1) Il est clair que chaque tour est un 1-arbre, d'où*

$$z^{TSP} \geq z^{1-arbre} = \min_{T \subseteq E} \left\{ \sum_{e \in T} c_e : T \text{ est un 1-arbre} \right\} \tag{2.7}$$

2) Un tour est un 1-arbre ayant deux arête incidentes de chaque sommet.

3) La relation (2.7), indique que le problème 1-arbre est une relaxation pour le TSP.

Trouver une solution 1-arbre optimale associée à un problème TSP est simple, comme le montre l'exemple suivant :

Exemple 2.1 *Considérons un problème TSP dont la matrice des coûts est*

$$C_e = \begin{bmatrix} - & 30 & 26 & 50 & 40 \\ - & - & 24 & 40 & 50 \\ - & - & - & 24 & 26 \\ - & - & - & - & 30 \\ - & - & - & - & - \end{bmatrix}$$

La solution optimale 1-arbre est calculée en prenant les deux arêtes à coût minimal sortant du sommet 1, i.e; l'arête $(1,2)$ *et* $(1,3)$*, plus les arêtes d' un arbre optimal des sommets* $\{2,3,4,5\}$*, qui est donné par les arêtes* $(3,4),(3,5)$ *et* $(2,3)$*.*

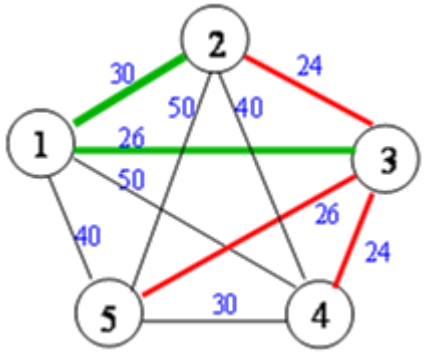


FIGURE 2.2 – Exemple TSP de Cinq villes

Considérons le problème TSP spécifié par le graphe $G=(V,E)$ et c_e le poids d'une arête $e \in E$. :

$$z = \min \sum_{e \in E} c_e x_e \tag{2.8}$$

$$s.c: \sum_{e \in \delta(i)} x_e = 2 \qquad \textit{pour tout } i \in V \tag{2.9}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad 2 \leq |S| \leq |V| - 1 \tag{2.10}$$

$$x_e \in \{0,1\} \qquad \textit{pour tout } e \in E. \tag{2.11}$$

La moitié des sous-tours introduit dans les contraintes (2.10) sont redondants. on a :

$$|S| - \sum_{e \in E(s)} x_e = \frac{1}{2} \sum_{i \in S} \sum_{e \in \delta(i)} x_e - \sum_{e \in E(s)} x_e = \frac{1}{2} \sum_{e \in \delta\left(S,\overline{S}\right)} x_e,$$

où $\delta\left(S,\overline{S}\right)$ est l'ensemble des arêtes avec un point final dans S et un autre dans $\overline{S} = V/S$, et donc comme $\delta\left(S,\overline{S}\right) = \delta\left(\overline{S},S\right)$, $|S| - \sum\limits_{e \in E(s)} x_e = \left|\overline{S}\right| - \sum\limits_{e \in E(\overline{S})} x_e$ donc,

$\sum\limits_{e \in E(s)} x_e \leq |S| - 1$ si seulement si $\sum\limits_{e \in E(\overline{s})} x_e \leq \left|\overline{S}\right| - 1,$

en additionnant les contraintes (2.9) et divisant par 2 , nous obtenons :

$$\sum_{e \in E} x_e = n.$$

Donc, supprimons toutes les contraintes sous-tours avec $1 \in S$. Nous obtenons une relaxation Lagrangienne, en dualisant toutes les contraintes de degré sur les sommets sauf le sommet 1.

$$(L_\lambda)\begin{cases} w(\lambda) = \min \sum_{e\in E}(c_e - \lambda_i - \lambda_j)x_e + 2\sum_{i\in V}\lambda_i \\ \sum_{e\in\delta(1)} x_e = 2 \\ \sum_{e\in E(s)} x_e \leq |S| - 1 \qquad 2 \leq |S| \leq |V| - 1, \quad 1 \notin S \\ \sum_{e\in E} x_e = n \\ x_e \in \{0,1\}, \end{cases}$$

les solutions réalisables de (L_λ) sont précisément les 1-arbre.

Exemple 2.2 *Reprenant l'exemple cité ci-dessus*

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

Soit la valeur dual $\lambda = (0, 0, -15, 0, 0, 0)$*, la matrice* $\overline{c}_e$ *des coûts réduit est*

$$\overline{c}_e = c_e - \lambda_i - \lambda_j = \begin{pmatrix} - & 30 & 41 & 50 & 40 \\ - & - & 39 & 40 & 50 \\ - & - & - & 39 & 41 \\ - & - & - & - & 30 \\ - & - & - & - & - \end{pmatrix}.$$

La solution optimale 1-arbre est calculée en prenant les deux arêtes à coût minimal sortant du sommet 1, i.e; l'arête $(1, 2)$ *et* $(1, 5)$*, plus les arêtes d' un arbre optimal des sommets* $\{2, 3, 4, 5\}$*, qui est donné par les arrêtes* $(4, 5)$*,* $(2, 3)$ *et* $(3, 4)$*.*

Il se trouve que la solution optimale 1-arbre $1-2-3-4-5-1$*, est aussi un tour, donc une solution optimale.*

$w(\lambda) = (30 + 39 + 39 + 30 + 40) - 2\sum\lambda_i = 178 - 30 = 148$

$w(\lambda) = 148 \leq z$ *alors est une solution optimale.*

CHAPITRE 3

Méthodes sous-gradient en optimisation

Sommaire

Ce chapitre est consacré aux méthodes d'optimisation de type sous-gradient qui peuvent être utilisées pour résoudre le problème dual Lagrangien (D) (donné par (2.3)) de la programmation en nombres entiers, (ILP) définie par (2.1) où la fonction objectif w n'est pas différentiable en tout point.

3.1 Introduction

Les problèmes d'optimisation non différentiable, même en l'absence de contraintes, sont en général très difficiles à résoudre. Les méthodes de sous-gradient sont une généralisation naturelle des méthodes de gradient au cas non différentiable en remplaçant le gradient par un sous-gradient arbitraire. Elles ont été introduites dans les années 60 par N.Z. Shor [N.Z. Shor(1962)] pour la minimisation sans contrainte de fonctions convexes. Les premiers résultats de convergence sont dus à [B.T. Polyak (1967), B.T. Polyak (1969)] ou [A.S. Nemirovskii (1978)]. Une référence classique qui présente un aperçu général des premiers développements sur les méthodes sous-gradient est le livre de Shor [N.Z. Shor(1985)].

Le succès de ces méthodes est dû à la simplicité de leur mise en œuvre, en particulier en grande dimension. Bien qu'elles soient largement utilisées en optimisation non différentiable, elles souffrent de plusieurs inconvénients : contrairement aux méthodes de gradient en optimisation différentiable, ce ne sont pas des méthodes de descente. Cela peut conduire à des phénomènes oscillatoires (zig-zag) et à de mauvaises performances numériques. On ne dispose pas de critère d'arrêt naturel et implémentable permettant de détecter l'optimalité. Enfin, leur vitesse de convergence est en général assez faible (sous-linéaire).

La méthode d'optimisation de sous-gradient est une procédure itérative qui peut être utilisée pour résoudre le problème de la maximisation d'une fonction concave non-différentiable $w(\lambda)$ sur un ensemble convexe fermé Ω, i.e.,$\max_{\lambda\in\Omega} w(\lambda)$ en utilisant la procédure générique suivante :

1. Choisissez un point initial $\lambda_0 \in \Omega$.
2. Construire une séquence de points $\left(\lambda^k\right)_k \subset \Omega$ qui converge finalement à une solution optimale en utilisant la règle $\lambda^{k+1} = P_\Omega\left(\lambda^k + t_k s^k\right)$, où $P_\Omega(.)$ est un opérateur de projection sur l'ensemble Ω et t_k est un scalaire positif appelé pas et s^k est un vecteur(direction de déplacement), qui doit être déterminé à chaque itération k.
3. Remplacer k par $k+1$ et répétez le processus jusqu'à certains critères d'arrêt.

La direction s^k qui doit être déterminée à chaque itération de la procédure joue un rôle important afin d'être en mesure d'obtenir un résultat souhaité. Selon une stratégie particulière pour trouver la direction du déplacement, deux types de méthodes d'optimisation de sous-gradient sont discutés ici, à savoir la méthode de sous-gradient pur et la méthode de sous-gradient dévié.

La méthode du sous-gradient pur utilise un sous-gradient (la définition formelle sera donnée à la section 3.2) de la fonction objectif à chaque point comme direction de progression, pour générer une séquence d'itérations formant un angle obtus avec la direction précédente de déplacement, alors un phénomène de zig-zag va apparaitre (voir Figure 3.1). Un tel phénomène pourrait se manifester à n'importe quel étape de l'algorithme du sous-gradient provoquant une convergence lente de la procédure. A fin de surmonter cette difficulté, une procédure de sous-gradient dévié est utilisée dans laquelle la direction dévié est calculée en combinant le sous-gradient courant avec la direction précédente. Cette stratégie est appelée méthode du sous-gradient dévié et c'est l'objet de la section (4.2). Nous devons toujours projeter les points d'itération générés par la procédure de sous gradient pur ou dévié sur l'ensemble Ω afin de maintenir l'admissibilité.

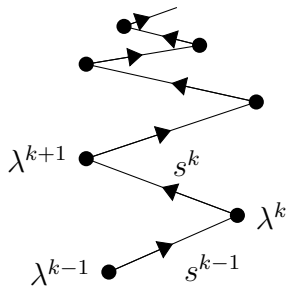


FIGURE 3.1 – Phénomene de zig-zag.

3.2 La méthode du sous-gradient pur

3.2.1 Introduction

La valeur maximale d'une fonction concave différentiable peut être généralement déterminée par les méthodes du gradient. Une méthode du gradient, par exemple la méthode de la plus grande pente, trouve une solution optimale du problème $\max_\lambda w(\lambda)$ par méthode itérative dans laquelle, à partir de λ^0, une suite de λ^k convergeant finalement vers une solution optimale est construite selon la relation

$$\lambda^{k+1} = \lambda^k + t_k \nabla w(\lambda^k),$$

où $t_k \in \mathbb{R}_+$ est un pas approprié et $\nabla w(\lambda^k)$ est le vecteur gradient de w en λ^k. Pour plus de détails sur ce sujet et sur des sujets connexes, le lecteur peut se référer à [J. Nocedal (1999)].

Dans le cas de notre problème, les gradients sont remplacés par des sous-gradients afin d'utiliser la structure de la concavité de la fonction duale.

Définition 3.1 *(**sous-gradient**) : Soit $f : \mathbb{R}^m \to \mathbb{R}$ une fonction concave*[1] *. Un vecteur $s \in \mathbb{R}^m$ est appelé sous-gradient de f au point $\overline{x} \in X$ si*

$$f(x) \leq f(\overline{x}) + s^\top (x - \overline{x}) \qquad \forall x \in \mathbb{R}^m. \tag{3.1}$$

Définition 3.2 *(**sous-différentiel**) : L'ensemble de touts les sous-gradient de f au point $\overline{x}$ notée $\partial f(\overline{x})$*

$$\partial f(\overline{x}) = \left\{ s : f(x) \leq f(\overline{x}) + s^\top (x - \overline{x}) \quad \forall x \in \mathbb{R}^m \right\}$$

est appelé le sous-différentiel de f au point $\overline{x}$.

1. Pour une fonction convexe l'inégilaté dans la relation (3.1) est inversée.

Si $\partial f(\overline{x})$ est non vide, on dit que f sous-différentiable en $\overline{x}$. On sait qu'une fonction concave est sous-différentiel à chaque point de l'intérieur de son domaine. En outre, son sous-différentiel est un ensemble convexe, fermé et borné non vide. Une fonction concave n'est pas nécessairement différentiable à tous les points de son domaine. Comme le montre le théorème suivant, si une fonction concave est différentiable en un point $\overline{x}$ alors $\nabla f(\overline{x})$ est un sous-gradient de f en ce point. En ce sens, le sous-gradient est considéré comme un gradient généralisé d'une fonction concave. Pour une étude exhaustive au calcul sous-différentiel nous renvoyons le lecteur aux ouvrages classiques [R.T. Rockafellar (1970)] et [V.F. Dem'yanov(1985)].

Théorème 3.1 *Soit $f : \mathbb{R}^m \to \mathbb{R}$ concave et différentiable. Alors, $\nabla f(\overline{x}) \in \partial f(\overline{x})\ \forall x \in \mathbb{R}^m$.*

Preuve Il suffit de montrer que pour tout $\overline{x} \in \mathbb{R}^m$

$$f(x) \leq \nabla f(\overline{x})(x - \overline{x}) + f(\overline{x}) \quad \forall x \in \mathbb{R}^m.$$

Pour $x = \overline{x}$, cette inégalité tient évidemment. Nous devons donc considérer seulement le cas $x \neq \overline{x}$. Puisque f est différentiable, la dérivée directionnelle de f en $\overline{x}$ dans la direction de $x - \overline{x}$, donnée par

$$lim_{t\to 0^+} \frac{f(\overline{x} + t(x - \overline{x})) - f(\overline{x})}{t},$$

existe et est égal à $\nabla f(\overline{x})(x-\overline{x})$. Puisque f est concave, ce qui suit est valable pour $t \in (0, 1)$

$$\begin{aligned} f(x) - f(\overline{x}) &= \frac{tf(x) + (1-t)f(\overline{x}) - f(\overline{x})}{t} \\ &\leq \frac{f(tx + (1-t)\overline{x}) - f(\overline{x})}{t} \\ &= \frac{f(\overline{x} + t(x - \overline{x})) - f(\overline{x})}{t}, \end{aligned}$$

ce qui implique

$$f(x) - f(\overline{x}) \leq lim_{t\to 0^+} \frac{f(\overline{x} + t(x - \overline{x})) - f(\overline{x})}{t} = \nabla f(\overline{x})(x - \overline{x}).$$

Ceci termine la preuve. ■

Notons que, la définition de la concavité signifie qu'un sous gradient est un vecteur de l'hyperplan supportant **l'hypographe** de f en $(\overline{x}, f(\overline{x}))$, où **l'hypographe** de f est

$$hyp\, f = \left\{(x, z) \in \mathbb{R}^{m+1} : z \leq f(x)\right\},$$

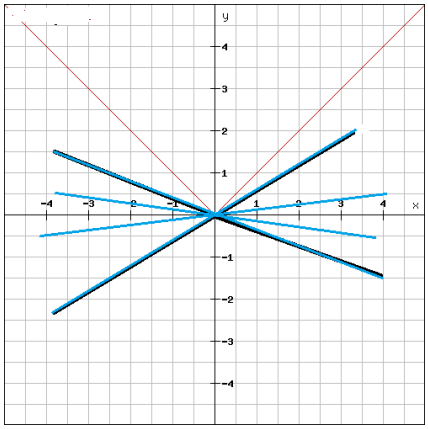
qui est un ensemble convexe fermé.

Si la fonction concave f est également différentiable en $\overline{x}$, alors on sait qu'un tel hyperplan support est uniquement déterminé par le gradient $\nabla f(\overline{x})$. Cela signifie que le sous-gradient de f en $\overline{x}$ est déterminé de manière unique et donné par $\nabla f(\overline{x})$. Nous avons donc le théorème suivant.

Théorème 3.2 *Si $\nabla f(x)$ existe, alors $\partial f(x)$ est un singleton et $\partial f(x) = \{\nabla f(x)\}$.*

Cependant, à un point $\overline{x}$ où la fonction est non différentiable, nous pouvons avoir un nombre infini d'éléments dans l'ensemble sous-différentiel.

Exemple 3.1 *Soit $f : x \in \mathbb{R} \mapsto |x|$. Calculons les sous-gradients de f (convexe) en tout point x de $\mathbb{R}$.*



Dans $\mathbb{R}^2$, les hyperplans d'appui sont des droites et les sous-gradients associés sont leurs coefficients directeurs.

$$\partial f(x) = \begin{cases} \{1\} & \text{si } x < 0 \\ \{-1\} & \text{si } x > 0 \\ [-1, 1] & \text{si } x = 0. \end{cases}$$

Théorème 3.3 *Le sous-différentiel $\partial f(x)$ de f en $x \in \mathbb{R}^n$ est un ensemble convexe.*

Preuve Supposons que $s_1, s_2 \in \partial f(x)$. Alors

$$s_1^T(y - x) \geq f(y) - f(x), \quad \forall y \in \mathbb{R}^n$$

et

$$s_2^T(y - x) \geq f(y) - f(x), \quad \forall y \in \mathbb{R}^n$$

Donc, pour tout $t \in [0, 1]$ et $y \in \mathbb{R}^n$ nous avons,

$$\begin{aligned} [ts_1 + (1-t)s_2]^T (y - x) &= ts_1^T(y - x) + (1-t)s_2^T(y - x) \\ &\geq t\,(f(y) - f(x)) + (1-t)\,(f(y) - f(x)) \\ &= f(y) - f(x), \end{aligned}$$

qui, par la définition d'un sous-gradient, on a

$$ts_1 + (1-t)s_2 \in \partial f(x),$$

et ceci complète la preuve. ■

Théorème 3.4 *Une condition nécessaire et suffisante pour que $x^* \in \mathbb{R}^n$ soit un maximum d'une fonction concave f sur $\mathbb{R}^n$ est $0 \in \partial f(x^*)$.*

Preuve Par la définition du sous-gradient, $0 \in \partial f(x^*)$ pour $x \in \mathbb{R}^n$ si et seulement

$$f(x) \leq f(x^*) + 0(x - x^*) \qquad \forall x \in \mathbb{R}^n,$$

Ce qui est équivalent à

$$f(x) \leq f(x^*) \qquad \forall x \in \mathbb{R}^n.$$

■

Remarque 3.1 *La condition $0 \in \partial f(x^*)$ est une généralisation de la condition stationnaire usuelle $\nabla f(x^*) = 0$ dans le cas différentiable.*

3.2.2 L'algorithme de sous-gradient pur

Le schéma standard de la méthode sous-gradient de base pour résoudre (D) est le suivant :

On commence par un certain point λ^0 et on construit une séquence de points $\{\lambda^k\}$ selon la règle

$$\lambda^{k+1} = P^+\left(\lambda^k + t_k s^k\right), \quad k = 0, 1, 2, ... \tag{3.2}$$

où s^k est le sous gradient de w au point λ^k, t_k est le pas, et P^+ est l'opérateur projection, de $\mathbb{R}^m$ sur $\mathbb{R}^m_+$ c-à-d :

$$P^+(\lambda) = \max(0, \lambda) = (\max(0, \lambda_1), ..., \max(0, \lambda_m))^\top.$$

Le théorème suivant donne une méthode simple de calcul des vecteurs sous-gradient à chaque point.

Théorème 3.5 *Soit x_λ une solution optimale du problème (L_λ). Alors : $s = A_1 x_\lambda - b_1$ est un sous-gradient de w au point λ.*

Preuve On a :

$$w(\lambda) = c^T x_\lambda + \lambda^\top (A_1 x_\lambda - b_1)$$

Pour $\mu \in \mathbb{R}^m_+$, nous avons :

$$w(\mu) = \min_{x \in X} (c^T x + \mu^\top (A_1 x - b_1)$$

$$\begin{aligned} &\leq c^T x_\lambda + \mu^\top (A_1 x_\lambda - b_1) \\ &= c^T x_\lambda + \lambda^\top (A_1 x_\lambda - b_1) + \underbrace{(\mathbf{A_1 x_\lambda - b_1})^\top}(\mu - \lambda) \\ &= \quad w(\lambda) + s^\top (\mu - \lambda)\,, \end{aligned}$$

donc $s = A_1 x_\lambda - b_1$ est un sous-gradient de la fonction dual w. ■

À un point donné, nous n'avons aucun sous-gradient unique de la fonction. Cela pose certaines difficultés en ce qui concerne la construction d'une bonne procédure itérative qui utilise un vecteur de sous gradient comme direction de progression.

En général dans le cas différentiable, il est bien connu que le vecteur gradient est un vecteur de descente, ce n'est pas le cas pour un vecteur de sous-gradient et donc la procédure d'itération de la méthode de sous-gradient n'améliore pas nécessairement la valeur de la fonction objectif à certaines étapes. En conséquence, les techniques de recherche linéaire dans le cas différentiable ne sont pas applicables pour déterminer une longueur de pas appropriée t_k dans la procédure de sous-gradient.

3.3 Choix des pas t_k

L'algorithme suivant de type sous-gradient est une méthode itérative qui génère une suite de points $\{\lambda^k\}$ où λ^1 est un point initial choisi dans X.

Algorithm 1 (Algorithme sous-gradient pour le dual Lagrangien)

1. Choisissons un vecteur initiale λ^1, et posons $v^1 = -\infty$, $k = 1$.
2. Déterminer un vecteur du sous-gradient s^k à x^k en résolvant le sous-problème Lagrangien
$$(L_\lambda) \qquad \begin{cases} w(\lambda^k) = \min c^T x^k + (\lambda^k)^T (A_1 x^k - b_1) \\ x^k \in X. \end{cases} \tag{3.3}$$
soit x^k la solution de sous-problème (L_λ). Alors $s^k = A_1 x^k - b_1$,
3. Si $\|s^k\| \leq \epsilon (\epsilon = 10^{-4})$. STOP. $w^* = v^k$. Sinon aller à l'étape 4.
4. Soit
$$\lambda^{k+1} = P^+(\lambda^k + t_k s^k), \tag{3.4}$$
et
$$v^{k+1} = \max(v^k, w(\lambda^k)), \tag{3.5}$$
posons $k = k + 1$, et aller à l'étape 2.

Nous avons le lemme de base suivante de la procédure 1.

Lemme 3.1 *soit $\lambda^* \geq 0$ une solution optimal pour (D), alors pour tout k ,on a*

$$w(\lambda^*) - v^k \leq \frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k}\left(\frac{t_i}{\|s^i\|}\right)}. \tag{3.6}$$

Preuve puisque s^i est un sous gradient de w à λ^i nous avons :

$$w(\lambda^*) \leq w\left(\lambda^i\right) + \left(s^i\right)^\top \left(\lambda^* - \lambda^i\right).$$

Ainsi,

$$\begin{aligned} \|\lambda^{i+1} - \lambda^*\|^2 &= \left\|P^+\left(\lambda^i + t_i\frac{s^i}{\|s^i\|}\right) - P^+(\lambda^*)\right\|^2 \\ &\leq \left\|\lambda^i + t_i\frac{s^i}{\|s^i\|} - \lambda^*\right\|^2 \\ &= \|\lambda^i - \lambda^*\|^2 + 2\left(\frac{t_i}{\|s^i\|}\right)(s^i)^\top(\lambda^i - \lambda^*) + t_i^2 \\ &\leq \|\lambda^i - \lambda^*\|^2 + 2\left(\frac{t_i}{\|s^i\|}\right)(w(\lambda^i) - w(\lambda^*)) + t_i^2 \end{aligned}, \tag{3.7}$$

additionnons (3.7) pour $i = 1, ..., k$, on obtient :

$$\begin{aligned} 0 &\leq \left\|\lambda^{k+1} - \lambda^*\right\|^2 \\ &\leq \left\|\lambda^1 - \lambda^*\right\|^2 + 2\sum_{i=1}^{k}\left(\frac{t_i}{\|s^i\|}\right)\left[w(\lambda^i) - w(\lambda^*)\right] + \sum_{i=1}^{k} t_i^2, \end{aligned}$$

donc

$$\begin{aligned} w(\lambda^*) - v^k &= w(\lambda^*) - \max_{i=1,...,k} w(\lambda^i) \\ &\leq \frac{\sum_{i=1}^{k}\left(\frac{t_i}{\|s^i\|}\right)[w(\lambda^*) - w(\lambda^i)]}{\sum_{i=1}^{k}\left(\frac{t_i}{\|s^i\|}\right)} \\ &\leq \frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k}\left(\frac{t_i}{\|s^i\|}\right)}. \end{aligned}$$

■

Plusieurs approches ont été proposées dans la littérature pour choisir la suite des pas t_k pour que la méthode de type sous-gradient converge. Dans la suite, nous allons discuter certains d'entre eux.

Règle (1) : Pour un pas $t_k = \varepsilon$ où $\varepsilon > 0$ (on a pas constant indépendant de k).

Règle (2) : Une règle classique garantissant la convergence théorique de l'algorithme consiste à choisir une série divergente de carré sommable pour le pas qui satisfait :

$$\sum_{k=1}^{+\infty} {t_k}^2 < +\infty \quad et \quad \sum_{k=1}^{+\infty} t_k = +\infty,$$

Règle (3) : Dans [B.T. Polyak (1967)], Polyak à démontré que la suite $\left\{w(\lambda^k)\right\}$ converge vers la valeur optimale w^* recherchée si la suite des pas t_k satisfait les conditions suivantes

$$t_k \longrightarrow 0, k \longrightarrow \infty \; et \sum_{k=1}^{+\infty} t_k = +\infty. \tag{3.8}$$

Notons qu'il existe $M > 0$ tel que $\left\|s^k\right\| = \left\|A_1 x^k - b_1\right\| \leq M$ pour tout k puisque $x^k \in X$ et X est un ensemble fini de nombre entier .

Théorème 3.6 (i) *Si la règle* (1) *est utilisée dans l'algorithme (1) alors*

$$\lim_{k \longrightarrow +\infty} \inf \; v^k \geq w(\lambda^*) - \frac{1}{2}\varepsilon M \quad . \tag{3.9}$$

(ii) *Si la règle* (2) *ou la règle* (3) *pour le choix du pas est utilisée dans l'algorithme (1) alors*

$$\lim_{k \longrightarrow +\infty} v^k \geq w(\lambda^*) \qquad . \tag{3.10}$$

Preuve (i) On note que $\left\|s^k\right\| \leq M$ pour tout k, par la relation (3.6) nous avons :

$$w(\lambda^*) - v^k \leq \frac{\|\lambda^1 - \lambda^*\|^2 + \varepsilon^2 k}{2\frac{\varepsilon k}{M}} \longrightarrow \frac{1}{2}\varepsilon M \quad , (k \longrightarrow +\infty)$$

d'où (3.9).

(ii) Si le pas (2) est utilisé, alors par le lemme (3.1) nous avons :

$$0 \leq w(\lambda^*) - v^k \leq \frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k}\left(\frac{t_i}{M}\right)} \longrightarrow 0 \quad , (k \longrightarrow +\infty) \quad . \tag{3.11}$$

Donc (3.11) est vérifiée .

Si le pas (3) est utilisé, nous confirmons que le côté droit de (3.11) converge vers 0. Sinon, il doit exister $\eta > 0$ tels que :

$$\frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k}\left(\frac{t_i}{M}\right)} \geq \eta \quad , \forall k,$$

ou

$$\sum_{i=1}^{k} t_i^2 - 2\left(\frac{\eta}{M}\right)\sum_{i=1}^{k} t_i \geq -\left\|\lambda^1 - \lambda^*\right\|^2 \ , \forall k. \tag{3.12}$$

Comme $t_i \to 0$ $(i \to \infty)$, il existe N_1 telle que $t_i \leq \frac{\eta}{M}$ où $i > N_1$. Donc :

$$\begin{aligned}
& \sum_{i=1}^{k} t_i^2 - 2\left(\frac{\eta}{M}\right)\sum_{i=1}^{k} t_i \\
= & \left(\sum_{i=1}^{N_1} t_i^2 + \sum_{i=N_1+1}^{k} t_i^2\right) - \left(\frac{\eta}{M}\right)\left(\sum_{i=1}^{k} t_i + \sum_{i=1}^{k} t_i\right) \\
\leq & \sum_{i=1}^{N_1} t_i^2 + \left(\frac{\eta}{M}\right)\sum_{i=N_1+1}^{k} t_i - \left(\frac{\eta}{M}\right)\left(\sum_{i=1}^{k} t_i + \sum_{i=N_1+1}^{k} t_i\right) \\
= & \sum_{i=1}^{N_1} t_i^2 - \left(\frac{\eta}{M}\right)\sum_{i=1}^{k} t_i \to -\infty \quad (k \to \infty)\,.
\end{aligned}$$

Ceci contredit 3.12. ■

Evidemment, un algorithme de sous-gradient basé sur la régle 3 n'offre pas un grand intérêt en pratique, car la convergence vers un point optimal est généralement excessivement lente.

C'est pourquoi nous étudierons d'autres procédés de choix des paramètres de déplacement t_k qui permettent, sous certaines conditions, d'obtenir de meilleures vitesses de convergence, et, plus précisément, la règle suivante :

Règle 4 [1] : Held, Wolfe et Crowder ont proposé d'utiliser la suite de pas définie comme suit :

$$t_k = \rho_k \frac{\overline{w} - w(\lambda^k)}{\|s^k\|^2}, \tag{3.13}$$

où ρ_k est un coefficient de relaxation satisfaisant $0 < \rho_k \leq 2$, s^k dénote un sous-gradient de w à λ^k et $\overline{w}$ est une estimation [2] de la valeur optimale w^*, $\overline{w} \geq w(\lambda^k)$ et $s^k \neq 0$ $\forall k$.

Sous ces conditions, ils ont démontré que $\{w(\lambda^k)\} \longrightarrow \overline{w}$ ou qu'il existe un k tel que $w(\lambda^k) \leq \overline{w}$. La justification de cette formule est trouvé dans [M.H. Held(1974)].

1. Le pas donnée par (3.13) est généralement connue sous le nom de relaxation ou aussi de pas de Polyak.
2. En pratique, comme on ne connaît pas w^* (c'est précisément la valeur cherché) on devra se contenter d'une estimation $\overline{w}$ de w^* qui peut être obtenue en appliquant une heuristique au problème primale (ILP)

3.4 Application au problème TSP

Algorithm 2 (Algorithme de sous gradient pour le dual Lagrangienne au problème TSP)

1. (Initialisation) : Choisissons n'importe quel $\lambda^1 \geq 0$, posons $v^1 = -\infty$, $k = 1$.
2. Résoudre le problème Lagrangien
   (L_{λ^k}) $\quad w(\lambda^k) = \min \sum_{e\in E}(c_e - \lambda_i^k - \lambda_j^k)x_e + 2\sum_{i\in V}\lambda_i^k$,
   et obtenir un x_e^k solution optimale,
   calculons $\quad s^k = 2 - \sum_{e\in\delta(i)} x_e$ et $\quad v^{k+1} = \max(v^k, w(\lambda^k))$,
   si $s^k = 0$ on s'arrête et λ^k est la solution optimale de (D).
3. Calculons
   $$\lambda^{k+1} = \lambda^k + t_k \frac{s^k}{\|s^k\|},$$
   où
   $$t_k = \rho_k \frac{\overline{w} - w(\lambda^k)}{\|s^k\|^2}, 0 < \rho_k \leqslant 2.$$
4. Posons $k = k + 1$, et allons en l'étape 2.

Exemple 3.2

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

Supposons que nous avons trouvé le tour $1-2-3-4-5-1$ *de coût* 148. *. Nous utilisons* $\rho = 1$ *et* $\overline{w} = 148$.

$$\begin{aligned}
\lambda^{k+1} &= \lambda^k + t_k \frac{s^k}{\|s^k\|} \\
&= \lambda^k + \rho \frac{\overline{w} - w(\lambda^k)}{\| s^k \|} \frac{s^k}{\|s^k\|} \\
&= \lambda^k + \frac{\overline{w} - w(\lambda^k)}{\| s^k \|^2} s^k \\
&= \lambda^k + \frac{148 - w(\lambda^k)}{\left\| \left(2 - \sum\limits_{e \in \delta(i)} x_e \right) \right\|^2} \left(2 - \sum_{e \in \delta(i)} x_e \right) \\
&= \lambda^k + \frac{148 - w(\lambda^k)}{\left\| \left(2 - \sum\limits_{e \in \delta(i)} x_e \right) \right\|^2} \left(2 - \sum_{e \in \delta(i)} x_e \right).
\end{aligned}$$

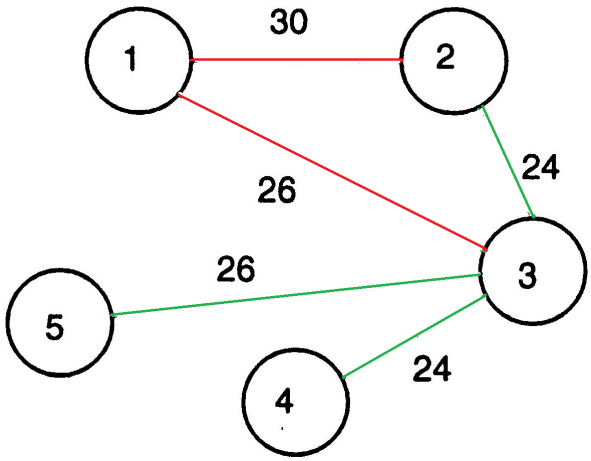
Itération 1 : $\lambda^1 = (0,0,0,0,0)$,

$$c_e - \lambda_i^1 - \lambda_j^1 = (c_e)\,,$$

$$w(\lambda^1) = 30+26+24+26+24 = 130,$$

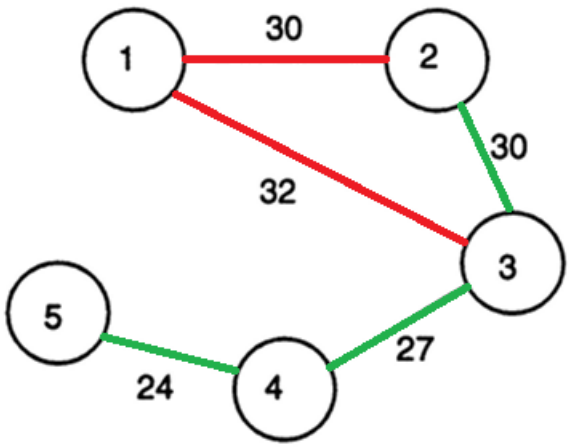
$$\left(2 - \sum_{e \in \delta(i)} x_e \right) = (0,0,-2,1,1),$$

$$\begin{aligned}
\lambda^2 &= \lambda^1 + \frac{148 - w(\lambda^1)}{\|(0,0,-2,1,1)\|^2} (0,0,-2,1,1) \\
&= \lambda^1 + \frac{148-130}{6} (0,0,-2,1,1) \\
&= (0,0,-6,3,3)\,.
\end{aligned}$$



Itération 2 : $\lambda^2 = (0, 0, -6, 3, 3)$,

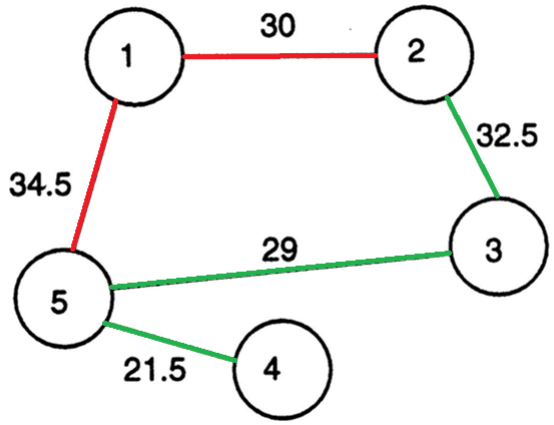
$$\left(c_e - \lambda_i^2 - \lambda_j^2\right) = \begin{pmatrix} - & 30 & 32 & 47 & 37 \\ - & - & 30 & 37 & 47 \\ - & - & - & 27 & 29 \\ - & - & - & - & 24 \\ - & - & - & - & - \end{pmatrix},$$



$$w(\lambda^2) = 143 + \sum_{i=1}^{6} \lambda_i^2 = 143,$$
$$\left(2 - \sum_{e \in \delta(i)} x_e\right) = (0, 0, -1, 0, 1)\,,$$
$$\lambda^3 = \lambda^2 + \frac{148 - w(\lambda^2)}{\|(0,0,-1,0,1)\|^2}\,(0, 0, -1, 0, 1)$$
$$= \lambda^2 + \frac{148-143}{2}\,(0, 0, 1, 0, -1)$$
$$= (0, 0, -8.5, 3, 5.5)\,.$$

Itération 3 : $\lambda^3 = (0, 0, -8.5, 3, 5.5)$,

$$\left(c_e - \lambda_i^2 - \lambda_j^2\right) = \begin{pmatrix} - & 30 & 43.5 & 47 & 34.5 \\ - & - & 32.5 & 37 & 44.5 \\ - & - & - & 29.5 & 29 \\ - & - & - & - & 21.5 \\ - & - & - & - & - \end{pmatrix},$$



$$w(\lambda^3) = 147.5 + \sum_{i=1}^{6} \lambda_i^3 = 147.5.$$

Comme les données de notre problème sont entières, $w^* \leq \lceil w(\lambda^3) \rceil = \lceil 147.5 \rceil = 148$ *est solution admissible de coût 148, on obtient alors le tour optimale* $1 - 2 - 3 - 4 - 5 - 1$.

CHAPITRE 4

Une nouvelle méthode de sous gradient dévié

Sommaire

4.1 Introduction

Dans ce chapitre, nous reconsidérons le programme linéaire en nombre entier suivant :

$$(ILP)\begin{cases} z^* = \min\ c^\top x \\ \text{s.t.} \quad A_1 x \leq b_1 \\ x \in X = \left\{x \in \mathbb{Z}_+^n : A_2 x \leq b_2\right\}, \end{cases} \tag{4.1}$$

où x est un vecteur $n \times 1$, $\mathbb{Z}_+^n$ est l'ensemble des entiers positifs, c , b_1, b_2, A_1 et A_2 sont des matrices $n \times 1$, $m \times 1$, $k \times 1$, $m \times n$ et $k \times n$ respectivement. Nous supposons que le problème (ILP) est réalisable et que X est un ensemble borné et fini. Nous rappelons que le problème (ILP) est le problème primal et z^* est la valeur optimale primale. ainsi les contraintes $A_2 x \leq b_2$ sont les contraintes faciles, dans le sens où un programme linéaire en nombre entier avec seulement ces contraintes est facile à résoudre. Nous rappelons aussi que la dualité Lagrangienne [M.S. Bazaraa (1981)] est la méthode la plus utile pour résoudre (ILP). Le problème dual Lagrangien est obtenu par l'approche de la relaxation Lagrangienne [M.L. Fisher (1985)], où les contraintes $A_1 x \leq b_1$ qui sont appelées les "contraintes compliquées", sont relâchées en introduisant un vecteur multiplicateur $\lambda \in \mathbb{R}_+^m$, appelé "multiplicateur de Lagrange".

Nous rappelons les formulations de problème de relaxation Lagrangienne (L_λ) et le problème dual (D) :

$$(L_\lambda)\begin{cases} w(\lambda) = \min c^\top x \ + \lambda^\top (A_1 x - b_1) \\ \text{s.t.} \qquad x \in X, \end{cases} \tag{4.2}$$

$$(D)\begin{cases} w^* = \max w(\lambda) \\ \lambda \geq 0. \end{cases} \tag{4.3}$$

Comme nous l'avons mentionné dans les chapitres précédents et avec certaines hypothèses convenables, la valeur optimale duale w^* est égale à z^* (la dualité forte). En général, w^* fournit une borne inférieure de z^*. Ces bornes peuvent être utiles dans l'évaluation des nœuds dans des méthodes exactes telles que l'algorithme "Branch and Bound" pour trouver la solution optimale de (ILP). La fonction $w(\lambda)$ est continue et concave mais non différentiable. La méthode la plus largement adoptée pour résoudre le problème duale est l'optimisation du sous-gradient, voir par exemple [B.T. Polyak (1967)], [N.Z. Shor(1985)], [A. Nedic(2010)], [Y. Nestrov (2014)] et [Y. Hu(2015)]. La méthode d'optimisation du sous-gradient pur est une procédure itérative qui peut être utilisée pour résoudre le problème de maximisation (minimisation) d'une fonction concave (convexe) non différentiable $w(\lambda)$ sur un ensemble convexe fermé Ω. Cette procédure est utilisé dans divers domaines de la science et l'ingénierie [S. Sra(2012)].

Dans le contexte de la relaxation Lagrangienne, le calcul de la direction du sous-gradient s^k et de la projection $P_\Omega\left(\lambda^k + t_k s^k\right)$ $(\Omega = \mathbb{R}^m_+)$ est un problème relativement facile. Puisque le sous-gradient s^k n'est pas nécessairement une direction de descente, le choix de pas t_k diffère de ceux donnés dans des méthodes de descente. En fait, ce choix assure la diminution de la sous-séquence $\left(\left\|\lambda^k - \lambda^*\right\|\right)_k$ ainsi que la convergence de $\left(\lambda^k\right)_k$ à λ^*. Cependant, il est impossible de connaître à l'avance la valeur de w^* pour la plupart des problèmes. Pour ce faire, le moyen le plus efficace consiste à utiliser les méthodes de la variable cible ("*the variable target value methods*") développées dans [S. Kim (1990), F. Fumero (2001)] and [H.D. Sherali(2000)].

4.2 Algorithme sous-gradient dévié

Un autre défi dans l'optimisation de sous-gradient est le choix de la direction de recherche qui affecte la performance de calcul de l'algorithme. On sait que le choix de la direction du sous-gradient s^k conduit au phénomène de zig-zag qui pourrait ralentir la procédure d'exploration vers l'optimalité [M.S. Bazaraa (2006)]. Pour surmonter cette situation, dans

l'esprit de la méthode du gradient conjugué [J. Nocedal (1999), R. Fletcher (1964)], nous pouvons adopter une direction de recherche qui dévie la direction du sous-gradient pure. Par conséquent, la direction de recherche d^k à λ^k est calculée comme suit :

$$d^k = s^k + \Psi_k d^{k-1}, \tag{4.4}$$

où $\Psi_k \geq 0$ est un paramètre de déviation, s^k est un sous-gradient de la fonction w à λ^k et d^{k-1} est la direction précédente ($d^0 = 0$). Ensuite, la nouvelle itération est calculée comme suit :

$$\lambda^{k+1} = P^+\left(\lambda^k + t_k d^k\right). \tag{4.5}$$

Définition 4.1 *Soit t_k un scalaire positif et $d^k \in \mathbb{R}^n$. On dit que la procédure*

$$\lambda^{k+1} = P_\Omega(\lambda^k + t_k d^k), k = 0, 1, 2, ...$$

*forme un **zig-zag**, si $\lambda^k, \lambda^{k+1} \in \Omega$, l'angle entre les directions d^{k+1} et d^k est obtus, i.e., $d^{k+1} d^k < 0$.*

L'algorithme 3 donne une description détaillée de la méthode du sous gradient dévié.

Algorithm 3 (sous gradient dévié)

1. Choisir un vecteur initiale λ^1, et soit $k = 1$, $d^k = 0$.
2. Déterminer un sous-gradient $s^k \in \partial w(u^k)$

$$d^k = s^k + \Psi_k d^{k-1} \tag{4.6}$$

$$\lambda^{k+1} = P^+(\lambda^k + t_k d^k) \tag{4.7}$$

   Les règles pour déterminer ψ_k et t_k seront données plus tard.

   Répéter la procédure à partir de l'étape 2 tant que le test d'arrêt n'est pas vérifie.
3. Si une condition d'arrêt n'est pas encore prise,on revient à l'étape 2.$\hookrightarrow Test$

Un comportement important de la procédure du sous-gradient est que, à chaque itération, la direction du sous gradient s^k forme un angle aigu avec $(\lambda^* - \lambda^k)$. Toutefois, selon plusieurs travaux (voir par exemple,[P.M. Camerini(1975)] et [H.D. Sherali(1989)], l'angle entre la direction du sous-gradient s^k peut former un angle obtus avec la direction précédente s^{k-1}. Ce qui peut forcer le prochain itéré de devenir proche du précédent.

Ce phénomène peut évidemment ralentir la convergence de la procédure.

De tel phénomène de **zig-zag** pourrait se manifester à n'importe quelle étape de la procédure du sous-gradient et ralentir le processus de recherche. Afin d'éviter un tel comportement, P.M. Camerini et al [P.M. Camerini(1975)] ont proposé une modification de la méthode de sous-gradient pur dans lequel la direction du sous-gradient s^k à l'itération k est remplacé par une direction sous-gradient dévié d^k_{TGM}, donnée par :

$$d^k_{MGT} = s^k + \Psi^{MTG}_k d^{k-1}_{MGT}. \tag{4.8}$$

Avec le paramètre de déviation Ψ^{MTG}_k est donné par :

$$\Psi^{MGT}_k = \begin{cases} -\eta_k \frac{s^k d^{k-1}_{MGT}}{\|d^{k-1}_{MGT}\|^2} & \text{if } s^k d^{k-1}_{MGT} < 0, \\ 0 & \text{sinon,} \end{cases} \tag{4.9}$$

où $0 < \eta_k \leq 2$ et $d^{k-1}_{TGM} = 0$ pour $k = 0$.

Il existe diverses formes du choix du paramètre de déviation différentes de celle proposée par Camerini et al, H.D. Sherali et al [H.D. Sherali(1989)] ont proposé de choisir la direction de déviation comme la bissectrice de l'angle formé par le sous-gradient actuel s^k et la direction précédente. Pour obtenir cette direction, le paramètre de déviation est calculé selon

$$\Psi^{ADS}_k = \frac{\|s^k\|}{\|d^{k-1}_{ADS}\|}. \tag{4.10}$$

Avec ce choix du paramètre de déviation Ψ^{ADS}_k, la direction devient :

$$d^k_{ADS} = s^k + \Psi^{ADS}_k d^{k-1}_{ADS}. \tag{4.11}$$

où $d^{k-1}_{ADS} = 0$ pour $k = 0$.

On appelle cette stratégie direction moyenne(Average Direction Strategy).

Dans le reste de ce chapitre, nous présentons une nouvelle direction de recherche déviée comme une combinaison convexe de la direction d^k_{MGT} (4.8) et de la direction d^k_{ADS} (4.11). Notre principal résultat est l'identification du paramètre de combinaison convexe qui force l'algorithme pour une meilleure recherche de déviation que ceux donnés dans le sous-gradient pur, MGT et ADS. Pour une comparaison numérique de notre approche et les deux techniques concurrentes MGT et ADS, nous avons opté pour le problème de voyageur de commerce (TSP) où son importance vient de la richesse de son application et le fait que c'est un exemple typique d'autres problèmes d'optimisation combinatoire [M. Diaby (2016)] et [N.A. El-Sherbeny(2010)].

4.3 La nouvelle modification de la méthode de sous-gradient déviée

Dans cette section, nous présentons une nouvelle méthode de sous-gradient dévié modifié $(NMDS)$ qui détermine la direction de recherche comme suit :

$$d^k = (1-\alpha_k)d^k_{MGT} + \alpha_k d^k_{ADS}\ , \quad \alpha_k \in (0,1)\,. \tag{4.12}$$

On obtient alors le paramètre de déviation suivant :

$$\Psi_k = \begin{cases} \dfrac{-\eta_k(1-\alpha_k)s^k d^{k-1} + \alpha_k \left\|s^k\right\| \left\|d^{k-1}\right\|}{\left\|d^{k-1}\right\|^2} & \text{si} \quad s^k d^{k-1} < 0, \\ 0 \qquad\qquad\qquad\qquad\quad \text{sinon,} \end{cases} \tag{4.13}$$

Par conséquent $d^k = s^k + \Psi_k d^{k-1}$.

Considérons l'algorithme de sous-gradient dévié donné dans l'algorithme 3. La proposition suivante étend les propriétés importantes du vecteur de sous-gradient s^k et la direction du sous-gradient dévié d^k_{MGT} à la nouvelle direction de sous-gradient déviée d^k(4.12). Avec un meilleur choix du paramètre t_k, d^{k-1} construit un angle aigu avec $\lambda^* - \lambda^k$ et d^k. Nous obtenons également la diminution de la sous-suite $(\|\lambda^* - \lambda^k\|)_k$.

Proposition 4.1 *Soit $s^k \in \partial w(\lambda^k)$, d^k la nouvelle direction de sous-gradient déviée donnée par (4.12) et (4.13) et soit $\{\lambda^k\}$ la suite générée par le schéma du sous-gradient dévié. Si nous prenons $0 < \eta_k \leq 2$ avec un pas t_k tel que*

$$0 < t_k < \frac{w^* - w(\lambda^k)}{\left\|d^k\right\|^2}, \quad \forall k = 0, 1, 2, ... \tag{4.14}$$

alors,

1. $$d^{k-1}\left(\lambda^* - \lambda^k\right) \geq 0, \tag{4.15}$$

2. $$\left\|\lambda^{k+1} - \lambda^*\right\| \leq \left\|\lambda^k - \lambda^*\right\|, \tag{4.16}$$

3. $$d^k d^{k-1} \geq 0.$$

pour tout k tel que les λ^k sont des points non optimaux et λ^* est une solution optimale.

Preuve

1. La preuve est établie par récurrence sur k. Puisque nous commençons par $d^0 = 0$, le cas $k = 1$ est trivial. Maintenant, supposons que nous avons

$$d^{k-2}\left(\lambda^* - \lambda^{k-1}\right) \geq 0, \forall k \geq 2, \tag{4.17}$$

et montrons la relation (4.15). En utilisant la définition de d^{k-1}, on obtient que

$$\begin{aligned} d^{k-1}\left(\lambda^* - \lambda^k\right) &= d^{k-1}\left(\lambda^* - \lambda^{k-1} + \lambda^{k-1} - \lambda^k\right) \\ &= d^{k-1}\left(\lambda^* - \lambda^{k-1}\right) + d^{k-1}\left(\lambda^{k-1} - \lambda^k\right) \\ &= \left(s^{k-1} + \Psi_{k-1}d^{k-2}\right)\left(\lambda^* - \lambda^{k-1}\right) + d^{k-1}\left(\lambda^{k-1} - \lambda^k\right) \\ &= s^{k-1}\left(\lambda^* - \lambda^{k-1}\right) + \Psi_{k-1}d^{k-2}\left(\lambda^* - \lambda^{k-1}\right) + d^{k-1}\left(\lambda^{k-1} - \lambda^k\right). \end{aligned}$$

A partir de la concavité de la fonction $w(\cdot)$, l'hypothèse de récurrence (4.17) et les inégalités dans (4.14), nous obtenons

$$d^{k-1}\left(\lambda^* - \lambda^k\right) \geq \left(w^* - w(\lambda^{k-1})\right) - d^{k-1}\left(\lambda^k - \lambda^{k-1}\right). \tag{4.18}$$

Comme le vecteur $\lambda^k - P_\Omega(\lambda^{k-1} + t_{k-1}d^{k-1})$ est perpendiculaire à l'hyperplan de support de $\Omega = \mathbb{R}^m_+$ à λ^k, l'angle à λ^k est obtus (voir Figure 4.1). On en déduit que

$$d^{k-1}(\lambda^k - (\lambda^{k-1} + t_{k-1}d^{k-1})) \leq 0,$$

ce qui est équivalent à

$$-d^{k-1}(\lambda^k - \lambda^{k-1}) \geq -t_{k-1}\|d^{k-1}\|^2 \tag{4.19}$$

En substituant (4.19) dans (4.18) nous obtenons

$$d^{k-1}\left(\lambda^* - \lambda^k\right) \geq \left(w^* - w(\lambda^{k-1})\right) - t_{k-1}\|d^{k-1}\|^2 \geq 0. \tag{4.20}$$

2. On a

$$\begin{aligned} \left\|\lambda^* - \lambda^{k+1}\right\|^2 &= \left\|\lambda^* - P_\Omega(\lambda^k + t_k d^k)\right\|^2 \\ &\leq \left\|\lambda^* - \lambda^k - t_k d^k\right\|^2 \\ &= \left\|\lambda^* - \lambda^k\right\|^2 + t_k^2\left\|d^k\right\|^2 - 2t_k d^k\left(\lambda^* - \lambda^k\right) \\ &= \left\|\lambda^* - \lambda^k\right\|^2 + t_k\left[t_k\left\|d^k\right\|^2 - 2d^k\left(\lambda^* - \lambda^k\right)\right]. \end{aligned}$$

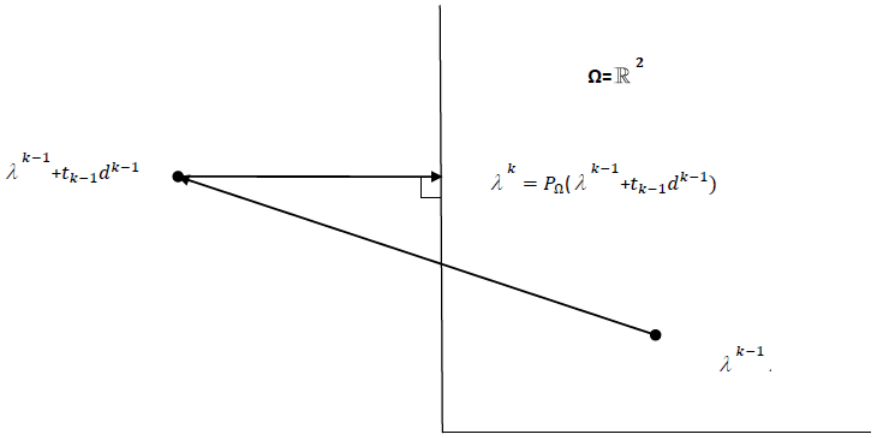


FIGURE 4.1 – Illustration dans un cas bidimensionnel.

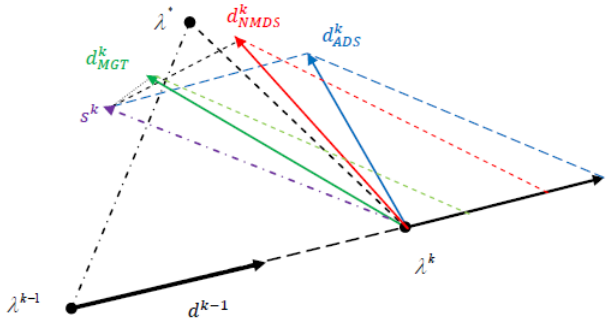


FIGURE 4.2 – Cas où s^k est dévié car il a formé un angle obtus avec d^{k-1} et la direction d^k_{NMDS} est meilleure par rapport aux autres directions d^k_{ADS} et d^k_{MGT}.

De la concavité de w, les inégalités dans (4.14) et en appliquant (4.15) dans la proposition (4.1), on obtient respectivement les relations suivantes :

$$\begin{aligned} t_k \left\|d^k\right\|^2 &\leq w^* - w(\lambda^k) \leq 2\left(w^* - w(\lambda^k)\right) \\ &\leq 2s^k\left(\lambda^* - \lambda^k\right) \\ &\leq 2d^k\left(\lambda^* - \lambda^k\right). \end{aligned}$$

Il en résulte que

$$t_k \left\|d^k\right\|^2 - 2d^k\left(\lambda^* - \lambda^k\right) \leq 0.$$

Par conséquent,

$$\left\|\lambda^* - \lambda^{k+1}\right\| \leq \left\|\lambda^* - \lambda^k\right\|.$$

3. Si $s^k d^{k-1} \geq 0$ alors $d^k = s^k$ et par conséquent, la relation suit. Ainsi, considérons le cas $s^k d^{k-1} < 0$. Nous avons alors

$$\begin{aligned} d^k d^{k-1} &= \left(s^k + \Psi_k d^{k-1}\right) d^{k-1} \\ &= s^k d^{k-1} + \Psi_k \left\|d^{k-1}\right\|^2 \\ &= s^k d^{k-1} - \eta_k\left(1-\alpha_k\right) s^k d^{k-1} + \alpha_k \left\|s^k\right\| \left\|d^{k-1}\right\| \\ &= \left(-\alpha_k + \eta_k\alpha_k(1-\alpha_k) + \alpha_k\right) \left\|s^k\right\| \left\|d^{k-1}\right\| \\ &= \eta_k\alpha_k\left(1-\alpha_k\right) \left\|s^k\right\| \left\|d^{k-1}\right\| \\ &\geq 0. \end{aligned}$$

Cela complète la prouve. ■

L'importance de la proposition (4.1) repose sur le fait que le choix du paramètre de déviation Ψ_k en utilisant la règle (4.13) avec $0 < \eta_k \leq 2$ force la direction de sous-gradient déviée actuelle à former toujours un angle aigu avec la direction précédente et, par conséquent, cette méthode réduire le phénomène de zig-zag. Notons que le choix du vecteur de direction déviée d^k_{MGT} est toujours au moins aussi bon que la direction du vecteur sous gradient s^k. Si $1 \leq \eta_k \leq 2$, alors $d^k_{MGT} d^{k-1}_{MGT} \geq 0$. [P.M. Camerini(1975)].

Le théorème (4.1) ci-dessous montre qu'avec un choix particulier du paramètre de combinaison convexe α_k et du paramètre η_k, la direction du vecteur de sous gradient dévié d^k est toujours au moins aussi bonne que la direction d^k_{MGT} dans le sens où d^k peut former un angle plus aigu avec la meilleure direction vers une solution optimale que d^k_{MGT} (voir Figure 4.2), qui améliore la vitesse de convergence. Les deux lemmes ci-dessous sont nécessaires pour la preuve de notre résultat principal.

Lemme 4.2 *Soit $s^k \in \partial w(\lambda^k)$ et fixé $\alpha_k = -\cos(s^k, d^{k-1})$ si $s^k d^{k-1} < 0$. Avec l'hypothèse dans (4.14) et laisser*

$$0 < \eta_k \leq \frac{1}{2-\alpha_k}, \tag{4.21}$$

alors

$$d^k\left(\lambda^* - \lambda^k\right) \geq d^k_{MGT}\left(\lambda^* - \lambda^k\right) \text{ pour tout } k. \tag{4.22}$$

Preuve En utilisant (4.8), (4.11) et (4.12) on obtient la relation suivante :

$$\begin{aligned} d^k\left(\lambda^* - \lambda^k\right) - d^k_{MGT}\left(\lambda^* - \lambda^k\right) &= (1-\alpha_k) d^k_{MGT}\left(\lambda^* - \lambda^k\right) + \alpha_k d^k_{ADS}\left(\lambda^* - \lambda^k\right) - d^k_{MGT}\left(\lambda^* - \lambda^k\right) \\ &= \alpha_k\left[d^k_{ADS}\left(\lambda^* - \lambda^k\right) - d^k_{MGT}\left(\lambda^* - \lambda^k\right)\right] \\ &= \alpha_k\left[\left(s^k + \Psi^{ADS}_k d^{k-1}\right)\left(\lambda^* - \lambda^k\right) - \left(s^k + \Psi^{MGT}_k d^{k-1}\right)\left(\lambda^* - \lambda^k\right)\right] \\ &= \alpha_k\left(\Psi^{ADS}_k - \Psi^{MGT}_k\right) d^{k-1}\left(\lambda^* - \lambda^k\right). \end{aligned}$$

De (4.9) et (4.10) il s'ensuit que :

$$\Psi^{ADS}_k - \Psi^{MGT}_k = \frac{\left\|s^k\right\|\left\|d^{k-1}\right\|}{\left\|d^{k-1}\right\|^2}\left[1 + \eta_k cos(s^k, d^{k-1})\right]. \tag{4.23}$$

En utilisant la dernière égalité et l'application de la proposition (4.1) nous obtenons (4.22).

■

Lemme 4.3 *Sous la même hypothèse du lemme (4.2), nous avons*

$$\left\|d^k\right\| \leq \left\|d^k_{MGT}\right\| \qquad \text{pour tout } k. \tag{4.24}$$

Preuve Si $s^k d^{k-1} \geq 0$ alors $\Psi_k = 0$ et donc (4.24) détient évidemment et on a simplement $d^k = d^k_{MGT}$. Dans le cas où $s^k d^{k-1} < 0$, alors :

$$\begin{aligned} \left\|d^k\right\|^2 - \left\|d^k_{MGT}\right\|^2 &= \left\|s^k + \Psi_k d^{k-1}\right\|^2 - \left\|s^k + \Psi^{MGT}_k d^{k-1}\right\|^2 \\ &= \left(\Psi^2_k - \left(\Psi^{MGT}_k\right)^2\right)\left\|d^{k-1}\right\|^2 + 2\left(\Psi_k - \Psi^{MGT}_k\right) s^k d^{k-1} \\ &= \left(\Psi_k - \Psi^{MGT}_k\right)\left[\left(\Psi_k + \Psi^{MGT}_k\right)\left\|d^{k-1}\right\|^2 + 2 s^k d^{k-1}\right]. \end{aligned}$$

Puisque $s^k d^{k-1} = \left\|s^k\right\|\left\|d^{k-1}\right\| cos(s^k, d^{k-1})$ on trouve :

$$\left\|d^k\right\|^2 - \left\|d^k_{MGT}\right\|^2 = \alpha^2_k\left\|s^k\right\|^2\left(-\eta_k\alpha_k + 1\right)\left[\eta_k\left(2-\alpha_k\right) - 1\right].$$

Par le choix de η_k tel que nous obtenons

$$\left\|d^k\right\| \leq \left\|d^k_{MGT}\right\|.$$

■

Théorème 4.1 *Avec les mêmes hypothèses du lemme (4.2), nous avons*

(*i*)

$$\frac{d^k\left(\lambda^* - \lambda^k\right)}{\|d^k\|} \geq \frac{d^k_{MGT}\left(\lambda^* - \lambda^k\right)}{\left\|d^k_{MGT}\right\|}. \tag{4.25}$$

(*ii*) *Si les vecteurs* d^k *et* d^k_{MGT} *forment un angle* θ_{d^k} *et* $\theta_{d^k_{MGT}}$, *respectivement, avec le vecteur* $\lambda^* - \lambda^k$, *alors*

$$0 \leq \theta_{d^k} \leq \theta_{d^k_{MGT}} \leq 90°.$$

Preuve Conséquence directe des lemmes précédents. ■

4.4 Résultats numériques

L'algorithme proposé a été appliqué à l'un des problèmes standard de programmation linéaire en nombre entier dans le domaine de la recherche opérationnelle, à savoir le problème de voyageur de commerce symétrique ($TSPs$). Le problème de voyageur de commerce est un problème d'optimisation combinatoire classique "*NP-difficile*" [M.R. Garey (1990)]. Il peut être formulé comme suit : en donnant un ensemble de villes, et les distances entre elles, le but est de trouver le tour le plus court chemin, tel que visitant chaque ville une seule fois et revenant à la ville de départ. Pour plus de détails voire [E.L. Lawler (1985)]. Le problème du TSP peut être énoncé comme suit :

$$\min \sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq i}}^{m} c_{ij}x_{ij}, \tag{4.26}$$

sous contraintes

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i = 1, ..., m, \tag{4.27}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, ..., m, \tag{4.28}$$

$$\sum_{i\in Q}^{m}\sum_{j\in Q}^{m} x_{ij} \leq |Q| - 1, \quad \forall Q : 2 \leq |Q| \leq m - 2, \tag{4.29}$$

$$x_{ij} = \{0, 1\} \quad pour\ tout\, i, j = 1, ..., m, \tag{4.30}$$

où les c_{ij} sont les coûts du lien (i, j) et $Q \subset \{1, ..., m\}$. Si X est l'ensemble des 1-arbres [M.H. Held(1970)], les contraintes de sous-tour (4.29) peuvent être éliminées en insistant sur le fait qu'un vecteur x satisfaisant les contraintes (4.27), (4.28) et (4.30) doit aussi appartenir à X.

En particulier pour le cas symétrique, les contraintes (4.27) et (4.28) peuvent être remplacées par (4.31) conduisant à la formulation équivalente suivante de TSPs :

$$\min \quad \sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq i}}^{m} c_{ij}x_{ij},$$

sous contraintes

$$\sum_{\substack{j=1\\ j\neq i}}^{m} x_{ij} + \sum_{\substack{j=1\\ j\neq i}}^{m} x_{ji} = 2 \quad \text{for } i = 1, ..., m, \tag{4.31}$$

$$x \in X.$$

De là, on obtient la fonction duale suivante, qui doit être maximisé :

$$w(\lambda) = \min\left\{\sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq i}}^{m}(c_{ij} + \lambda_i + \lambda_j)\, x_{ij},\ x \in X\right\} - 2\sum_{i=1}^{m}\lambda_i, \tag{4.32}$$

où $\lambda \in \mathbb{R}^m$ est le vecteur des multiplicateurs de Lagrange.

Étant donné un vecteur $\overline{\lambda}$, si $\overline{x}$ optimise $w(\overline{\lambda})$, alors un vecteur $\overline{s}$ dont la $i^{ème}$ composante.

$$\overline{s_i} = \left(\sum_{\substack{j=1\\ j\neq i}}^{m} x_{ij} + \sum_{\substack{j=1\\ j\neq i}}^{m} x_{ji} - 2\right) \tag{4.33}$$

est un sous-gradient de $w(\lambda)$ à $\overline{\lambda}$ ([M.S. Bazaraa (2006)] et [M.H. Held(1974)]).

Pour valider la faisabilité et l'efficacité de l'approche proposée, nous l'avons appliquée sur certaines instances de TSP pris de TSPLIB [1]. L'algorithme proposé, MGT et ADS, ont été implémentés par Matlab et exécutés sur un processeur Intel (R) Core (TM) i517U @ 1.70GHz 1.70GHz RAM 4.00GO.

Pour toutes les instances symétriques et pour une comparaison équitable entre les trois algorithmes, les paramètres suivants ont été choisis :

- Le même multiplicateur initial $\lambda^1 = (0, 0, ..., 0)^T$ a été utilisé pour les trois algorithmes.
- Les conditions d'arrêt sont le nombre maximum d'itérations $iterMAX = 1000$, ou $|w^* - w(\lambda^k)| \leq \varepsilon$, où ε est une petite tolérance ($\varepsilon = 10^{-2}$).
- La taille de pas t_k est défini selon la formule (3.13).
- Le paramètre δ_k suit la suggestion Held et Karp, [M.H. Held(1974)], qui fait $0 < \delta_k \leq 2$, en commençant par $\delta_k = 2$. Si après 20 itérations $w(\lambda^k)$ n'augmente pas, δ_k est mis à jour $\delta_k = \dfrac{\delta_k}{2}$.

1. http ://comopt.ifi.uniidelberg.de/software/TSPLIB95/

- Pour l'algorithme MGT, comme mentionné dans [P.M. Camerini(1975)], l'utilisation de $\eta_k = 1.5$ est recommandé et sa justification intuitive ainsi que des résultats de calcul sont également donnés, ce qui explique les performances de la stratégie MGT par rapport à l'algorithme de sous-gradient pur, en pratique.
- Pour notre algorithme, la valeur de η_k dépend du paramètre optimal de combinaison convexe α_k comme il est indiqué dans le lemme 4.2, où $\alpha_k = -\cos(s^k, d^{k-1})$ si $s^k d^{k-1} < 0$. Nous avons utilisé $\eta_k = \dfrac{1}{2-\alpha_k} - \varepsilon$, où ε est une petite valeur arbitraire.

Le tableau 4.1 montre les résultats expérimentaux obtenus par : Stratégie MGT, stratégie ADS et en appliquant notre algorithme NMDS proposé dans cette thèse avec 11 tests des instances symétriques entre n = 6 et n = 101 sommets pris à partir de TSPLIB. Pour les trois stratégies, l'écart de la dualité " GAP" de ces 11 exemples est nul. Cependant, toujours l'algorithme NMDS surperforme les autres en nombre d'itérations et on temps d'exécution. Le tableau 4.2 donne des résultats numériques pour 19 tests des instances symétriques entre 131 et 3056 sommets. Ce tableau montre également que notre algorithme donne des résultats quasi-optimaux pour plusieurs instances. Les en-têtes de colonne sont les suivants : :

- Nom : Indique le nom de l'instance.
- n : Indique la taille du problème.
- w^* : La meilleure solution connue.
- LB : La meilleure valeur (borne inférieure) obtenue par chaque stratégie.
- $Iter$: Nombre d'itérations pour lesquelles la meilleure valeur LB est obtenue (limitée à 1000).
- $GAP = \dfrac{w^* - LB}{w^*}$.
- CPU : Temps total d'exécution, en seconde pour calculer la meilleure valeur LB obtenue par chaque stratégie.

Nom	n	w^*	Stratégie											
			Stratégie NMDS				Stratégie ADS				Stratégie MGT			
			LB	GAP	Iter	CPU	LB	GAP	Iter	CPU	LB	GAP	Iter	CPU
tsp6	6	207	207	0	2	0.027862 s	207	0	2	0.032690 s	207	0	2	0.028533 s
tsp7	7	106.4	106.4	0	3	0.028651 s	106.4	0	3	0.028819 s	106.4	0	3	0.029059 s
tsp8	8	100	100	0	3	0.032122 s	100	0	3	0.032360 s	100	0	3	0.033069 s
tsp10	10	378	378	0	3	0.077130 s	378	0	18	0.085784 s	378	0	15	0.099755 s
ulysses16	16	68.59	68.59	0	16	0.077255 s	68.59	0	30	0.099056 s	68.59	0	26	0.090043 s
gr21	21	2707	2707	0	19	0.077098 s	2707	0	26	0.086654 s	2707	0	22	0.080609 s
ulysses22	22	70.13	70.13	0	21	0.219532 s	70.13	0	70	0.467253 s	70.13	0	28	0.253197 s
tsp33	33	10861	10861	0	18	0.27955 s	10861	0	19	0.303524 s	10861	0	22	0.304997 s
eil76	76	538	538	0	12	0.728258 s	538	0	23	1.285156 s s	538	0	18	1.071510 s
rat99	99	1211	1211	0	19	1.817552 s	1211	0	20	1.902072 s	1211	0	34	3.111573 s
eil101	101	629	629	0	13	1.253360 s	629	0	16	1.523116 s	629	0	15	1.422569 s

TABLE 4.1 – Résultats numériques pour $6 \leq n \leq 101$ [R. Belgacem (2017)].

Name	n	w^*	Strategy											
			Our Method				ADS Strategy				MGT Strategy			
			LB	GAP	Iter	CPU	LB	GAP	Iter	CPU	LB	GAP	Iter	CPU
xq131	131	564	555.96	0.0159	350	38.134934 s	555.52	0.0167	350	39.009219 s	555.64	0.0165	350	42.417376 s
ch150	150	6528	6498.3	0.0045	193	57.562539 s	6463.2	0.0099	200	58.301731 s	6490.4	0.0057	202	59.548258 s
tsp237	237	1019	1004.8	0.0139	190	76.942993 s	1002.4	0.0162	200	79.046667 s	1003.9	0.0148	197	77.288710 s
a280	280	2579	2569.8	0.0035	251	128.401733 s	2568.3	0.0041	251	134.245723 s	2569.8	0.0035	251	129.376815
linhp318	318	41345	41345	0	188	135.792252 s	41330	0.0003	200	148.125054 s	41337	0.0001	195	140.734334 s
pbk411	411	1343	1443	0	250	328.562376 s	1338.3	0.0034	250	376.626413 s	1338.6	0.0032	250	235.626147 s
pbn423	423	1365	1365	0	251	344.519801 s	1360.3	0.0034	251	349.933052 s	1361.5	0.0025	251	370.504702 s
pbm436	436	1443	1423.9	0.0132	207	241.663478 s	1422.3	0.0143	251	334.741944 s	1421.4	0.0149	207	255.055079 s
rat575	575	6773	6721.2	0.0076	350	1287.476161 s	6716	0.0084	350	1266.572875 s	6718.9	0.0079	350	1138.613357 s
rbx711	711	3115	3099.6	0.0049	500	1648.379114 s	3094.3	0.0066	500	1668.377875 s	3096.2	0.0060	500	1701.099452 s
rat783	783	8806	8652.2	0.0197	121	496.527482 s	8609.5	0.0223	180	744.089813 s	8642.8	0.0185	131	592.200505 s
dkg813	813	3199	3164.7	0.0107	200	126.290411 s	3147.5	0.0203	200	129.330427 s	3154.7	0.0138	200	138.698584 s
pbd984	984	2797	2797	0	500	2216.458302,s	2772	0.0089	500	4059.009584 s	2774.3	0.0081	500	3394.524016,s
xit1083	1083	3558	3551.1	0.0019	550	3394.524016 s	3525.1	0.0092	600	3394.524016 s	3549.6	0.0023	600	3726.7668 s
dka1376	1376	4666	4666	0	500	7321.695698 s	4659.6	0.0013	500	14632.080221 s	4662	0.0008	500	14704.797585 s
dja1436	1436	5257	5194.2	0.0437	500	10959.322745 s	5189.4	0.0128	500	12014.302471 s	5184.4	0.0138	500	11486.802618 s
dcc1911	1911	6396	6367.9	0.0043	500	10602.013548 s	6338.3	0.0097	500	11742.144423 s	6357.2	0.0060	500	11127.0759858 s
djb2103	2103	6197	6197	0	430	13566.845650 s	6133.8	0.0101	500	14954.211430 s	6169.3	0.0044	500	15955.063763 s
pia3056	3056	8285	8285	0	500	51124.513213 s	8192	0.0112	500	59123.013454 s	8228.2	0.0068	500	58072.083303 s

1

TABLE 4.2 – Résultats numériques pour $131 \leq n \leq 3056$ [R. Belgacem (2017)].

Conclusion

Dans cette thèse, nous avons proposé une nouvelle approche de sous gradient dévié qui repose sur une combinaisons convexe de la direction du gradient modifié d^k_{MGT} et la direction moyenne d^k_{ADS}. En identifiant le paramètre optimal de combinaison convexe, la nouvelle direction déviée réduit à chaque itération, le phénomène de zig-zag. Les études d'analyse sont cohérentes avec les expériences numériques. De plus, cette méthode peut être utilisée pour améliorer la convergence dans le domaine de la méthode de sous-gradient dévié en utilisant la dualité Lagrangienne augmentée [R.S. Burachik (2010)] et les méthodes dual sous-gradient [E. Gustavsson(2015)]. On peut aussi suivre [C. Lim (2006)] et combiner cette méthode avec une technique de variable cible "target value method" afin d'avoir une bonne performance. Enfin, la méthode du sous-gradient est généralement utilisée comme sub-routine dans l'optimisation exacte, heuristique et méta heuristique, ce qui justifie le large spectre d'applications de notre approche.

Appendices

ANNEXE A

Théorie des graphes

Dans cette thèse nous manipulerons des objets mathématiques : **les graphes**. Nous allons donc donner un bref aperçu de la terminologie utilisée dans cette thèse. Plus d'informations de fond peuvent être trouvées dans [D.B. West (1996)], [F. Harary(1996)] et [G. Chartrand (2005)].

A.1 Notations de base

Graphe : un graphe $G = (V, E)$ est constitué de deux ensembles finis : V, l'ensemble des *sommets* de G (ou *nœud*), qui est un ensemble non vide d'éléments appelés *sommets* et $E \subseteq V \times V$ est appelé ensemble *d'arêtes*. La cardinalité de l'ensemble de sommets V est appelée *l'ordre* de G, communément désigné par $|V|$. La cardinalité de l'ensemble des arêtes E est la *taille* de G, notée $|E|$. Un graphe d'ordre n et de taille m est souvent appelé un (n, m)-*graphe*. Deux sommets distincts u et v sont *adjacents* (ou *voisins*) s'il existe une arête uv qui les relie. On dit qu'une arête uv est *incident* aux sommets u et v. Deux arêtes sont adjacentes si elles sont incidentes à un même sommet. Un graphe est *simple* s'il y a au plus un arête entre deux sommets. Dans ce document, les graphes considérés seront des graphes simples finis.

Degré et voisinage : Soit v un sommet d'un graphe G. Un voisin de v est un sommet u tel que $uv \in E$.

Le voisinage d'un sommet v dans G, noté $N_G(v)$ ou $N(v)$ s'il n'y a pas d'ambiguïté sur le graphe considéré, est l'ensemble des voisins de v dans G :

$$N_G(v) = \{u \in V, uv \in E\}.$$

Le voisinage fermé d'un sommet v dans G, noté $N_G^+(v)$, est le voisinage de v plus le sommet v lui même : $N_G^+(v) = N_G(v) \cup \{uv\}$.

Le degré d'un sommet v dans G, noté $d_G(v)$ est le nombre d'arêtes incidentes à v, c'est-à-dire le nombre d'arêtes ayant une extrémité égale à v. On a donc $d_G(v) = |N_G(v)|$. Le degré de v est parfois noté $d(v)$ s'il n'y a pas d'ambiguïté sur le graphe considéré. Si $d(v) = 0$, cela signifie que v n'est adjacent à aucun autre sommet, alors v est appelé un sommet *isolé*. Un sommet de un degré est appelé un point d'extrémité ou d'un sommet de pendentif ou une feuille. Le degré minimal d'un graphe G est $\delta(G) = \min\{d(v) : v \in V\}$ et le degré maximal d'un graphe G est noté $\triangle(G) = \max\{d(v) : v \in V\}$.

Ensembles indépendants : un ensemble indépendant de G est un sous-ensemble de sommets $U \subseteq V$, tel que deux sommets dans U ne sont pas adjacents. Un ensemble indépendant est dit maximal si aucun ensemble indépendant ne le contient correctement. Un ensemble indépendant de cardinalité maximum est appelé un ensemble indépendant maximum.

Cliques et stables : Une clique dans un graphe G est un sous-ensemble de sommets $C \subseteq V$ tel que tous les deux sommets de C sont adjacents dans G. Le nombre de cliques $\omega(G)$ est l'ordre d'une clique maximale de G.

Un stable est un ensemble non vide de sommets $S \subseteq V$ non voisins deux à deux. Autrement dit, pour tous sommets s et t de S, $st \notin E$.

Distance et connectivité : Un chemin d'un graphe G est une suite non vide de sommets $(v_1, v_2, ..., v_k)$ telle que pour tout $1 \leq i \leq k$, $v_i v_{i+1}$ est une arête de G. Un chemin constitue un parcours dans le graphe G. Un chemin $u_0 u_1 ... u_k$ est un cycle si et seulement si $k \geq 3$et $u_0 = u_k$. Un chemin est appelé simple si tous ses sommets sont distincts. Un cycle est donc un parcours dans lequel on revient au point de départ. La longueur d'un chemin ou d'un cycle est le nombre k d'arêtes de celui-ci.

Un graphe G est connexe s'il existe un chemin entre deux sommets distincts de G. Sinon, le graphe G est déconnecté. La distance entre deux sommets u, v dans G, notée $dist(u, v)$, est le nombre minimum d'arêtes dans un chemin les reliant. Le diamètre de G, notée $diam(G)$ est la distance maximale entre deux sommets de G. Un graphe est $k-$arête-connexe si il y a k chemins disjoints entre chaque paire de sommets ou, de manière équivalente, on ne peut séparer deux sommets en enlevant moins de k arêtes.

Isomorphes des graphes : G et H deux graphes. On dit que G et H sont isomorphes et on note $(G \cong H)$ s'il y a une bijection $\varphi : V \to V$ telle que pour tous sommets u et v de G on a :

$$uv \in E(G) \Leftrightarrow \varphi(u)\varphi(v) \in E(H).$$

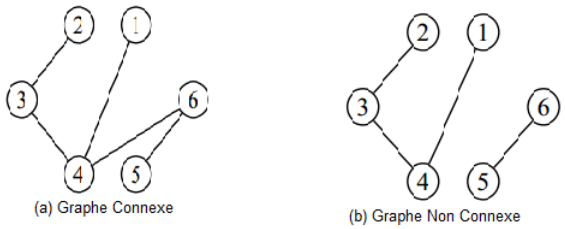


FIGURE A.1 – Graphe connexe et non Connexe

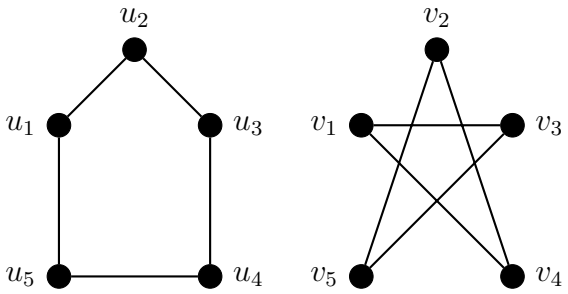


FIGURE A.2 – Exemple de graphes isomorphes : $f(u_1) = v_1, f(u_2) = v_3, f(u_3) = v_5, f(u_4) = v_2$ et $f(u_5) = v_4$.

Nous fournissons un exemple dans la Figure (A.2).

Sous-graphes : un graphe G' est un sous-graphe partiel de G, noté s'il existe $S \subseteq V(G)$ et $F \subseteq E(G)$ tels que $G' = (S, F)$. Cette relation entre G et G' est notée $G' \subseteq G$. Un graphe G' est un sous-graphe induit de G si et seulement s'il existe un ensemble S de sommets de G tel que $G' = G[S]$. Un sous-graphe H est un sous-graphe couvrant ("*spanning subgraph*") de G si H contient tous les sommets de G. Étant donné $V' \subseteq V$, le sous-graphe $G' = G[V'] = (V', E')$ désigne le sous-graphe de G induit par V', c'est-à-dire que E' contient tous les arêtes de E qui ont les deux extrémités dans V'.

A.2 Opérations sur les graphes

Dans les définitions suivantes, nous détaillons quelques opérations sur les graphes bien connues.

Complément d'un graphe : si G est un graphe simple avec un ensemble de sommets $V(G)$, son complément $\overline{G}$ est le graphe simple avec un ensemble de sommets $V(G)$ dans

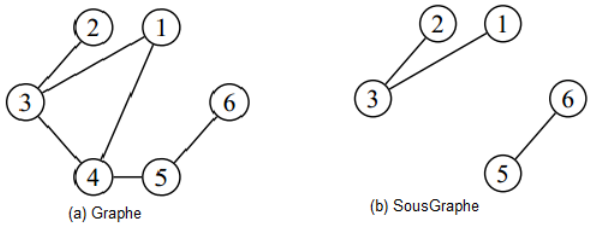


FIGURE A.3 – Graphe et Sous Graphe

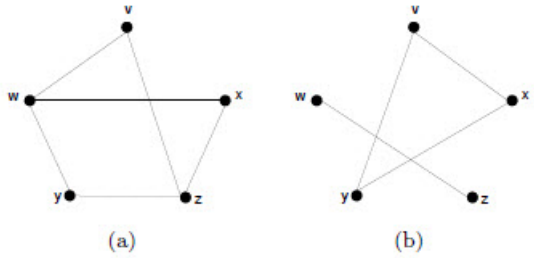


FIGURE A.4 – (a) G, (b) $\overline{G}$.

lequel deux sommets sont adjacents si et seulement s'ils ne sont pas adjacents dans G. La Figure (A.4) illustre un graphe et son complément.

Supprimer un sommet : si v est un sommet du graphe $G = (V, E)$, alors $G - v$ est le sous-graphe de G induit par l'ensemble des sommets $V \backslash \{v\}$.

Supprimer une arête : de manière similaire à l'opération précédente, si e est une arête du graphe $G = (V, E)$, alors $G - e$ est le graphe (V, E'), où E' est obtenu en supprimant e de E. Notez que les extrémités de e ne sont pas supprimés de G.

Union de graphe : l'union $G = G_1 \cup G_2$ des graphes G_1 et G_1 avec les ensembles de sommets disjoints V_1 et V_2 et les ensembles des arrêtes E_1 et E_2 est le graphe avec $V = V_1 \cup V_2$ et $V = E_1 \cup E_2$. Cette opération est parfois également connue explicitement sous le nom d'union disjointe.

Produit cartésien : le produit cartésien de deux graphes G_1 et G_2, notée $G_1 \square G_2$, est le graphe simple K contenant $|V(G_1)| \times |V(G_2)|$ sommets et construit comme suit :

- $V(K) = \{(u, u'), u \in V(G), u' \in V(H)\}$,
- $E(K) = \{(u, u')(v, v'), u = v \; et \; u'v' \in E(H)\} \cup \{(u, u')(v, v'), u' = v' \; et \; uv \in E(G)\}$.

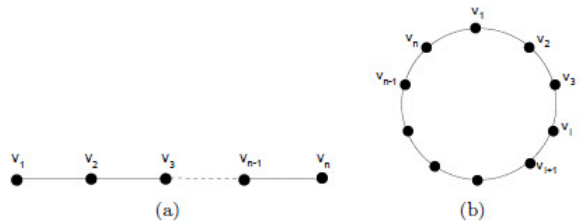


FIGURE A.5 – (a) Chemin, (b) Cycle.

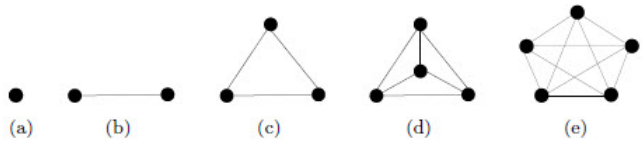


FIGURE A.6 – Graphes complets (a) K_1, (b) K_2, (c) K_3, (d) K_4, (e) K_5.

A.3 Quelques classes de graphes

Dans cette section nous allons définir quelques familles de graphes.

Chemin : un graphe G est appelé chemin, noté P_n, s'il est de la forme :
$V(P) = \{v_1, ..., v_n\}$, $E(P) = v_1v_2, ..., v_{n-1}v_n$ (voir la Figure (A.5)(a)).

Cycle : un graphe G est appelé cycle, noté C_n, s'il est de la forme :
$V(C) = \{v_1, ..., v_n\}$, $E(C) = v_1v_2, ..., v_{n-1}v_n, v_nv_1$ (voir la Figure (A.5) (b)).

Graphe hamiltonien : un cycle de Hamiltonien est un cycle contenant tous les sommets du graphe. Un graphe est hamiltonien s'il a un cycle hamiltonien.

Graphe complet : un graphe K est complet si chaque sommet du graphe est relié directement à tous les autres sommets. Pour $n \geq 0$, le graphe complet d'ordre n est noté K_n. Dans la Figure (A.6), nous donnons cinq exemples de graphes complets.

Graphe planaire : un graphe planaire est un graphe pouvant être dessiné de façon planaire, c'est-à-dire dessiné sur le plan de telle sorte qu'aucune arête n'en croise une autre. Par exemple, le graphe complet K_5 est un graphe planaire.

Arbre : un *arbre* A est une graphe connexe sans cycle. Une *forêt* est un graphe non connexe dont les composantes connexes sont toutes des arbres. Le graphe de la Figure (A.7) est un arbre.

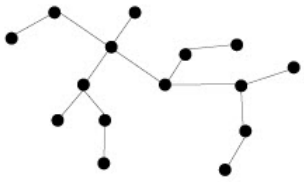


FIGURE A.7 – Arbre

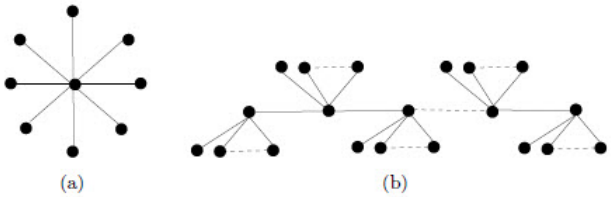


FIGURE A.8 – (a) étoile, (b) Chenille.

Graphe étoile : Une étoile généralisée E (appelée simplement étoile) est un arbre où au plus un sommet est de degré supérieur ou égal à 3. Le degré de ce sommet est le nombre de branches de E (voir la Figure (A.6) (a))

Chenille : Une *chenille* C est une arbre pour lequel il existe un chemin $P = u_0u_1u_2...u_k$ tel que pour tout sommet v il existe un sommet u_i de P tel que $d_C(v, u_i) \leq 1$. Une chenille est représentée sur la Figure (A.8) (b), avec le chemin P constitué par les sommets du haut de la Figure. On observe que ce chemin n'est généralement pas unique.

Graphe biparti : un graphe G est dit *biparti* s'il existe une partition de $V(G)$ en deux ensembles non vides U et W telle que toute arête de G relie un sommet de U avec un sommet de W. Un graphe G est *biparti complet* si G est biparti en deux ensembles U et W, et si pour tous $u \in U$ et $v \in W$ il existe une arête entre u et v. On note $K_{n,p}$ le graphe biparti complet tel que $n = |U|$ et $p = |W|$. La Figure (A.9) est un exemple de graphe biparti complet.

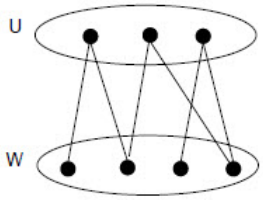


FIGURE A.9 – Graphe biparti

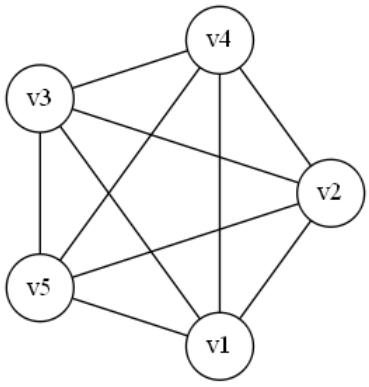


FIGURE A.10 – Un graphe 4-régulier

Graphe régulier : un graphe *régulier* (resp. k-régulier) est un graphe où tous les sommets ont le même degré (resp. degré k). Le graphe de la Figure (A.10) est un graphe 4-régulier.

ANNEXE B

Introduction à la théorie de la NP-complétude

B.1 Introduction

Les problèmes NP-complets recouvrent une très grande partie dans l'optimisation combinatoire et la théorie du graphes. Imaginons que l'on ait un ensemble fini d'une part et une propriété logique "facile" à vérifier d'autre part, la NP-complétude serait alors la question de savoir s'il existe ou non un élément de cet ensemble satisfaisant cette propriété. Il s'agit donc d'un problème décisionnel. Le premier problème NP-complet, la satisfiabilité d'une formule logique binaire, a été établi en 1971 par *Stephen Cook*. Depuis lors, de nombreux autres problèmes NP-complets ont été identifiés ou prouvés mais certains (beaucoup en réalité) sont considérés comme intraitables par les ordinateurs (voir la satisfaction d'une formule de logique, la coloration d'un graphe, la couverture des sommets, les cycles hamiltoniens). L'impossibilité de traiter certains problèmes théoriques est liée au fait que $P = NP$ est toujours au stade de conjecture (elle n'a pas encore été démontré).

B.2 Classification de problèmes : la classe P et de la classe NP

Pour des raisons de simplicité et techniques aussi, la théorie de la complexité se limite juste à l'étude des problèmes de décision. L'étude de l'NP-complétude nécessiterait une connaissance de terminologie (et pas seulement) pratiquement illimitée, de plus qu'il s'agit d'un domaine de recherche et donc non borné. Ici on fournit uniquement les définitions que l'on a réputé “ de base ” pour une analyse des problèmes informatiques.

B.2.1 Les différents classements

Nous fournirons une image panoramique de diverses classifications de problèmes informatiques. Tous les problèmes de résolution sont effectivement mis dans des groupes de complexité similaire : les classes de complexité. La liste suivante ne veut pas être exhaustive, mais elle montre une très grande partie

Définition B.1 *(Instance d'un problème) :soit X un problème caractérisé par l'ensemble E de ses données, $E = \{e_1, e_2, ..., e_n\}$, une instance de X serait alors caractérisée par un ensemble concret $E' = \{e'_1, e'_2, ..., e'_n\}$, une deuxième instance serait caractérisée par un deuxième ensemble $E" = \{e_1", e_2", ..., e_n"$concret et ainsi de suite.*

L'invariant suivante est toujours vérifié : $|E| = |E'| = |E''|$.

Définition B.2 *(Problème abstrait) : relation binaire entre un ensemble d'instances d'un problème et un ensemble de solutions à ce problème.*

Définition B.3 *(Problème de décision ou décisionnel) : Un problème de décision est un problème dont la solution est formulée en termes oui/non.*

On en déduit qu'un problème décisionnel abstrait fait correspondre à toute instance du problème l'ensemble des solutions vrai, faux.

Exemple B.1 *Étant donné un graphe $G = (V, E)$, existe t-il un chemin de longueur $\leq L$?*

Exemple B.2 *Étant donné un graphe $G = (V, E)$, les sommets de V peuvent-il être colorés par au plus m couleurs de telle manière que les sommets adjacents soient de couleurs différentes.*

Le but de la théorie de la complexité est la classification des problèmes de décision suivant leur degré de difficulté de résolution. Dans la littérature, il existe plusieurs classes de complexité, mais les plus connues sont les suivantes.

Définition B.4 *(Résolution en temps polynomial) : il existe un algorithme permettant la résolution en $O(n^k)$ pour une constante k.*

Définition B.5 *Un problème est dit de la classe **P** s'il peut être résolu en un temps polynomial. C'est la classe des problèmes dits facile. Sinon les problèmes sont dits difficiles.*

Définition B.6 *Un algorithme non déterministe est muni (à l'inverse des déterministes) d'une instruction qui permette, chaque fois qu'elle est appliquée, de faire le bon choix.*

Définition B.7 *Un problème appartient à la classe **NP** si il peut être résolu par un algorithme polynomial non déterministe. NP signifie non déterministe polynomial*

Exemple B.3 ***le problème SAT (de satisfiabilité).***

Soit $F(x_1, x_2, ..., x_n)$ une expression logique de n variables. Le problème SAT consiste à trouver des valeur vrai ou faux pour chacune des variables x_i de telle manière à rendre vrai l'expression $F(x_1, x_2, ..., x_n)$.

Une autre manière, plus informelle pour définir la classe NP : c'est la classe des problèmes pour lesquels les seuls algorithmes connus de résolution sont ceux ayant une complexité exponentielle.

B.3 Problématiques de théorie de la complexité

La théorie de la complexité s'intéresse à l'étude de classification de problèmes et les frontières existant entre ces différentes classes. Elle étudie aussi les limites de calcul pour résoudre un problème donné. La problématique centrale, la plus connue, dans cette théorie est la fameuse question : **P = ? NP**. En d'autres termes, s'il est toujours facile de vérifier une solution, est-il aussi facile de trouver une solution ? la réponse à cette question résout plusieurs autre grandes questions de cette discipline, mais bien entendu pas toutes les questions.

Il n'y a pas de raison de croire que l'égalité (**P = NP**) soit vraie. L'attente de la plupart des informaticiens et mathématiciens est que l'égalité soit fausse. Malheureusement, il n'existe pas de preuve pour cette assertion. Par conséquent, toute une théorie est construite sur les classes **P** et **NP** sans que l'on puisse affirmer s'il existe un problème dans **NP** qui ne soit pas dans **P**. Par conséquent, quelqu'un pourrait dire, à juste titre d'ailleurs, quelle est l'intérêt d'une théorie si elle ne peut répondre à la question de connaître le degré de difficulté d'un problème donné. Une réponse à cette question vient de la notion de **NP-complétude**.

B.4 La NP-complétude

La théorie de la **NP-complétude** concerne la reconnaissance des problèmes les plus durs de la classe **NP**. La notion de la difficulté d'un problème qui est introduite dans cette

classe est celle d'une classe de problème qui sont équivalents en ce sens que si l'un d'eux est prouvé être facile alors tous les problèmes de **NP** le sont. Inversement, si l'un d'eux est prouvé être difficile, alors la classe **NP** est distincte de la classe **P**.

Définition B.8 *Un problème de décision est dit **NP-complet** si tout problème de la classe **NP** lui est polynomialement réductible.*

Notons qu'il existe des problèmes dans **NP** qui ne sont pas dans **P** et qui, vraisemblablement, ne seront pas dans la classe **NPC** (pour dire **NP-complet**).

Le concept central relié à la définition de la **NP-complétude** est celui de la réduction entre problèmes.

Définition B.9 *(Réductibilité) : un problème X peut être ramené (réduit) à un autre problème X' si une instance quelconque de X peut être facilement reformulée comme instance de X' dont la solution sera aussi solution pour X.*

Définition B.10 *(Réductibilité en temps polynomial) : la fonction qui réduit le problème X en X' peut être obtenue en temps polynomial.*

Théorème B.1 *(Théorème de Cook) : Le problème de satisfiabilité est **NP-complet** .*

Cook (1971) a montré que tous les problèmes de la classe **NP** sont réductibles au problème de la satisfiabilité d'une expression logique quelconque.

Autrement dit, si jamais quelqu'un venait à trouver un algorithme polynomial pour le problème de **SAT**, alors la question de savoir si $\mathbf{P} = \mathbf{NP}$ est résolu de fait !

Un problème (P) quelconque est dit " **NP-dur** " s'il existe une réduction polynomial du problème de satisfaisabilité à (P). Les problèmes **NP-durs** sont des problèmes au moins aussi difficile que les problèmes **NP-complets**. Le problème de voyageur de commerce dont le problème de reconnaissance associé est **NP-complet**, est **NP-dur**.

Il est utile de signaler que montrer q"un problème est **NP-complet** signifie qu'il existe au moins une instance pour laquelle le seul algorithme connu pour résoudre sur cette instance est exponentiel.

Par ailleurs, réduire un problème (P_1) à un autre problème (P_2) revient à montrer que (P_1) est au moins plus facile à résoudre que (P_2), et (P_2) est au moins plus difficile à résoudre que (P_1). Autrement dit, cette réduction signifie que (P_1) est un cas particulier de (P_2).

Sachant que la réduction polynomiale est transitive, pour montrer qu'un nouveau problème (P) est **NP-complet**, on procède comme suit :

1. Montrer (P) est dans **NP**.
2. Choisir un problème, (P_2), **NP-complet** approprié.
3. Construire une réduction f, qui transformant (P_2) vers (P) telle que

$$I \text{ une oui-instance de } (P_2) \Leftrightarrow f(I) \text{ une oui-instance de } (P).$$

ANNEXE C

Éléments d'analyse convexe

Pour étudier les problèmes d'optimisation non-différentiable, il est nécessaire de recourir à des outils spécifiques dont l'étude est basée sur l'analyse convexe [J.B. Hiriart-Urruty (2001)]. Dans ce chapitre, nous présentons rapidement quelques éléments de l'analyse convexe requis pour l'étude de l'optimisation non-différentiable.

On se place sur l'espace vectoriel $\mathbb{R}^n, n \in \mathbb{N}$. On le munit d'un produit scalaire $\langle ., . \rangle$. La norme associée au produit scalaire est notée $\|.\|_2$. Elle est définie pour tout $x \in \mathbb{R}^n$ par $\|x\|_2 = \sqrt{\langle x, x \rangle}$.

Définition C.1 ***(Ensemble convexe)*** *Soit $X \subseteq \mathbb{R}^n$ un ensemble. On dit que X est convexe si :*

$$\forall x_1, x_2 \in X, \forall \alpha \in [0,1], \alpha x_1 + (1-\alpha) x_2 \in X. \tag{C.1}$$

Définition C.2 ***(Fonction convexe)*** *Une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}$ est convexe si pour tout $x, y \in \mathbb{R}^n$ et $\lambda \in [0,1]$ l'inégalité suivante est vérifiée :*

$$f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y)$$

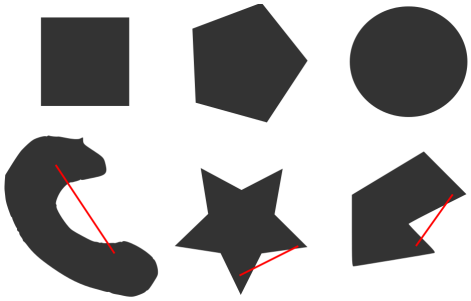


FIGURE C.1 – En haut : quelques exemples d'ensembles convexes en 2 dimensions. En bas : quelques exemples d'ensembles non convexes.

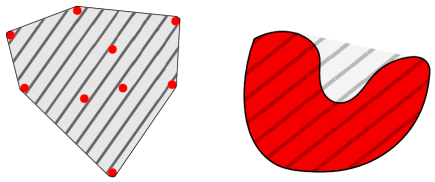


FIGURE C.2 – Exemples d'enveloppes convexes. A gauche : enveloppe convexe d'un ensemble discret. A droite : enveloppe convexe d'un ensemble continu.

Définition C.3 ***(Fonction concave)*** *Une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}^n$ est concave si pour tout $x, y \in \mathbb{R}^n$ et $\lambda \in [0,1]$ l'inégalité suivante est vérifiée :*

$$f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda)f(y)$$

Définition C.4 ***(Combinaison convexe)*** *Soient $x_1, x_2, ..., x_m$, m éléments de $\mathbb{R}^n$. On dit que x est combinaison convexe de ces points s'il existe $\alpha_1, \alpha_2, ..., \alpha_m$ tels que :*

$$\forall i \in \{1, ..., m\}, \alpha_i \in \mathbb{R}_+, \sum_{i=1}^{m} \alpha_i = 1, x = \sum_{i=1}^{m} \alpha_i x_i.$$

Définition C.5 ***(Enveloppe convexe)*** *Soit $X \subseteq \mathbb{R}^n$ un ensemble. On appelle enveloppe convexe de X et on note $conv(X)$ l'ensemble convexe le plus petit contenant X. En dimension finie, c'est aussi l'ensemble des combinaisons convexes d'éléments de X :*

$$conv(X) = \{x \in \mathbb{R}^n : x = \sum_{i=1}^{m} \alpha_i x_i \text{ où } x_i \in X, m \in \mathbb{N} \text{ et} \sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0\}.$$

Théorème C.1 *f est convexe si et seulement si son épigraphe*

$$epi(f) = \{(x,t) \in \mathbb{R}^n \times \mathbb{R}, f(x) \leq t\}$$

est convexe.

Preuve Si $(x_1, t_1) \in epi(f)$ et $(x_2, t_2) \in epi(f)$ alors pour tout $\alpha \in [0,1]$ on a

$$\alpha t_1 + (1-\alpha)t_2 \geq \alpha f(x_1) + (1-\alpha)f(x_2) \geq f(\alpha x_1 + (1-\alpha)x_2).$$

Ainsi, $(\alpha x_1 + (1-\alpha)x_2, \alpha t_1 + (1-\alpha)t_2) \in epi(f)$.

Réciproquement, si $epi(f)$ est convexe, alors pour $x_1, x_2 \in \mathbb{R}^n$, on a

$$(x_1, f(x_1)) \in epi(f), (x_2, f(x_2)) \in epi(f)$$

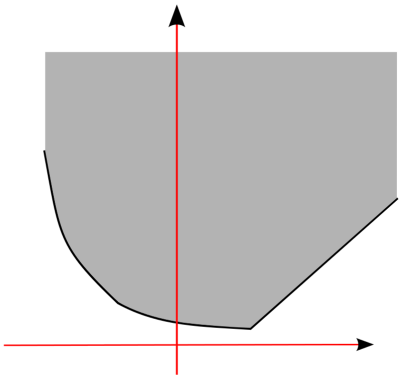


FIGURE C.3 – L'épigraphe de la fonction est la zone grisée au-dessus du graphe de la fonction.

Ainsi$(\alpha x_1 + (1-\alpha)x_2, \alpha t_1 + (1-\alpha)t_2) \in epi(f)$. , soit encore

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2).$$

■

Définition C.6 ***(Sous-gradient et sous-différentiel)*** *Soit f une fonction convexe. Un vecteur $s \in \mathbb{R}^n$ est appelé sous-gradient de f au point $x_0 \in dom(f)$ si*

$$\forall x \in dom(f), f(x) \geq f(x_0) + \langle s, x - x_0 \rangle.$$

L'ensemble de tous les sous-gradients en x_0 est appelé sous-différentiel de f. Il est noté $\partial f(x_0)$.

L'interprétation géométrique du sous-différentiel est la suivante. Il est formé par toutes les directions des hyperplans qui passent par le point $(x, f(x))$ et restent "sous" le graphe de la fonction f. Ces hyperplans sont appelés hyperplans support ou hyperplans d'appui au graphe de f en x.

Lemme C.1 *Le sous-différentiel $\partial f(x_0) = \{s \in \mathbb{R}^n, \forall x \in \mathbb{R}^n, f(x) \geq f(x_0) + \langle s, x - x_0 \rangle\}$ est un ensemble convexe fermé.*

Preuve Soient s_1 et s_2 des éléments de $\partial f(x_0)$. On a $\forall y \in \mathbb{R}^n$:

$$f(y) \geq f(x) + \langle s_1, y - x \rangle$$

$$f(y) \geq f(x) + \langle s_2, y - x \rangle$$

Soit $\alpha \in [0, 1]$. En multipliant la première inégalité par α, la deuxième par (1_α) et en sommant, on voit que $\alpha s_1 + (1-\alpha)s_2 \in \partial f(x_0)$. ■

Bibliographie

[E. Allen (1987)] E. Allen, R. Helgason, al. Kennington, *A generalization of Poliak's convergence results for subgradient optimization. Mathematical Programming, 37, 309-317, 1987.*

[S. Arora (1998)] S. Arora, *Polynomail time approximation shemes for eculiden traveling salsman and geometric problems. Journal of the ACM (JACM), 45(5), 735–782, 1998.*

[M.S. Bazaraa (1981)] M.S Bazaraa, H.D., Sherali, *On the choice of step sizes in subgradient optimization. Europ. J. Oper. Res. 7, 380-388, 1981.*

[M.S. Bazaraa (2006)] M.S. Bazaraa, H.D. Sherali, C.M. Shetty, *Non linear programming : Theory and algorithms. Wiley-Interscience series in discrete mathematics and optimization, 2006.*

[R. Belgacem (2017)] R. Belgacem, A. Amir, *A new modified deflected subgradient method. Journal of King Saud University - Science, In Press, 2017.*

[R.S. Burachik (2010)] R.S. Burachik, C.Y Kaya, *A Deflected Subgradient Method Using a General Augmented Lagrangian Duality with Implications on Penalty Methods. In : Burachik R., Yao JC. (eds) Variational Analysis and Generalized Differentiation in Optimization and Control. Springer Optimization and Its Applications, 47. Springer, New York, NY, 2010.*

[P.M. Camerini(1975)] P.M. Camerini, L. Fratta et F. Maffioli, *On improving relaxation methods by modified gradient techniques. In : Balinski M.L., Wolfe P. (eds) Nondifferentiable Optimization. Math. Program. Studies. 3. Springer, Berlin, Heidelberg., 1975*

[G. Chartrand (2005)] G. Chartrand and P. Zhang *Introduction to graph theory. McGraw-Hill, New York, 2005.*

[J.C. Culioli (1994)] J.C. Culioli, *Introduction à l'optimisation, Edition Ellipses, 1975.*

[G.B. Dantzig (1951)] G.B. Dantzig *Maximization of a linear function of variables subject to linear inequalities. In : Activity Analysis of Production and Allocation (T.C. Koopmans, ed.), Wiley, New York, pp. 359–373, 1951.*

[G.B. Dantzig (1963)] G.B. Dantzig *Linear Programming and Extensions, Princeton University Press, Princeton, 1963.*

[V.F. Dem'yanov(1985)] V.F. Dem'yanov and L.V. Vasil'ev, *Nondifferentiable Optimization. Optimization Software, Inc. Publications Division, New York, 1985.*

[M. Diaby (2016)] M. Diaby and M.H. Karwan, *Advances in combinatorial optimization : linear programming formulation of the travelling salesman and other hard combinatorial optimization problems, 2016.*

[N.A. El-Sherbeny(2010)] N.A. El-Sherbeny, *Vehicle routing with time windows : An overview of exact, heuristic and metaheuristic methods. J. King Saud Univ. Sci. 22, 123-131, 2016.*

[Y.M. Ermoliev(1966)] Y.M. Ermoliev, *Methods for solving nonlinear extremal problems. Cybernetics, 16, 1-14, 1966.*

[M.L. Fisher (1985)] M.L. Fisher, *An application oriented guide to Lagrangian relaxation. Interfaces. 15, 10-21, 1985.*

[R. Fletcher (1964)] R. Fletcher, C.M. Reeves, *minimization by conjugate gradients. Comput. J. 7, 149-154, 1964.*

[F. Fumero (2001)] F. Fumero, *A modified subgradient algorithm for Lagrangian relaxation. Comput. Oper. Res.. 28, 33-52, 2001.*

[M.R. Garey (1990)] M.R. Garey, D.S. Johnson, *Computers and intractability : A guide to the theory of NP completeness. W. H. Freeman & Co., New York, NY, USA, 1990.*

[J.L. Goffin (1977)] J.L. Goffin, *On Convergence Rates of Subgradient Optimization Methods, Mathematical Programming 13, 329-347, 1977.*

[A.M. Geoffrion (1974)] A.M. Geoffrion, *Lagrangian relaxation for integer programming. Mathematical Programming Study 2,(1974), 82– 114, 1974.*

[D. Goldfarb (1989)] D. Goldfarb and M.J. Todd, *Linear programming. In M.J. Todd G.L. Nemhauser A.H.G. Rinnooy Kan, éditeur, Handbooks in operations research and management science, 1, pages 73–170. Elsevier, 1989.*

[B.L. Golden(1988)] B.L. Golden and A.A. Assad, *Vehicle routing : methods and studies, North-Holland, 1988.*

[E. Gustavsson(2015)] E. Gustavsson, M. Patriksson, A.B. Strömberg, *Primal convergence from dual subgradient methods for convex optimization. Math. Program. 150, 365-390, 2015.*

[B. Guta(2003)] B. Guta, *Subgradient Optimization Methods in Integer Programming with an Application to a Radiation Therapy Problem, Ph.D Thesis, University of Kaiserslautern, 2003.*

[F. Harary(1996)] F. Harary, *Survey of methods of automorphism destruction in graphs. Invited address, Eighth Quadrennial International Conference on Graph Theory, Combinatorics, Algorithms and Applications, Kalamazoo, Michigan, 1996.*

[K.H. Helbig-Hansen(1974)] K.H. Helbig-Hansen and J. Krarup, *J. Improvements of the Held-Karp algorithm for the symmetric traveling salesman problem. Mathematical Programming, 7, 87-96, 1974.*

[M.H. Held(1970)] M.H. Held, R.M Karp, *The travelling salesman problem and minimum spanning trees. Oper. Res. 18, 1138-1162, 1970.*

[M.H. Held(1971)] M.H. Held, R.M Karp, *The Traveling salesman problem and minimum spanning trees : Part II. Mathematical Programming 1, 6-25, 1971.*

[M.H. Held(1974)] M.H. Held, P. Wolfe, H.D. Crowder, *Validation of subgradient optimization. Math. Program, 6, 62-88, 1974.*

[J.B. Hiriart-Urruty (2001)] J.B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of convex analysis, Grundlehren Text Editions, Springer, 2001.*

[Y. Hu(2015)] Y. Hu, X. Yang, C.-K. Sim, *Inexact subgradient methods for quasi-convex optimization problems. Europ. J. Oper. Res. 240, 315-327, 2015.*

[D.S. Jonhson(1997)] D.S. Jonhson, L.A. McGeoch, *The traveling salesman problem : a case study in local optimization. In Local search in Combinatorial optimization, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, New York, 1997.*

[N. Karmarkar (1984)] N. Karmarkar, *A New Polynomial-time Algorithm for Linear Programming, Combinatorica 4, 373-395, 1984.*

[L.G. Khachiyan (1979)] L.G. Khachiyan, *A Polynomial Algorithm in Linear Programming, Soviet Mathematics Doklady 20, 191-194,1979.*

[S. Kim (1990)] S. Kim, H. Ahh, S.-C. Cho, *Variable Target Value Subgradient Method. Math. Program. 49, 359-369, 1990.*

[S. Kim (1993)] S. Kim, and B.S. Um, *An improved subgradient method for constrained nondifferentiable optimization, Operations Research Letters 14,61-64, 1993.*

[V. Klee (1972)] V. Klee, G.J. Minty, *How good is the simplex algorithm ? In : Inequalities III (O. Shisha, ed.), Academic Press, New York, pp. 159–175, 1972.*

[G. Laporte (1992)] G. Laporte, *The traveling salesman problem : an overview of exact and approximate algorithms. European Journal of Operational Research, 59, 231-247, 1992.*

[E.L. Lawler (1985)] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan et D.B. Shmoys, *The traveling salesman problem : a guided tour of combinatorial optimization, Wiley, 1985.*

[M. Lebbar (2000)] M. Lebbar, *Résolution de problèmes combinatoires dans l'industrie, apport de la programmation mathématique et des techniques de décomposition. Thèse de doctorat, École Centrale, Paris, 2000.*

[C. Lim (2006)] C. Lim, H.D. Sherali, *Convergence and computational analyses for Some variable target value and subgradient deflection methods. Comput. Optim. Appl., 34, 409-428, 2006.*

[S. Martello(1990)] S. Martello et P. Toth, *Knapsack Problems : Algorithm and Computer Implementations, John Wiley and Son, New York, (1990).*

[P. Merchandani(1990)] P. Merchandani et R. Francis, *Discreted Location Theory, John Wiley and Son, New York, (1990).*

[A. Nedic(2010)] A. Nedic, D.P. Bertsekas, *The effect of deterministic noise in subgradient methods. Math. Program. 125, 75-99, 2010.*

[A.S. Nemirovskii (1978)] A.S. Nemirovskii and D.B. Yudin, *Cesaro convergence of gradient method approximation of saddle points for convex-concave functions, Doklady Akademii Nauk SSSR, 239,1978.*

[Y. Nestrov (2014)] Y. Nestrov, *Subgradient Methods for Huge-Scale Optimizations Problems. Math. Program. 146, 275-297, 2014.*

[J. Nocedal (1999)] J. Nocedal and S.J. Wright, *Numerical Optimization. Springer- Verlag New York, Inc., 1999.*

[B.T. Polyak (1967)] B.T. Polyak, *A general method of solving extremum problems. Sov. Math. Dokl. 8, 593-597, 1967.*

[B.T. Polyak (1969)] B.T. Polyak, *Minimization of unsmooth functionals, USSR Comutational Mathematics and Mathematical Physics 9, 14–29,1969.*

[G. Reinelt (1994)] G. Reinelt, *The traveling salesman problem : computational solutions for TSP applications. Lecture Notes in Computer Science 840, Springer Verlag, Berlin, 1994.*

[R.T. Rockafellar (1970)] R.T. Rockafellar, *Rockafellar, R.T. : Convex Analysis. Princeton University Press, 1970.*

[A. Schrijver (1986)] A. Schrijver, *Theory of linear and Integer Programming, Wiley, 1986.*

[H.D. Sherali(2000)] H.D. Sherali, G. Choi, C.H.Tuncbilek, *A Variable target value method for nondifferentiable optimization. Oper. Res. Lett. 26, 1-8,2000.*

[H.D. Sherali(1989)] H.D. Sherali, O. Ulular, *A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems. Appl. Math. Optim. 20, 193-221,1989.*

[N.Z. Shor(1962)] N.Z. Shor, *Cut-off method with space extension in convex programming problems. Cybernetics 13, 94–96, 1977.*

[N.Z. Shor(1985)] N.Z. Shor(1985), *Minimization methods for non differentiable functions, volume 3 of Springer Series in Computational Mathematics. Springer-Verlag, Berlin. Translated from the Russian by K. C. Kiwiel and A. Ruszczynski, 1985.*

[S. Sra(2012)] S. Sra, S. Nowozin, S.J. Wright, *Optimization For Machine Learning. MIT Press,2012.*

[J. Teghem (2003)] J. Teghem, *Programmation linéaire, Édition de l'université de Bruxelles, 2003.*

[C.L. Valenzuela (1995)] C.L. Valenzuela, A J. Jones, *Estimating Held-Karp lower bond for the geometric TSP, 1995.*

[T. Volgenant (1982)] T. Volgenant, R. Jonker, *A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. European Journal of Operational Research, 9, 83-89, 1982.*

[D.B. West (1996)] D.B.West, *An Introduction to Graph Theory. Prentice-Hall, 1996.*

[L.A. Wolsey (1998)] L.A. Wolsey, *Integer Programming, John Wiley and sons, Inc, 1998.*