Méthodes optimales et sous-optimales d'allocation de ressources efficace en codage numérique

Axel Le Poupon

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

présentée pour obtenir le grade de docteur

de Telecom ParisTech

Spécialité : Electronique et Communications

Axel Le Poupon

Méthodes optimales et sous-optimales d'allocation de ressources efficace en codage numérique

soutenue le 4 septembre 2009 devant le jury composé de

Pierre DUHAMEL	Président
Inbar FIJALKOW	Rapporteurs
Fabrice LABEAU	
Jean-Claude BELFIORE	Examinateurs
Pierre SIOHAN	
Riadh ABDELFATTAH	
Olivier RIOUL	Directeur de thèse

À Telecom ParisTech

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

PhD thesis

Telecom ParisTech

Communications and Electronics department

Digital Communications group

Axel Le Poupon

Optimal and sub-optimal methods for efficient resource allocation in digital coding

Defense date : September, 4^{th} 2009

Committee in charge :

Pierre DUHAMEL	Chairman
Inbar FIJALKOW	Reporters
Fabrice LABEAU	
Jean-Claude BELFIORE	Examiners
Pierre SIOHAN	
Riadh ABDELFATTAH	
Olivier RIOUL	Advisor

At Telecom ParisTech

Chercher n'est pas une chose et trouver une autre,
mais le gain de la recherche, c'est la recherche même.
Saint Grégoire de Nysse

À tous ceux qui cherchent.

Remerciements

L'expérience est le nom que chacun donne à ses erreurs.
Oscar Wilde

Cette citation résume bien à elle seule l'ensemble des années passées sur ces travaux. Quelle que soit l'expérience que j'ai pu y acquérir, cette thèse a été pour moi l'occasion de multiples fourvoiements et d'aveuglements passagers. Ces remerciements sont donc l'occasion de manifester ma reconnaissance à tout ceux qui m'ont apporté un peu de la lumière qui éclaire ce manuscrit.

Je tiens tout particulièrement à remercier Olivier Rioul sans qui tout ceci ne serait qu'un vague balbutiement sans valeur. Il a su me faire croire en ce travail et à dissiper mes doutes pour me pousser un peu plus vers le bout du tunnel. Je suis reconnaissant du temps et de l'énergie qu'il a pu me consacrer. Il a ma plus grande estime pour ses nombreuses compétences aussi bien humaines qu'intellectuelles.

Je remercie chaleureusement Bernard Robinet, Christian Queinnec et Henri Maître. J'ai pu trouver en chacun d'eux une oreille attentive et un conseiller pertinent. J'ai un immense respect pour les aptitudes dont ils font preuve dans la gestion du vaste navire que représente une école doctorale.

Je souhaite exprimer toute ma gratitude à Pierre Duhamel. Ce fut un honneur pour moi d'avoir un président du jury de son envergure. Sa confiance et ses encouragements m'ont profondément touché et je l'en remercie.

Je suis aussi particulièrement reconnaissant envers Fabrice Labeau et Inbar Fijalkow pour avoir rempli la lourde tâche de rapporteurs. Leurs comptes-rendus respectifs, par leur finesse d'analyse et leurs remarques judicieuses, m'ont permis de prendre un bien meilleur recul sur mon mémoire. Je leur exprime mes sincères remerciements.

Je remercie Jean-Claude Belfiore, Pierre Siohan et Riadh Abdelfattah pour leur active participation au jury. La vision originale que chacun a portée sur mes travaux a été des plus enrichissantes.

J'aimerais spécialement remercier Sylvain Chaillou et Elie Danjou pour leur grande assistance morale et intellectuelle. J'ai partagé avec eux les joies et les peines de la thèse et j'espère avoir

encore de nombreuses choses à réaliser en leur compagnie. Je remercie Bruno, Sébastien et David, compagnons sans lesquels Telecom aurait été bien vide. Merci à Florent, Tristan, Ol et Cédric pour les instants mélomanes. Merci enfin à mes chers camarades du Ministère de la Défense. Tous, à leur manière, m'ont permis de rendre ce rêve réel.

Et pour conclure, il est tout à fait impensable d'oublier ici celle qui m'a donné la force d'aller jusqu'au bout. Il m'est difficile de trouver les mots justes pour souligner son indéfectible soutien. Elle restera pour toujours le fourreau de mon âme. Claire, merci.

Paris, le 9 septembre 2009

Résumé

Le problème d'allocation de ressources consiste à optimiser le coût global d'un système en répartissant selon diverses composantes une ressource contrainte par un budget. Chaque composante du système est régie par une loi reliant la ressource au coût qu'elle engendre. Cette description s'applique à de nombreux contextes dont les deux principaux évoqués ici sont issus du codage en communications numériques. Ainsi, les systèmes de codage de source par transformée ou de codage de canal à modulation multi-porteuses intègrent ce problème dans la recherche d'un point de fonctionnement optimal, s'adaptant bien à un formalisme commun.

En communications numériques, différentes méthodes de résolution ont été élaborées au cours des quarante dernières années. Leur validité se limite souvent à des cas non-exhaustifs et leurs performances dépendent des caractéristiques des fonctions considérées. De plus, complexité et optimalité revêtent une importance particulière mais sont rarement optimisées de concert en raison des techniques de modélisation et des principes de recherche employés. Un des problèmes majeurs provient en particulier de l'existence de points dit "cachés" n'appartenant pas à l'enveloppe convexe du nuage global et souvent ignorés par les méthodes actuelles d'optimisation.

Cette thèse propose d'aborder le problème d'allocation de ressources en communications numériques sous un angle nouveau et global. La reformulation complète du problème et de l'ensemble des concepts sous-jacents permet ainsi d'énoncer des critères propices à la description de quatre algorithmes novateurs.

Les deux premiers algorithmes établissent de nouvelles techniques de recherche optimales. L'une des méthodes s'apparente à une technique de "branche et limite" et développe le concept de chemins convexes "admissibles" au travers d'un nuage de points. La seconde méthode établit une relation d'ordre dont la mise en œuvre sélectionne rapidement des points de fonctionnement optimaux. Les deux autres algorithmes augmentent leur rapidité de résolution en relâchant la contrainte d'optimalité. Le premier algorithme élimine localement des points de fonctionnements afin de trouver de nouvelles solutions globales sur les enveloppes convexes alors générées. Le second algorithme dérive le concept de chemins convexes dans un cas sous optimal mais proche de l'optimalité.

Tous ces algorithmes sont largement décrits dans cette thèse et leurs performances expérimentalement éprouvées. De plus, leur application, initialement motivée par des problèmes de codage en communications numériques, ne se limite pas uniquement à ceux-ci. Quelques perspectives envisageables ultérieurement sont suggérées en fin de manuscrit.

Abstract

The resource allocation problem in digital coding consists in the global cost optimization of a system distributing in several components a resource constrained by a budget. Each component follows a rule linking the resource to the cost it spends. We meet this description in different contexts. We especially deal with two major ones issued from the coding domain in digital communications. In one hand, transform source coding systems, on the other hand, channel coding with multi-carrier modulation systems, corresponds to the same notions of optimization.

In digital communications, different solving methods have been studied for forty years. Their performance depends on the studied functions and their validity is usually limited to some non-exhaustive cases. Moreover, complexity and optimality are essential but rarely managed jointly due to modelization techniques and searching methods. One of the major problem comes from the existence of so called "hidden" points. They are not on the convex hull of the global feasible cloud and usually ignored by the current optimisation methods.

This thesis proposes to deal with the resource allocation problem in digital communications over a new and global point of view. The complete rewriting of the problem and the underlying concepts allows to state the criterion for the description of four innovative algorithms.

Two of these algorithms built new techniques of optimal research. The first is related to "branch and bound" method and develops the concept of "feasible" convex path through a global cloud. The second algorithm uses an ordering relation which classifies the points in a way that the optimal ones are quicker to find. Two other algorithms decrease their complexity by relaxing the optimality constraint. The first algorithm eliminates local points in a component in order to find optimal points deduced from the new convex hull. The second algorithm derives the concept of convex path in a sub but quasi-optimal case.

All of these algorithms and their experimental performances are well and largely described in this thesis. Moreover, their application, initially motivated by digital communications coding problems, is not limited to this domain. Some prospects are proposed in the final part.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations et notations

Abréviations

Pour des raisons de lisibilité, la signification d'une abréviation ou d'un acronyme n'est souvent explicitée qu'à sa première apparition dans le document. Nous présentons donc ici les principales abréviations employées dans ce document. Par ailleurs, il arrive que l'abréviation la plus usuelle soit issue d'un terme anglais, auquel cas nous joignons ici la traduction française.

DMT	Discrete Multi Tone	Multi-ton discrète
CU		Convexe Uniforme
CUT		Convexe Uniforme à Trous
CNU		Convexe Non-Uniforme
NCU		Non-Convexe Uniforme
NCNU		Non-Convexe Non-Uniforme
RDPO		Recherche Directe de Points Optimaux
RDPE		Recherche Directe de Points d'Enveloppe

Notations

Nous avons regroupé ci-dessous les principales notations employées dans ce document. Dans la mesure du possible, nous avons tenté de conserver les mêmes notations d'un chapitre à l'autre. Certaines notations, apparaissant uniquement de manière ponctuelle, ont été omises.

N	nombre de composantes du problème d'allocation considéré
M_i	nombre de points dans la composante i
R_i	taux de la composante i
D_i	distorsion de la composante i
R	taux global
D	distorsion globale
k_i	index d'allocation de la composante i
$\underline{k}$	vecteur d'allocation global
$\underline{1}$	vecteur d'allocation minimal ne contenant que des 1
$\underline{M}$	vecteur d'allocation maximal contenant les M_i
σ_i^2	variance d'une composante i
R_b	taux budget

(R^*, D^*)	point optimal
(R^e, D^e)	point d'enveloppe
(R^h, D^h)	point caché
$\mathscr{L}$	lagrangien
λ	multiplicateur critique, multiplicateur de Lagrange
$\vec{\Delta}$	opération élémentaire
$\vec{\Delta}_{k_i}$	opération élémentaire associée à un point $(R_i(k_i), D_i(k_i))$
$\vec{\Delta}_{\underline{k}}$	opération élémentaire associée à un point $(R(\underline{k}), D(\underline{k}))$
$\vec{\Delta}_{\lambda}$	opération élémentaire associée à un multiplicateur critique λ
∇^h	triangle caché
∇^h_b	triangle où reside la solution optimale pour le budget R_b
D_{virt}	distorsion d'un point virtuel de taux R_b situé sur l'enveloppe convexe
$\mathcal{P}_D$	perte en distorsion
$\mathcal{H}$	enveloppe convexe
T	opérateur de translation
$\mathcal{R}$	région de recherche des points optimaux dans le plan R-D
$\mathcal{R}_{\text{mort}}$	région complémentaire de $\mathcal{R}$ au dessus de l'enveloppe convexe dans le plan R-D
$\mathcal{S}_{\text{mort}}$	sous-région de $\mathcal{R}_{\text{dead}}$ définie pour un opérateur de translation T donné
$\prec$	relation d'ordre total basée sur un ordre lexicographique de couple $(D + \lambda R, D)$
$\mathcal{I}$	ensemble d'indexation
$\mathcal{E}_\kappa$	opération d'omission d'index
$\lambda^{\mathcal{E}}$	nouveau multiplicateur critique issu d'une omission d'index

Introduction

Alors que depuis maintenant une trentaine d'années la planète subit une crise mondiale énergétique liée à l'épuisement progressif de ressources très convoitées, le problème de la répartition de ces ressources s'impose clairement aux yeux du grand public comme un défi majeur. De même, le processus croissant de mondialisation et la globalisation des marchés internationaux engendrent une nouvelle approche de la distribution des ressources, certains y voyant un prétexte à une disparité grandissante. Plus social mais tout autant d'actualité en France, la réforme du régime des retraites par répartition représente un exemple concret de tentative de résolution d'un problème où le besoin s'accroît alors que la ressource s'amenuise.

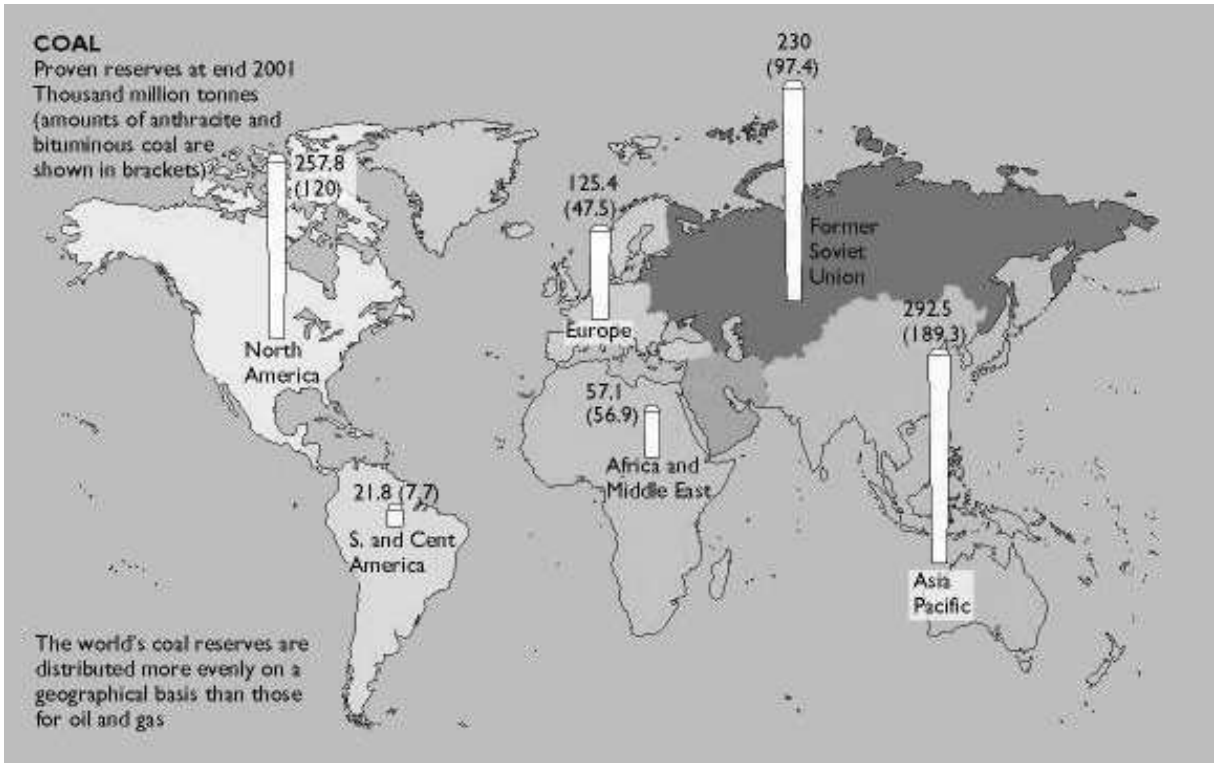


FIG. 1: Exemple de la répartition d'une matière première dans le monde : le charbon (*source BGR 2001*)

Quel que soit le domaine d'étude, le problème reste toujours le même : "Comment répartir efficacement une ressource limitée ?". Chacun prend aujourd'hui conscience de la fragilité d'un

système où le partage des ressources serait mal établi. Une ressource, par essence (sans jeu de mot facile) limitée, demande à être distribuée avec parcimonie, partagée avec efficacité, repartie à moindre coût.

Le terme d'*allocation de ressources* est bien connu du monde des économiciens. Il s'agit d'un concept relatif à l'utilisation des ressources rares et notamment aux facteurs de production (travail, capital, matières premières) pour satisfaire à court et long terme les besoins de consommation de la population. D'autre part, les informaticiens emploient couramment le terme d'allocation de ressources afin de définir la mise à disposition de mémoire pour une utilisation au profit de variables. Ces deux concepts, issus de mondes bien différents, s'approchent du problème que nous allons traiter dans ce manuscrit. Nous allons toutefois l'aborder de manière quelque peu plus spécifique en établissant les fondements d'une étude relative au domaine du traitement du signal, et plus particulièrement dans le cadre de systèmes de codage numérique.

La littérature originellement concernée par l'application de notre problème mentionne celui-ci sous les termes anglo-saxons *rate loading*, *bit loading*, *power loading*, *power allocation*, etc. Ces termes reflètent la notion d'attribution d'une ressource de taux ou de puissance donnée dans un système de codage (de source ou de canal). En effet, le système est décomposé de telle manière que la ressource peut être attribuée suivant plusieurs composantes, chacune ayant ses spécificités.

Par contre, ces appellations ne font pas mention de la seconde variable qui donne tout son sens au problème d'allocation : la fonction de coût associée à la ressource. C'est cette fonction de coût qui va nous permettre de déterminer la répartition *optimale* répondant au problème d'allocation de ressources. En effet, la ressource, couplée à cette fonction dont les caractéristiques dépendent de l'origine du problème considéré, est contrainte à ne pas dépasser un certain budget. Cette limitation nécessite par conséquent la recherche du point de fonctionnement optimal répondant au mieux au critère imposé, c'est à dire à la dépense minimale.

L'optimalité possède dans notre vocabulaire un sens double, sens qui dépendra du terme auquel on le rattache. Nous parlerons ainsi de *solution optimale* lorsque celle-ci est la meilleure solution au problème d'allocation de ressources pour un budget particulier. Il s'agit alors de caractériser un point dans un ensemble global. Ce point sera qualifié d'optimal indépendamment d'une quelconque méthode de résolution, l'optimalité étant alors considérée comme une caractéristique absolue du point.

Cependant nous parlerons aussi d'*algorithme optimal*. Un algorithme optimal n'est pas seulement un algorithme qui trouve des points optimaux, mais surtout un algorithme trouvant à chaque fois la solution optimale au problème d'allocation de ressource *quel que soit* le budget. Nous caractérisons alors l'optimalité d'une méthode par les solutions qu'elle propose. L'optimalité d'un algorithme est reliée au problème pour lequel il s'applique, c'est à dire qu'elle est relative aux caractéristiques des fonctions considérées. A contrario, nous nous référerons aux procédures trouvant des points non-optimaux sous le terme de méthodes *sous-optimales*.

Remarquons que l'optimalité n'est toutefois pas l'unique critère de performance pour les procédures d'allocation de ressources. Une des préoccupations essentielles de tout algorithme est l'efficacité de celui-ci en terme de complexité. Un algorithme efficace se doit de résoudre le problème d'allocation le plus rapidement possible afin de répondre aux contraintes pratiques d'implémentation. À l'heure actuelle, le compromis entre optimalité et complexité est un enjeu fondamental, d'autant plus qu'il n'a été que difficilement maîtrisé par les algorithmes développés jusqu'à présent. L'optimalité de certaines méthodes est atteinte au prix d'une complexité excessive, inadaptée aux contraintes spécifiques du domaine du codage en communications numériques. On utilisera donc exclusivement le terme d'*efficacité* lorsque l'on souhaitera évoquer la complexité de ces algorithmes.

L'un des principaux moyens pour réduire la complexité consiste à sélectionner des points sous-optimaux possédant des propriétés particulières facilement exploitables. De telles propriétés permettent ainsi de calculer ces points à moindre coût. Les méthodes efficaces actuelles se basent essentiellement sur des propriétés de linéarité et de convexité afin d'atteindre une complexité raisonnable. Toute la finesse de la méthode réside alors dans la proximité de ces points sous-optimaux à la solution optimale, ce qui dépend largement des caractéristiques des fonctions mises en jeu.

La proximité est ici une notion subjective dont il nous faudra fixer une méthode de mesure plus formelle. D'une telle mesure découlera un critère de comparaison capable de classifier les méthodes sous-optimales autrement que par leur complexité. Nous établirons dans ce manuscrit un critère positif, appelé par la suite *perte en distortion*, et sa minimisation sera un de nos objectifs, celui-ci s'annulant pour des méthodes optimales.

L'ensemble de ces concepts guidera notre raisonnement tout au long de cette thèse dont l'organisation est résumée de la manière suivante.

Organisation du document

Ce manuscrit se divise en 6 chapitres et 3 annexes. Les annexes contiennent des calculs et des figures que nous n'avons pas voulu laisser dans le corps principal de ce document pour des raisons de légèreté.

Le chapitre 1 de ce document débute par une description générale du problème d'allocation de ressources. Nous évoquerons les domaines concernés par cette thèse et leurs connections au problème initial. Nous présenterons l'ensemble des concepts nécessaires à notre étude, que ce soit les définitions et les théorèmes fondamentaux, les ensembles de simulations ou les critères de comparaisons des méthodes étudiées.

Au chapitre 2, nous verrons l'ensemble des travaux publiés sur le thème de l'allocation de ressources telle que nous l'avons présentée. Nous introduirons un certain nombre d'algorithmes

qui serviront de référence en terme de performance. Nous justifierons ainsi la nécessité d'aller plus loin dans la recherche de points optimaux ou dans la rapidité de résolution du problème.

Nous proposerons au chapitre 3 un algorithme visant l'optimalité. Fondée sur la mise en évidence de chemins convexes au sein du nuage global des points admissibles, cette méthode sera largement détaillée et ses performances seront évoquées en fin de chapitre.

A l'instar du chapitre précédent, le chapitre 4 développera un autre algorithme optimal. Nous verrons alors l'efficacité de cet algorithme basé sur le réordonnancement du nuage global suivant une relation d'ordre particulière.

Le chapitre 5 traitera d'un algorithme pour répondre de manière sous-optimale au problème d'allocation de ressource. Cette méthode, découlant d'un principe d'omission d'index au sein des composantes étudiées, verra son principal gain résider dans la proximité des solutions trouvées avec la solution optimale au prix d'une complexité, quant à elle, grandement réduite en comparaison des procédures optimales.

Le chapitre 6 développera une adaptation de notre algorithme de chemins convexes présenté précédemment. Dans ce cas, la méthode sera sous-optimale mais offrira des solutions proches de l'optimalité avec une complexité se rapprochant des méthodes les plus rapides à l'heure actuelle.

À l'issue de ces six chapitres, nous ferons un bilan de tous ces travaux et évoquerons certaines des perspectives qu'ouvre cette thèse.

L'annexe A évoquera l'équivalence de deux problèmes dont les énoncés sont proches et duaux. Cette équivalence est démontrée dans l'annexe.

L'annexe B présentera des exemples graphiques des différents ensembles de fonctions considérés pour notre corpus de tests des algorithmes.

L'annexe C proposera quelques développements de calculs de complexité et de coût mémoire d'algorithmes spécifiques.

Contribution de la thèse

Cette thèse est en premier lieu l'occasion d'établir un bilan global unifié autour du problème d'allocation de ressources en communications numériques. Ce problème, traité de manière disparate dans des contextes variés, est ainsi énoncé indépendamment de son domaine d'étude sous une forme synthétique. Il en découle que les concepts et les enjeux majeurs sont mieux appréhendés. Nous ne retenons que l'essentiel et la comparaison des algorithmes existants en est plus aisée.

D'autre part, cette thèse propose deux nouveaux algorithmes optimaux pour résoudre le problème d'allocation de ressources pour un budget donné. Le premier s'applique aux fonctions convexes et le second à des fonctions quelconques. Leurs complexités respectives sont inférieures à celles des algorithmes de performance équivalente.

Dans un souci d'implémentation efficace, nous avons aussi développé dans cette thèse deux méthodes de complexité équivalente aux algorithmes les plus rapides, mais avec des performances supérieures. La première méthode s'applique à des fonctions convexes, mais peut s'adapter à des ensembles quelconques. La seconde méthode ne réclame pas de contraintes sur les fonctions et s'applique à tout problème.

Au total, quatre nouveaux algorithmes, découlant de méthodes complètement novatrices, sont donc proposés, chacun d'eux apportant un gain important dans l'un des critères que nous étudierons. Alors que les vingt dernières années n'ont vu que des adaptations et remaniements mineurs des principes d'allocation de ressources, cette thèse attaque sous un angle neuf un problème récurrent des communications numériques, amenant autant de perspectives nouvelles.

Chapitre 1

Le problème d'allocation de ressources

La clarté, c'est une juste répartition d'ombres et de lumières.
Johann Wolfgang Von Goethe

1.1 Introduction

Le problème d'allocation de ressources consiste à répartir une variable nommée *ressource* selon différentes composantes et dont la performance est mesurée par une variable de *coût*. Cette ressource est contrainte par un *budget* qui limite celle-ci à un domaine de définition précis. Les variables représentant notre ressource et notre critère de performance sont essentielles, et leurs définitions conditionnent le problème lui-même. Nous nous devons d'expliciter celles-ci et de détailler les conditions d'étude ainsi que le champ d'application de notre problème. Ce premier chapitre s'attelle par conséquent à la description complète du problème et de ses implications.

Nous définirons tout d'abord au paragraphe 1.2 le problème d'allocation de ressources tel que nous l'étudierons tout au long de ce manuscrit. Nous y introduirons les notations et les principaux concepts qui serviront de base à l'ensemble des travaux qui suivent.

Nous aborderons ensuite rapidement au paragraphe 1.3 les variantes éventuelles de ce problème. Nous chercherons ainsi à nous affranchir des complications dans l'écriture de nos algorithmes. Des transformations simples nous ramènerons au problème tel qu'il a été initialement formulé.

Nous verrons les applications concrètes qui motivent notre étude et leur contexte respectif au paragraphe 1.4. Les domaines d'application seront scindés en deux parties, l'une se référant au codage de source, l'autre au codage de canal.

Dans un long paragraphe 1.5, nous définirons l'ensemble des concepts, des propriétés et des théorèmes fondamentaux que nous utiliserons au cours des chapitres suivants. Ainsi décrits, tous ces résultats seront essentiels à la justification et la compréhension des algorithmes développés aux chapitres suivants.

Nous présenterons dans le paragraphe 1.6, un corpus de test, c'est à dire les ensembles de fonctions que l'on voudra représentatives des cas concrets étudiés et sur lesquels nous effectuerons les simulations des différents algorithmes étudiés.

Le paragraphe 1.7 traitera des critères de comparaison qui nous permettront de vérifier les performances respectives des algorithmes sur le corpus de test. Ces critères vont mesurer l'efficacité des méthodes décrites, certains des critères servant de références, d'autres d'indicateurs.

Finalement nous tirerons quelques conclusions au paragraphe 1.8.

1.2 Présentation du problème

Considérons un ensemble fini d'indices $i = 1, \ldots, N$ que nous nommerons *composantes* où N correspond à la *dimension* globale de notre problème. Pour chaque composante i, un ensemble de M_i valeurs $\{R_i\}$, nommées *taux*, et de M_i valeurs $\{D_i\}$, nommées *distorsions*, nous est donné. Cette terminologie provient de la théorie taux-distorsion en codage de source, mais n'est en aucun cas restrictive à son seul domaine d'application. Les R_i et D_i peuvent tout aussi bien être considérés comme un ensemble quelconque de valeurs réelles (positives ou negatives). Nous avons donc un total de $\sum_i M_i$ couples de taux et de distortions qui sont les données de notre problème. Le cardinal d'un ensemble $\{(R_i, D_i)\}$ peut éventuellement être infini, auquel cas $M_i = \infty$.

À partir de ces couples rangés par composantes, on définit des variables globales associées comme suit :

Définition 1.2.1
On appelle R, taux global, *l'expression*

$$R = \sum_{i=1}^{N} R_i \tag{1.1}$$

Définition 1.2.2
On appelle D, distorsion globale, *l'expression*

$$D = \sum_{i}^{N} D_i \tag{1.2}$$

Notons le comportement additif des variables globales de taux et de distorsion. Cette caractéristique est essentielle dans le développement ultérieur de ce document, les propriétés de linéarité étant abondamment employées dans la suite. Les points globaux résultants (R, D) sont représentées dans le plan R-D et forment un ensemble qu'on nommera par la suite "nuage de points".

Dans certains cas, il sera parfois plus judicieux d'adopter une notation par indexation afin de simplifier les expressions littérales ou d'éviter des ambiguïtés dans l'écriture des formules. Soit un ensemble $\{k_1, k_2, \ldots, k_n\}$ d'indices entiers indexant l'ensemble des points taux-distorsion $\{(R_i, D_i)\}$ à la composante i. Chaque k_i prend par conséquent ses valeurs dans $\{1, \ldots, M_i\}$. Nous écrirons le $k_i^{\text{ème}}$ point taux-distorsion de la composante i sous la forme :

$$(R_i(k_i), D_i(k_i)) \tag{1.3}$$

Ainsi décrites, les valeurs globales de taux et de distorsions seront indexées par un vecteur d'allocation de dimension N de coordonnées $\underline{k} = (k_1, \ldots, k_N)$:

$$\left(\begin{array}{rcl} R(\underline{k}) & = & \sum_{i=1}^{N} R_i(k_i) \\ D(\underline{k}) & = & \sum_{i=1}^{N} D_i(k_i) \end{array} \right) \tag{1.4}$$

De cette manière, l'unique description des valeurs de $\underline{k}$ suffit à déterminer un point dans l'espace des ensembles $\{(R_i, D_i)\}$ et donc à définir les distorsions et taux globaux associés. Il est alors pratique et moins lourd d'identifier $(R(\underline{k}), D(\underline{k})))$ à son vecteur d'allocation $\underline{k}$ correspondant et de parler de point $\underline{k}$ dans le plan global R-D.

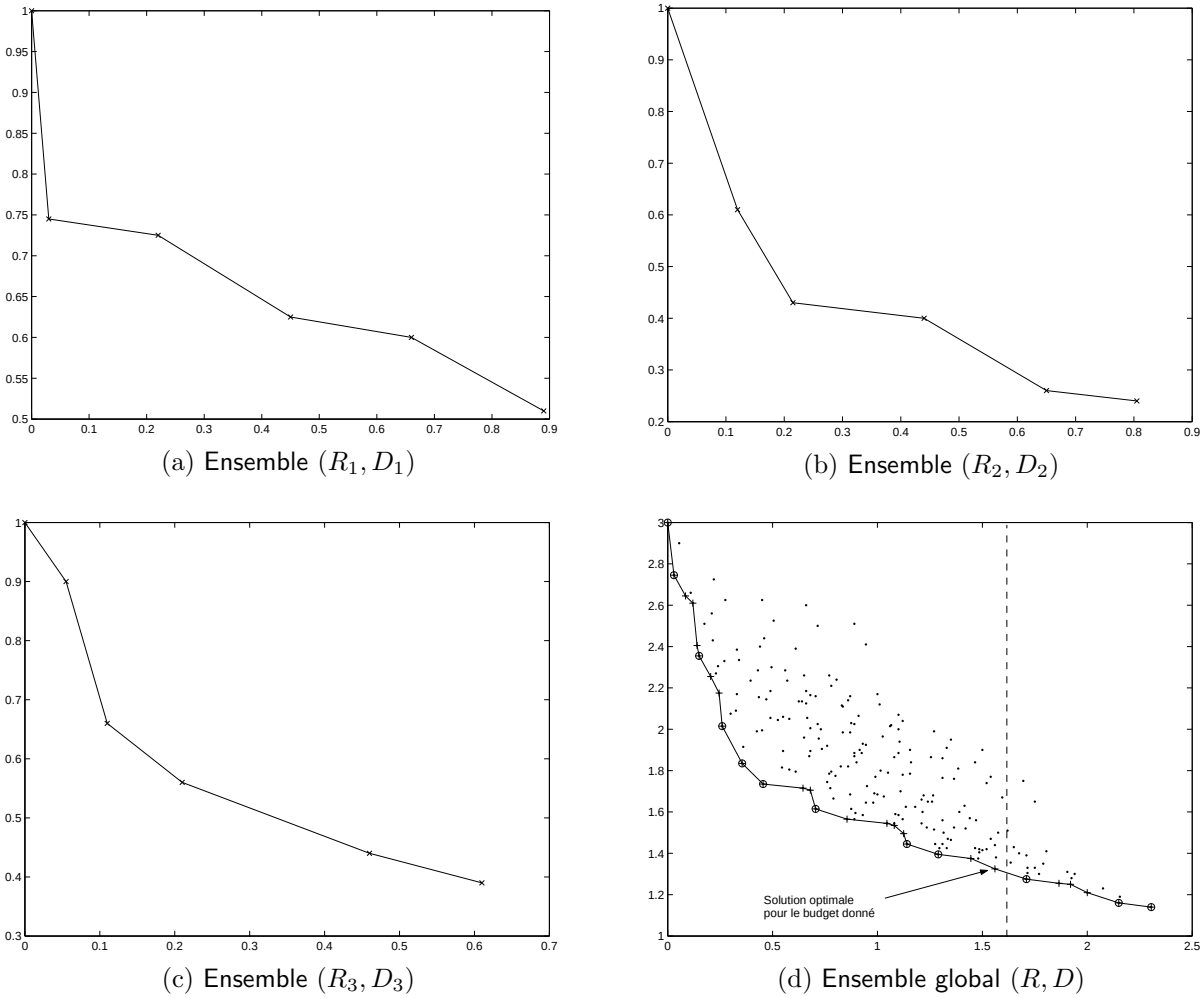
Le problème d'optimisation taux-distorsion global que nous appellerons par la suite *problème d'allocation de ressources* sous la contrainte d'un budget donné peut être ainsi décrit par la définition :

Définition 1.2.3
Le problème d'allocation de ressources :
Minimiser la distorsion D de telle sorte que le taux R soit inférieur ou égal à un taux donné appelé budget R_b *:*

$$\min_{R \leqslant R_b} D$$

Le problème énoncé peut effectivement être interprété comme un problème d'allocation de ressources. Il s'agit de distribuer une ressource limitée (par un budget) selon différentes composantes de telle sorte que la performance qui en découle soit maximale, c'est à dire, que le prix qu'il en coûte (la distorsion) soit minimal. Exprimé différemment, c'est trouver la meilleure combinaison d'indices $\underline{k} = (k_1, \ldots, k_N)$ telle que la fonction $D(\underline{k})$ soit minimisée sous la contrainte $R(\underline{k}) \leqslant R_b$. Notons qu'en cas d'égalité sur les distorsions globales D, nous considé-

rerons par convention la solution (R, D) au problème d'allocation de ressources comme celle qui minimise le taux global R.



(a) Ensemble (R_1, D_1) (b) Ensemble (R_2, D_2)

(c) Ensemble (R_3, D_3) (d) Ensemble global (R, D)

Fig. 1.1: Exemple d'un ensemble (R, D) avec $N = 3$ composantes (a), (b) et (c) de $M_i = 6$ points chacune.

Le groupe de figures 1.1 illustre l'exemple d'un nuage de points (R, D) issu de $N = 3$ composantes de $M_i = 6$ points chacune. Examinons quelques instants la figure 1.1 (d). Les solutions au problème d'allocation de ressources pour un budget R_b glissant sont décrites dans le plan global R-D en trait plein, chaque solution étant matérialisée par un $'+'$. Nous pouvons en particulier y voir la solution optimale pour le budget représenté en ligne pointillé verticale : il s'agit du premier point optimal à la gauche de la ligne verticale de budget. D'autre part, nous comptons un total de 28 points distincts solutions du problème pour un budget donné, sur un nuage global de $\prod_i M_i = 6^3 = 216$ points. Nous notons aussi que la courbe décrivant les solutions est décroissante ainsi que non convexe. Nous établirons mathématiquement dans un prochain paragraphe les caractéristiques spécifiques de cette courbe.

Nous remarquerons qu'on peut toujours trouver une solution au problème d'allocation de

ressources sous la contrainte d'un budget dès lors que tous les M_i sont supposés finis. On est alors confronté à un problème combinatoire discret qu'une étude exhaustive des $\prod_{i=1}^{N} M_i$ allocations disponibles permettra de résoudre. On déconseillera toutefois ce procédé quelque peu lourd ! Si tous les M_i sont infinis, et que les composantes (R_i, D_i) sont sous la forme de fonctions $D_i(R_i)$ continues, on peut aussi trouver une solution.

Pour simplifier et en première approche, nous supposerons qu'il n'existe pas d'*alignements* non verticaux de trois points ou plus, que ce soit dans les plans R_i-D_i ou dans le plan global R-D. Il est toutefois évident que tout alignement de points dans l'un des plans R_i-D_i implique un alignement similaire dans le plan global R-D. Nous pouvons donc nous restreindre à l'hypothèse générale qu'aucun alignement non vertical de trois points ne se produit dans le plan global R-D. Nous reviendrons plus amplement sur de telles considérations dans le paragraphe 1.5 afin de justifier ce choix.

1.3 Problèmes duaux

Parallèlement au problème d'allocation de ressources, il est possible de considérer le problème réciproque obtenu en inversant les rôles de R et de D. Ce problème se définit comme suit :

Définition 1.3.1
Le problème d'allocation de ressources réciproque :
Minimiser le taux R de telle sorte que la distorsion D soit inférieure ou égale à une distorsion donnée appelée budget D_b *:*

$$\min_{D \leqslant D_b} R$$

Évidemment, ce problème est résolu en échangeant R et D dans tout ce qui suit.

D'autre part, il est facile d'adapter toute cette présentation à la résolution d'un des problèmes suivants :

$$\begin{array}{ll} \min\limits_{R \leqslant R_b} D & \min\limits_{R \geqslant R_b} D \\ \max\limits_{R \leqslant R_b} D & \max\limits_{R \geqslant R_b} D \end{array}$$

au moyen des transformations suivantes :

$$R \leqslant R_b \Longleftrightarrow (-R) \geqslant (-R_b) \tag{1.5}$$

et

$$\max D \Longleftrightarrow -\min(-D) \tag{1.6}$$

Dorénavant, nous considérerons notre problème selon les conditions sous lesquelles nous l'avons initialement énoncé. Les problèmes duaux ou réciproques ne seront plus abordés.

1.4 Applications

Ce qui suit est une brève description des applications qui motivèrent initialement notre étude. Les deux contextes possibles sont l'optimisation taux-distorsion pour des applications en codage de source et l'optimisation capacité-coût pour des applications en codage de canal.

1.4.1 Codage de source par transformée

Un signal est décomposé en N composantes au moyen d'une transformée inversible ; Ces composantes sont ensuite quantifiées (scalairement ou vectoriellement) et peuvent éventuellement subir un codage entropique ; la transformée inversible est appliquée à la reconstruction (e.g. codage en sous-bandes, codage en ondelettes ou paquets d'ondelettes d'un signal 1D ou d'images 2D). Le taux global est le nombre moyen de bits codés par échantillons, et se trouve être toujours additif :

$$R = \sum_i \alpha_i r_i \tag{1.7}$$

où r_i est le nombre moyen de bits codés par échantillons de la $i$ème composante. Par conséquent, le facteur α_i est le nombre moyen d'échantillon dans une composante par échantillon du signal. Les valeurs des α_i peuvent être différentes, puisque la transformée peut introduire des taux d'échantillonnage différents sur chaque composante.

La distorsion globale (par échantillon) est additive, à condition que la transformée soit orthogonale :

$$D = \sum_i \alpha_i d_i \tag{1.8}$$

où d_i est l'erreur quadratique moyenne par échantillon dans une composante. Parfois, des pondérations supplémentaires w_i sont rajoutées afin d'introduire une disparité dans la contribution de chaque composante. Pour une classe donnée de quantificateurs et de codeurs il est souvent admis que d_i est une fonction particulière de r_i :

$$d_i = \sigma_i^2 \delta_i(r_i) \tag{1.9}$$

où σ_i^2 est la variance de la $i$ème composante ; dans ce modèle il est parfois même admis que les fonctions $\delta_i = \delta$ sont les même pour chaque composantes, e.g. $\delta(r_i) = c2^{-2r_i}$. Pour un codage optimal d'une source gaussienne sans mémoire, $c = 1$, pour une quantification scalaire optimale en haute résolution , $c = \pi\sqrt{3}/2$, et pour un codage entropique en haute résolution, $c = \pi e/6$.

Bien entendu, la description générale ci-dessus s'applique avec $R_i = \alpha_i r_i$ et $D_i = \alpha_i d_i$ ou $\alpha_i w_i d_i$. Chaque codeur possible pour la composante i est indexé par k_i et le problème R-D revient à trouver la combinaison de codeurs $\underline{k} = (k_1, \ldots, k_N)$ telle que la distorsion globale est minimisée pour un budget donné. Ce problème est largement connu sous le nom de problème d'optimisation d'allocation de bits.

Si chaque composante est décomposée en sous-composantes, nous obtenons un problème de taux-distorsion en deux dimensions où $R = \sum_{i,j} R_{i,j}$ et $D = \sum_{i,j} D_{i,j}$. Par un réarrangement adroit des indices, la description en une dimension s'applique aussi à ce cas.

1.4.2 Codage multi-canaux

L'optimisation d'un système de modulation multi-porteuses introduit un problème d'optimisation d'*allocation de puissance* où

$$P = \sum_{i=1}^{N} P_i \tag{1.10}$$

est la puissance moyenne totale (additive sur les sous-canaux i) et où

$$C = \sum_{i=1}^{N} C_i \tag{1.11}$$

est la capacité globale du canal à maximiser (c'est le nombre moyen de bits d'information par échantillon qui atteint une probabilité d'erreur négligeable). Dans ce cas, le problème d'allocation de ressources précédent doit être traduit en un problème capacité-puissance :

$$\max_{P \leqslant P_b} C \tag{1.12}$$

Pour des canaux gaussiens, la formule de la capacité de Shannon est souvent utilisée :

$$C_i = \frac{1}{2} \log_2(1 + \frac{P_i}{N_i}) \tag{1.13}$$

où N_i est la puissance de bruit moyenne sur chaque porteuse i.

Notre description précédente du problème d'allocation de ressources s'applique en prenant $C = -D$ et $P = R$, comme présenté pour les problèmes duaux. Remarquons que les C_i peuvent ne pas être bornées lorsque P_i croît, comme dans la formule de Shannon. C'est dans cette optique que notre description générale ne suppose pas les R_i ou les D_i non-négatifs ou minorés.

1.5 Définitions et propriétés fondamentales

Reprenons maintenant le problème d'allocation de ressources général et traitons de quelques propriétés fondamentales qui seront couramment employées par la suite.

1.5.1 Points optimaux

Définition 1.5.1
Nous appelons un point global (R^*, D^*) optimal *lorsque aucun autre point* (R, D) *n'est trivialement meilleur, c'est à dire, satisfaisant à* $R \leqslant R^*$ *et* $D \leqslant D^*$.

Un point optimal est évidemment une solution particulière au problème d'allocation de ressources (e.g, avec $R_b = R^*$). À l'inverse, un point qui n'est pas optimal ne peut être solution, car un autre point meilleur peut toujours être trouvé. Nous avons donc le théorème suivant :

Théorème 1.5.1
Un point (R, D) *est optimal si et seulement si il est solution du problème d'allocation de ressources pour un budget particulier.*

Nous pouvons donc aussi considérer le problème sous-jacent consistant à trouver l'ensemble des points optimaux du plan global R-D :

Définition 1.5.2
Le problème de la courbe optimale R-D ***:***
Trouver tous les points optimaux (R^*, D^*).

L'ensemble des points optimaux (R^*, D^*) sera dorénavant appelé *courbe optimale* R-D. La courbe optimale R-D est dessinée en solide dans la figure 1.1 (d). Puisque la définition de l'optimalité est symétrique suivant (R, D), la courbe duale D-R correspondant au problème d'allocation de ressources réciproque est identique à celle du problème R-D (voir annexe A pour preuve).

Par définition, deux points optimaux (R^*, D^*) et (R^{**}, D^{**}) ne peuvent avoir le même taux, et si $R^* < R^{**}$ alors $D^* > D^{**}$. Nous avons donc le lemme suivant :

Lemme 1.5.2
La courbe optimale R-D *est de la forme* $D(R)$ *où* D *est une fonction strictement décroissante de* R.

Nous adopterons la même définition de l'optimalité pour une composante i donnée :

Définition 1.5.3
Nous appelons un point (R_i^*, D_i^*) optimal *lorsque aucun autre point* (R_i, D_i) *ne satisfait à* $R_i \leqslant R_i^*$ *et* $D_i \leqslant D_i^*$.

Bien évidemment, si un des (R_i, D_i) n'est pas optimal alors $(R = \sum_i R_i, D = \sum_i D_i)$ ne peut non plus pas être optimal.

Théorème 1.5.3
Si $(R^*(\underline{k}), D^*(\underline{k}))$ *est optimal, alors chaque* $(R_i^*(k_i), D_i^*(k_i))$ *est optimal pour* $i = 1, \ldots, N$.

Cela a pour conséquence la possibilité de restreindre les ensembles de points étudiés dans les différentes composantes : seuls les points (R_i, D_i) optimaux seront conservés pour résoudre le problème d'allocation de ressources initial ou le problème de la courbe optimale R-D. Grâce au lemme 1.5.2 appliqué à chaque composante i, les (R_i, D_i) optimaux forment une courbe $D_i(R_i)$ strictement décroissante. Par conséquent, les (R_i, D_i) optimaux peuvent être trouvés au moyen de l'algorithme de recherche A.

Algorithme A Recherche directe de points optimaux (RDPO)

Cet algorithme sélectionne l'ensemble $\mathcal{O}_i$ des points optimaux d'une composante i.

1. *[Initialisation]*
   Ranger dans $\mathcal{P}$ les (R_i, D_i) selon les R_i croissants.
   Initialiser $\mathcal{O}_i$ avec le premier point de $\mathcal{P}$.
2. *[Boucle]*
   Sélectionner dans $\mathcal{P}$ le point (R_i^*, D_i^*) dont la distorsion est strictement inférieure à celle du dernier point de $\mathcal{O}_i$.
3. *[Mise à jour]*
   Rajouter (R_i^*, D_i^*) dans $\mathcal{O}_i$.
   Retirer de $\mathcal{P}$ le point (R_i^*, D_i^*) ainsi que tout les points le précédant.
4. Retourner $\mathcal{O}_i$ si $\mathcal{P}$ est vide.
   Revenir à l'étape 2 sinon.

Dorénavant, nous supposerons que l'algorithme RDPO est préalablement appliqué à chaque composante afin de ne garder que les (R_i, D_i) optimaux. On assimilera alors les (R_i^*, D_i^*) aux (R_i, D_i), et on n'utilisera plus la notion (devenue implicite) d'optimalité pour une composante i. Ainsi, les fonctions $D_i(R_i)$ sont strictement décroissantes, et en supposant les taux R_i indexés par entiers croissants k_i, on remarque que $R_i(k_i)$ et $D_i(k_i)$ sont des fonctions respectivement strictement croissantes et décroissantes de k_i.

En particulier, l'*origine* $(R(\underline{1}), D(\underline{1}))$ défini par un vecteur d'allocation $\underline{1} = [1, \ldots, 1]$ est toujours optimal, avec un taux minimal. Réciproquement, le point d'allocation $\underline{M} = [M_1, \ldots, M_N]$ est aussi toujours optimal, avec une distorsion minimale.

1.5.2 Points d'enveloppe

Définition 1.5.4
Un point global (R^e, D^e) *est appelé* point d'enveloppe *lorsque qu'il appartient à une droite de pente négative* $-\lambda$ *au dessous de laquelle aucun autre point* (R, D) *ne peut être trouvé.*

Mathématiquement, cela signifie qu'il existe $\lambda \in [0, \infty]$ tel que

$$D^e + \lambda R^e \leqslant D + \lambda R \tag{1.14}$$

pour tout point global (R, D).

Il est alors facile de voir qu'un point d'enveloppe est nécessairement optimal. Les points d'enveloppe apparaissent comme des cercles dans la figure 1.1 (d).

L'ensemble des multiplicateurs $\lambda \in [0, \infty[$ qui satisfont la définition d'un point d'enveloppe (R^e, D^e) donné est caractérisé par l'inégalité (1.14) pour tout (R, D). On peut aisément l'écrire :

$$\max_{\substack{R>R^e \\ D<D^e}} \frac{D^e - D}{R - R^e} \leqslant \lambda \leqslant \min_{\substack{R<R^e \\ D>D^e}} \frac{D - D^e}{R^e - R} \tag{1.15}$$

avec comme convention pour l'ensemble vide : $\max(\oslash) = 0$ et $\min(\oslash) = \infty$

Cette condition définit toujours un intervalle fermé $\subset [0, \infty]$. Puisque chaque λ correspond à au moins un point d'enveloppe (défini comme celui qui minimise $D + \lambda R$), ces intervalles couvrent l'ensemble $[0, \infty]$. Si λ définit deux points d'enveloppe distincts (R^e, D^e) et $(R^{e'}, D^{e'})$, alors $D^e + \lambda R^e = D^{e'} + \lambda R^{e'}$. Cela signifie que :

$$\lambda = \frac{D^e - D^{e'}}{R^{e'} - R^e} \tag{1.16}$$

se situe nécessairement sur les bords des intervalles (1.15). Par conséquent, ces intervalles forment une partition de l'ensemble $[0, \infty]$: il existe une suite de valeurs *critiques* de λ (avec indice à l'exposant)

$$\lambda^0 = \infty > \lambda^1 > \lambda^2 > \cdots > \lambda^n = 0 \tag{1.17}$$

telle que chaque intervalle (1.15) est de la forme $[\lambda^k, \lambda^{k-1}]$ pour un $0 < k \leqslant n$ particulier. Étant donné notre hypothèse de non-alignement de 3 points, les λ^k sont tous distincts, et chaque point d'enveloppe correspond à un unique intervalle $[\lambda^k, \lambda^{k-1}]$. Notons que n est le nombre total (éventuellement infini) de points d'enveloppe. Inversement, chaque λ non critique correspond à un unique point d'enveloppe, et chaque $\lambda^k \in (0, \infty)$ critique est de la forme (1.16)

pour deux points d'enveloppe successifs (R^e, D^e) et $(R^{e'}, D^{e'})$. Les pentes correspondantes $-\lambda^k$ forment une séquence de valeurs négatives strictement croissante. Ainsi :

Lemme 1.5.4
La courbe optimale R-D constituée de tous les points d'enveloppe est de la forme $D(R)$ où D est une fonction de R strictement convexe.

On dit que les points d'enveloppe résident sur l'*enveloppe convexe* de l'ensemble $\{(R, D)\}$.

La valeur $\mathscr{L} = D + \lambda R$ est communément nommée *lagrangien* associé au point (R, D) pour le multiplicateur λ. Par définition, un point d'enveloppe minimise le lagrangien. Pour une composante i, on écrit de même $\mathscr{L}_i = D_i + \lambda R_i$. Ici, λ joue le role d'un multiplicateur de Lagrange. Pour un λ donné, on a :

$$\mathscr{L}(\underline{k}) = \sum_{i=1}^{N} \mathscr{L}_i(k_i) \tag{1.18}$$

ce qui montre bien que $\mathscr{L}$ est minimal si et seulement si $\mathscr{L}_i$ est minimal pour chaque i. Par conséquent, nous avons le théorème suivant :

Théorème 1.5.5
$(R^e(\underline{k}), D^e(\underline{k}))$ est un point d'enveloppe si et seulement si $(R_i^e(k_i), D_i^e(k_i))$ sont points d'enveloppe pour chaque $i = 1, \ldots, N$ avec le même multiplicateur λ.

Ce théorème souligne l'intérêt du lagrangien qui transforme le problème d'optimisation avec contrainte en un problème sans contrainte, lui même ensuite décomposé en N sous-problèmes sans contrainte. Ce principe sert de base au développement d'algorithmes efficaces pour trouver des points d'enveloppe (et donc optimaux).

Le théorème 1.5.5 nous permet de toujours nous restreindre aux points d'enveloppe (R_i, D_i) quand on applique de tels algorithmes fondés sur cette méthode lagrangienne. Grâce aux lemmes 1.5.2 et 1.5.4 appliqués à chaque composante i, ces points d'enveloppe forment une courbe convexe strictement décroissante $D_i(R_i)$. Ceux-ci peuvent être trouvés par une recherche directe sous la forme de l'algorithme B.

En effet, c'est de cette valeur de λ_i que découle la borne inférieure écrite dans (1.15) pour la composante i. Cet algorithme trouve donc aussi la suite de multiplicateurs critiques :

$$\infty > \lambda_i^1 > \lambda_i^2 > \cdots > \lambda_i^{n_i - 1} > 0 \tag{1.19}$$

pour la composante i, où n_i est le nombre de points d'enveloppe (R_i^e, D_i^e).

Algorithme B Recherche directe de points d'enveloppe (RDPE)

Cet algorithme sélectionne l'ensemble $\mathcal{H}_i$ des points d'enveloppe d'une composante i.

1. *[Initialisation]*
   Ranger dans $\mathcal{P}$ les (R_i, D_i) selon les R_i croissants.
   Initialiser $\mathcal{H}_i$ avec le premier point de $\mathcal{P}$.
2. *[Boucle]*
   Sélectionner dans $\mathcal{P}$ le point (R_i^e, D_i^e) tel que

$$\frac{D_i^{e'} - D_i^e}{R_i^e - R_i^{e'}}$$

   est maximal, où $(R_i^{e'}, D_i^{e'})$ est le dernier point de $\mathcal{H}_i$.
3. *[Mise à jour]*
   Rajouter (R_i^e, D_i^e) dans $\mathcal{H}_i$.
   Retirer de $\mathcal{P}$ le point (R_i^e, D_i^e) ainsi que tout les points le précédant.
4. Retourner $\mathcal{H}_i$ si $\mathcal{P}$ est vide.
   Revenir à l'étape 2 sinon.

Maintenant, soit λ^k le multiplicateur critique appartenant à la séquence (1.17). λ^k correspond donc à deux points d'enveloppe successifs et distincts (R^e, D^e) et $(R^{e'}, D^{e'})$. Ces deux points satisfont l'équation (1.16). Or, par le théorème 1.5.5, tous les points (R_i^e, D_i^e) et $(R_i^{e'}, D_i^{e'})$ dans chaque composante sont des points d'enveloppe pour le même λ^k. Pour une composante i, si λ^k n'appartient pas à la séquence des multiplicateurs critiques $\{\lambda_i^p\}$ décrite par (1.19), alors le point d'enveloppe correspondant est forcément unique : $(R_i^e, D_i^e) = (R_i^{e'}, D_i^{e'})$. Sinon, on a $\lambda^k = \lambda_i^p$ et, dans ce cas, (R_i^e, D_i^e) et $(R_i^{e'}, D_i^{e'})$ sont des points d'enveloppe successifs dans le plan R_i-D_i. Le dernière situation arrive au moins une fois, puisque (R^e, D^e) et $(R^{e'}, D^{e'})$ sont distincts, et au plus une fois, puisque nous avons supposé qu'aucun alignement de trois points n'était possible. En conclusion, nous avons le résultat fondamental suivant :

Théorème 1.5.6
Chaque multiplicateur critique dans (1.17) *appartient à une unique séquence critique* (1.19). *Donc* (1.17) *est la réunion des ensembles de séquences* (1.19). *De plus, deux points d'enveloppe successifs* (R^e, D^e) *et* $(R^{e'}, D^{e'})$ *ont des allocations équivalentes à l'exception d'une unique composante.*

Remarquons que nous n'avons pas seulement :

$$\lambda^k = \frac{D^e - D^{e'}}{R^{e'} - R^e} = \frac{D_i^e - D_i^{e'}}{R_i^{e'} - R_i^e} = \lambda_i^l \tag{1.20}$$

mais aussi :

$$D^e - D^{e'} = D_i^e - D_i^{e'} \qquad R^{e'} - R^e = R_i^{e'} - R_i^e \tag{1.21}$$

Notons en particulier que les allocations $\underline{1}$ et $\underline{M}$ sont toutes les deux des allocations de points d'enveloppe. On peut aussi déduire aisément du théorème 1.5.6 le nombre total de points d'enveloppe dans le plan global R-D. Si on a n_i^e points d'enveloppe dans la composante i, alors il y a $n_i^e - 1$ multiplicateurs critiques dans cette composante, d'où la liste exhaustive des multiplicateurs critiques compte $\sum_i (n_i^e - 1)$ valeurs distinctes. Au final, cela représente $\sum_i (n_i^e - 1) + 1$ points d'enveloppe dans le plan global R-D.

1.5.3 Considérations autour des alignements

Comme mentionné précédemment, nous supposerons qu'il n'existe pas d'alignements non verticaux de trois points ou plus, que ce soit dans le plan global R-D, ou dans une des composantes. Nous allons tenter ici de justifier cette hypothèse et d'en étudier les impacts majeurs sur la suite de nos travaux.

Tout d'abord, nous décomposerons les cas d'alignements en deux catégories :
– les alignements dans une même composante i,
– les alignements dans le plan global R-D.
L'alignement de 3 points dans une même composante i signifie qu'il existe (R_i^e, D_i^e), $(R_i^{e'}, D_i^{e'})$ et $(R_i^{e''}, D_i^{e''})$ vérifiant :

$$\frac{D_i^e - D_i^{e'}}{R_i^{e'} - R_i^e} = \frac{D_i^e - D_i^{e''}}{R_i^{e''} - R_i^e} = \frac{D_i^{e''} - D_i^{e'}}{R_i^{e'} - R_i^{e''}} = \lambda_{\text{align}} \tag{1.22}$$

où $0 \leqslant \lambda_{\text{align}} < \infty$.

Le multiplicateur critique λ_{align} caractériserait alors non pas deux, mais trois points d'enveloppe. Si ce type d'alignement est envisageable théoriquement, il reste de probabilité nulle de par l'origine des fonctions $R_i(D_i)$ considérées. Si l'on se retrouvait cependant confronté à un tel alignement, il suffirait d'introduire, sans ne rien perdre à la généralité de notre exposé, une déviation infinitésimale dans notre fonction $R_i(D_i)$ de sorte que les trois points soient caractérisés par deux valeurs de multiplicateurs critiques distinctes.

Maintenant, l'alignement de 3 points dans le plan global R-D (non issu d'un alignement dans une même composante) signifie qu'il existe (R_i^e, D_i^e) et $(R_i^{e'}, D_i^{e'})$, d'une part, et (R_j^e, D_j^e) et $(R_j^{e'}, D_j^{e'})$ d'autre part, vérifiant :

$$\frac{D_i^e - D_i^{e'}}{R_i^{e'} - R_i^e} = \frac{D_j^e - D_j^{e'}}{R_j^{e'} - R_j^e} = \lambda_{\text{align}} \tag{1.23}$$

où $0 \leqslant \lambda_{\text{align}} < \infty$.

Ce type d'égalité proviendrait de la contribution équivalente de deux composantes distinctes, ce qui est encore une fois théoriquement imaginable, mais de probabilité nulle de par l'origine du problème. Il est ici aussi possible de modifier infinitésimalement et localement l'une des deux composantes afin d'éliminer une telle singularité.

En conclusion, de tels alignements, qu'on les considère comme des événements possibles ou pas, ne sont pas une fatalité et seront simplement considérés comme inexistants dans toute la suite de cet exposé, et ce afin ne pas surcharger inutilement les descriptions des algorithmes. Nous remarquerons aussi que nous n'avons pas supposé l'absence d'alignements verticaux. Ces alignements sont effectivement immédiatement retirés des composantes (R_i, D_i) par l'algorithme RDPO initialement appliqué à chacune d'elle. Si il s'agit d'alignements verticaux dans le plan global R-D, c'est à dire de points de taux équivalents, ce qui est tout à fait possible lorsque les composantes $\{(R_i, D_i)\}$ sont à taux R_i entiers, la définition du problème nous permet de ne pas les considérer. En effet, le point de distorsion la plus forte sera vu comme non-optimal, et ne sera pas la source de cas particuliers pour les algorithmes ultérieurement décrits.

1.5.4 L'opération élémentaire $\vec{\Delta}$

Nous avons vu qu'en raison du théorème 1.5.6 chaque multiplicateur critique s'exprime sous la forme de l'équation (1.20). On peut encore dire que les multiplicateurs critiques s'écrivent de deux manières équivalentes suivant qu'on les considère dans le plan global R-D ou dans leur plan R_i-D_i d'origine. On a ainsi l'expression suivante dans un plan R_i-D_i :

$$\lambda_i(k_i) = \frac{D_i(k_i) - D_i(k_i+1)}{R_i(k_i+1) - R_i(k_i)} = \frac{\Delta D}{\Delta R} \tag{1.24}$$

où $\Delta R = R_i(k_i+1) - R_i(k_i)$ et $\Delta D = D_i(k_i) - D_i(k_i+1)$ sont des quantités positives.

Et puisque ce multiplicateur critique $\lambda_i(k_i)$ correspond à deux points d'enveloppe consécutifs $(R(\underline{k}), D(\underline{k})) = (R, D)$ et $(R(\underline{k}'), D(\underline{k}')) = (R', D')$, on a l'expression suivante dans le plan global R-D :

$$R' = R + \Delta R \qquad D' = D - \Delta D \tag{1.25}$$

En reprenant la notation par indexation, cela revient à l'opération suivante :

$$\underline{k} = (k_1, \ldots, k_i, \ldots, k_N) \longmapsto \underline{k}' = (k_1, \ldots, k_i + 1, \ldots, k_N) \tag{1.26}$$

qui incrémente k_i mais conserve les autres composantes inchangées.

On définit ainsi une opération élémentaire $\vec{\Delta}$ comme suit :

Définition 1.5.5
On appelle opération élémentaire $\vec{\Delta}$ toute opération qui transforme :
– *un point $(R_i(k_i), D_i(k_i))$ en un point $(R_i(k_i+1), D_i(k_i+1))$*
– *une allocation k_i en k_i+1*
– *une allocation $\underline{k} = (k_1, \ldots, k_i, \ldots, k_N)$ en $\underline{k}' = (k_1, \ldots, k_i+1, \ldots, k_N)$*
– *un point $(R(\underline{k}), D(\underline{k}))$ en un point $(R(\underline{k}'), D(\underline{k}'))$*
– *un point (R, D) en $(R+\Delta R, D-\Delta D)$*

Cette opération $\vec{\Delta}$ est alors à la fois associable à :
– un couple unique (k_i, k_i+1)
– un ensemble de couples $(\underline{k}, \underline{k}')$
– un couple unique $(\underline{k}^e, \underline{k}^{e'})$
– un couple unique $(\Delta R, \Delta D)$
De plus, dans certains cas (de convexité), l'opération $\vec{\Delta}$ sera aussi associable à un unique multiplicateur critique λ :

$$\lambda = \frac{\Delta D}{\Delta R} = \frac{D(\underline{k}) - D(\underline{k}')}{R(\underline{k}') - R(\underline{k})} = \frac{D_i(k_i) - D_i(k_i+1)}{R_i(k_i+1) - R_i(k_i)}$$

On écrira de manière équivalente : $\vec{\Delta} \equiv \vec{\Delta}_{\underline{k}} \equiv \vec{\Delta}_{k_i} \equiv \vec{\Delta}_{\lambda}$. Cette opération élémentaire s'applique aussi bien dans un des plans R_i-D_i que dans le plan global R-D.

1.5.5 Points cachés

Nous avons vu qu'un point d'enveloppe était toujours un point optimal. Cependant le contraire n'est pas vrai : un point optimal n'est pas forcément un point d'enveloppe. Nous verrons même par la suite que les points d'enveloppe représentent une minorité des points optimaux.

Définition 1.5.6
Nous appellerons un point (R^h, D^h) caché *tout point optimal qui n'est pas un point d'enveloppe.*

Lorsque l'on applique des algorithmes basés sur une méthode lagrangienne pour résoudre le problème de la courbe optimale R-D, on obtient un résultat sous-optimal : les points cachés sont inaccessibles pour ces méthodes. C'est d'ailleurs pour cette raison que nous les nommons "points cachés" (sous entendu cachés aux méthodes lagrangiennes).

En combinant les lemmes 1.5.2 et 1.5.4, il est facile de voir que les points cachés (R^h, D^h) se situent entre deux points d'enveloppe successifs (R^e, D^e) et $(R^{e'}, D^{e'})$, où $R^e < R^h < R^{e'}$ et

$D^e > D^h > D^{e'}$. De plus, pour le multiplicateur critique (1.16), le lagrangien correspondant satisfait l'inégalité $\mathscr{L}^e = \mathscr{L}^{e'} < \mathscr{L}^h$. Nous avons donc le lemme suivant :

Lemme 1.5.7
Un point caché, point optimal de la courbe optimale R-D situé entre les points d'enveloppe (R^e, D^e) et $(R^{e'}, D^{e'})$,se trouve dans un triangle défini par les inégalités suivantes :

$$D < D^e \qquad R < R^{e'} \qquad \mathscr{L} = D + \lambda R > \mathscr{L}^e = \mathscr{L}^{e'} \tag{1.27}$$

où $\lambda = \frac{D^e - D^{e'}}{R^{e'} - R^e}$

D'où la définition suivante :

Définition 1.5.7
Nous appellerons triangle caché ∇^h tout triangle du plan global R-D situé entre deux points d'enveloppe successifs tel que décrit au lemme 1.5.7 (voir Fig. 1.2).

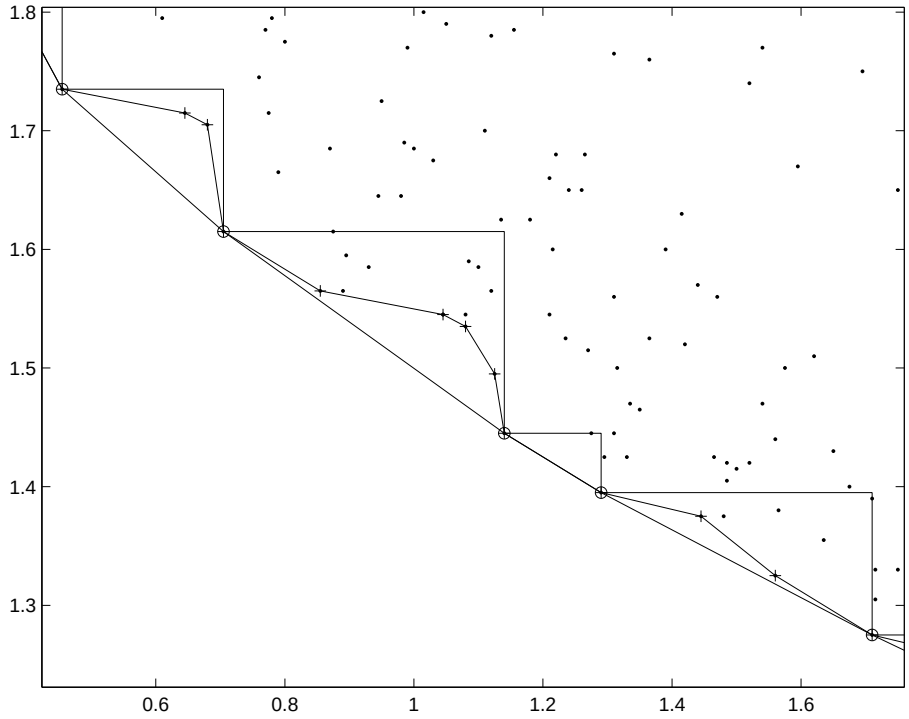


Fig. 1.2: Zoom de la figure 1.1 (d) montrant des triangles cachés entre des points d'enveloppe successifs (cerclés).

Comme le montre la figure 1.2, un point caché ne peut résider que dans un triangle caché ; hors triangles cachés, aucun point optimal n'existe. Cependant, il n'y aucun moyen de prévoir le nombre de points cachés dans un triangle donné. Un tel triangle peut aussi bien ne contenir aucun point caché ou en comporter un nombre important. D'autre part, un tel triangle peut aussi contenir des points non-optimaux.

D'autre part, lorsque nous étudions le problème d'allocation de ressource sous la contrainte d'un budget R_b, la zone où se situent les points cachés potentiels, dont fait partie la solution du problème, est un triangle ∇_b^h de définition :

Définition 1.5.8
Nous appellerons ∇_b^h, triangle caché associé au budget R_b, le triangle du plan global R-D défini par :

$$D < D^e \qquad R < R_b \qquad \mathscr{L} = D + \lambda R > \mathscr{L}^e = \mathscr{L}^{e'} \tag{1.28}$$

où les points (R^e, D^e) et $(R^{e'}, D^{e'})$ sont les deux points d'enveloppe consécutifs dont les taux vérifient $R^e < R_b \leqslant R^{e'}$.

Pour un budget donné, le triangle ∇_b^h est contenu dans un des triangles ∇^h (voir Fig. 1.3).

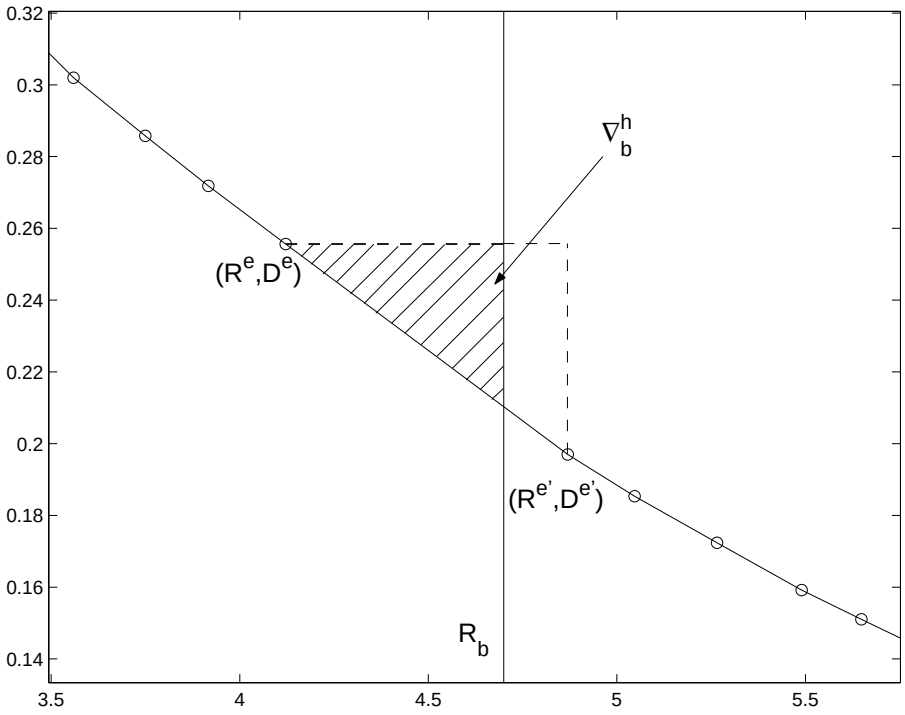


Fig. 1.3: Exemple de triangle ∇_b^h.

De la définition découle directement le lemme :

Lemme 1.5.8
Un point caché, solution au problème d'allocation de ressource pour le budget R_b, se trouve dans le triangle ∇_b^h.

Le théorème 1.5.5 nous amène qu'un point caché (R^h, D^h) peut être obtenu lorsque un des (R_i^h, D_i^h) n'est lui-même pas un point d'enveloppe pour le même multiplicateur λ. Cependant, alors que tous les (R_i^h, D_i^h) sont des points d'enveloppe pour différentes valeurs de λ, le point global (R^h, D^h) peut très bien ne pas être un point d'enveloppe tout en étant opti-

mal. En d'autres termes, se restreindre à des $D_i(R_i)$ convexes n'implique pas forcément une convexité de la courbe optimale R-D ; des points cachés peuvent apparaître (voir les exemples paragraphe 1.6).

Il y a malgré tout un cas particulier important pour lequel des $D_i(R_i)$ convexes amènent un $D(R)$ convexe. Ce cas est traité par le théorème suivant :

Théorème 1.5.9
Supposons tous les $D_i(R_i)$ convexes, et les R_i uniformément espacés (avec le même écart entre chaque point dans les différentes composantes). Alors $D(R)$ est convexe.
En d'autres termes, si aucun point caché n'existe dans des composantes à taux R_i régulièrement espacés, alors aucun point caché global ne peut apparaître. Dans ce cas, tous les points optimaux sont sur l'enveloppe convexe, et les méthodes de résolution lagrangiennes sont optimales.

À notre connaissance, ce cas n'a jamais été explicitement décrit dans la littérature, bien qu'il ait été employé dans les simulations de [SG88].

Preuve : Nous pouvons toujours nous limiter à des taux à valeurs entières $R_i = k_i$ en utilisant un facteur d'échelle adéquat. Les taux globaux R sont alors des entiers.

Par le théorème 1.5.6 deux points d'enveloppe successifs sont tels que $R^{e'} - R^e = R_i^{e'} - R_i^e$ où R_i^e et $R_i^{e'}$ correspondent à deux points successifs dans le plan R_i-D_i. Par hypothèse du théorème, $R_i^{e'} - R_i^e = 1$ et donc : $R^{e'} - R^e = 1$.

Maintenant un point caché quelconque (R^h, D^h) se trouverait entre deux points d'enveloppe successifs (lemme 1.5.7), et en particulier $R^e < R^h < R^{e'}$. Puisque les taux sont entiers, cela contredit notre égalité $R^{e'} - R^e = 1$. ■

1.6 Corpus de test

Pour illustrer, valider et comparer nos algorithmes, nous avons besoin d'effectuer des simulations sur des fonctions représentatives des différents type de problèmes rencontrés en allocation de ressources. Nous avons donc créé plusieurs ensembles de fonctions de simulation $D_i(R_i)$ en essayant de couvrir la plupart des situations.

Pour ce faire, nous avons mis en place des ensembles $D_i(R_i)$ comprenant $M_i = 25$ points par composantes dans l'intervalle $0 \leqslant R_i \leqslant 6$. Nous avons créé $N = 8$ composantes de formes différentes. La création s'est faite à partir d'un contexte de codage de source où différents types de quantificateurs se sont vus appliqués à une source gaussienne sans mémoire de variance $\sigma^2 = 1$. En choisissant différents coefficients σ_i^2 qui représentent la variance de chaque composante i, on obtient un nombre virtuellement infini d'ensembles possibles $(R_i, \sigma_i^2 D_i)$.

Les taux uniformément espacés (e.g. entiers) ont été obtenus par des quantificateurs scalaires de type Lloyd-Max et un ensemble de quantificateurs vectoriels optimisés de dimension 2, 4 ou 8. Nous avons obtenu des taux non-uniformément espacés en utilisant des codeurs entropiques. Notons qu'après selection des points d'enveloppe (par l'algorithme RDPE) d'un ensemble de points régulièrement espacés, mais non convexe, $D_i(R_i)$, nous obtenons un ensemble convexe à taux non régulièrement espacés. Dans ce cas, nous parlons de taux "uniformément espacé à trous".

En résumé, nous avons classé nos ensembles de fonctions de simulations selon le caractère convexe (ou pas) des fonctions $D_i(R_i)$, et la régularité des taux R_i (à savoir : taux uniformément espacés avec ou sans "trous" ou taux non-uniformément espacés). Nous considérerons donc les cinq types de fonctions :
– Convexe à taux Uniformément répartis (CU),
– Convexe à taux Uniformément répartis avec trous (CUT),
– Non-Convexe à taux Uniformément répartis (NCU),
– Convexe à taux Non-Uniformément répartis (CNU),
– Non-Convexe à taux Non-Uniformément répartis (NCU).

Des exemples sont illustrés aux figures B.1–B.8, regroupées en annexe B.

Nous observons que pour B.4 et B.7 les courbes optimales R-D sont non-convexes même si tous les $D_i(R_i)$ sont convexes. Cela est dû à la répartition non uniforme des taux R_i. On peut ainsi remarquer un nombre important de points optimaux n'appartenant pas à l'enveloppe convexe : ce sont les points cachés.

1.7 Critères de comparaison

Afin de mettre en regard les performances des différents algorithmes que nous introduirons dans les chapitres suivants, nous nous devons de définir des critères appropriés. Pour certains, ils consistent en des critères théoriques classiques (comme la complexité d'un algorithme) ou de mesures expérimentales plus pratiques (comme le temps de calcul moyen par point). Pour d'autres, il s'agit de critères plus spécifiques définis pour les besoins de notre étude.

1.7.1 Critères théoriques

Complexité

La complexité d'un algorithme est l'un des critères essentiels de comparaison. Elle permet de déterminer la viabilité de l'implémentation d'un tel algorithme. Cependant la complexité théorique d'un algorithme n'est pas toujours évaluable. Nous essaierons donc de fournir une

approximation de la complexité des algorithmes pour lesquels le calcul théorique est possible. Ces calculs sont regroupés dans l'annexe C.

Coût mémoire

Le coût mémoire d'un algorithme est un autre critère de comparaison. Celui-ci permet de mesurer et contrôler l'empreinte mémoire nécessaire à l'algorithme. Cependant, à l'instar de la complexité, le coût mémoire théorique d'un algorithme n'est pas toujours facile à exprimer. Nous essaierons de fournir une évaluation théorique du coût mémoire des algorithmes pour lesquels cela est possible. Ces calculs sont aussi regroupés dans l'annexe C.

1.7.2 Critères expérimentaux

Les critères expérimentaux sont déterminés par l'intermédiaire de simulations. Celles-ci ont consisté en un nombre important d'itérations des différents algorithmes sur des ensembles et des budgets aléatoires. A l'issue de ces simulations, nous avons calculé les valeurs moyennes des différents critères décrits ci-dessous.

À titre indicatif, la totalité des simulations ont été réalisées par programmation sous Matlab sur une machine équipée d'un processeur Pentium 4 cadencé à 3GHz et de 1Go de mémoire RAM. Les calculs sous Matlab s'effectuent sur des mots *double*, codés sur 64 bits.

Temps de calcul moyen

Le temps de calcul moyen correspond à la résolution du problème d'allocation de ressource pour un budget unique donné. Les simulations réalisées ont été "chronométrées" au moyen d'une fonction s'appuyant sur l'horloge interne du CPU. Le temps de processeur alloué aux différents algorithmes a pu ainsi être récupéré et nous avons fait une moyenne sur l'ensemble des itérations afin d'obtenir un temps moyen d'exécution. Les temps fournis dans ce manuscrit seront exprimés en secondes.

Pourcentage de points optimaux

Si certaines procédures sont dites optimales, d'autres ne retournent que des solutions sous-optimales, à cause entre autre de l'existence de points cachés. Un de nos critères consiste donc à calculer pour chaque algorithme le pourcentage points qui sont des points optimaux parmi les solutions proposées. Seul un algorithme qui trouve 100% de points optimaux parmi ses solutions est décrété optimal.

Perte en distorsion moyenne

Le pourcentage de points optimaux est un critère de comparaison utile et nécessaire, mais ne permet pas de quantifier les performances relatives entre algorithmes sous-optimaux. Nous allons donc nous intéresser à un critère plus complexe permettant de mesurer cette notion de performance relative. En effet, certaines méthodes, bien que sous optimales, atteignent à moindre coût des résultats très proches des procédures optimales. La perte en distorsion va permettre d'évaluer cet écart de performance entre algorithmes, qu'ils soient optimaux ou sous-optimaux. Nous devons, pour décrire plus précisément ce critère, préalablement définir une valeur que nous nommerons D_{virt}.

Soit R_b le budget à allouer, et considérons les deux points d'enveloppe consécutifs (R^e, D^e) et $(R^{e'}, D^{e'})$, où $R^e \leqslant R^b < R^{e'}$.

Définition 1.7.1
On définit la distorsion D_{virt} comme la distorsion d'un point virtuel de taux R_b se situant sur le segment reliant (R^e, D^e) et $(R^{e'}, D^{e'})$:

$$D_{\text{virt}} = (1-\alpha)D^e + \alpha D^{e'} \tag{1.29}$$

où $\alpha = \frac{R_b - R^{e'}}{R^e - R^{e'}}$.
On peut voir un exemple à la figure 1.4.

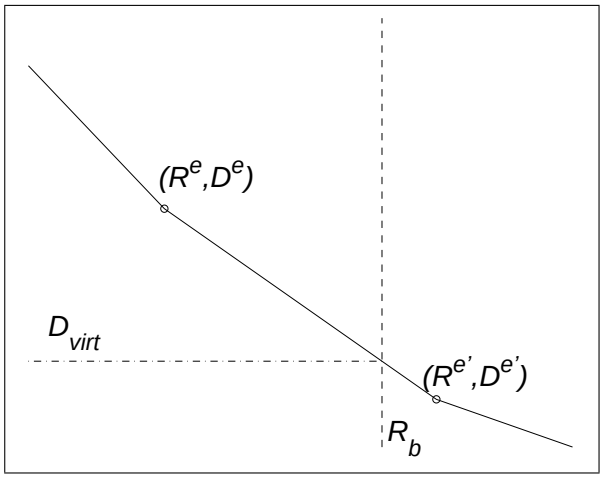


FIG. 1.4: Représentation d'une portion d'enveloppe convexe avec mise en évidence de D_{virt}

La distorsion D_{virt} représente la limite virtuelle accessible. En effet, quelles que soient les performances d'un algorithme, la solution $(R_{\text{sol}}, D_{\text{sol}})$ qu'il proposera sera minorée en distorsion par ce point : $D_{\text{sol}} \geqslant D_{\text{virt}}$. Par extension, la solution optimale $(R_{\text{opt}}, D_{\text{opt}})$ au problème d'allocation sous la contrainte du budget R_b vérifiera de même $D_{\text{opt}} \geqslant D_{\text{virt}}$, avec égalité si et seulement si $R_b = R^e$.

Définition 1.7.2
On définit la perte en distorsion $\mathcal{P}_D$ d'une solution $(R_{\mathrm{sol}}, D_{\mathrm{sol}})$ pour le budget R_b comme la valeur :

$$\mathcal{P}_D = 10.log_{10}\left(\frac{D_{\mathrm{sol}} - D_{\mathrm{virt}}}{D_{\mathrm{opt}} - D_{\mathrm{virt}}}\right) \tag{1.30}$$

Si le budget correspond au taux d'un point d'enveloppe donné, on a $R_b = R^e$ et $D_{\mathrm{opt}} = D_{\mathrm{virt}} = D^e$. Dans ce cas, nous prendrons par convention $\mathcal{P}_D = 0$ si $D_{\mathrm{sol}} = D^e$ et nous considérerons que $\mathcal{P}_D$ n'est pas défini pour les solutions de distortion $D_{\mathrm{sol}} > D^{\mathrm{SG}}$.

On calculera la moyenne $\bar{\mathcal{P}}_D$ de cette perte en distorsion sur l'ensemble des simulations. Remarquons que si l'algorithme est optimal, c'est à dire si $D_{\mathrm{sol}} = D_{\mathrm{opt}}$ pour tout budget, alors la perte en distortion moyenne vaut $\bar{\mathcal{P}}_D = 0$. De manière plus générale, étant donné les inégalités $D_{\mathrm{sol}} \geqslant D_{\mathrm{opt}} \geqslant D_{\mathrm{virt}}$, nous avons toujours $\bar{\mathcal{P}}_D \geqslant 0$.

1.8 Conclusions

Nous avons présenté le problème général d'allocation de ressources. Celui-ci, sous sa forme la plus générale, consiste en la minimisation d'une variable de "coût", additive, selon la répartition d'une variable de ressource, elle aussi additive, pour un budget donné. L'étude de ce problème général est motivée par l'existence de problèmes similaires dans le domaine du codage de source ou du codage de canal, mais ne se cantonnent pas à cet unique cadre. Ces problèmes, en partie déjà résolus, sont restés toutefois sans réponse optimale et efficace dans leur contexte le plus général.

Nous avons aussi énuméré les différents problèmes équivalents à notre problème original. Ainsi, en présentant les moyens de se ramener quoi qu'il arrive au problème initialement décrit, nous pouvons restreindre notre présentation à la seule description des méthodes de résolution de celui-ci.

Nous avons explicité les applications qui nous poussent à étudier ce problème général d'optimisation. Aussi bien issu du monde du codage de source que du codage de canal, ces applications réclament une résolution efficace du problème d'allocation de ressources.

Nous avons ensuite établi un certain nombre de définitions et de théorèmes préalables à nos travaux. Ceux-ci nous ont permis d'aborder des notions telles que l'optimalité, le lagrangien, l'enveloppe convexe et l'existence de points optimaux appelés points cachés. Deux algorithmes de recherche, nommés (RDPO) et (RDPE), ont été présentés afin de sélectionner efficacement des points d'intérêt. Ceux-ci seront couramment réutilisés par la suite au profit des différents algorithmes de résolution du problème d'allocation.

Nous avons mis en place un ensemble de courbes et de critères pertinents afin d'évaluer les performances des algorithmes étudiés ou proposés dans la suite de ce manuscrit. Ces critères

seront utilisés tout au long de notre exposé afin d'appuyer nos propos et de justifier notre démarche.

Désormais, nous sommes en mesure de présenter l'ensemble des travaux déjà effectués dans le domaine de l'allocation de ressources. Nous allons décrire de manière unifiée les différents algorithmes existants afin d'avoir une vue d'ensemble de l'état de l'art sur le thème de l'allocation.

Chapitre 2

L'état de l'art

L'art est long et le temps est court.
Charles Cros

2.1 Introduction

Le problème d'allocation de ressources maintenant décrit, nous allons nous atteler à l'étude des méthodes et procédures ayant été développées pour résoudre celui-ci. Nous allons voir en particulier que ce problème d'optimisation n'est pas récent, et a été traité sous différentes formes et selon des approches variées.

Nous examinerons tout d'abord au paragraphe 2.2 les principales approches concrètement employées pour étudier le problème d'allocation de ressource. Nous verrons alors les différents domaines d'étude privilégiés, dont font partie le codage de source et le codage de canal, et les deux grandes familles d'algorithmes recensées, issues d'approches continues ou lagrangiennes du problème.

Nous approfondirons alors au paragraphe 2.3 les méthodes consistant à trouver les points d'enveloppe du plan global. Ces algorithmes décrivent l'enveloppe convexe du nuage global et permettent de résoudre (mais pas forcément de manière optimale) le problème d'allocation de ressource contraint par un budget. Nous verrons alors que l'optimalité de telles méthodes dépend de la nature des fonctions considérées. Dans certains cas, bien que la solution soit un point optimal (puisque c'est un point d'enveloppe), elle ne correspondra pas à la solution optimale.

C'est dans cette optique que nous décrirons dans le paragraphe 2.4 les méthodes de recherche de points cachés. Ces points optimaux, rarement étudiés, et quasiment jamais atteints par

les algorithmes classiques, nécessitent des procédures particulières. Nous verrons le principe et l'intérêt de telles méthodes dont les performances dépendent largement de l'origine des fonctions étudiées.

Le paragraphe 2.5 nous permettra de récapituler les performances des algorithmes présentés dans ce chapitre. Nous confronterons ces performances dans le but d'établir de manière exhaustive les avantages et les inconvénients respectifs de chacune des méthodes. Nous y présenterons par ailleurs quelques résultats anticipés sur les algorithmes présentés aux chapitres 3 à 6.

Finalement, au paragraphe 2.6, nous résumerons en quelques conclusions les résultats présentés dans ce chapitre. Nous en tirerons un bilan global de cet état de l'art et nous ouvrirons les perspectives de nos algorithmes pour les chapitres suivants.

2.2 Approches envisagées

2.2.1 Approches suivant le domaine considéré

Approche taux-distorsion

Le problème d'allocation de ressources sous contrainte est, en codage de source, un problème classique depuis longtemps mis en évidence et étudié. Initialement, Huang et Schultheiss [HS63] l'abordèrent dans le cadre de leurs travaux prouvant l'optimalité de la transformée de Karhunen-Loeve, et depuis, le problème a été abondamment traité, sous différentes formes et hypothèses, dans des conditions diverses.

De par la définition que nous avons faite du problème d'allocation de ressources, inspirée du monde du codage de source, l'ensemble des éléments théoriques et des méthodes de cette thèse s'appliquent pratiquement de manière directe. Nous avons essayé de rendre l'écriture des algorithmes présentés la plus générale possible en gardant un contexte et des notations unifiés.

Approche taux-puissance

Les systèmes de modulation discrète multi-tons, *Discrete Multi-Tone modulation* (DMT) en anglais, furent les premiers à introduire cette notion d'allocation de ressources en codage de canal. C'est Hughes-Hartogs [Huga] qui, au travers d'un *patent*, mis le premier en œuvre ce genre de méthode. Plus généralement, les systèmes de modulation multi-porteuses sont aujourd'hui à l'origine d'une conséquente bibliographie sur le problème d'allouer une puissance (ou un taux) sous une contrainte de budget. Cela est essentiellement dû au succès du domaine

d'application : systèmes de communication xDSL et systèmes de transmission sans fil sont en développement constant et représentent des enjeux commerciaux considérables.

Il est important de noter que le *taux* a été très souvent considéré comme *entier* (on y fait référence comme des *bits*). Cela n'est pourtant pas une obligation. Cette hypothèse permet de grandement simplifier les méthodes de résolution du problème d'allocation de ressource mais elle n'est pas forcément exclusive. D'autre part, les fonctions considérées sont toujours implicitement supposées convexes. Si cette hypothèse paraît vraisemblable, elle ne suffit pas à elle seule à garantir l'optimalité de certaines méthodes.

Autres approches

Nous avons essentiellement étudié les problèmes issus des communications numériques. Il existe toutefois des articles plus généraux [Eve63, Fox66] ou provenant de domaines différents qui tentent de résoudre le même problème. On l'utilise ainsi pour modéliser des situations (quelquefois en tant que sous-problème) telles que :
- dans des systèmes d'aide à la gestion de portefeuille : on souhaite alors équilibrer sélectivité et diversification afin de trouver le meilleur rapport entre rendement et risque pour un capital placé sur plusieurs actifs financiers ;
- dans le chargement de bateau ou d'avion : tous les bagages à destination doivent être amenés, sans être en surcharge ;
- dans la découpe de matériaux : pour minimiser les chutes lors de la découpe de sections de longueurs diverses dans des barres en fer ;
- ...

Autant de problèmes plus ou moins originaux qui, de par leur modélisation, se ramènent à la description faite dans ce manuscrit du problème d'allocation de ressource.

Une autre raison de s'intéresser à ce type de problème d'allocation est son apparition dans certaines utilisations de méthodes de génération de colonnes (ainsi pour le problème de « bin packing »). Les variables ont alors une réalité toute autre, et les contraintes de fonctions sont parfois différentes. Nous citerons à titre d'exemple le problème bien connu du sac à dos à variables binaires qui est un cas particulier de notre étude pour laquelle les $M_i = 2$ (N quelconque) et où les fonctions employées sont de taux régulièrement espacés à trous.

Nous pourrons de toute manière toujours ramener ces problèmes à la définition initiale faite au premier chapitre du problème d'allocation de ressources.

2.2.2 Approches suivant la méthode considérée

Les différents algorithmes proposés pour résoudre le problème d'allocation de ressources peuvent être regroupés en trois catégories bien distinctes : les méthodes *continues*, les méthodes dites

lagrangiennes et les méthodes basées principalement sur des procédés de *branch and bound*. Les premières considèrent le problème discret comme continu et appliquent des théorèmes et algorithmes classiques d'analyse ([KT51] par exemple), puis reviennent aux contraintes de taux initiales en discrétisant la solution. Les méthodes lagrangiennes, introduisant un lagrangien et son multiplicateur associé, transforment le problème avec contrainte initial en un problème sans contrainte (en utilisant le résultat du théorème 1.5.5). Les dernières sont la mise en œuvre d'un procédé de *branch and bound* : par méthode itérative, on génère un ensemble de points qu'on vient ensuite décimer suivant un critère, les points restants servant de base à la génération suivante.

Approche continue

Le problème d'allocation de ressources en codage de source fut initialement traité dans la littérature pour des fonctions $D_i(R_i)$ *continues*, où R_i pouvait prendre n'importe quelle valeur réelle. Il existe des expressions littérales bien connues [HS63] pour des solutions optimales lorsque $D_i = c\sigma_i^2 2^{-2R_i}$, mais celles-ci peuvent retourner des valeurs de R_i négatives. Plus récemment, des modèles analytiques *ad-hoc* de $D_i(R_i)$ ont été proposés [Raj04] pour du codage d'images basse résolution. Le défaut des méthodes par modèles continus est leur spécificité qui les contraint à des problèmes particuliers, un modèle ne pouvant pas forcément s'appliquer à un problème différent de celui pour lequel il est élaboré.

La contrainte additionnelle de positivité des taux donne lieu à des conditions de Kuhn-Tucker pour des fonctions $D_i(R_i)$ strictement décroissantes et convexes ; la solution étant ensuite obtenue par un algorithme classique de *reverse waterfilling* [Dav72, Seg76]. Pour le problème dual en capacité-puissance appliqué à des canaux gaussiens parallèles, il est bien connu [Gal68] que la solution est donné par un algorithme de waterfilling. C'est d'ailleurs l'approche aussi employée dans [CCB95, FH96, LC97, BFB02].

L'avantage principal de l'approche continue est sa rapidité. Cependant, ces méthodes s'appliquent relativement mal à des fonctions $D_i(R_i)$ non convexes et/ou à des taux donnés contraints à un ensemble de valeurs particulières (e.g., dans un domaine spécifique, ou défini sur des entiers). Dans [HS63], les valeurs entières de R_i sont recherchés par tâtonnement (sous le terme anglophone *trial and error*) par rapport à la valeur réelle "optimale" correspondante. Cela limite grandement l'optimalité de telles méthodes.

Approche lagrangienne

Les performances résultant des algorithmes basés sur une approche continue sont faibles spécialement lorsque l'on se retrouve avec un ensemble limité de codeurs disponibles. Il est par conséquent important de prendre en compte la nature discrète des taux accessibles pour résoudre le problème d'allocation de ressources. L'aspect discret des taux et le caractère linéaire des variables globales nous amènent naturellement à aborder la résolution du problème contraint

sous la forme d'une optimisation lagrangienne et de le ramener à un problème non-contraint : l'approche lagrangienne est directement motivée par le théorème 1.5.5.

L'approche lagrangienne, par sa caractérisation de convexité, favorise la sélection efficace des points d'enveloppe. La majeure partie des algorithmes implémentés aujourd'hui sont fondés sur ce type d'approche.

Nous verrons dans la suite de ce chapitre uniquement les méthodes basées sur une approche lagrangienne du problème. En effet, comme expliqué précédemment, les méthodes continues ne peuvent s'exprimer sur des problèmes généraux et nécessitent une spécificité des fonctions $D_i(R_i)$ que nous ne voulons pas introduire. Nous préférons garder l'aspect général de notre propos, englobant le maximum de problèmes possible.

Autres approches

Parmi les autres approches existantes, une grande partie d'entre elles sont issues ou dérivées des méthodes de programmation dynamique [Bel57, GG61, GG63, GG66] ou de techniques de *branch and bound* [Kol67, HS74]. Ces approches sont à l'origine de nombreux travaux algorithmiques, mais tirent majoritairement profit de la répartition des taux. En effet, ces méthodes s'intéressent généralement à des taux entiers ce qui limite la portée de celles-ci à des problèmes plus généraux.

On peut aussi citer des approches basées sur les réseaux de neurones ou des algorithmes génétiques, dont la mise en œuvre n'a pas encore été exploitée dans le domaine du codage numérique. Ces algorithmes sont essentiellement déclinés dans le cadre de la recherche opérationnelle que nous n'évoquerons pas ici.

2.3 Méthodes de recherche de points d'enveloppe du plan global

Everett [Eve63] fut apparemment le premier à faire des recherches sur les méthodes lagrangiennes discrètes dans le contexte d'analyse opérationnelle pour résoudre les problèmes du type $\max_{A \leqslant A_b} B$.

Everett prouva le théorème 1.5.5 et mentionna l'existence de régions *inaccessibles* contenant des points cachés : les triangles cachés ∇^h (voir Fig. 1.2). Comme mentionné plus tôt, tous les algorithmes basés sur des méthodes lagrangiennes sont généralement sous-optimaux puisque les points cachés ne peuvent être atteints par ce moyen.

Du théorème 1.5.5 nous déduisons immédiatement l'algorithme C, qui, pour une valeur donnée

de multiplicateur λ, trouve le point d'enveloppe correspondant sur l'enveloppe convexe :

Algorithme C Everett

Cet algorithme retourne le point d'enveloppe (R^e, D^e) d'allocation $\underline{k}^e$ associé à un multiplicateur λ.

1. Pour $i = 1, \ldots, N$, trouver k_i^e qui minimise $\mathscr{L}_i = D_i(k_i^e) + \lambda R_i(k_i^e)$
2. Retourner $\underline{k}^e = [k_1^e, \ldots, k_N^e]$.

Maintenant, le problème est de trouver la valeur adéquate de λ. Pour résoudre le problème de la courbe optimale R-D, c'est-à-dire, trouver l'ensemble des points d'enveloppe sur l'enveloppe convexe, on peut commencer de $\lambda = 0$ et itérer en accroissant $\lambda \longrightarrow \lambda + \delta\lambda$ à chaque pas. Si $\delta\lambda$ est suffisamment petit, tous les points d'enveloppe seront finalement trouvés par cette procédure. Cependant, prendre un $\delta\lambda$ très petit conduit à une complexité élevée et un choix a priori de $\delta\lambda$ ne garantit pas de trouver tous les points d'enveloppe. Un compromis intéressant a été trouvé expérimentalement par [SG88]. Notons que certains points ne pourront de toute manière pas être trouvés par cette méthode.

Pour résoudre le problème d'allocation de ressources sous la contrainte d'un budget R_b, la même procédure de balayage du type $\lambda \longrightarrow \lambda + \delta\lambda$ peut être employée. On trouve alors le point d'enveloppe (R, D) (éventuellement sous-optimal) pour lequel $R \leqslant R_b$ est le plus proche de R_b ; la recherche s'arrête en fait dès que R_b est dépassé.

Une autre procédure efficace a été proposée dans le cadre du codage de source par Ramchandran et Vetterli [RV93] puis repris par Krongold, Ramchandran et Jones [KRJ00] dans le cas de modulations multi-porteuses. Ils utilisent un algorithme pour trouver une racine sécante sur $R(\lambda) - R_b$ où $R(\lambda)$ est le taux pour lequel le lagrangien $\mathscr{L} = D + \lambda R$ est minimum. Lee and Ra [LR96] ont proposé une modification pour mettre à jour λ dans la procédure, fondée sur l'hypothèse que les $D_i(R_i)$ sont exponentiels décroissants. La recherche de Ramchadran et Vetterli converge après quelques itérations, et est bien adaptée aux situations dans lesquelles les indices $\underline{k}$ sont organisés sous la forme d'un arbre (e.g. pour une quantification vectorielle à structure en arbre ou pour la décomposition en paquets d'ondelettes) ; dans ce contexte, c'est une application spécifique de l'algorithme BFOS généralisé [CLG89].

Notons que dans l'algorithme C, il n'est pas nécessaire de faire la minimisation effective de $\mathscr{L}_i(k_i)$ pour un lambda donné λ, puisque nous savons que la solution $(R_i(k_i), D_i(k_i))$ est un point d'enveloppe pour ce λ. Ainsi, il est suffisant d'appliquer l'algorithme RDPE une fois pour toute afin de trouver la suite décroissante $\{\lambda_i(k_i)\}$ de multiplicateurs critiques pour chaque composante i. Ensuite, la solution k_i qui minimise $\mathscr{L}_i(k_i) = D_i(k_i) + \lambda R_i(k_i)$ doit vérifier $\lambda \in [\lambda_i(k_i - 1), \lambda_i(k_i)]$ où $\lambda_i(k_i)$ est le multiplicateur critique :

$$\lambda_i(k_i) = \frac{D_i(k_i) - D_i(k_i + 1)}{R_i(k_i + 1) - R_i(k_i)} \tag{2.1}$$

La solution est trouvée par une simple comparaison à la suite de multiplicateurs critiques $\{\lambda_i(k_i)\}$. Une remarque similaire apparaît dans [YGS02].

En se servant des remarques précédentes, il est possible d'obtenir l'ensemble des points de l'enveloppe convexe en se restreignant au calcul des multiplicateurs critiques $\lambda = \lambda_i(k_i)$. Cette approche, qui se trouve être la plus efficace pour rechercher tous les points d'enveloppe, fut proposée par Shoham et Gersho [SG88]. Leur algorithme est basé sur le théorème 1.5.6 qu'ils furent les premiers à énoncer explicitement. Il en découle l'algorithme D.

Algorithme D Shoham-Gersho

Cet algorithme retourne l'ensemble des points d'enveloppe $\mathcal{H}$.

1. *[Initialisation]*
   Appliquer l'algorithme RDPE à chaque composante i.
   Fusionner l'ensemble des multiplicateurs critiques (1.19) en une unique liste triée par ordre décroissant $\mathcal{L} = \{\lambda^j\}_j$ (voir (1.17)).
   Initialiser $\mathcal{H} = \underline{1}$ et $j = 1$.
2. *[Boucle]*
   Appliquer au dernier point de $\mathcal{H}$ l'opération élémentaire $\vec{\Delta}_{\lambda^j}$.
3. *[Mise à jour]*
   Rajouter le point obtenu dans $\mathcal{H}$.
   Actualiser $j \longmapsto j + 1$.
4. Retourner $\mathcal{H}$ si $\mathcal{L}$ a été entièrement parcourue.
   Revenir à l'étape 2 sinon.

La présentation initiale de Shoham et Gersho supposait les taux à valeurs entières, mais ils firent remarquer eux-même que leur procédure s'appliquait de façon similaire à des taux non-entiers. Des présentations équivalentes se retrouvent dans des articles tels que [WBB88] et [Ris91] qui présentent le problème d'allocation dans un contexte de structure en arbre de paquets d'ondelettes. Cette présentation paraît toutefois "surdimensionnée" au regard de la formulation initiale du problème.

Appliqué au problème d'allocation de ressources sous la contrainte d'un budget R_b, cet algorithme s'arrête dès que R_b est dépassé. Nous noterons qu'à la même époque, Hughes-Hartogs publia une série de brevets [Huga, Hugb, Hugc] dont le cadre d'emploi est la modulation multi-porteuses pour des modems et qui décrit une méthode d'allocation de ressources équivalente à celle de Shoham et Gersho.

La procédure de recherche du λ critique peut être améliorée par restriction du domaine d'étude des multiplicateurs :

$$\lambda^l > \lambda^{l+1} > \cdots > \lambda^r \tag{2.2}$$

où les points d'enveloppe (R^l, D^l) et (R^r, D^r), correspondant respectivement aux multiplicateurs critiques λ^l et λ^r, sont tels que $R^l < R_b < R^r$. Afin de trouver ces points d'enveloppe encadrant le budget, nous pouvons appliquer la méthode de balayage $\lambda \longrightarrow \lambda + \delta\lambda$ avec des valeurs empiriques de pas $\delta\lambda$ et de λ initial [SG88]. Une alternative plus efficace et moins aléatoire consiste à mettre en œuvre la procédure de Ramchandran et Vetterli [RV93] mentionnée précédemment.

En s'appuyant sur le résultat du théorème 1.5.9, l'algorithme de Shoham-Gersho est effectivement optimal pour des taux uniformément espacés et des fonctions $D_i(R_i)$ convexes, puisque dans ce cas tous les points optimaux se situent sur l'enveloppe convexe. En supposant, pour alléger l'écriture, que les taux sont à valeurs entières : $R_i = k_i - 1$, l'expression des multiplicateurs critiques se résume à des différences de distorsions $\lambda_i(k_i) = D_i(k_i) - D_i(k_i + 1) = \Delta D$ et l'algorithme de Shoham-Gersho D s'exprime de manière simplifiée sous la forme de l'algorithme E.

Algorithme E Fox

Cet algorithme retourne l'ensemble des points d'enveloppe $\mathcal{H}$ pour des fonctions convexes et des taux entiers.

1. *[Initialisation]*
   Fusionner l'ensemble des $\Delta D = D_i(R_i) - D_i(R_i + 1)$ en une unique liste triée par ordre décroissant $\mathcal{L} = \{\Delta D^j\}_j$.
   Initialiser $\mathcal{H} = (0, D(\underline{1}))$ et $j = 1$.
2. *[Boucle]*
   Appliquer $(R \longmapsto R + 1, D \longmapsto D - \Delta D^j)$.
3. *[Mise à jour]*
   Rajouter le point (R, D) dans $\mathcal{H}$.
   Actualiser $j \longmapsto j + 1$.
4. Retourner $\mathcal{H}$ si $\mathcal{L}$ a été entièrement parcourue.
   Revenir à l'étape 2 sinon.

Fox proposa cet algorithme [Fox66], et prouva son optimalité dans le cadre de ses conditions d'étude, de nombreuses années avant Shoham et Gersho. Son étude portait en fait sur des problèmes du type $\max_{A \leqslant A_b} B$ qui apparaissent en analyse marginale, avec des applications dans les domaines des jeux de guerre et de choix de cibles stratégiques pour des missiles tactiques (on rapprochera aisément la date de parution de l'article à la situation géopolitique de l'époque...). Fox n'apporta, par contre, pas de réponse au cas des fonctions non convexes, son algorithme aboutissant à des solutions non optimales (voire loin de l'optimalité).

Une simplification supplémentaire peut être apportée à l'algorithme précédent si l'on suppose une relation de la forme $D_i = c\sigma_i^2 2^{-2R_i}$, e.g. pour le codage en sous-bandes de signaux avec la distorsion assimilée comme l'erreur quadratique moyenne. Dans ce cas, la différence $\Delta D = D_i(R_i) - D_i(R_i + 1) = \frac{3}{4} D_i$ est proportionnelle à la valeur $\sigma_i^2 2^{-2R_i} = (\sigma_i / 2^{R_i})^2$ et

chaque incrémentation $R_i \longrightarrow R_i + 1$ réduit de moitié l'expression $\sigma_i / 2^{R_i}$. Cette considération amène l'algorithme F.

Algorithme F Ramstad

Cet algorithme décrit l'ensemble des points d'enveloppe pour des taux R_i régulièrement espacés et des fonctions $D_i = c\sigma_i^2 2^{-2R_i}$.

1. *[Initialisation]*
   Établir la liste $\mathcal{L} = \{\widetilde{\sigma}_i = \sigma_i\}$.
2. *[Boucle]*
   Allouer un bit dans la composante i de plus grand $\widetilde{\sigma}_i$.
3. *[Mise à jour]*
   Remplacer $\widetilde{\sigma}_i$ par $\widetilde{\sigma}_i / 2$ dans $\mathcal{L}$.
4. Revenir à l'étape 2.

Pour résoudre le problème d'allocation de ressources, Ramstad [Ram82] proposa cet algorithme avec un facteur de réduction r quelconque et observa à raison qu'un facteur proche de 2 permettait de retourner un point à proximité de la solution optimale. Il fit aussi le rapprochement pour $r = 2$ à des solutions continues dans [Ram86]. Notre dérivation de son algorithme, dont la description rappelle celle de l'algorithme D, prouve que son choix intuitif, basé sur des observations empiriques, est en effet réellement optimal. Nous voyons ainsi que l'algorithme de Shoham-Gersho est d'une certaine manière la synthèse de plusieurs méthodes antérieures qui traitaient des cas particuliers d'un même problème.

Dans un contexte de taux entiers (interprétés comme des bits) et de fonctions convexes de la forme $D_i(R_i) = c\sigma_i^2 2^{-2R_i}$, Campello [Cam98, Cam99] proposa un moyen original de restreindre le tri des multiplicateurs critiques à un domaine compris entre des valeurs particulières. La méthode consiste à se servir d'une caractérisation spécifique d'une allocation dite alors *efficace* et définie comme suit :

Définition 2.3.1
Soit $\underline{k} = [k_1, \ldots, k_N]$ un vecteur d'allocation quelconque. On considère l'ensemble des différences de distorsion (équivalentes aux multiplicateurs critiques dans le cas de taux entiers) dans chaque composante :

$$\{\Delta D_i(k)\}_{(i,k)} = \{D_i(k) - D_i(k-1)\}_{(i,k)} \tag{2.3}$$

$\underline{k}$ est dit efficace *si ses coordonnées vérifient :*

$$\min_i \Delta D_i(k_i) \geqslant \max_i \Delta D_i(k_i + 1) \tag{2.4}$$

Campello annonce que si $\underline{k}$ est efficace (au sens de la définition 2.3.1) et que $R(\underline{k}) = R_b$, alors $\underline{k}$ est optimal. Il propose alors une formule directe d'allocation de bits $\underline{k}$ qui vérifie cette

caractérisation d'*efficacité* :

Théorème 2.3.1
Soit l'allocation définie par :

$$k_i = [\lfloor log_2(\sigma_i) \rfloor - p]_1^{M_i} \qquad i = 1, \ldots, N \tag{2.5}$$

où p est un entier quelconque, et $[x]_a^b = \begin{cases} b & \text{si } b < x \\ x & \text{si } a \leqslant x \leqslant b \\ a & \text{si } x < a \end{cases}$
Alors, le point $\underline{k} = [k_1, \ldots, k_N]$ est efficace au sens de la définition 2.3.1.

On peut vérifier que ces points sont efficaces en remplaçant tout simplement la formule (2.5) dans la définition 2.3.1. Afin de justifier cette formule par des considérations théoriques, nous allons étudier plus précisément les positions respectives des $\{\sigma_i\}$.

Preuve : Appliquons l'algorithme F au problème. À chaque étape, lorsqu'on divise un $\widetilde{\sigma}_i$ par 2 et qu'on augmente k_i de 1, cela revient à retrancher 1 de $log_2(\widetilde{\sigma}_i)$. L'algorithme F, après un certain nombre d'itérations, nous amène donc à avoir tous les $\widetilde{\sigma}_i$ contenus dans un intervalle de longueur 1 : $p \leqslant \widetilde{\sigma}_i < p + 1, \forall i$ avec $p \in \mathbb{N}$. Cela signifie bien que $\lfloor log_2(\widetilde{\sigma}_i) \rfloor = \lfloor log_2(\sigma_i) \rfloor - k_i = p$. En se limitant aux intervalles de définition des différentes composantes, on retrouve finalement la formule (2.5) avancée par Campello. Celle-ci met bien en évidence des points d'enveloppe particuliers, donc optimaux. ■

Au final, Campello sélectionne des points d'enveloppe caractérisés par (2.4), et encadre la solution optimale par deux de ses points efficaces. Cette procédure a l'avantage de limiter le domaine de recherche et se révèle être alors plus rapide qu'un tri exhaustif des multiplicateurs critiques.

Quelle que soit la méthode lagrangienne employée, nous voyons que les points obtenus sont toujours des points d'enveloppe. Ces méthodes trouvent des points optimaux, mais ces points ne sont pas forcément la solution optimale au problème d'allocation de ressources. Pour être exhaustif, il faut donc aussi s'intéresser aux méthodes qui recherchent des points cachés.

2.4 Méthodes de recherche de points cachés du plan global

A notre connaissance, mis à part le cas très restrictif des fonctions $D_i(R_i)$ convexes à taux entiers, le problème d'allocation de ressources n'a pas encore été efficacement (d'un point de vue complexité) résolu. Un tel algorithme devrait non seulement être capable de trouver les points d'enveloppe, mais aussi les points optimaux cachés. Étrangement, très peu de recherches ont été menées dans ce but dans le domaine du codage numérique, l'existence de tels points

cachés étant parfois complètement occultée. La littérature n'en fait en tout cas quasiment pas mention.

Il existe bien entendu un algorithme "évident" fondé sur l'itération de l'algorithme RDPO. Soit $I \cup J$ une partition de $\{1, 2, \ldots, N\}$ en deux ensembles d'index. Soient $R_I = \sum_{i \in I} R_i$, $D_I = \sum_{i \in I} D_i$ et, de manière similaire, (R_J, D_J) les points correspondant à ces ensembles d'index. Considérons alors $R = R_I + R_J$ et $D = D_I + D_J$. Par application immédiate du théorème 1.5.3 (pour $N = 2$) tous les points optimaux (R^*, D^*) correspondent à des (R_I^*, D_I^*) et des (R_J^*, D_J^*) optimaux et peuvent être obtenus par application de l'algorithme RDPO aux nuages de points $\{(R_I, D_I)\}$ et $\{(R_J, D_J)\}$. Par partitions successives I et J nous obtenons l'algorithme itératif G.

Algorithme G Fusion de nuages

Cet algorithme retourne l'ensemble des points optimaux $\mathcal{O} = \{(R^*, D^*)\}$.

1. *[Initialisation]*
   Initialiser $\mathcal{O} = \{(R_1, D_1)\}$ et $n = 2$.
2. *[Boucle]*
   Fusionner les nuages $\mathcal{O}$ et $\{(R_n, D_n)\}$ dans $\mathcal{O}$.
3. *[Mise à jour]*
   Appliquer l'algorithme RDPO à $\mathcal{O}$.
   $n \longmapsto n + 1$.
4. Retourner $\mathcal{O}$ si $n > N$.
   Revenir à l'étape 2 sinon.

De plus, appliqué au problème d'allocation de ressources sous la contrainte d'un budget R_b, il est suffisant de ne conserver à chaque étape n que les points (R^n, D^n) vérifiant $R^n \leqslant R_b$.

Pour des taux R_i à valeurs entières, il s'agit d'une méthode classique de *programmation dynamique*, et sa dérivation, d'une conséquence directe du principe d'optimalité de Bellman. Toutes les solutions possibles peuvent être représentées sous la forme d'un treillis dont l'état de la $n^{\text{ième}}$ étape est donné par les valeurs de R^n, chaque état R^n donné de l'étape n génère un chemin survivant dans le treillis.

Malheureusement, l'algorithme G possède une complexité prohibitive pour notre problème d'allocation. L'annexe C.5 détaille ce calcul, à comparer avec les performances de l'algorithme de Fox (voir annexe C.4).

Fox [Fox66] étendit son algorithme E au cas d'ensembles arbitraires $\{(R_i, D_i)\}$ afin de tenter de trouver des points cachés. Mais, bien qu'optimal pour des fonctions $D_i(R_i)$ convexes, il a été démontré [SG88] que cette optimalité s'effondre pour des fonctions $D_i(R_i)$ non convexes : les solutions sont souvent loin de la solution optimale.

Trushkin [Tru80] proposa une simplification de la méthode par programmation dynamique

pour des taux entiers et des fonctions $D_i(R_i)$ convexes. Cela est précisément le cas dans lequel l'algorithme de Fox est optimal, et au final, la dérivation de Trushkin se révèle être équivalente à celle de Fox pour chaque étape de l'algorithme G. Une implémentation directe de l'algorithme de Fox E est par conséquent bien supérieure.

Plus récemment, Yoo *et al.* [YOR96] proposèrent une autre modification de l'algorithme G pour résoudre le problème d'allocation sous contrainte d'un budget R_b. Leur présentation suppose les taux R_i entiers mais celle-ci peut être facilement étendue à des ensembles arbitraires de valeurs de taux. L'idée est d'appliquer une fusion de nuages en se limitant à des nuages restreints $\{(R_i, D_i)\}$ tels que $|R_i - \bar{R}_i| \leqslant \Delta$. Ici, $\bar{R}_i$ est obtenu soit par le taux moyen $\bar{R}_i = R_b/N$, soit par la solution de l'algorithme D pour le même budget. On obtient l'algorithme H.

Algorithme H **Fusion de nuages à fenêtre Θ**

Cet algorithme retourne une solution au problème d'allocation de ressource pour un budget R_b.

1. *[Initialisation]*
   Initialiser $\bar{R}_i$ pour tout i.
   Retirer des composantes $\{(R_i, D_i)\}$ tous les points ne vérifiant pas $|R_i - \bar{R}_i| \leqslant \Theta\}$.
   Initialiser $\mathcal{P} = \{(R_1, D_1)\}$ et $n = 2$.
2. *[Boucle]*
   Fusionner les nuages $\mathcal{P}$ et $\{(R_n, D_n)\}$ dans $\mathcal{P}$.
3. *[Mise à jour]*
   Appliquer l'algorithme RDPO à $\mathcal{P}$.
   $n \longmapsto n + 1$.
4. Aller à l'étape 5 si $n > N$.
   Revenir à l'étape 2 sinon.
5. Sélectionner le point $(R_{\text{sol}}, D_{\text{sol}})$ dans $\mathcal{P}$ vérifiant $R_{\text{sol}} = \max_{\mathcal{P}}(R \leqslant R_b)$.
   Retourner $(R_{\text{sol}}, D_{\text{sol}})$.

Pour la résolution du problème d'allocation de ressource, la complexité de cet algorithme est moindre que celle de l'algorithme G. Cependant, certains chemins du treillis étant éliminés par la condition $|R_i - \bar{R}_i| \leqslant \Theta$, nous n'avons aucune garantie d'optimalité. Même en augmentant Θ, il n'y a pas de moyen de garantir l'optimalité d'une telle méthode. De plus, le choix d'une valeur de Θ est relativement empirique et ne peut être ajusté qu'expérimentalement. On notera aussi que cette méthode ne pourra aucunement s'appliquer au problème de la courbe optimale R-D car il n'y est pas du tout adapté (même l'algorithme G peut travailler plus vite).

De nombreux articles mentionnent aussi l'existence d'un algorithme dit *avide* ou *avare* (évoqué sous le terme *greedy* en anglais). Sans vraiment décrire la procédure, ils la considèrent comme "un algorithme qui alloue *un bit à la fois* dans la composante qui atteindra le *meilleur résultat* pour l'allocation partielle courante". Une forme de procédure avare est détaillée dans [GG91] pour des taux entiers. Toute la subtilité réside dans la définition du terme "meilleur résultat" : si,

pour des taux à valeurs entières, les multiplicateurs critiques et les distorsions sont totalement équivalentes, ce n'est pas le cas de fonctions à taux irréguliers. Le critère reste donc à définir proprement. Une procédure avare pourrait être décrite selon l'algorithme I.

Algorithme I Procédure avare

Cet algorithme retourne une solution $\underline{k}$ au problème d'allocation de ressource pour un budget R_b donné.

1. *[Initialisation]*
   Initialiser $\underline{k} = \underline{1}$.
2. *[Boucle]*
   Appliquer à $\underline{k}$ l'opération élémentaire $\vec{\Delta}_\lambda$ de λ maximal vérifiant $R(\vec{\Delta}_\lambda(\underline{k})) \leqslant R_b$.
3. Retourner $\underline{k}$ si il n'y a plus aucune opération $\vec{\Delta}_\lambda$ candidate disponible.
   Revenir à l'étape 2 sinon.

L'optimalité d'une telle procédure n'est garantie que pour des fonctions convexes à taux entiers [FG86], auquel cas elle s'assimile à l'algorithme E. Cependant, dans le cas des fonctions les plus générales de type CNU, les résultats obtenus par cet algorithme sont meilleurs que ceux obtenus par l'algorithme de Shoham-Gersho, car celui-ci ne se restreint pas aux seuls points d'enveloppe. Pour être plus précis, l'algorithme I décrit les même points que l'algorithme D jusqu'à la solution de Shoham-Gersho pour ensuite parcourir une suite de points dans le triangle caché ∇_b^h. Ce constat est uniquement valable pour des fonctions convexes. Lorsque nous considérons des fonctions non convexes, le principe d'allocation d'un seul bit à la fois amène des résultats particulièrement dégradés. En effet, dans ce cas, la procédure ne parcourt plus les points d'enveloppe en "bifurquant" dès la première non-convexité rencontrée.

Shoham et Gersho [SG88] tentèrent de même de trouver des points cachés à partir de la solution de l'algorithme D, en allouant les bits restants aux composantes introduisant le plus grand différentiel en distorsion. Il s'agit en fin de compte d'appliquer à l'issue de l'algorithme D une procédure avare. La méthode décrite par Shoham et Gersho n'est toutefois envisageable que pour des taux entiers, et bien que les points trouvés soient à l'intérieur des triangles cachés, il n'y a aucune garantie que ceux-ci soient optimaux.

2.5 Performances

Un résumé des algorithmes de l'état de l'art apparaît dans la table 2.1. Ce tableau récapitulatif présente les algorithmes regroupés en trois catégories : méthodes continues, méthodes lagrangiennes discrètes, et méthode de recherche de points cachés. Dans chaque catégorie, les algorithmes sont rangés par ordre de parution. Concernant la complexité, les $+$ indiquent des algorithmes rapides, les $-$ indiquent des algorithmes lents.

TAB. 2.1: Récapitulatif des conditions d'emploi et des performances des algorithmes

Références	Algorithme	Contexte R_i	Contexte $D_i(R_i)$	Solution	Complexité
[HS63]		continu	$D_i = c_i\sigma_i^2 2^{-2R_i}$	approx.	$+++$
[Dav72]	(waterfilling)	positif	$D_i = \sigma_i^2 2^{-2R_i}$	approx.	$+++$
[Seg76]	(waterfilling)	positif	strict. convexe	approx.	$+++$
[Fox66, Tru80]	E	entiers	strict. convexe	**optimal**	$+++$
[Ram82, Ram86]	F	entiers	$D_i = c\sigma_i^2 2^{-2R_i}$	**optimal**	$+++$
[SG88, WBB88, Ris91]	D	**tous**	**tous**	enveloppe	$+++$
[RV93, LR96, YGS02]	D+ sécante	**tous**	**tous**	enveloppe	$+++$
	G	**tous**	**tous**	**optimal**	$---$
	I	**tous**	**tous**	sous-optimal	$+++$
[YOR96]	H	**tous**	**tous**	sous-optimal	$-/++$
§ 3	J	**tous**	strict. convexe	**optimal**	$++$
§ 4	K	**tous**	**tous**	**optimal**	$+$
§ 5	M	**tous**	strict. convex	sous-optimal	$+++$
§ 6	N	**tous**	**tous**	sous-optimal	$+++$

Dans ce tableau récapitulatif, les considérations sur la complexité sont tirées entre autre des calculs regroupés dans l'annexe C. Nous pouvons remarquer qu'il n'existe pas aujourd'hui d'algorithmes de complexité raisonnable répondant de manière optimale au problème d'allocation de ressources. La recherche des points cachés est payée au prix fort.

Pour l'anecdote historique, nous noterons que dès 1963, Everett donna dans son article précurseur [Eve63] deux pistes intéressantes pour élaborer des méthodes nouvelles de recherche de points cachés :

1. utiliser des valeurs de multiplicateurs critiques λ_i "composites" pour différentes composantes i,
2. rechercher des points possédant un lagrangien $\mathscr{L} = D + \lambda R$ sous optimal.

Ces deux idées ont été totalement sous-exploitées durant ces quarante dernières années. Dans les chapitres 3 à 6 suivants, nous proposerons des algorithmes qui mettent enfin en œuvre ces pistes prometteuses. À titre de comparaison, nous avons inséré dans le tableau 2.1 une quatrième série de lignes représentant les performances de ces algorithmes. On peut donc déjà apprécier par anticipation les résultats des méthodes ainsi développées. Pour des comparatifs de performances plus poussés, nous nous reporterons aux paragraphes 3.5, 4.6, 5.5 et 6.4 des chapitres correspondants.

Nous avons appliqué quelques uns des principaux algorithmes de l'état de l'art aux différents ensembles du corpus de test, et nous avons calculé les performances de ceux-ci du point de vue des critères décrits au chapitre précédent. Nous avons pris $N = 16$ composantes pour

l'ensemble des tests et $M_i = 25$ valeurs par composantes pour les ensembles de fonctions CNU, NCU, NCNU et $N = 16$ et $M_i = 13$ pour l'ensemble de fonctions CUT. Les tests ont été effectués pour une séquence de 1000 budgets répartis sur l'ensemble de la dynamique de taux ($R_b \in [R(\underline{1}), R(\underline{M})]$). Les performances expérimentales sont ainsi récapitulées dans les tableaux 2.2- 2.5, rangées par ensemble de fonctions.

TAB. 2.2: Performances des algorithmes pour l'ensemble CUT.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.03	51.8	2.48
G	3.8	100	0
$H_{(\Theta = 2)}$	0.6	36.8	4.40
$H_{(\Theta = 5)}$	2.0	95.5	0.27
I	0.03	72.5	0.56

TAB. 2.3: Performances des algorithmes pour l'ensemble CNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	13.3	5.1
G	112	100	0
$H_{(\Theta = 2)}$	2.5	7.6	14.0
$H_{(\Theta = 5)}$	17.5	50.0	6.0
I	0.07	23.3	3.1

TAB. 2.4: Performances des algorithmes pour l'ensemble NCU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	49.9	2.84
G	7.7	100	0
$H_{(\Theta = 2)}$	0.36	3.5	11.8
$H_{(\Theta = 5)}$	1.52	56.4	2.69
I	0.07	7.7	8.6

TAB. 2.5: Performances des algorithmes pour l'ensemble NCNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	0.3	6.89
G	71.8	100	0
$H_{(\Theta = 2)}$	1.9	1.3	19.7
$H_{(\Theta = 5)}$	11.6	33.3	8.84
I	0.07	3.7	13.7

Les tableaux 2.2- 2.5 sont présentés par ensemble de test. Ils confirment les résultats théoriques du tableau 2.1 et permettent de vérifier le caractère sous-optimal des algorithmes D, H et I.

Seul l'algorithme G est vraiment optimal pour tout type de fonctions. Nous noterons cependant le temps prohibitif nécessaire à cet algorithme pour trouver une solution optimale ; les pires cas étant atteints pour des fonctions à taux non uniformément répartis. En effet, l'algorithme de fusion de nuages souffre énormément de l'irrégularité des taux, car le nombre de points optimaux dans un tel cas est bien plus important, tandis que des taux réguliers limitent fortement la "dispersion" des points cachés.

L'algorithme classique de Shoham-Gersho est le plus rapide de tous, mais la procédure avare se révèle plus performante sur les fonctions convexes pour un temps de calcul moyen à peine supérieur. Notons l'importante chute en performance de l'algorithme I pour des fonctions non convexes, comme précédemment évoqué.

L'amélioration de Yoo et al. (algorithme H) est plus rapide et prend moins de mémoire que l'algorithme G duquel elle découle, mais la contrainte de fenêtre réduit en général considérablement le pourcentage de points optimaux trouvés. Nous voyons que le choix de Θ est difficile à appréhender suivant le type de fonctions sur lesquelles s'applique la méthode. Dès lors, cette méthode demande un ajustement empirique de la fenêtre d'étude. Cet aspect rédhibitoire fait perdre à la méthode beaucoup de son intérêt initial.

Globalement, aucun des algorithmes étudiés ne propose une résolution optimale ou quasi optimale efficace du problème d'allocation de ressources.

2.6 Conclusions

Nous avons présenté les principales approches employées en allocation de ressources. Nous avons mis en évidence les différents domaines d'étude et les différentes méthodes de résolution. Nous avons rappelé que le codage de source et le codage de canal possèdent des problèmes théoriquement proches dans leur écriture, s'apparentant tous au problème d'allocation de ressources. Nous avons décrit les principes de résolution, tantôt selon une approche continue, tantôt selon une approche discrète introduisant un lagrangien.

Nous avons ensuite largement récapitulé l'ensemble des algorithmes existants dans le domaine pour la recherche des points d'enveloppe. Aussi bien issues du codage de source que du codage de canal, ces méthodes, basées sur la mise en évidence d'une enveloppe convexe, sacrifient la performance au profit d'une faible complexité. Leur champ d'application est parfois restrictif, et leur optimalité rarement vérifiée dans les cas les plus généraux. De plus, la diversité initiale des méthodes, due à la variété des approches utilisées, se révèle finalement faible, l'ensemble des méthodes lagrangiennes étant des variantes de l'algorithme présenté par Shoham et Gersho.

Nous avons aussi décrit les principales méthodes de recherche de points cachés. Ces méthodes varient en performances suivant que nous souhaitons optimiser le taux de points optimaux trouvés ou la complexité de l'algorithme. Cela nous permet d'identifier l'absence flagrante

de procédure optimale efficace (d'un point de vue complexité) pour la résolution du problème d'allocation sous la contrainte d'un budget. C'est la lacune majeure en allocation de ressource, et c'est que qui va principalement motiver notre étude au fil des chapitres 3 et 4 . Nous allons maintenant pouvoir nous intéresser à la recherche efficace de points cachés, et partir ainsi à la poursuite de l'optimalité.

Chapitre 3

Recherche optimale par chemins convexes

Laetitia est hominis transitio a
minore ad majorem perfectionem.
Spinoza

3.1 Introduction

Comme mentionné dans le précédent chapitre, les points cachés apparaissent dans le plan R-D pour des taux non-uniformément espacés (voir Figs. B.7, B.4 et B.8) ou des fonctions $D_i(R_i)$ non-convexes (voir Fig. B.5). Le but à atteindre pour répondre complètement à notre problème d'allocation original est donc de trouver *tous* les points optimaux, même si ceux-ci se trouvent être cachés. Nous souhaiterons par ailleurs atteindre ce résultat à une complexité moindre que les procédures basées sur une programmation dynamique coûteuse. Dans ce chapitre, nous allons proposer une nouvelle méthode originale pour résoudre de manière optimale le problème d'allocation de ressources dans le cas où les fonctions $D_i(R_i)$ sont convexes.

Le paragraphe 3.2 introduit donc cette méthode (qui s'applique pour des fonctions $D_i(R_i)$ *convexes*, mais par ailleurs générales)en présentant le concept de *chemin convexe*. Fondés sur l'utilisation des opérations élémentaires employées dans le chapitre précédent, les chemins convexes généralisent et complètent astucieusement le principe d'enveloppe convexe.

Quelques propriétés fondamentales de ces chemins convexes vont permettre de développer une méthode de recherche efficace par la mise en place au paragraphe 3.3 d'un critère d'élimination de chemins non pertinents.

Nous décrirons finalement un algorithme de recherche de points optimaux basé sur cette méthode au paragraphe 3.4.

Nous ferons ensuite un point au paragraphe 3.5 sur les performances de cet algorithme et nous les comparerons à celles des algorithmes déjà cités. Nous verrons ainsi le gain obtenu en terme de complexité.

Nous finirons enfin notre chapitre sur quelques conclusions au paragraphe 3.6.

3.2 Définition du concept de chemin convexe

Considérons l'opération élémentaire $\vec{\Delta}$ appliqué à un point $\underline{k}$, qui incrémente k_i mais conserve les autres composantes inchangées. Cela transforme le point $(R, D) = (R(\underline{k}), D(\underline{k}))$ en un point $(R', D') = (R(\underline{k}'), D(\underline{k}'))$ tel que $\underline{k}' = [k_1, \ldots, k_i + 1, , \ldots, k_N]$, en accord avec (1.26). Rappelons que cette opération élémentaire correspond à un unique multiplicateur critique $\lambda = \lambda_i(k_i)$, et inversement.

Il est facile de voir que n'importe quel point $\underline{k}$ peut être obtenu en partant de l'origine $\underline{1} = (1, \ldots, 1)$ en appliquant des opérations élémentaires $\vec{\Delta}$ successives. L'ensemble des points ainsi obtenus forment un *chemin* joignant $\underline{1}$ à $\underline{k}$ dans le plan R-D. À ce chemin correspond une unique séquence de multiplicateurs critiques :

$$\lambda_1(1), \ldots, \lambda_1(k_1 - 1), \lambda_2(1), \ldots, \lambda_2(k_2 - 1), \ldots, \lambda_N(1), \ldots, \lambda_N(k_N - 1). \tag{3.1}$$

Bien sur, nous n'avons pas besoin de nous limiter à un arrangement particulier de la séquence. Si nous appliquons des opérations élémentaires $\vec{\Delta}$ dans un ordre différent, nous obtenons un chemin différent reliant $\underline{1}$ à $\underline{k}$.La seule contrainte est la nécessité pour un $\lambda_i(j+1)$ d'arriver après le $\lambda_i(j)$ pour tout i et $j > 0$.

Maintenant, étant donné que tous les $D_i(R_i)$ sont supposés convexes, les multiplicateurs critiques

$$\lambda_i(1) > \lambda_i(2) > \cdots > \lambda_i(k_i - 1) \tag{3.2}$$

forment une sous-séquence décroissante pour tout i. On peut toujours les réunir pour en faire une suite décroissante de multiplicateurs critiques

$$\infty > \lambda^1 > \lambda^2 > \cdots > \lambda^m > 0 \tag{3.3}$$

de telle manière que le chemin correspondant soit *convexe*. À un point $\underline{k}$ correspond une

unique séquence décroissante de multiplicateurs critiques, et inversement. En d'autres termes, ce résultat s'écrit sous la forme du théorème suivant :

Théorème 3.2.1
Tout point $\underline{k}$ se trouve sur un unique chemin convexe reliant $\underline{1}$ à $\underline{k}$.

Selon l'algorithme de Shoham-Gersho, le chemin convexe correspondant à la séquence décroissante de *tous* les multiplicateurs critiques caractérise tous les points d'enveloppe (ce chemin définit l'enveloppe convexe). Mais le théorème 3.2.1 avance plus encore : nous obtenons tous les points optimaux (en fait tous les points du plan R-D) en cherchant tous les chemins convexes possibles partant de l'origine, comme illustré à la figure 3.1. Par unicité dans le théorème 3.2.1, chaque point $\underline{k}$ est obtenu une fois et une seule par cette procédure.

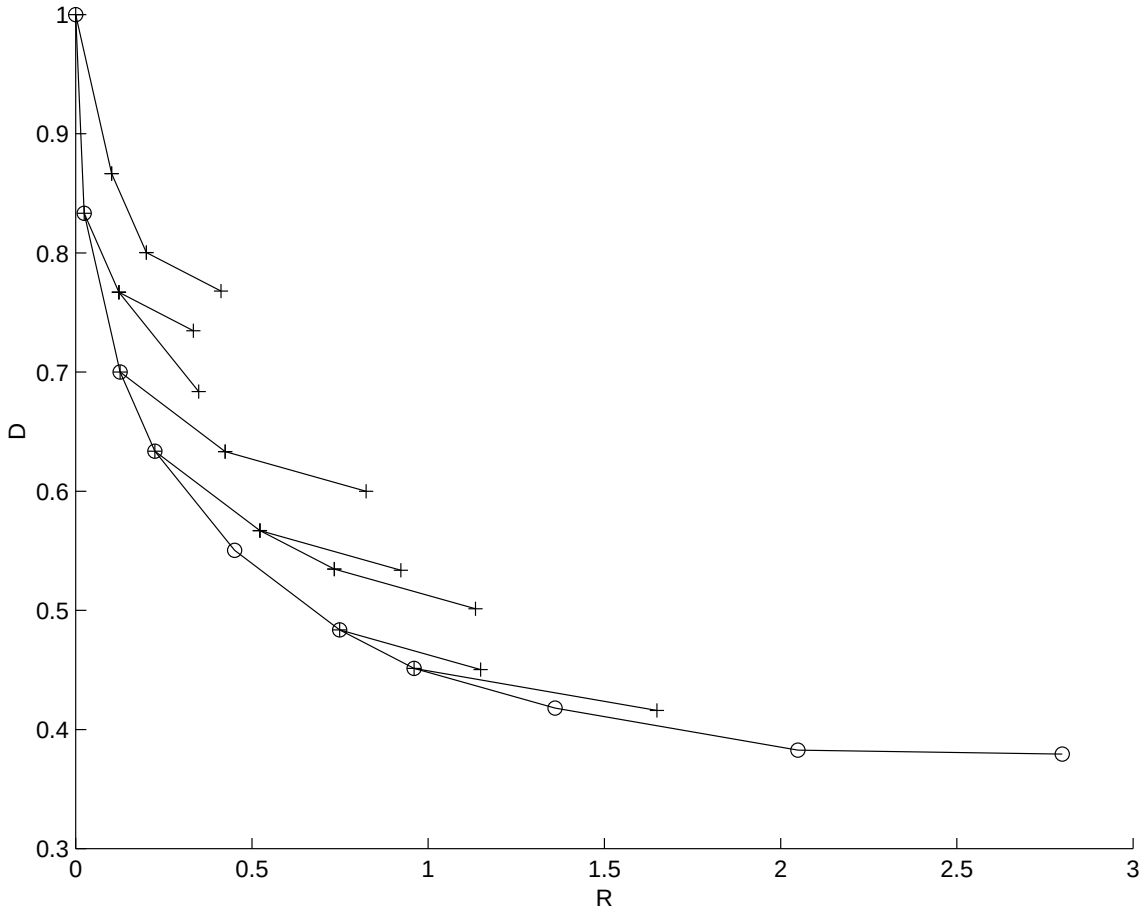


Fig. 3.1: Exemples de plusieurs chemins convexes décrits depuis l'origine $(R, D) = (0, 1)$

3.3 Mise en place d'un critère de recherche efficace

En pratique, les points optimaux sont recherchés uniquement dans une région spécifique $\mathcal{R}$. Par exemple, pour résoudre le problème d'allocation de ressources sous la contrainte d'un budget R_b, nous recherchons un point optimal avec un taux $R \leqslant R_b$ dans le triangle caché ∇_b^h.

Afin d'effectuer une recherche efficace des point cachés, nous devons nous donner un critère sur $\underline{k}$ qui garantit que la totalité des points à venir dans les chemins convexes qui passent par ce point $\underline{k}$ ne se situe pas dans la région spécifique de recherche $\mathcal{R}$.

Définition 3.3.1
Soit un point d'allocation $\underline{k}$. Considérant le problème d'allocation de ressources sous la contrainte d'un budget R_b, $\underline{k}$ sera déclaré mort *si tout point issu d'un chemin convexe passant par $\underline{k}$ ne se situe pas dans la zone de recherche $\mathcal{R}$.*

Un point mort peut être immédiatement retiré de la recherche des points cachés ; la recherche continuera uniquement avec les chemins convexes *survivants*.

Soit $\mathcal{R}_{\text{mort}}$ la région complémentaire de $\mathcal{R}$ *au dessus* de l'enveloppe convexe dans le plan R-D (voir Fig. 3.2). Par définition, n'importe quel point dans $\mathcal{R}_{\text{mort}}$ ne peut être une solution optimale, et si un point $\underline{k}$ et tous ses sous-chemins à venir sont dans cette région, alors $\underline{k}$ est mort. Cela est aussi vrai pour toute sous-région de $\mathcal{R}_{\text{mort}}$. Puisque il est difficile de décrire $\mathcal{R}_{\text{mort}}$ mathématiquement, nous considérerons une sous-région $\mathcal{S}_{\text{mort}}$ à la place. Pour cela nous avons besoin de définir :

Définition 3.3.2
Nous appellerons opérateur de translation $\mathbf{T}(\delta R, \delta D)$, une fonction du type :

$$\mathbf{T} : \begin{cases} R \longmapsto R + \delta R \\ D \longmapsto D + \delta D \end{cases}$$

Nous pouvons ainsi décrire la sous-région $\mathcal{S}_{\text{mort}}$ du plan global au moyen de la définition suivante :

Définition 3.3.3
Soient $\mathcal{H}$ l'enveloppe convexe dans le plan R-D (incluant les segments linéaires par morceaux reliant les points d'enveloppe) et $\mathbf{T}(\delta R, \delta D)$ un opérateur de translation où $\delta R > 0$ et $\delta D > 0$, de telle sorte que la version translatée de l'enveloppe convexe $\mathbf{T}\mathcal{H}$ se situe entièrement dans $\mathcal{R}_{\text{mort}}$, comme dessiné figure 3.2.
$\mathcal{S}_{\text{mort}} \subset \mathcal{R}_{\text{mort}}$ est définie comme la région au dessus de *$\mathbf{T}\mathcal{H}$.*

Nous allons maintenant présenter deux lemmes préliminaires qui vont nous permettre d'établir un résultat essentiel (décrit au théorème 3.3.3).

Tout d'abord, nous savons que tout point $\underline{k}$ correspond à une unique suite décroissante de

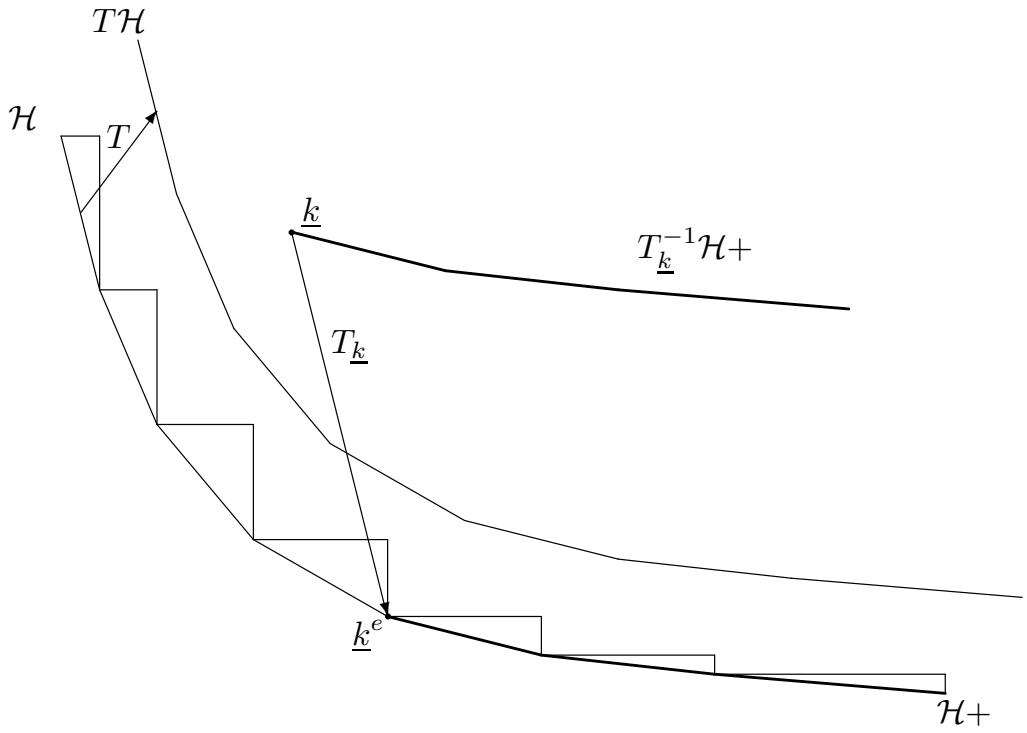


FIG. 3.2: Enveloppe convexe translatée $\mathbf{T}\mathcal{H}$ se situant dans la région $\mathcal{R}_{\text{mort}}$ au dessus de tous les triangles cachés ∇^h

multiplicateurs critiques :

$$\lambda^{j_1} > \cdots > \lambda^{j_m} \tag{3.4}$$

et tous les points $\underline{k}'$ appartenant à des sous-chemins convexes issus de $\underline{k}$ doivent correspondre à une séquence de multiplicateurs critiques du type :

$$\lambda^{j_1} > \cdots > \lambda^{j_m} > \lambda^{j_{m+1}} > \cdots \tag{3.5}$$

On a alors les inégalités $R(\underline{k}') > R(\underline{k})$ et $D(\underline{k}') < D(\underline{k})$ qui découlent de la décroissance de la courbe décrivant le chemin convexe.

Définissons $\underline{k}^e$ comme le point d'enveloppe de $\mathcal{H}$ correspondant à la séquence de tous les multiplicateurs $\lambda \geqslant \lambda^{j_m}$:

$$\lambda^1 > \lambda^2 > \cdots > \lambda^{j_m-1} > \lambda^{j_m} \tag{3.6}$$

Ce $\underline{k}^e$ est obtenu à partir de $\underline{k}$ en appliquant toutes les opérations élémentaires $\vec{\Delta}$ associées aux multiplicateurs critiques qui sont dans (3.6) mais pas dans (3.4).

Donc, $(R^e, D^e) = (R(\underline{k}^e), D(\underline{k}^e))$ est déduit de $(R, D) = (R(\underline{k}), D(\underline{k}))$ par un opérateur de translation $\mathbf{T}_{\underline{k}}(\delta_{\underline{k}}R, \delta_{\underline{k}}D)$ où $\delta_{\underline{k}}R > 0$ et $\delta_{\underline{k}}D < 0$. En appliquant $\mathbf{T}_{\underline{k}}$ aux points $\underline{k}'$ correspondant à (3.5), on obtient des points $\underline{k}^{e'}$ qui correspondent à des séquences de la forme

$$\lambda^1 > \lambda^2 > \cdots > \lambda^{j_m-1} > \lambda^{j_m} > \lambda^{j_{m+1}} > \cdots \tag{3.7}$$

Puisque tous les multiplicateurs critiques $\lambda \geqslant \lambda^{j_m}$ apparaissent dans ces séquences, tous les $\underline{k}^{e'}$ se trouvent sur des sous-chemins convexes passant par $\underline{k}^e$. Par conséquent, nous devons avoir $R(\underline{k}^{e'}) > R(\underline{k}^e)$ et $D(\underline{k}^{e'}) < D(\underline{k}^e)$. Sachant que $\underline{k}^e$ est sur l'enveloppe convexe $\mathcal{H}$, on en déduit de la définition de l'enveloppe convexe que tous les points $\underline{k}^{e'}$ doivent se trouver au-dessus de $\mathcal{H}$. En revenant aux points $\underline{k}'$ par application de la translation inverse $\mathbf{T}_{\underline{k}}^{-1}$, nous obtenons le lemme suivant :

Lemme 3.3.1
Tout point $\underline{k}'$ d'un sous-chemin convexe passant par $\underline{k}$ se situe au dessus de $\mathbf{T}_{\underline{k}}^{-1}\mathcal{H}$, un translaté particulier de l'enveloppe convexe passant par $\underline{k}$.

Notons que les $\underline{k}'$ sont par ailleurs tels que $R(\underline{k}') \geqslant R(\underline{k})$, c'est-à-dire, ils se situent au dessus

de la portion de $\mathbf{T}_{\underline{k}}^{-1}\mathcal{H}$ qui se trouve sur la droite de $(R(\underline{k}), D(\underline{k}))$. Nous noterons cette portion (droite) $\mathbf{T}_{\underline{k}}^{-1}\mathcal{H}+$ avec pour extrémité (gauche) $(R(\underline{k}), D(\underline{k}))$.

Maintenant, le deuxième lemme :

Lemme 3.3.2
Soient $\mathcal{C}$ une courbe convexe quelconque dans le plan R-D et $\mathbf{t}(\delta R, \delta D)$ un opérateur de translation vers la gauche, *c'est à dire avec $\delta R < 0$. Si l'extrémité gauche de $\mathbf{t}\mathcal{C}+$ se situe au dessus de $\mathcal{C}$, alors $\mathbf{t}\mathcal{C}+$ l'est aussi entièrement.*

Preuve : Soit $D(R)$ la fonction convexe définissant $\mathcal{C}$, de telle manière que $\mathbf{t}\mathcal{C}$ est définie par la fonction $D(R-\delta R)+\delta D$. Prenons l'extrémité gauche (R_0, D_0) de $\mathbf{t}\mathcal{C}+$ où $(R_0-\delta R, D_0-\delta D)$ se situe sur $\mathcal{C}$. Nous avons $D_0-\delta D = D(R_0-\delta R)$ et, par définition, $\delta R < 0$ et $D_0 > D(R_0)$. Nous devons prouver que $D(R-\delta R)+\delta D > D(R)$ pour tout $R \geqslant R_0$. Mais puisque $D(R)$ est convexe, nous devons avoir, pour tout $R \geqslant R_0$:

$$\frac{D(R_0-\delta R)-D(R_0)}{-\delta R} \leqslant \frac{D(R-\delta R)-D(R)}{-\delta R} \tag{3.8}$$

c'est à dire, $D_0-\delta D-D(R_0) \leqslant D(R-\delta R)-D(R)$. Par conséquent,

$$D(R-\delta R)+\delta D \geqslant D_0-D(R_0)+D(R) > D(R) \tag{3.9}$$

comme il devait être prouvé. ■

Grâce à ces deux lemmes, nous obtenons le principal théorème de ce paragraphe :

Théorème 3.3.3
Si $\underline{k}$ se situe dans $\mathcal{S}_{\text{mort}}$ alors $\underline{k}$ est mort.

Preuve : Premièrement, comme nous l'avons déjà fait remarquer plus haut, puisque $\mathcal{S}_{\text{mort}}$ est inclus dans $\mathcal{R}_{\text{mort}}$, si un point $\underline{k}$ et tous ses sous-chemins à venir se situent dans $\mathcal{S}_{\text{mort}}$ alors $\underline{k}$ est mort. Par conséquent, il est suffisant de montrer que si $\underline{k}$ se trouve dans $\mathcal{S}_{\text{mort}}$, alors c'est aussi le cas de tous les sous-chemins qui en découlent.

Maintenant, grâce au lemme 3.3.1 nous devons prouver que si $\underline{k}$ se trouve dans $\mathcal{S}_{\text{mort}}$, c'est à dire au dessus de $\mathbf{T}\mathcal{H}$, alors $\mathbf{T}_{\underline{k}}^{-1}\mathcal{H}+$ l'est aussi. C'est illustré figure 3.2 et cela revient à dire que $\mathbf{T}^{-1}\mathbf{T}_{\underline{k}}^{-1}\mathcal{H}+$ se situe au dessus de l'enveloppe convexe $\mathcal{H}$.

En reprenant les notations des deux lemmes précédents, $\mathbf{T}^{-1}\mathbf{T}_{\underline{k}}^{-1}$ est une translation vers la gauche : l'opérateur transforme R en $R-\delta_{\underline{k}}R+\delta R$ où $\delta_k R$ est positif et δR est une quantité négative. Ainsi, nous pouvons appliquer le lemme 3.3.2 à $\mathbf{t} = \mathbf{T}^{-1}\mathbf{T}_{\underline{k}}^{-1}$. ■

Remarquons que la force de ce résultat réside dans le fait qu'il est beaucoup plus simple de tester si $\underline{k}$ se situe ou pas dans $\mathcal{S}_{\text{mort}}$ que de tester des conditions similaires pour tous les sous-chemins à venir. De plus, la façon dont nous avons défini $\mathcal{S}_{\text{mort}}$ en utilisant $\mathbf{T}\mathcal{H}$ est primordiale dans la preuve ; notre résultat ne peut être étendu à la région plus grande $\mathcal{R}_{\text{mort}}$.

3.4 Description d'un algorithme optimal de recherche par chemins convexes

Ici nous appliquons le théorème 3.3.3 au problème d'allocation de ressources avec un budget R_b. Prenons le meilleur point d'enveloppe (R^e, D^e) que l'on trouve par exemple par l'algorithme de Shoham-Gersho, et $(R^{e'}, D^{e'})$ le prochain point d'enveloppe sur l'enveloppe convexe $\mathcal{H}$, de telle manière que $R^e \leqslant R_b \leqslant R^{e'}$. Le lemme 1.5.8 nous donne que la solution optimale doit se trouver dans le triangle ∇_b^h qui est donc la zone de recherche $\mathcal{R}$. Il suit que $\mathbf{T} : (R, D) \longmapsto (R + \delta R, D + \delta D)$ est choisie de telle sorte que la version translatée de l'enveloppe convexe $\mathbf{T}\mathcal{H}$ se situe au-dessus de $\mathcal{R}$. Pour simplifier, nous prenons $\delta R = 0$. La valeur δD doit alors être plus grande que la hauteur du triangle ∇_b^h, qui est :

$$\delta D_{\min} = \frac{R_b - R^e}{R^{e'} - R^e}(D^e - D^{e'}) = D^e - D_{\text{virt}} \tag{3.10}$$

où D_{virt} est la distorsion définie par (1.29).

Le point optimal est recherché en parcourant tous les chemins convexes possibles dans la région $R \leqslant R_b$. Il découle du théorème 3.3.3 que si un point (R, D) se situe au dessus de $\mathbf{T}\mathcal{H}$ alors il est retiré de la recherche. Puisque $\delta R = 0$, cette condition peut être facilement écrite comme

$$D > D_R + \delta D \tag{3.11}$$

où (R, D_R) est le point se situant sur le segment reliant deux points d'enveloppe consécutifs $(R^\varepsilon, D^\varepsilon)$, $(R^{\varepsilon'}, D^{\varepsilon'})$ avec $R^\varepsilon \leqslant R \leqslant R^{\varepsilon'}$. Nous trouvons :

$$D_R = \alpha D^\varepsilon + (1 - \alpha) D^{\varepsilon'} \tag{3.12}$$

où $\alpha = (R^{\varepsilon'} - R)/(R^{\varepsilon'} - R^\varepsilon)$.

Si, pendant la recherche, un point $(R^h, D^h) \in \mathcal{R}$ est trouvé, alors le point optimal sera nécessairement situé dans un triangle plus petit $\{(R, D) \mid R \leqslant R_b \text{ et } D \leqslant D^h\}$. Ainsi, $\mathcal{R}$

peut être mis à jour et δD par voie de conséquence diminué (voir Fig. 3.3). Cette procédure améliore grandement l'efficacité de la recherche car de plus en plus de points seront décrétés morts et donc retirés.

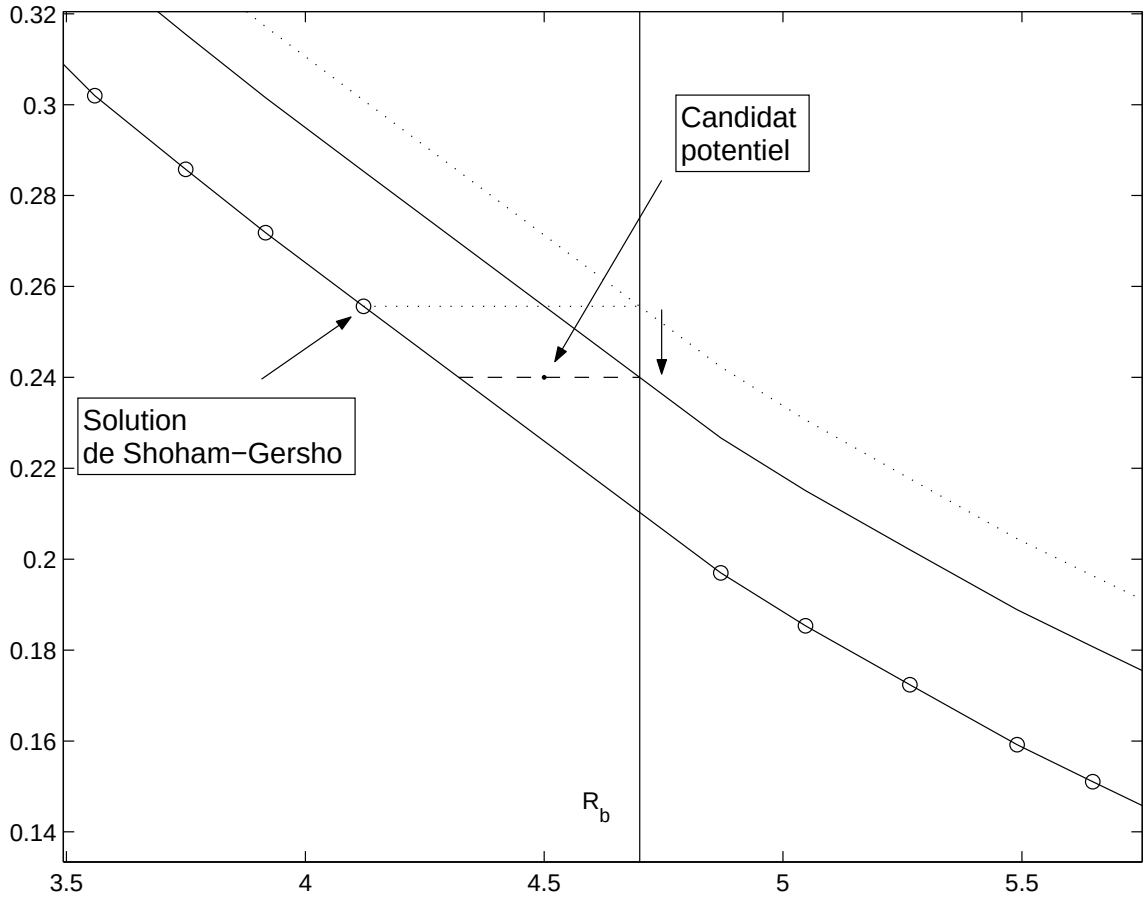


Fig. 3.3: Mise en évidence du triangle ∇_b^h et de l'impact d'un candidat potentiel sur la zone de recherche.

Afin d'obtenir un algorithme efficace, nous organisons la recherche par chemins convexes de telle sorte que $\mathcal{R}$ et δD soient fréquemment mis à jour. Au regard du théorème 3.2.1, il est facile de voir que n'importe quel point (R, D) peut être obtenu par un unique chemin convexe dont l'origine est un point d'enveloppe sur $\mathcal{H}$. Ainsi, nous commençons notre recherche depuis les points d'enveloppe sur $\mathcal{H}$ et nous remplaçons itérativement chaque point par ses successeurs qui n'appartiennent pas à l'enveloppe convexe $\mathcal{H}$ sur tous les chemins convexes possibles. Chaque successeur est trouvé par une unique opération élémentaire $\vec{\Delta}$, et on peut donc au maximum avoir N successeurs par point de départ. En pratique, ce nombre de successeurs potentiels est réduit par la présence dans le vecteur d'allocation $\underline{k}$ du point de départ de composantes déjà "saturées" (c'est à dire $k_i = M_i$) et par les contraintes de convexité du chemin passant par $\underline{k}$.

L'algorithme optimal de recherche par chemins convexes s'écrit finalement sous la forme de l'algorithme J.

Algorithme J Recherche optimale par chemins convexes

Cet algorithme sélectionne la solution optimale (R^*, D^*) au problème d'allocation de ressources sous la contrainte d'un budget R_b pour des fonctions $D_i(R_i)$ convexes.

1. *[Initialisation]*
   Initialiser $\mathcal{P}$ avec l'ensemble des points d'enveloppe de taux $R \leqslant R_b$.
   Initialiser (R^*, D^*) avec la solution de l'algorithme D.
   Initialiser δD avec $\delta D_{\min}$ donné par (3.10).
2. *[Boucle]*
   Remplacer chaque point dans $\mathcal{P}$ par ses successeurs de taux $R \leqslant R_b$.
   Aller à l'étape 4 si $\mathcal{P}$ est vide.
3. *[Mise à jour]*
   Soit (R^h, D^h) le point de plus faible distorsion dans $\mathcal{P}$.
   Si $D^h < D^*$, actualiser $\delta D \longmapsto \delta D - (D^* - D^h)$ et $(R^*, D^*) \longmapsto (R^h, D^h)$.
   Retirer les points de $\mathcal{P}$ satisfaisant à (3.11).
4. Retourner (R^*, D^*) si $\mathcal{P}$ est vide.
   Revenir à l'étape 2 sinon.

Cet algorithme peut être simplement adapté au problème de la courbe optimale R-D (trouver toutes les solutions optimales), en prenant pour $\mathcal{R}$ la réunion de tous les triangles cachés et fixer en conséquence $\delta D_{\min}$ comme la hauteur maximale de ces triangles. Malheureusement, l'algorithme ainsi utilisé se révèle être lent (bien que moins lent qu'un algorithme de programmation dynamique adapté au même problème).

Une amélioration de l'algorithme consiste à initialiser la solution courante avec le meilleur point sur un chemin convexe issu de la solution de l'algorithme I. Cela réduit couramment la région $\mathcal{S}_{\text{mort}}$ initiale, et ces chemins ne seront de toute manière plus à étudier. De manière plus générale, tout point caché pourrait initialiser l'algorithme afin de réduire la région de recherche.

À cause de l'implémentation particulière de la recherche, due au principe d'élimination de chemins convexes, il est difficile de donner une estimation de la complexité et du coût mémoire théoriques de l'algorithme J. Les performances de l'algorithme J seront évaluées par simulation sur les ensembles de courbes convexes du corpus de test : les résultats sont regroupés dans le paragraphe 3.5 de ce chapitre.

Par ailleurs, nous noterons que, bien qu'il faille nécessairement avoir des fonctions convexes pour appliquer l'algorithme J, on peut envisager de l'utiliser dans des cas de fonctions non convexes. En effet, employer la procédure sans modification quelconque de l'algorithme reste envisageable par application préalable d'une recherche directe de points d'enveloppe (RDPE) aux ensembles R_i-D_i de fonctions non-convexes. L'algorithme n'est bien entendu plus optimal dans ce cas, mais il pourrait être intéressant de s'en servir dans certains situations.

Dans le paragraphe 3.5 à venir, nous n'avons pas souhaité détailler les performances de cette version de la méthode (RDPE + recherche par chemins convexes) sur des fonctions non convexes (type NCU ou NCNU). Toutefois, nous retiendrons ce principe pour la suite de cette thèse. Ainsi, dans le chapitre 6, où l'optimalité n'est plus l'objectif principal, nous emploierons ce procédé sur une adaptation de notre algorithme J.

3.5 Performances

Nous avons effectué des simulations pratiques de l'algorithme J sur les différents ensembles de test suivants :
– CUT
– CNU

Nous avons pris $N = 16$ composantes pour l'ensemble des tests et $M_i = 25$ valeurs par composantes pour tous les ensembles de fonctions hormis CUT où $M_i = 13$. Les tests ont été réalisés pour une suite de 1000 budgets répartis sur l'ensemble de la dynamique de taux. Certains des algorithmes mentionnés dans le chapitre 1, et dont les performances ont déjà été testées sur les même fonctions et séquence de budgets, sont rappelés ici à titre de comparaison. Les résultats expérimentaux sont ainsi récapitulés dans les tableaux 3.1- 3.2, rangés par ensembles de fonctions.

TAB. 3.1: Performances des algorithmes pour l'ensemble CUT.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.03	51.8	2.48
G	3.8	100	0
$H_{(\Theta = 2)}$	0.6	36.8	4.40
$H_{(\Theta = 5)}$	2.0	95.5	0.27
I	0.03	72.5	0.56
J	4.9	100	0

TAB. 3.2: Performances des algorithmes pour l'ensemble CNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	13.3	5.1
G	112	100	0
$H_{(\Theta = 2)}$	2.5	7.6	14.0
$H_{(\Theta = 5)}$	17.5	50.0	6.0
I	0.07	23.3	3.1
J	16.4	100	0

Tout d'abord, que ce soit sur l'ensemble CUT ou sur l'ensemble CNU, nous vérifions bien l'optimalité de l'algorithme J. En trouvant la totalité des points optimaux, celui-ci permet effectivement de résoudre de manière optimale le problème d'allocation de ressources dans le cas de fonctions $D_i(R_i)$ convexes.

Nous remarquons que, pour l'ensemble CUT, l'algorithme G possède un temps de calcul légèrement inférieur à notre algorithme. Cela s'explique notamment par le fait que pour les fonctions à taux régulièrement espacés (CU, CUT et NCU) l'algorithme de programmation dynamique utilise avantageusement la régularité des taux. En effet durant les itérations, l'alignement vertical des points permet d'éliminer rapidement les candidats non optimaux. De son côté l'algorithme présenté dans ce chapitre ne profite pas directement de ce type d'alignement.

Concernant le test mené sur l'ensemble CNU, nous pouvons voir une différence majeure quant au temps de calcul entre la méthode de programmation dynamique et l'algorithme J. L'algorithme G est dans notre exemple, 8 fois moins rapide que l'algorithme J. Pour trouver un algorithme de même performance en temps, il faut regarder la version modifiée de la méthode de programmation dynamique avec un $\Theta = 5$. Mais ce dernier ne trouve que la moitié des points optimaux, et avec une perte en distortion encore moins bonne que celle de l'algorithme de Shoham-Gersho.

Ces résultats apportent donc une réflexion intéressante sur la relativité des performances de tel ou tel algorithme suivant le type de fonctions. Nous savions déjà que le caractère de convexité des fonctions permettait de prédire l'optimalité de certains algorithmes. Cette fois, nous voyons aussi que la rapidité des algorithmes est liée à la configuration du nuage dans le plan R-D, qui découle de la régularité des taux des fonctions $D_i(R_i)$. Ces configurations sont d'ailleurs bien visibles dans les différents graphiques présentés en annexe B.

L'algorithme J offre ainsi une alternative optimale à la méthode de programmation dynamique dans les cas de taux non réguliers.

3.6 Conclusions

Nous avons proposé dans ce chapitre un nouvel algorithme s'appliquant à des fonctions $D_i(R_i)$ convexes, mais par ailleurs générales, pour résoudre de manière optimale notre problème d'allocation de ressources.

Nous avons mis en évidence l'existence de chemins convexes, permettant de décrire de manière originale et unique un point du nuage global dans le plan R-D. Cette description offre de nombreuses propriétés qui ont été présentées et démontrées.

Nous avons ainsi mis en place un principe de sélection des chemins convexes dits admissibles au moyen d'un critère d'élimination des chemins non pertinents. C'est du parcours de ces

chemins convexes admissibles que découlent alors les points optimaux recherchés.

De plus, certaines priorités évoquées permettent à l'algorithme de procéder à une mise à jour de la zone de recherche au cours des itérations successives de parcours des chemins convexes. En effet, la zone de recherche des points optimaux potentiels étant bien localisée, nous avons pu développer des procédés de réduction de cette zone limitant notablement la complexité de l'algorithme.

La solution trouvée pour un budget donné est donc optimale et se trouve être atteinte dans des temps au pire équivalents à la méthode de programmation dynamique, et souvent meilleurs. Nous avons détaillé les performances de celui-ci comparativement aux autres algorithmes présentés.

Au vu des résultats obtenus par cette méthode, nous voyons pourquoi les concepts développés seront réutilisés au chapitre 6 dans des circonstances et un but différents. Lorsque notre objectif d'optimalité passe au second plan, nous verrons comment tirer pleinement parti du principe de recherche par chemins convexes développé à ce chapitre.

Toutefois, la limitation actuelle de l'optimalité de notre algorithme à des fonctions $D_i(R_i)$ convexes nous pousse à envisager d'autres méthodes. Le chapitre 4 suivant se concentre donc sur la description d'un algorithme répondant à notre problème d'allocation de ressources dans le cas le plus général de fonctions.

Chapitre 4

Recherche optimale par ordonnancement

Rien ne vaut la recherche
lorsqu'on veut trouver quelque chose.
John Ronald Reuel Tolkien

4.1 Introduction

Dans le chapitre précédent, nous avons résolu de manière optimale le problème d'allocation de ressources pour des fonctions $D_i(R_i)$ convexes. Toutefois, de nombreux points cachés apparaissent dans le plan R-D pour des fonctions $D_i(R_i)$ non-convexes, comme peuvent le souligner les figures B.5 et B.8.

Everett [Eve63] proposa en 1963 des moyens d'améliorer sa procédure, sans toutefois les investiguer. L'une d'elles consistait à rechercher des points possédant un lagrangien $\mathscr{L} = D + \lambda R$ sous optimal. Aucune méthode ou article publié depuis ne fonde son raisonnement sur cette idée pourtant originale. C'est cette idée qui va donc servir de base à ce chapitre. Nous développerons ainsi une nouvelle méthode totalement générale qui s'applique à un ensemble quelconque de fonctions $D_i(R_i)$.

Pour se faire, nous établirons une relation d'ordre qui nous permettra de décrire le nuage global R-D selon un ordre favorable à la recherche de points cachés. A l'instar du chapitre précédent, des considérations théoriques nous permettront de limiter la region d'investigation que cherchera à analyser l'algorithme. Ainsi, nous pourrons améliorer la complexité de la procédure de recherche pour la rendre efficace.

Au paragraphe 4.2, nous mettrons en place la nouvelle relation d'ordre total permettant de classer les points selon un tri adapté à notre recherche de points optimaux.

Nous en déduirons dans les paragraphe 4.3 et 4.4 quelques propriétés intéressantes et des améliorations pour l'algorithme qui en découlera au paragraphe 4.5 .

Ensuite, nous effectuerons, au paragraphe 4.6, un bilan sur les performances de l'algorithme développé en le comparant à celles des algorithmes de l'état de l'art.

Enfin, nous conclurons le chapitre au paragraphe 4.7 avec quelques remarques.

4.2 Définition d'une nouvelle relation d'ordre

Soit un multiplicateur λ donné, critique ou pas, de valeur positive ($0 < \lambda < \infty$). Considérons le couple $(D + \lambda R, D)$ formé des valeurs du lagrangien et de la distortion associés au point (R, D) et définissons la relation d'ordre total sur les points (R, D) du plan global de la manière suivante :

Définition 4.2.1
On définit la relation d'ordre $\prec$ s'écrivant sur les points du plan global comme l'ordre lexicographique sur les couples $(D + \lambda R, D)$:

$$(R, D) \prec (R', D') \Longleftrightarrow \begin{cases} D + \lambda R < D' + \lambda R' \\ \textit{ou} \\ D + \lambda R = D' + \lambda R' \textit{ et } D \leqslant D' \end{cases} \tag{4.1}$$

Notons que l'expression $\{D + \lambda R = D' + \lambda R' \text{ et } D \leqslant D'\}$ dans (4.1) est équivalente à l'expression $\{D + \lambda R = D' + \lambda R' \text{ et } R \geqslant R'\}$. Nous utiliserons par la suite les deux caractérisations sans distinction.

Cette relation d'ordre total classe les points (R, D) du plan global selon un ordre particulier qui met intuitivement en évidence les points des triangles cachés en premier. Ce classement va nous permettre de parcourir la liste pour sélectionner le point qui nous intéresse pour le problème considéré.

Observons un triangle caché ∇^h donné. Ce triangle est délimité par les deux points d'enveloppe consécutifs (R^e, D^e) et $(R^{e'}, D^{e'})$ (avec des taux tels que $R^e < R^{e'}$). Nous savons déjà que tout point optimal situé entre ces deux points d'enveloppe réside dans le triangle caché ∇^h correspondant. Soit Q_0 le point du plan global de valeur $(R^{e'}, D^e)$. Ce point fictif vérifie le théorème suivant :

Théorème 4.2.1
Tout point optimal P d'un triangle caché situé entre (R^e, D^e) *et* $(R^{e'}, D^{e'})$ *est tel que* $P \prec (R^{e'}, D^e)$.

Preuve : Soit un point optimal (R^*, D^*) situé entre (R^e, D^e) et $(R^{e'}, D^{e'})$. Nous avons $R^e < R^* < R^{e'}$ et $D^{e'} < D^* < D^e$. Donc $R^e + \lambda D^{e'} < R^* + \lambda D^* < R^{e'} + \lambda D^e$ d'où $(R^*, D^*) \prec Q_0$. ■

Lors de la recherche de points optimaux dans un triangle caché ∇^h donné, il ne sera donc pas nécessaire de trier les points au delà du point fictif Q_0 défini comme précédemment. D'autre part, nous remarquerons que lorsque nous trions l'ensemble des points du plan global R-D selon la relation d'ordre $\prec$ pour un multiplicateur critique λ donné, on trouve d'abord les deux points d'enveloppe correspondant à ce multiplicateur critique (puisque ceux-ci minimisent simultanément le lagrangien $\mathscr{L} = D + \lambda R$ pour cette valeur de λ). Ensuite, si nous nous intéressons uniquement au triangle ∇^h associé à ces deux points d'enveloppe, le premier point que nous trouvons à l'intérieur de ce triangle n'est pas un point d'enveloppe (ce qui est une évidence puisqu'il est contenu strictement dans le triangle caché). Par contre, ce point, qu'on notera P_1 dans la suite du paragraphe, vérifie le théorème suivant :

Théorème 4.2.2
Soit P_1 *le premier point trouvé dans le triangle caché* ∇^h *associé au lambda critique* λ *lors du tri des points globaux selon* $\prec$. *Alors* P_1 *est optimal.*

Preuve : Supposons qu'il existe un point $P = (R, D) \neq P_1 = (R^1, D^1)$ tel que $D \leqslant D^1$ et $R \leqslant R^1$. On a donc aussi $D + \lambda R \leqslant D^1 + \lambda R^1 \iff P \prec P_1$, ce qui est contraire au fait que P_1 est le premier point dans le triangle ∇^h. L'hypothèse initiale est par conséquent fausse. ■

Le point P_1 est un point caché du triangle ∇^h. On a bien une relation qui range les points du plan selon un ordre mettant en évidence les points optimaux. Notons que, dans l'absolu (i.e si on considère la totalité du plan global R-D et non pas la seule zone de recherche ∇^h), P_1 n'est pas nécessairement le premier point trouvé par la relation d'ordre définie. D'autres points avec un $D_i + \lambda R_i$ plus faible hors de ∇^h peuvent potentiellement exister. Certains de ces points peuvent même être optimaux, voire être des points d'enveloppe caractérisés par un multiplicateur critique différent. Ces points sont toutefois sans intérêt pour la résolution de notre problème dont la résolution consiste à trouver de points cachés dans le triangle ∇^h considéré. Nous omettrons donc ces points lors de notre recherche.

Nous avons représenté dans la figure 4.1 un tel point P_1 présent dans le triangle caché ∇^h. Dans cette figure, nous voyons la zone de recherche des points optimaux correspondant au triangle caché. Le point fictif Q_0 symbolise le 3^{eme} sommet du triangle caché déduit des points d'enveloppe. Nous avons hachuré la zone dans laquelle aucun point ne peut exister selon le le résultat du théorème 4.2.2.

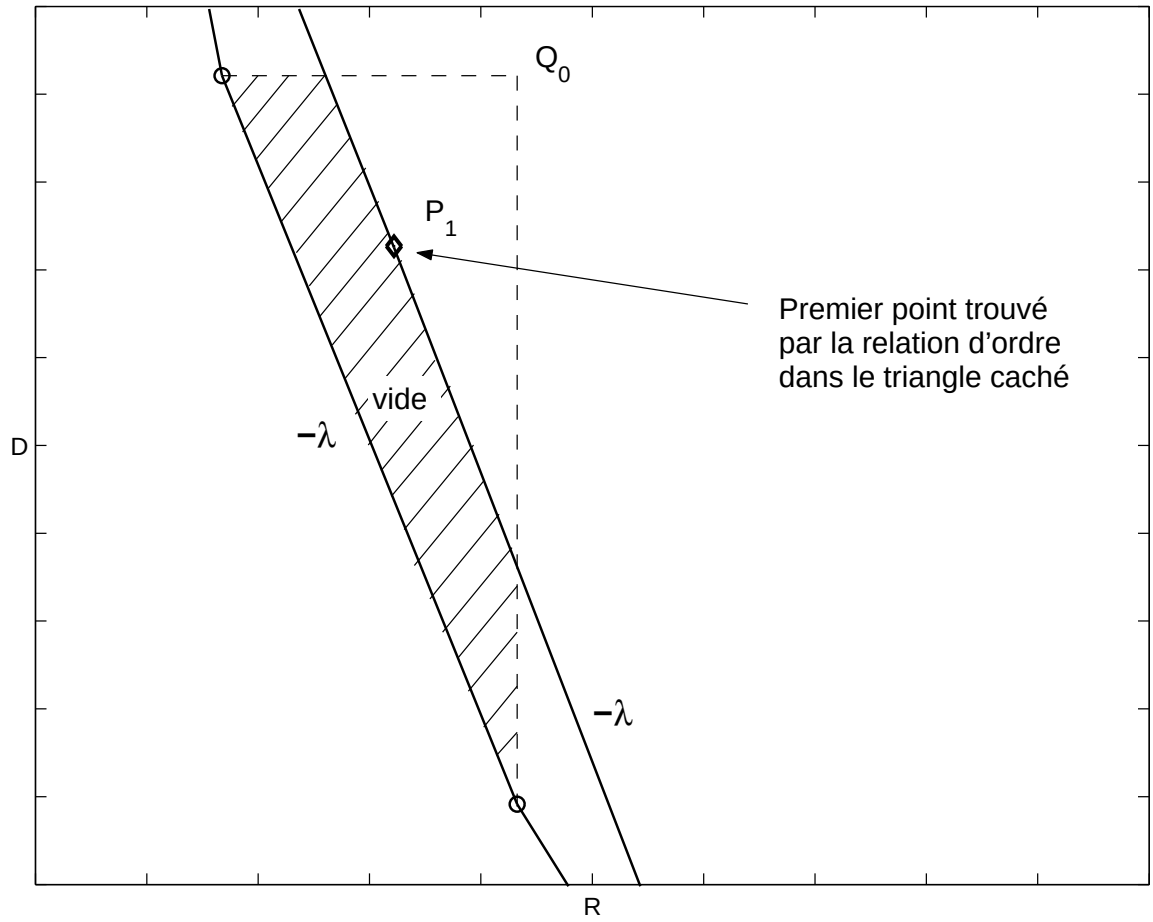


Fig. 4.1: Exemple de triangle caché dans lequel nous trouvons un point P_1, premier du triangle ∇^h par la relation d'ordre $\prec$

Nous allons maintenant pouvoir nous intéresser à la mise en œuvre de ce procédé de tri pour décrire un algorithme optimal efficace. Pour cela, nous allons devoir utiliser les propriétés de notre relation d'ordre, ainsi que les caractéristiques de la zone de recherche dont nous effectuerons une mise à jour dès lors que nous trouverons les premiers points cachés du triangle ∇^h concerné.

4.3 Principes de recherche de points optimaux

De la découverte d'un point optimal dans un triangle ∇^h par la relation $\prec$ découle une modification de la zone de recherche des points optimaux. En effet si on trouve par la relation d'ordre un point $P_1 = (R_1, D_1)$ dans le triangle caché, il est alors optimal (par le théorème 4.2.2). Dans ce cas, tout point $P = (R, D)$ appartenant à la région rectangulaire entre P_1 et Q_0 ne peut être optimal, puisque $R_1 < R$ et $D_1 < D$.

Ce résultat permet de réduire grandement la zone de recherche de points optimaux cachés. En effet, la zone de recherche exclut désormais la partie hachurée dans 4.1 (car déjà étudiée) et le rectangle entre Q_0 et P_1. Celle-ci se résume à la réunion de deux triangles rectangles. Ces traingles sont illustrés figure 4.2 et sont définis à partir des points Q_1 et Q_2 déduits de P_1.

Ces deux nouveaux points fictifs Q_1 et Q_2 reprennent le rôle de Q_0 dans leur triangle respectif. Ainsi, si $Q_0 = (R^0, D^0)$ et $P_1 = (R^1, D^1)$, les valeurs de ces points sont : $Q_1 = (R^1, D^0)$ et $Q_2 = (R^0, D^1)$. Les points cachés recherchés dans le triangle associé à Q_1 seront donc majorés (au sens de la relation d'ordre $\prec$) par Q_1 et les points cachés recherchés dans le triangle associé à Q_2 majorés par Q_2.

Nous noterons que la caractérisation du triangle initial de recherche de points optimaux $\{D < D^0, R < R^0, P \prec (R^0, D^0)\}$ peut simplement s'adapter à la caractérisation d'un des deux triangles en remplaçant R^0 par R^1 pour le triangle de Q_1 et et D^0 par D^1 pour le triangle de triangle de Q_2. Ainsi nous avons un procédé d'implémentation particulièrement simple et efficace. Au fur et à mesure que nous trouverons des points cachés $\{P_1, P_2, \ldots, P_j, \ldots\}$, nous définirons autant de nouveaux couples de points fictifs dont les taux et distortions dépendront des $\{R^j\}$ et $\{D^j\}$ et de leurs positions respectives. Nous aurons alors autant de triangles à étudier, la recherche se finissant lorsque le dernier triangle étudié est complètement parcouru.

Ainsi décrit, nous noterons que le principe de tri peut s'appliquer à la recherche de tous les points optimaux. Les zones de recherche sont alors les triangles cachés ∇^h successifs, caractérisés par des multiplicateurs critiques qui nous sont donnés par l'algorithme D. L'ensemble des points trouvés au final permet de résoudre le problème de la courbe optimale R-D.

Appliquée au problème d'allocation de ressource sous la contrainte d'un budget R_b, la méthode doit s'adapter à la zone de recherche spécifique. En effet, cette fois-ci, la zone de recherche de points optimaux n'est plus un triangle caché quelconque mais le triangle ∇_b^h. Le point Q_0

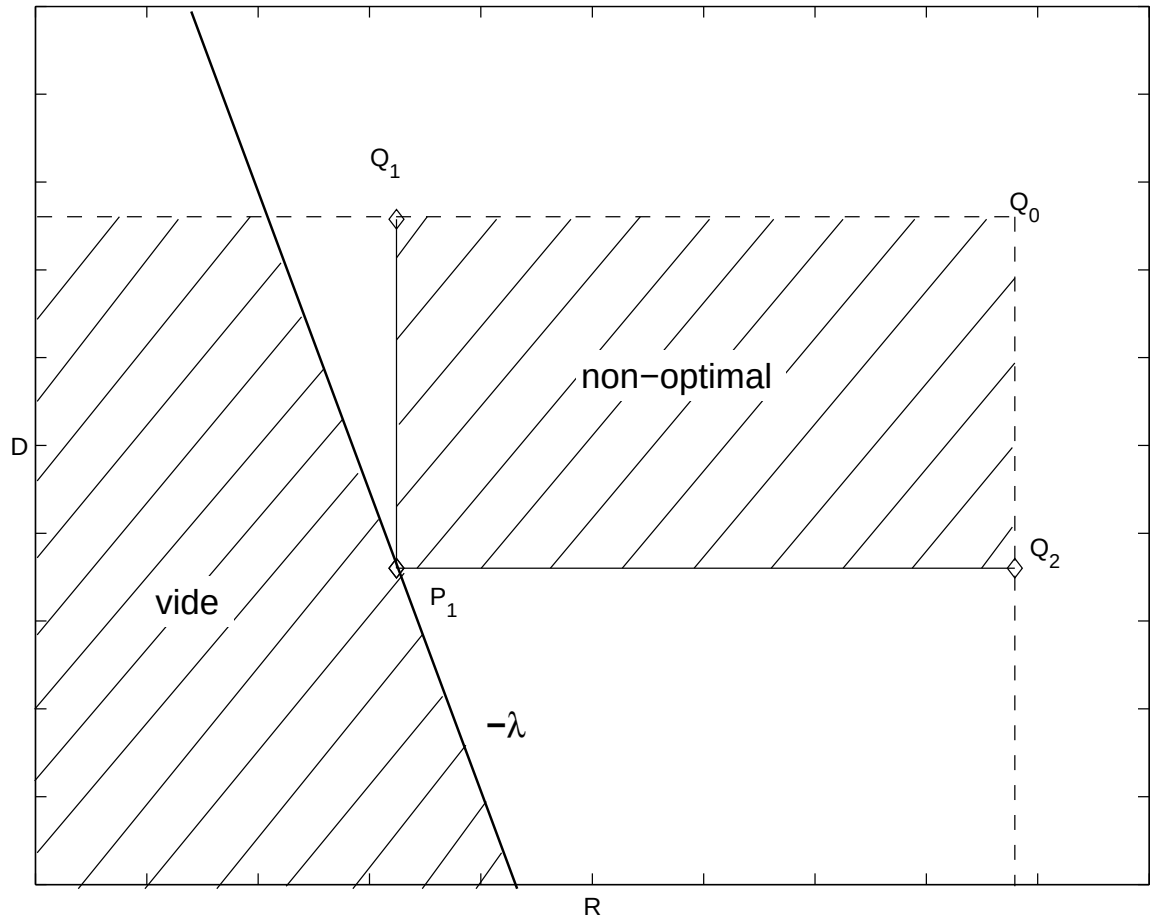


Fig. 4.2: Illustration de la modification de la zone de recherche des points optimaux lorsque P_1 est trouvé

est modifié en conséquence, son taux étant alors $R^0 = R_b$, sa distorsion restant inchangée. De plus, la mise à jour de la zone de recherche lors de la découverte d'un point caché P_0 s'effectue différemment puisque cette fois ci le triangle supérieur caractérisé par le point Q_1 n'a pas d'intérêt pour notre problème. Les seuls points éventuellement meilleurs que le point P_0 ont nécessairement une distorsion plus faible.

Nous pouvons de même améliorer la méthode par une considération simple sur les lagrangiens. A l'instar du taux majoré par le budget R_b, le lagrangien $\mathscr{L}^* = D^* + \lambda R^*$ d'un point optimal est borné par une valeur directement déduite de la relation $P^* \prec Q_0$. Ainsi, lors de la recherche de point optimaux par ordonnancement selon la relation $\prec$, nous pouvons ne pas tenir compte, dans chaque composante i, des points (R_i, D_i) tels que $D_i + \lambda R_i > D^0 + \lambda R^0$. Il est évident que de tels points ne peuvent générer des points globaux (R, D) tel que $D + \lambda R < D^0 + \lambda R^0$. En réduisant éventuellement le cardinal des ensembles des points (R_i, D_i) dans chaque composante, nous pouvons ainsi limiter dans certains cas la complexité du tri sur ces points. Nous pouvons d'ailleurs reproduire le procédé de majoration à chaque mise à jour de la zone de recherche.

Maintenant que nous savons efficacement sélectionner les points optimaux, nous devons désormais trouver un moyen simple et rapide de parcourir les points du plan global R-D dans l'ordre de la relation $\prec$.

4.4 Méthode efficace de parcours des points ordonnés du plan global

La procédure d'ordonnancement consiste à trier les points suivant la relation $\prec$. Cependant, trier directement les lagrangiens des points globaux $\mathscr{L}$ est impensable aux vues du cardinal du nuage R-D, engendrant une complexité résultante prohibitive. Nous devons donc trouver un moyen de traduire l'ordonnancement des points globaux sous une forme implémentable. Nous allons voir pour cela que nous pouvons ramener ce tri global sur les points du plan R-D à un tri par composantes grâce des considérations de linéarité.

Pour se faire, rangeons les lagrangiens $\mathscr{L}_i(k_i) = D_i(k_i) + \lambda R_i(k_i)$ par ordre croissant sur chaque composante en les indexant comme suit :

$$\mathscr{L}_i^1 < \mathscr{L}_i^2 < \ldots < \mathscr{L}_i^c < \ldots < \mathscr{L}_i^{M_i} \tag{4.2}$$

Nous obtenons N suites croissantes de lagrangiens $\{\mathscr{L}_i^c\}_c$. On peut voir aux figures 4.3 et 4.4 les lagrangiens de chaque composante suivant qu'ils sont rangés par taux croissants ou par valeurs croissantes. Les lagrangiens ainsi triés vont nous permettre de représenter les points dans un nouveau système de coordonnées.

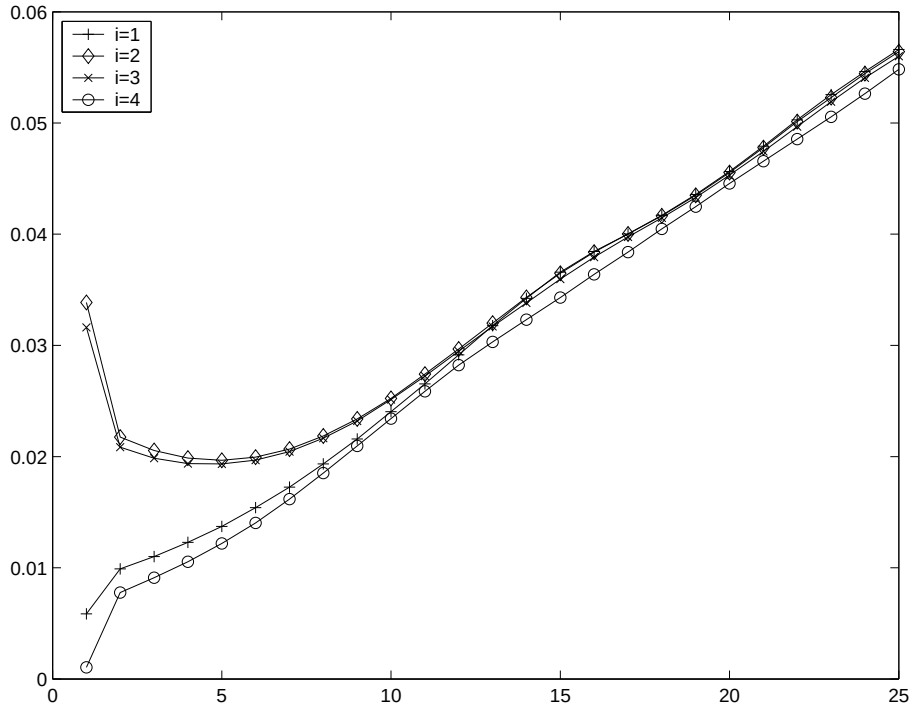


Fig. 4.3: Exemple de $N = 4$ composantes de $M = 25$ points dont les $\mathscr{L}_i(k_i) = D_i(k_i) + \lambda R_i(k_i)$ sont représentés rangés par k_i croissants.

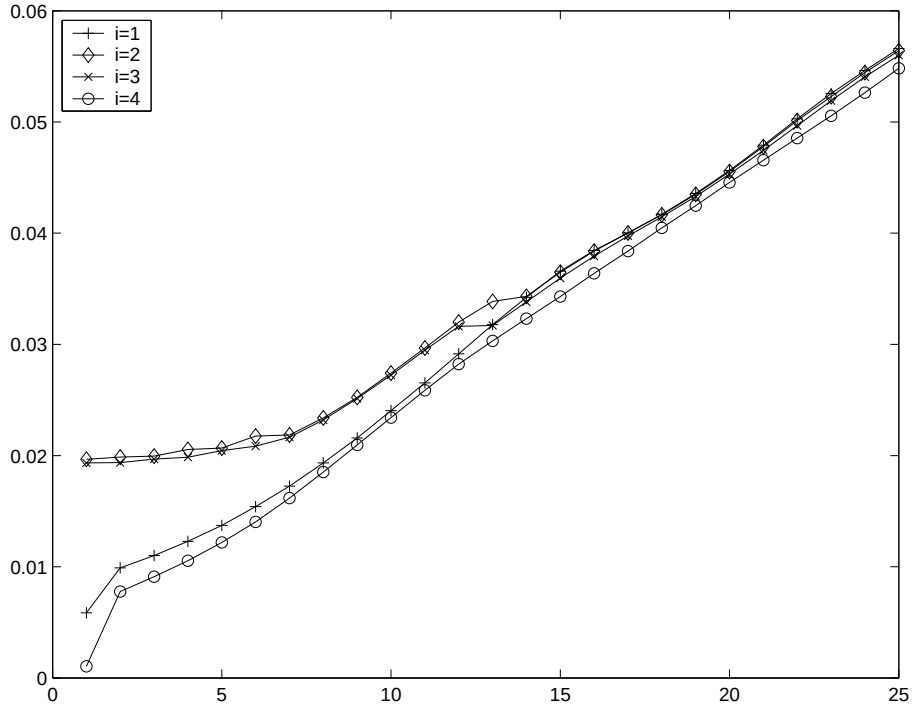


Fig. 4.4: Exemple de $N = 4$ composantes de $M = 25$ points dont les $\mathscr{L}_i(k_i) = D_i(k_i) + \lambda R_i(k_i)$ sont représentés rangés par $\mathscr{L}_i$ croissants.

Définition 4.4.1
Soit un indice $c_i \in \{1, \ldots, M_i\}$. On appelle coordonnée lagrangienne c_i du point (R_i, D_i), la valeur pour laquelle $\mathscr{L}_i = D_i + \lambda R_i = \mathscr{L}_i^{c_i}$.
On définit le vecteur $\underline{c} = [c_1, \ldots, c_N]$ comme le vecteur des coordonnées lagrangiennes du point $(R = \sum R_i, D = \sum D_i)$ du plan global.
En particulier, on notera le lagrangien de ce point : $\mathscr{L}^{\underline{c}} = \sum \mathscr{L}_i^{c_i}$.

Ainsi, lorsque c_i croît, $\mathscr{L}_i^{c_i}$ augmente. En particulier, le point d'enveloppe associé au multiplicateur λ a pour coordonnées lagrangiennes $\underline{c} = [1, \ldots, 1]$. En effet, le lagrangien $\mathscr{L}^e$ du point d'enveloppe $\underline{k}^e$ associé à λ est :

$$\mathscr{L}^e = \min_{\underline{k}}(\mathscr{L}(\underline{k})) = \min_{\underline{k}}(\sum_{i=1}^{N} \mathscr{L}_i(k_i)) = \sum_{i=1}^{N} \min_{k_i}(\mathscr{L}_i(k_i)) = \sum_{i=1}^{N} \mathscr{L}_i^1 \tag{4.3}$$

Considérons deux points P et P' du plan global tels que $P \prec P'$ et leurs coordonnées lagrangiennes respectives $\underline{c}$ et $\underline{c}'$. Si ces deux points se succèdent immédiatement pour la relation d'ordre $\prec$, alors nous pouvons affirmer que $\nexists P'' \notin \{P, P'\}$, vérifiant $P \prec P'' \prec P'$, c'est à dire vérifiant $\mathscr{L} \leqslant \mathscr{L}'' \leqslant \mathscr{L}'$. Nous en déduisons le résultat sur les lagrangiens suivant :

$$\begin{aligned}
\mathscr{L}' - \mathscr{L} &= \min_{\mathscr{L}'' > \mathscr{L}} (\mathscr{L}'' - \mathscr{L}) && (4.4)\\
&= \min_{\mathscr{L}'' > \mathscr{L}} \left(\sum_i \mathscr{L}_i'' - \sum_i \mathscr{L}_i \right) && (4.5)\\
&= \min_{\mathscr{L}'' > \mathscr{L}} \left(\sum_i (\mathscr{L}_i'' - \mathscr{L}_i) \right) && (4.6)\\
&= \sum_i \min_{\mathscr{L}'' > \mathscr{L}} (\mathscr{L}_i'' - \mathscr{L}_i) && (4.7)\\
&= \min_{\mathscr{L}_j'' > \mathscr{L}_j} (\mathscr{L}_j'' - \mathscr{L}_j) + \left(\sum_{i \neq j} 0 \right) && (4.8)\\
&= \min_{c_j'' > c_j} (\mathscr{L}_j^{c_j''} - \mathscr{L}_j^{c_j}) && (4.9)\\
&= \mathscr{L}_j^{c_j+1} - \mathscr{L}_j^{c_j} && (4.10)\\
&= \min_i (\mathscr{L}_i^{c_i+1} - \mathscr{L}_i^{c_i}) && (4.11)
\end{aligned}$$

où $j = \operatorname{Arg\,min}_i(\mathscr{L}_i'' - \mathscr{L}_i |\ \mathscr{L}_i'' > \mathscr{L}_i) = \operatorname{Arg\,min}_i(\mathscr{L}_i^{c_i+1} - \mathscr{L}_i^{c_i})$ est la composante qui introduit la plus faible croissance de lagrangien.

Parcourir les points du plan global dans l'ordre (de $\prec$) revient donc à sélectionner les points

dans l'ordre des lagrangiens $\mathscr{L}$ d'augmentation minimale , c'est à dire dans l'ordre des différences $\Delta\mathscr{L}_i^{c_i} = \mathscr{L}_i^{c_i+1} - \mathscr{L}_i^{c_i}$ minimales. On peut voir à la figure 4.5 une illustration de différences $\Delta\mathscr{L}_i^{c_i}$ définies à partir de l'exemple de la figure 4.3.

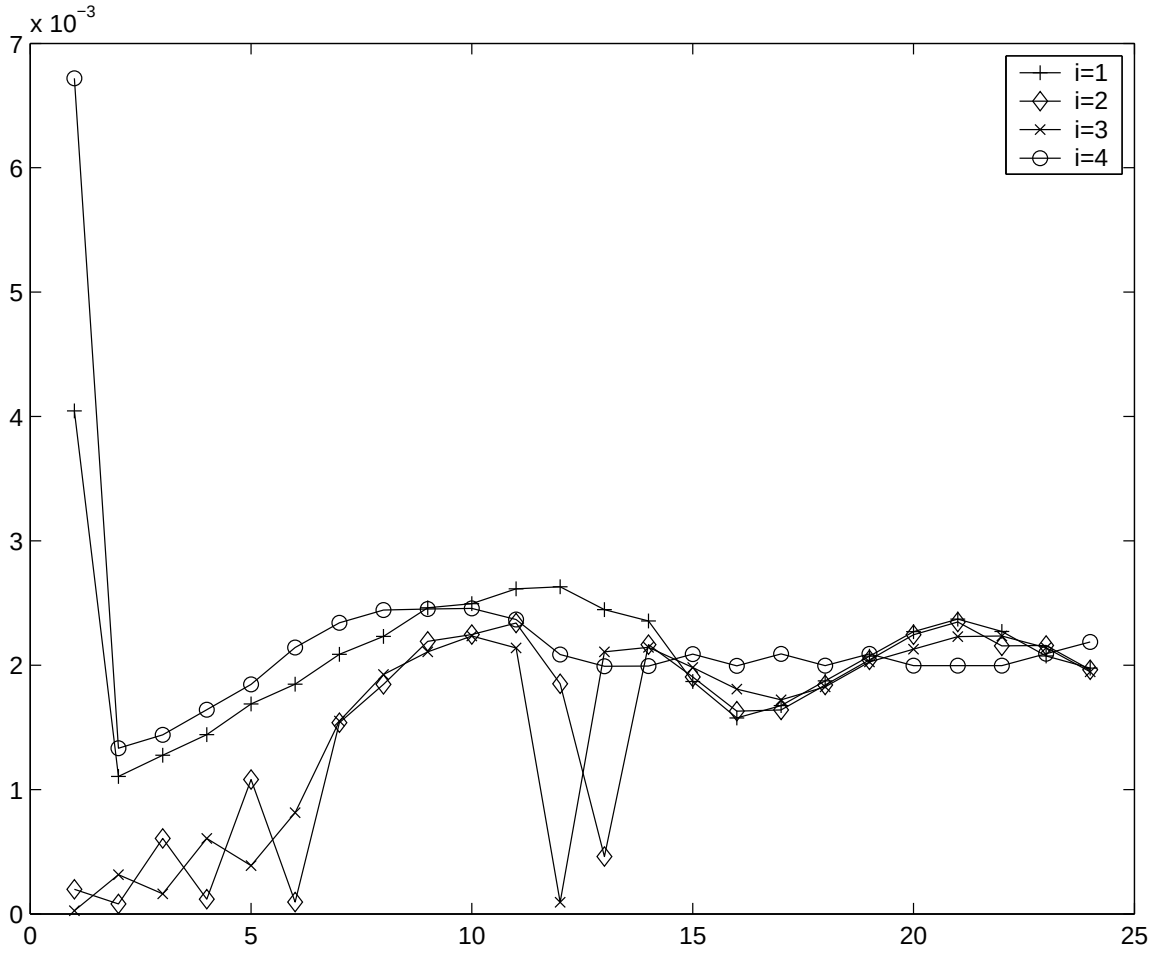


Fig. 4.5: Représentation des $\Delta\mathscr{L}_i^{c_i}$ par c_i croissants.

Nous observons que la sélection du point suivant (au sens de la relation d'ordre $\prec$) se résume à la recherche d'un minimum parmi N valeurs. À un instant donné, $\underline{c}$ étant le vecteur des coordonnées lagrangiennes du point courant, le prochain point sélectionné aura pour coordonnées lagrangiennes le vecteur $\underline{c}' = [c_1, \ldots, c_j + 1, \ldots, c_N]$ où le minimum de $\Delta\mathscr{L}_i^{c_i+1}$ est atteint pour la j^{eme} composante.

À l'issue des constatations des paragraphes précédents, nous pouvons décrire une procédure efficace de résolution du problème d'allocation de ressources sous la contrainte d'un budget R_b.

4.5 Description d'un algorithme optimal de recherche par ordonnancement

En appliquant la méthode de parcours des points du plan global et les techniques de caractérisation de l'optimalité et de mise à jour de la zone de recherche que nous avons développées, on obtient l'algorithme de recherche optimal K. Nous noterons que cet algorithme ne fait pas d'hypothèses sur les caractéristiques des fonctions sur lesquelles il s'applique : nous décrivons un algorithme adapté à tout type de courbes et de taux.

Algorithme K Recherche optimale par ordonnancement

Cet algorithme sélectionne la solution optimale (R^*, D^*) au problème d'allocation de ressources sous la contrainte d'un budget R_b.

1. *[Initialisation]*
   Initialiser (R^*, D^*) et calculer le multiplicateur critique λ correspondant au budget R_b avec l'algorithme D.
   Calculer $\mathscr{L}_i = D_i + \lambda R_i, \forall i$
   Ranger par ordre croissant, pour chaque i, les $\mathscr{L}_i < D^* + \lambda R_b$ pour obtenir $\{\mathscr{L}_i^c\}_c$.
   Calculer $\{\Delta\mathscr{L}_i^c\}_c$.
   Initialiser $\underline{c} = [1, \ldots, 1]$.
2. *[Boucle]*
   Sélectionner la composante i pour laquelle $\Delta\mathscr{L}_i^{c_i+1}$ est minimal.
3. *[Mise à jour]*
   Actualiser $c_i \longmapsto c_i + 1$ et calculer le point (R, D) de coordonnées lagrangiennes $\underline{c}$.
   Si $D < D^*$ et $R < R_b$, actualiser $(R^*, D^*) \longmapsto (R, D)$.
4. Retourner (R^*, D^*) si $(R_b, D^*) \prec (R, D)$.
   Revenir à l'étape 2 sinon.

L'algorithme K résout de manière optimale le problème d'allocation de ressource. A titre indicatif, il est à noter que le coût mémoire de cette méthode est particulièrement faible, puisque parmi les éléments conservés nous avons essentiellement les N listes $\{\Delta\mathscr{L}_i^c\}_c$.

4.6 Performances

Nous avons effectué des simulations pratiques de l'algorithme K sur les différents ensembles de test suivants :

– CUT
– CNU
– NCU
– NCNU

Nous avons pris $N = 16$ composantes pour l'ensemble des tests et $M_i = 25$ valeurs par composantes pour tous les ensembles de fonctions hormis CUT où $M_i = 13$. Les tests ont été réalisés pour une suite de 1000 budgets répartis sur l'ensemble de la dynamique de taux. Certains des algorithmes mentionnés dans le chapitre 1, et dont les performances ont déjà été testées sur les même fonctions et séquence de budgets, sont rappelés ici à titre de comparaison. Les résultats expérimentaux sont ainsi récapitulés dans les tableaux 4.1- 4.4, rangés par ensembles de fonctions.

Tab. 4.1: Performances des algorithmes pour l'ensemble CUT.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.03	51.8	2.48
G	3.8	100	0
$H_{(\Theta = 2)}$	0.6	36.8	4.40
$H_{(\Theta = 5)}$	2.0	95.5	0.27
I	0.03	72.5	0.56
K	4.0	100	0

Tab. 4.2: Performances des algorithmes pour l'ensemble CNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	13.3	5.1
G	112	100	0
$H_{(\Theta = 2)}$	2.5	7.6	14.0
$H_{(\Theta = 5)}$	17.5	50.0	6.0
I	0.07	23.3	3.1
K	5.2	100	0

Tab. 4.3: Performances des algorithmes pour l'ensemble NCU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	49.9	2.84
G	7.7	100	0
$H_{(\Theta = 2)}$	0.36	3.5	11.8
$H_{(\Theta = 5)}$	1.52	56.4	2.69
I	0.07	7.7	8.6
K	35.7	100	0

TAB. 4.4: Performances des algorithmes pour l'ensemble NCNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	0.3	6.89
G	71.8	100	0
$H_{(\Theta = 2)}$	1.9	1.3	19.7
$H_{(\Theta = 5)}$	11.6	33.3	8.84
I	0.07	3.7	13.7
K	0.16	100	0

Comme nous l'avions prévu, nous pouvons voir que l'algorithme K que nous avons décrit est optimal. Quel que soit le type de fonctions $D_i(R_i)$ utilisé, nous obtenons l'ensemble des points optimaux.

Dans le cas des fonctions de type CUT, nous retrouvons le même constat qu'au chapitre précédent : le temps de calcul des algorithmes K et G sont similaires. Comme déjà évoqué, l'algorithme de programmation dynamique profite de la régularité des taux, ce qui lui confère un avantage par rapport à notre algorithme K.

Ensuite, pour les fonctions de type CNU, nous remarquons que l'absence de régularité des taux qui pénalise l'algorithme G n'a pas d'effet sur l'algorithme décrit dans ce chapitre. Nous obtenons un temps de calcul 22 fois inférieur à la méthode de programmation dynamique. Nous voyons que la méthode hybride (algorithme H), dont le temps de calcul correspond a peu près à celui de l'algorithme K, trouve en comparison très peu de points optimaux.

Concernant les fonctions non-convexes à taux régulièrement espacés (NCU), le temps de calcul de notre algorithme est relativement mauvais. Ainsi, il est 5 fois moins rapide que l'algorithme de programmation dynamique. Encore une fois, la régularité des taux joue en faveur de l'algorithme G, ce qui n'est pas le cas pour notre méthode. Toutefois, nous nous devons de soulever un détail à propos d'un élément nullement exprimé par les tableaux récapitulatifs, mais qui est clairement ressorti des simulations effectuées. Si le temps de calcul de l'algorithme G est relativement stable quelque soit le budget, celui de l'algorithme K est lui très variable sur des ensembles à taux réguliers : il dépend énormément de la position du budget par rapport au triangle caché dans lequel il se situe. Ainsi, certaines résolutions sont instantanées alors que d'autres prennent beaucoup plus de temps, le temps de calcul *moyen* n'étant pas forcément représentatif pour tout budget. À titre d'exemple, l'écart-type des temps de calcul de l'algorithme K sur l'ensemble NCU est de 382, celui-ci descendant à 63 en retirant *seulement* 2 échantillons particuliers parmi les 1000 obtenus. Le temps de calcul moyen passe alors de 35.7 à une valeur de 12.8.

Ce comportement, difficile à prévoir, peut poser problème occasionnellement, mais pourrait être évité par une considération sur la régularité des taux. En effet, étant donné que ce phénomène est exclusivement présent pour des ensembles à taux régulièrement espacés, nous pourrions profiter de la connaissance de cette régularité pour améliorer notre recherche. Or, jusqu'ici,

les budgets tests étaient choisis indépendamment des taux étudiés. La régularité sur les composantes induisant des taux globaux uniformément espacés d'un pas donné connu, le budget pourrait alors être ramené au taux global directement inférieur, ce qui rendrait l'algorithme K indépendant de cet aléa.

Nous remarquerons pour finir le résultat spectaculaire obtenu par l'algorithme K sur l'ensemble de fonctions de type NCNU. Dans ce cas précis, la procédure de reduction de la zone de recherche est très bien adaptée à la situation (car de nombreux points optimaux sont situés dans le triangle caché), ce qui permet d'atteindre un temps de calcul moyen très faible.

Nous pouvons en déduire que la rapidité de l'algorithme réside dans l'efficacité de cette procédure de réduction de la zone de recherche, et donc dans la densité de points optimaux présents dans les triangles cachés. Plus l'algorithme trouve de points cachés, plus il opère vite. Cela explique en particulier la différence importante de performance de l'algorithme K suivant que nous l'appliquons à des ensembles à taux réguliers ou pas. En effet, dans le cas de fonctions à taux régulièrement espacés, les points globaux se retrouvent alignés sur les taux globaux eux-même réguliers, et la recherche par ordonnancement ne profite pas au mieux des considérations faites au paragraphe 4.4.

Pour les fonctions de type NCNU, l'algorithme D, seulement 4 fois plus rapide que notre algorithme K, trouve par contre 300 fois moins de points optimaux. Nous pouvons ainsi conclure que dans les cas les moins favorables à l'algorithme de Shoham-Gersho, notre algorithme offre une réelle plus value.

4.7 Conclusions

Nous avons proposé dans ce chapitre un algorithme totalement novateur. Applicable dans la totalité des cas possibles de fonctions (convexes ou non) et de taux (réguliers ou pas), cette méthode atteint l'optimalité et résout le problème d'allocation de ressources avec une complexité souvent plus faible que celle des méthodes existantes de performance équivalente.

La recherche consiste à réordonner le nuage global suivant une loi relative au lagrangien. Réputée très efficace, l'utilisation du lagrangien permettait déjà d'offrir des méthodes de recherche de points de l'enveloppe convexe telles que présentées au chapitre 2. La relation d'ordre total présentée n'était donc qu'un prolongement naturel aux raisonnements habituels.

A l'instar du chapitre précédent, la méthode développée s'est appuyée sur une actualisation de la zone d'investigation afin de limiter la durée de recherche. Encore une fois, la connaissance précise de la localisation potentielle des solutions optimales a pu apporter des priorités efficacement exploitables.

Les performances résultantes ont été analysées. Ainsi, l'optimalité de la méthode s'adjoint

dans certains cas d'un temps de calcul remarquablement faible et s'adapte en particulier très bien aux ensembles de fonctions à taux irrégulièrement espacés.

Malgré tout, nous voyons au travers de cette méthode, ainsi que de celle du chapitre précédent, qu'il faut généralement payer cher l'optimalité. L'approche consistant à considérer l'optimalité comme primordiale paraît alors illusoire, ou tout au moins trop ambitieuse par rapport aux contraintes diverses d'une implémentation pratique. Il peut s'avérer plus intéressant et efficace de sélectionner des points sous-optimaux dont la recherche est bien moins coûteuse en temps. Si nous arrivons à atteindre à moindre de coût des performances proches des méthodes optimales, alors l'optimalité deviendra accessoire.

C'est pourquoi les deux chapitres suivants proposent des méthodes sous optimales d'implémentation moins lourde et capables d'atteindre des performances intéressantes en terme de perte en distortion.

Chapitre 5

Recherche sous-optimale par omission d'index

Apprécier la paresse est impossible,
à moins d'avoir beaucoup de travail.
Anonyme

5.1 Introduction

Jusqu'ici, nous avons exploré différentes méthodes afin d'atteindre l'optimalité, que ce soit dans des cas les plus généraux possibles ou dans des conditions de fonctions particulières. Nous avons, par cette démarche, favorisé la performance en terme de pourcentage de points optimaux trouvés, laissant à la complexité le rôle de critère comparatif. Il est toutefois envisageable d'adopter une approche différente plus pragmatique. Elle consiste cette fois à s'intéresser à l'atteinte de solutions sous-optimales à moindre coût, c'est à dire, à considérer un compromis entre complexité et performance. Le taux d'optimalité aura alors un rôle annexe non déterminant et sera simplement consulté comme moyen de comparaison en cas de performances proches de certains algorithmes. Le critère fondamental de comparaison sera la perte en distorsion qui nous donnera une mesure de la *proximité* de notre solution avec la solution optimale pour le budget considéré.

Bien que rarement quantifié dans la littérature, le compromis complexité/performance a déjà été évoqué. Il reste actuellement l'une des préoccupations sous-jacentes majeures. Nombres d'algorithmes, tels ceux inspirés de l'algorithme I ou [YOR96], s'y intéressent et proposent des alternatives aux méthodes optimales de complexité plus grande. Les algorithmes fondés sur une approche lagrangienne offre entre autre une piste intéressante d'amélioration de ce compromis. Notre but dans ce chapitre va être de *densifier* à moindre coût l'ensemble des points optimaux

trouvés à partir de l'algorithme de Shoham-Gersho. Nous pourrons ainsi fournir des solutions, bien que non-optimales, *proches* de l'optimalité par le biais d'algorithmes de faible complexité.

Le présent chapitre découle d'un constat simple édicté au théorème 1.5.5 : l'enveloppe convexe du nuage global dépend uniquement de l'ensemble des points d'enveloppe de chaque composante. En modifiant un point d'enveloppe d'une de ces composantes, on altère l'enveloppe convexe dans le plan global R-D. Notre but sera donc d'omettre des points donnés dans des composantes afin d'atteindre simplement des solutions potentielles à notre problème d'allocation de ressources.

Nous commencerons le chapitre par le paragraphe 5.2 où nous présenterons ce principe d'omission d'index.

La description de l'enveloppe convexe issue de cette omission fera l'objet du paragraphe 5.3 suivant. En découlera une procédure efficace utilisant le principe et permettant de densifier localement notre connaissance du nuage global pour des fonctions $D_i(R_i)$ convexes.

Cette procédure servira de fondement à un nouvel algorithme que nous développerons amplement au paragraphe 5.4. Ainsi, nous obtiendrons un moyen rapide pour améliorer la solution proposée par l'algorithme de Shoham-Gersho.

Nous ferons enfin au paragraphe 5.5 un bilan des performances de cet algorithme et nous le comparerons à celles des algorithmes déjà cités au chapitre 2. Nous verrons alors l'intérêt de cette méthode.

Nous finirons notre chapitre par quelques conclusions au paragraphe 5.6.

5.2 Définition de l'opération d'omission d'index

Considérons un ensemble de fonctions $D_i(R_i)$ convexes quelconque. Prenons une composante i et son ensemble d'indexation $\mathcal{I}_i$ (identifiable à $\{1, \ldots, M_i\}$). Nous appelons opération d'omission $\mathcal{E}_\kappa$ la fonction décrite ainsi :

Définition 5.2.1
On définit $\mathcal{E}_\kappa$ comme l'opération qui consiste à retirer un index κ de l'ensemble $\mathcal{I}_i$:

$$\mathcal{E}_\kappa : \mathcal{I}_i \longmapsto \mathcal{I}_i \setminus \{\kappa\} \tag{5.1}$$

Appliquer $\mathcal{E}_\kappa$ à une composante i consiste à retirer le point d'index κ de l'ensemble R_i-D_i, et donc à réduire de un le choix des allocations disponibles ($M_i \longmapsto M_i - 1$). Cette modification est toutefois locale, puisque les autres allocations de cette composante n'en sont nullement modifiées. La figure 5.1 illustre un exemple d'omission d'index sur une composante donnée avec $M_i = 10$ et $\kappa = 6$. Les points de la courbe complète sont marqués par des croix et ceux

de la courbe tronquée sont représentés par des ronds.

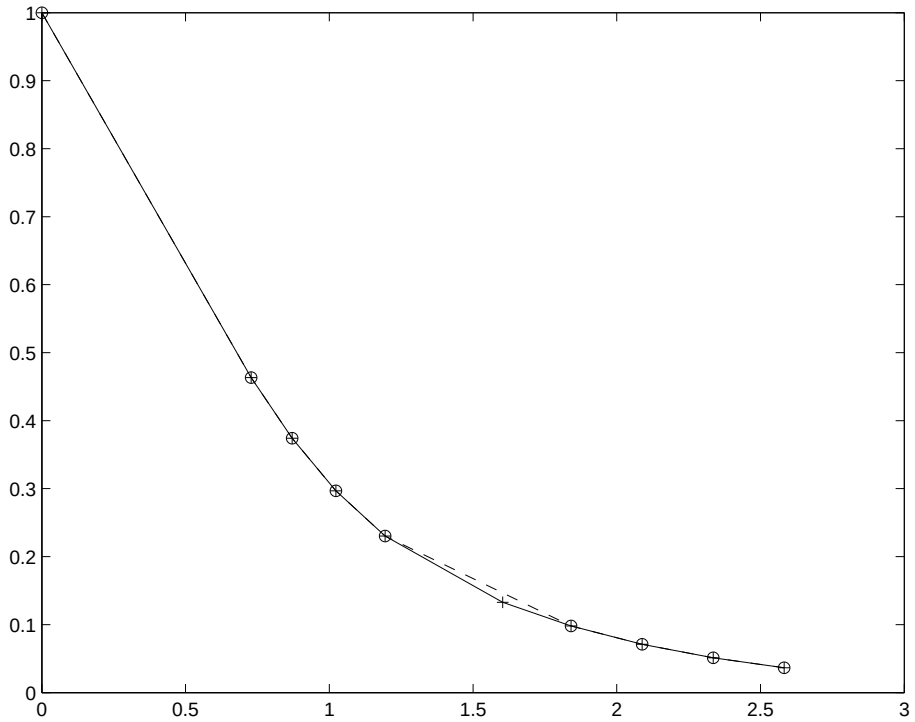


Fig. 5.1: Exemple de composante avec la courbe complète (trait plein) et la courbe tronquée (en pointillés)

Cette opération $\mathcal{E}_\kappa$ a pour conséquence de modifier l'ensemble des multiplicateurs critiques $\{\lambda_i\} = \{\lambda_i(1), \ldots, \lambda_i(M_i - 1)\}$ d'une composante. Si $\kappa = 1$ (resp. M_i), on retire simplement la valeur $\lambda_i(1)$ (resp. $\lambda_i(M_i - 1)$) de cet ensemble. Si $1 < \kappa < M_i$, on remplace $\lambda_i(\kappa)$ et $\lambda_i(\kappa + 1)$ dans cet ensemble par un nouveau multiplicateur $\lambda^{\mathcal{E}}$:

$$\lambda^{\mathcal{E}} = \frac{D_i(\kappa - 1) - D_i(\kappa + 1)}{R_i(\kappa + 1) - R_i(\kappa - 1)} \tag{5.2}$$

Notons qu'étant donné la convexité des fonctions $D_i(R_i)$, $\lambda^{\mathcal{E}}$ vérifie :

$$\lambda_i(\kappa) < \lambda^{\mathcal{E}} < \lambda_i(\kappa + 1) \tag{5.3}$$

La liste globale des multiplicateurs critiques subit ce changement et l'énumération des valeurs croissantes de λ est donc localement modifiée. Cette liste modifiée caractérise une nouvelle enveloppe convexe telle que la définit l'algorithme D. De par le caractère local de la modification, la nouvelle enveloppe convexe calculée coïncidera en partie avec l'ancienne (voir Fig. 5.2- 5.3), la zone de bifurcation se situant entre $\lambda_i(\kappa)$ et $\lambda_i(\kappa+1)$. Les cercles représentent les nouveaux points, alors que les anciens sont matérialisés par des croix.

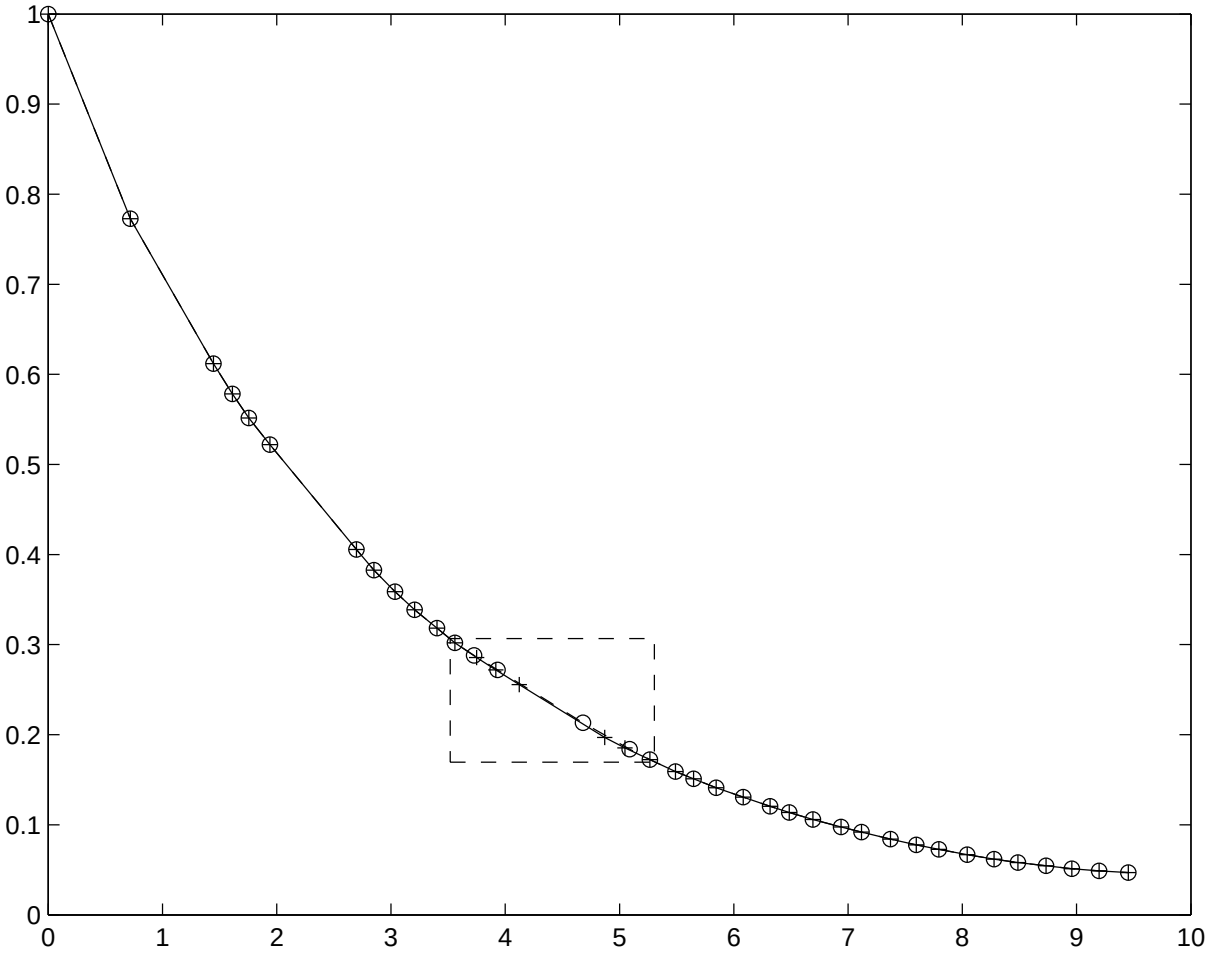


Fig. 5.2: Exemple d'enveloppe convexe (pour $N = 4$ et $M_i = 10$) avec la courbe normale (trait plein) et la version tronquée (en pointillés) après application d'une opération $\mathcal{E}_\kappa$ sur l'une des composantes (voir Fig. 5.1).

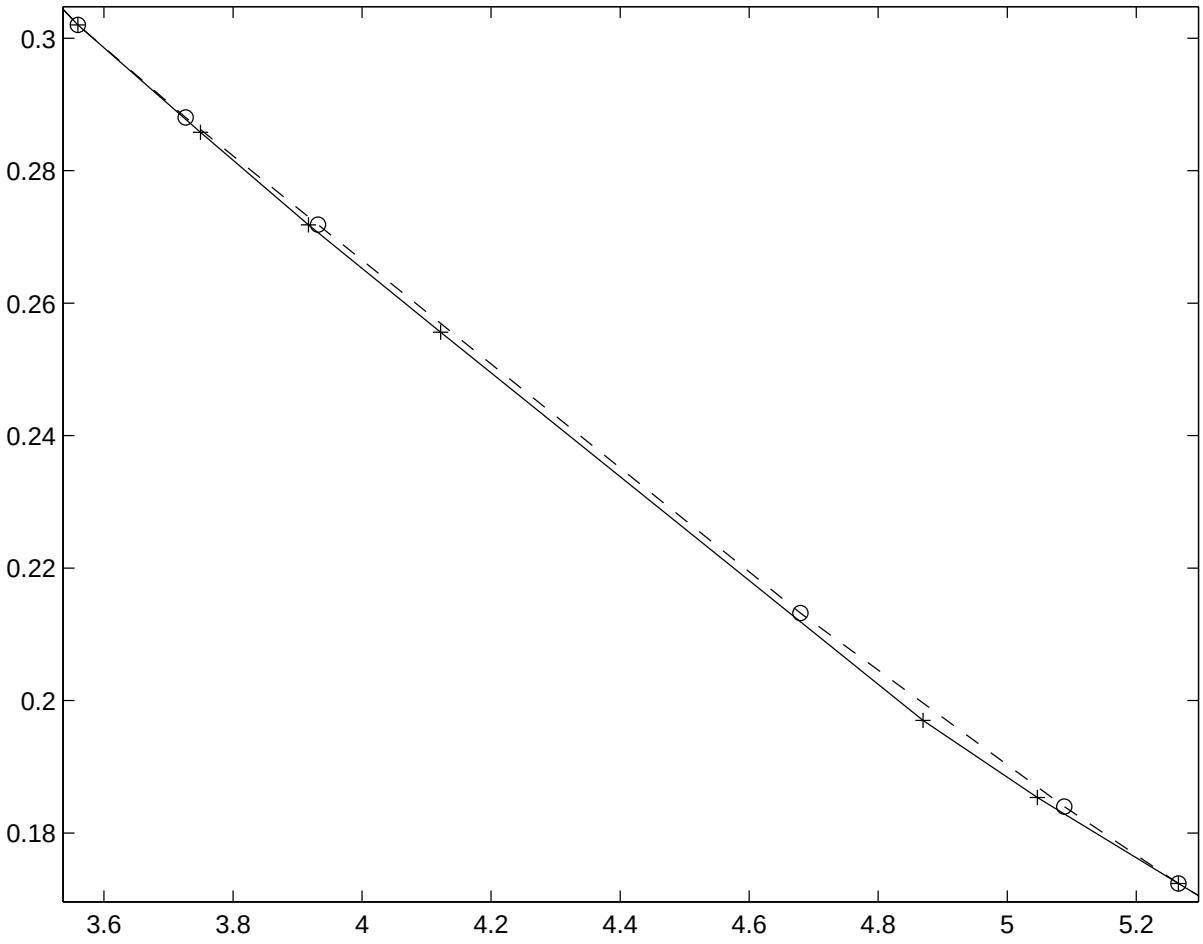


Fig. 5.3: Zoom de la figure 5.2

La nouvelle enveloppe convexe se situe bien entendu au dessus de l'ancienne. Nous noterons cependant dans la figure 5.3 la proximité des deux courbes et le fait que les points de la nouvelle enveloppe convexe sont situés dans les triangles ∇^h de l'enveloppe initiale. Cette remarque essentielle est le fondement qui motive notre raisonnement et la construction de notre nouvel algorithme.

5.3 Nouvelle enveloppe convexe par omission d'index

Nous avons vu que lorsqu'on applique une opération d'omission d'index $\mathcal{E}_\kappa$ sur une composante i, on modifie le nuage global ainsi que l'enveloppe convexe correspondante. Déduire la nouvelle enveloppe convexe de l'ancienne est cependant très simple et quasiment immédiat. On décrira l'enveloppe convexe $\mathcal{H}$ initiale en utilisant les vecteurs d'allocation des points d'enveloppe rangés par taux croissants, et donc par multiplicateurs critiques correspondants décroissants :

$$\mathcal{H} = \begin{bmatrix} \vdots & & \vdots & & \vdots \\ k_1^e & \dots & \kappa - 1 & \dots & k_N^e \\ k_1^e & \dots & \kappa & \dots & k_N^e \\ \vdots & & \vdots & & \vdots \\ k_1^{e'} & \dots & \kappa & \dots & k_N^{e'} \\ \vdots & & & & \vdots \\ k_1^{e''} & \dots & \kappa & \dots & k_N^{e''} \\ k_1^{e''} & \dots & \kappa + 1 & \dots & k_N^{e''} \\ \vdots & & \vdots & & \vdots \end{bmatrix} \tag{5.4}$$

où les notations utilisées sont les suivantes :

- $\underline{k}^e$ correspond au multiplicateur critique $\lambda_i(\kappa)$, c'est à dire la dernière allocation pour laquelle $k_i = \kappa - 1$,
- $\underline{k}^{e'}$ correspond au multiplicateur critique vérifiant $\min(\lambda > \lambda^{\mathcal{E}})$,
- $\underline{k}^{e''}$ correspond au multiplicateur critique $\lambda_i(\kappa + 1)$, c'est à dire la dernière allocation pour laquelle $k_i = \kappa$.

On notera que l'application d'une opération d'omission d'index $\mathcal{E}_\kappa$ sur une composante i modifiera la liste des multiplicateurs critiques uniquement entre les valeurs $\lambda_i(\kappa)$ et $\lambda_i(\kappa + 1)$ et donc la nouvelle enveloppe convexe $\mathcal{H}'$ sera différente de $\mathcal{H}$ seulement entre les vecteurs d'allocations $\underline{k}^e$ et $\underline{k}^{e''}$. Le reste de $\mathcal{H}'$ sera identique à $\mathcal{H}$ et ne nécessite aucun calcul supplémentaire.

Pour obtenir la liste des allocations des points d'enveloppe après application d'une omission

d'index, nous utiliserons par conséquent une procédure simple et rapide décrite dans l'algorithme L.

Algorithme L Nouvelle enveloppe convexe par omission d'index

Cet algorithme retourne l'ensemble des allocations $\{\underline{k}\}$ de l'enveloppe convexe déduite par application d'une opération $\mathcal{E}_\kappa$ à une composante i.

1. *[Initialisation]*
   Initialiser $\{\underline{k}\}$ avec l'ensemble des allocations des points d'enveloppe rangées par taux croissants.
   Calculer le $\lambda^{\mathcal{E}}$ correspondant à $\mathcal{E}_\kappa$.
   Trouver dans $\{\underline{k}\}$ les positions respectives (voir (5.4)) des allocations :
   – $\underline{k}^e$ correspondant à $\lambda_i(\kappa) : p^e$,
   – $\underline{k}^{e'}$ correspondant à $min(\lambda > \lambda^{\mathcal{E}}) : p^{e'}$,
   – $\underline{k}^{e''}$ correspondant à $\lambda_i(\kappa+1) : p^{e''}$.
2. *[Mise à jour]*
   Remplacer dans $\{\underline{k}\}$:
   – κ par $\kappa - 1$ de $p^e + 1$ à $p^{e'}$
   – κ par $\kappa + 1$ de $p^{e'} + 1$ à $p^{e''}$
   Retirer l'allocation en position $p^{e''}$.
   Insérer $[k_1^{e'}, \ldots, k_{i-1}^{e'}, \kappa+1, k_{i+1}^{e'}, \ldots, k_N^{e'}]$ entre les positions $p^{e'}$ et $p^{e'}+1$.
   Retirer l'allocation en position $p^e + 1$.
3. Retourner l'ensemble des allocations $\{\underline{k}\}$.

Cette nouvelle enveloppe convexe énumère un ensemble de points potentiellement cachés dans le plan global R-D, c'est à dire des points situés dans les triangles cachés définis par l'enveloppe convexe initiale. Il n'est pas possible de prévoir le comportement de la nouvelle enveloppe convexe, mais sa proximité avec l'ancienne enveloppe convexe permet d'envisager ces points comme solution pertinente à notre problème d'allocation de ressources. Nous avons ainsi un moyen rapide de *densifier* l'ensemble des allocations optimales et sous-optimales à notre disposition et de fournir une solution efficace au problème d'allocation de ressource. D'autre part la complexité de cet algorithme est très faible car il s'agit essentiellement du réarrangement d'une liste de multiplicateurs critiques.

5.4 Description d'un algorithme sous-optimal de recherche par omission d'index

L'algorithme L s'applique sur une unique composante i par l'intermédiaire de l'opération d'omission d'index $\mathcal{E}_\kappa$. En appliquant l'opération $\mathcal{E}_\kappa$ sur chacune des composantes i avec des

κ différents quelconques pour chacune d'elle, on obtient autant d'enveloppes convexes. Ces enveloppes diffèrent de l'enveloppe convexe initiale dans une région définie par les points d'enveloppe dont les multiplicateurs critiques correspondent aux points retirés par $\mathcal{E}_\kappa$. Si nous appliquons $\mathcal{E}_\kappa$ à un point d'enveloppe $\underline{k}^e$ en prenant successivement $\kappa = k_i^e$ avec $i = \{1, \ldots, N\}$, nous génèrons alors un ensemble de points autour de la zone de $\underline{k}^e$. En particulier, nous trouvons un certain nombre de points dans le triangle caché situé entre $\underline{k}^e$ et son successeur.

Ainsi, lorsque que nous recherchons une solution au problème d'allocation de ressources pour un budget R_b, nous nous intéressons à un triangle particulier défini par un point d'enveloppe $\underline{k}^e$ et son successeur (le triangle ∇_b^h), et l'application successive de $\mathcal{E}_{k_i^e}$ sur chaque composante i permet de générer un certain nombre de points cachés, et donc de solutions potentielles. Il ne restera plus qu'à faire un tri sur l'ensemble de ces points dont le nombre est relativement restreint. Nous pouvons donc décrire une procédure efficace de recherche sous la forme de l'algorithme M.

Algorithme M Recherche sous-optimale par omission d'index

Cet algorithme retourne une solution au problème d'allocation de ressources sous la contrainte d'un budget R_b pour des fonctions $D_i(R_i)$ convexes.

1. *[Initialisation]*
   Appliquer l'algorithme D pour calculer l'ensemble de l'enveloppe convexe.
   Initialiser la liste $\mathcal{L}$ avec la solution de Shoham-Gersho.
2. *[Boucle]*
   Appliquer l'algorithme L pour chaque composante i.
3. *[Mise à jour]*
   Ajouter les points trouvés de taux $R \leqslant R_b$ à la liste $\mathcal{L}$.
4. Retourner le point de $\mathcal{L}$ de distorsion minimale.

De par son principe de recherche, cet algorithme peut très bien retourner la même solution que celle de Shoham-Gersho. En effet, il n'y a même aucune garantie que les nouvelles enveloppes convexes passe par le triangle où se situent des points candidats. Toutefois, nous verrons au travers des résultats présentés au paragraphe 5.5 qu'en pratique l'intérêt de cette méthode est avéré et qu'elle offre une amélioration et une alternative convaincante des algorithmes efficace encore utilisés aujourd'hui.

D'autre part, il n'existe aucun moyen de garantir l'optimalité d'une telle méthode. Même si les points trouvés soient situés dans le triangle adéquat, aucune propriété ne permet d'affirmer qu'ils sont optimaux, et il peut très bien exister des points meilleurs. Cependant, nous remarquerons encore en pratique que le procédé d'omission employé propose des solutions proches de l'optimalité.

Nous noterons qu'il est possible d'appliquer le procédé d'omission d'index à plusieurs reprises à des rangs supérieurs. En effet, nous pouvons appliquer l'algorithme L à chaque nouvelle enveloppe convexe générée. Nous décrivons ainsi des courbes de plus en plus loin de l'enveloppe convexe initiale, mais les points, pour certains, peuvent cependant toujours être des points cachés, voire des points optimaux. On peut jouer de cette manière sur le compromis complexité/efficacité de la recherche.

Toutefois, il est difficile de déterminer en pratique l'impact de cette récurrence et le gain de performance que l'on en retire par rapport au temps de calcul supplémentaire que ce procédé implique. Nous n'avons donc pas souhaité inclure de résultats correspondant à cette méthode dans le paragraphe suivant.

5.5 Performances

Nous avons effectué des simulations pratiques de l'algorithme M sur les ensembles de test suivants :
– CUT
– CNU

Nous avons pris $N = 16$ composantes pour l'ensemble des tests et $M_i = 25$ valeurs par composantes pour tous les ensembles de fonctions hormis CUT où $M_i = 13$. Les tests ont été réalisés pour une suite de 1000 budgets répartis sur l'ensemble de la dynamique de taux. Certains des algorithmes mentionnés dans le chapitre 1, et dont les performances ont déjà été testées sur les même fonctions et séquence de budgets, sont rappelés ici à titre de comparaison. Les résultats expérimentaux sont ainsi récapitulés dans les tableaux 5.1- 5.2, rangés par ensembles de fonctions.

TAB. 5.1: Performances des algorithmes pour l'ensemble CUT.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.03	51.8	2.48
G	3.8	100	0
$H_{(\Theta = 2)}$	0.6	36.8	4.40
$H_{(\Theta = 5)}$	2.0	95.5	0.27
I	0.03	72.5	0.56
M	0.09	79.4	0.44

Tout d'abord, nous remarquons que l'algorithme M possède un temps de calcul moyen 3 à 4 fois supérieur à celui de l'algorithme de Shoham-Gersho ou de la procédure avare. Ce temps reste toutefois très inférieur à celui des autres méthodes (optimales ou pas).

Tab. 5.2: Performances des algorithmes pour l'ensemble CNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	13.3	5.1
G	112	100	0
$H_{(\Theta = 2)}$	2.5	7.6	14.0
$H_{(\Theta = 5)}$	17.5	50.0	6.0
I	0.07	23.3	3.1
M	0.19	32.7	2.6

Dans le cas des fonctions de type CUT, l'algorithme de recherche par omission d'index apporte 28% de points optimaux supplémentaires par rapport à l'algorithme de Shoham-Gersho. Pour les fonctions de type CNU, l'algorithme M trouve 19% de points optimaux de plus que l'algorithme D. Nous voyons donc que, malgré l'absence de garantie d'optimalité de la procédure, celle-ci offre effectivement une plus value conséquente.

En terme de perte en distortion, le gain de notre algorithme est relativement important : celui-ci est divisé par 5 pour les fonctions de type CUT, et par 2 pour les fonctions de type CNU, par rapport à l'algorithme de Shoham-Gersho.

Au final, l'algorithme M s'approche en performance de la procédure avare, en proposant un équilibre légèrement différent dans le compromis "temps de calcul"/"perte en distortion" : en prenant un peu plus de temps, notre algorithme trouve un peu plus de points optimaux.

5.6 Conclusions

Dans ce chapitre, nous avons présenté un nouvel algorithme pour répondre au problème d'allocation de ressources dans le cas de fonctions $D_i(R_i)$ convexes. Bien que non optimal, l'algorithme apporte une alternative intéressante aux procédures existantes.

Le principe, basé sur l'omission de points spécifiques dans des composantes R_i-D_i, a permis de mettre en lumière des enveloppes convexes possédant des points dans la zone d'intérêt des solutions optimales. Ces points sont alors triés pour conserver uniquement les éléments pertinents.

Cette méthode a l'avantage de limiter grandement les calculs effectués, l'algorithme consistant essentiellement à recalculer des portions d'enveloppes convexes. Un autre intérêt de cette méthode réside dans la possibilité de réitérer le principe à des rangs supérieurs afin d'améliorer le résultat. On obtient alors un algorithme pouvant éventuellement s'adapter à la géométrie du problème.

Cet algorithme, bien que non-optimal, approche en performance, c'est à dire en perte en

distortion, les algorithmes optimaux. La complexité atteinte est elle relativement faible, à la mesure de nos attentes. Nous n'avions pourtant aucune garantie du niveau de performance de l'algorithme, que seule l'application du principe au travers de tests est venu confirmer.

On notera que la limitation à des fonctions $D_i(R_i)$ convexes restreint toutefois la portée de cette méthode. Nous allons donc aborder au chapitre suivant une adaptation de la méthode présentée au chapitre 3 pour laquelle nous apporterons des modifications permettant son implémentation efficace et couvrant tous les cas de fonctions.

Chapitre 6

Recherche sous-optimale par chemins convexes

C'est une perfection de n'aspirer point à être parfait.
Fénelon

6.1 Introduction

Les performances évoquées au paragraphe 3.5 du chapitre 3 nous ont permis d'évaluer l'efficacité de la procédure de recherche par chemins convexes en terme d'optimalité. Cette méthode, bien plus rapide que l'algorithme G, est toutefois encore relativement lourde. En effet, les chemins convexes sont très nombreux, et si le nombre N de composantes est grand, on peut rapidement se retrouver débordé : le principe d'élimination appliqué conserve encore un nombre de chemins convexes relativement élevé.

Nous avons donc intérêt à limiter les chemins étudiés si l'on souhaite réduire ce temps de calcul. À l'instar d'une procédure avare, on peut expérimentalement vérifier que les chemins convexes issus de la solution de Shoham-Gersho contiennent des candidats dignes d'attention. Nous allons par conséquent adapter l'algorithme J à partir de cette constatation, en ne considérant que les sous-chemins convexes issus du point d'enveloppe solution de l'algorithme D. En limitant les points de départ de l'algorithme à un nombre faible d'allocations, nous allons ainsi réduire considérablement la complexité de notre algorithme tout en conservant un niveau de performance intéressant en terme de perte en distorsion.

Nous commencerons le chapitre par le paragraphe 6.2 où nous détaillerons des considérations liées à notre adaptation de l'algorithme initial J.

Nous décrirons l'algorithme résultant au paragraphe 6.3. Nous en profiterons pour adapter notre procédure afin de l'appliquer à un ensemble quelconque de fonctions $D_i(R_i)$.

Nous ferons ensuite au paragraphe 6.4 un bilan des performances de ce nouvel algorithme et les comparerons à celles des algorithmes déjà cités au chapitre 2.

Nous finirons notre chapitre par quelques conclusions au paragraphe 6.5.

6.2 Considérations pratiques d'adaptation de la recherche par chemins convexes

Prenons le problème d'allocation de ressources contraint par un budget R_b où les fonctions $D_i(R_i)$ sont convexes. Comme mentionné précédemment, nous allons modifier l'algorithme J pour ne parcourir que les chemins convexes issus de la solution de Shoham-Gersho pour le budget R_b. Dans la procédure décrite au chapitre 3, il faut alors modifier l'étape d'initialisation en ne prenant dans $\mathcal{P}$ que le point solution de l'algorithme D.

Ce choix de restreindre les points de départ à un unique vecteur d'allocation nous permet d'aller encore plus loin, car la zone d'étude s'en trouve immédiatement très réduite. Nous savons que les points candidats à la résolution du problème sont forcément situés dans le triangle ∇_b^h. La zone de recherche de points cachés $\mathcal{R}$ initiale correspond donc ici à ∇_b^h. Or, notre algorithme va successivement appliquer des opérations élémentaires $\vec{\Delta}$ en partant du point $(R(\underline{k}^{\mathrm{SG}}), D(\underline{k}^{\mathrm{SG}}))$ solution de Shoham-Gersho. On n'aura aucun intérêt à appliquer des opérations élémentaires $\vec{\Delta}$ telle que $\vec{\Delta}(\underline{k}^{\mathrm{SG}}) = \underline{k}' \notin \nabla_b^h$: celles-ci ne sont donc pas à considérer pour la construction des chemins convexes employés dans l'algorithme.

Considérons les quantités R_Δ et D_Δ suivantes :

$$R_\Delta = R_b - R(\underline{k}^{\mathrm{SG}}) \tag{6.1}$$
$$D_\Delta = D(\underline{k}^{\mathrm{SG}}) - D_{\mathrm{virt}} \tag{6.2}$$

où D_{virt} est défini par (1.29).

Les opérations élémentaires qui sont intéressantes doivent être telles que les ΔR et ΔD correspondants vérifient :

$$\Delta R \leqslant R_\Delta \qquad \Delta D \leqslant D_\Delta \tag{6.3}$$

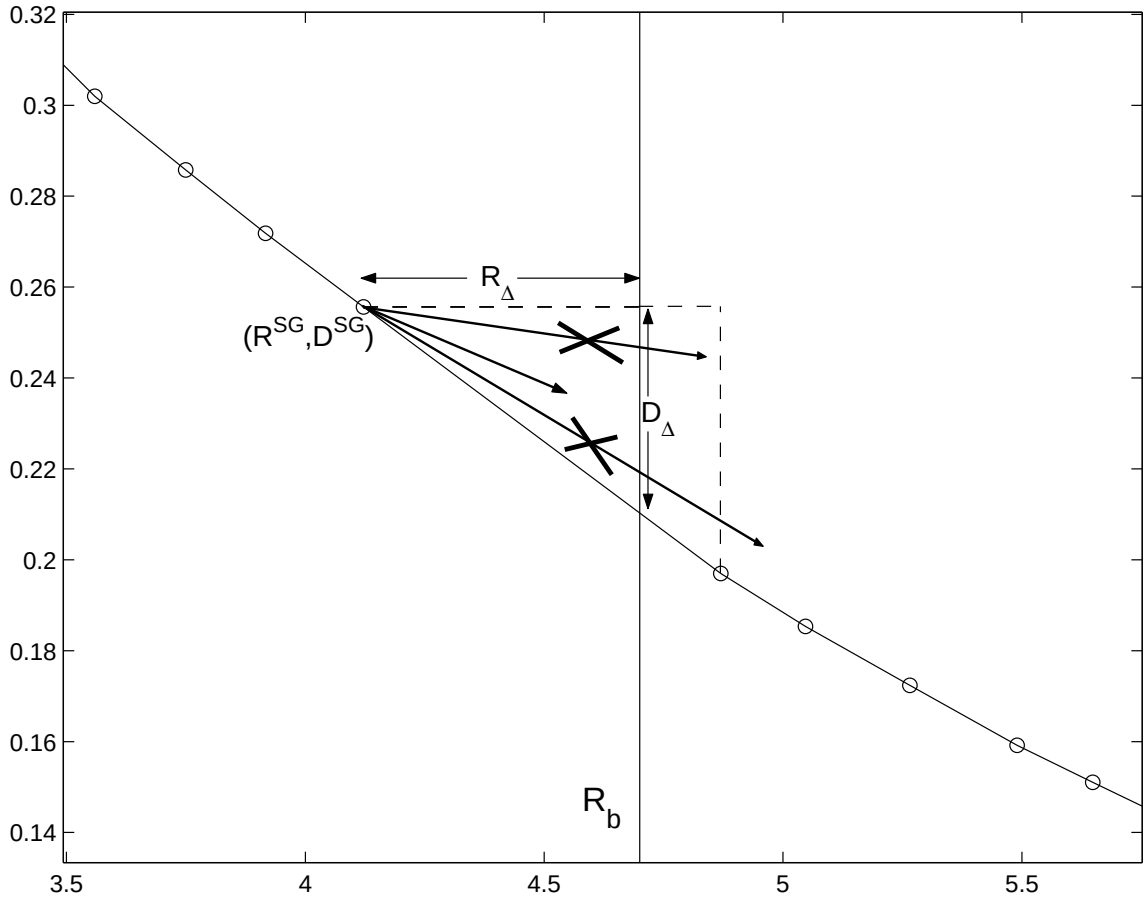


Fig. 6.1: Illustration des opérations élémentaires $\vec{\Delta}$ admissibles et non-admissibles

On peut voir un exemple d'opérations élémentaires $\vec{\Delta}$ admissibles et non-admissibles à la figure 6.1. L'opération à ne pas considérer a été barrée. On note que la condition sur les distorsions est en fait superflue. En effet, puisque l'on ne peut pas trouver de points au dessous de l'enveloppe convexe, si une opération élémentaire est telle que $\Delta D > D_\Delta$, on a aussi $\Delta R > R_\Delta$. On peut donc se restreindre à la seule condition sur les taux pour l'admissibilité d'une opération élémentaire.

Étendons maintenant le raisonnement à l'application d'une séquence d'opérations élémentaires $\{\vec{\Delta}^1, \ldots, \vec{\Delta}^j, \ldots\}$ (indexées à l'exposant) correspondant aux valeurs $\{\Delta R^1, \ldots, \Delta R^j, \ldots\}$ et $\{\Delta D^1, \ldots, \Delta D^j, \ldots\}$ associées. Ainsi, pour obtenir des points dans ∇_b^h, le même principe que précédemment nous permet de nous intéresser uniquement aux séquences d'opérations élémentaires vérifiant :

$$\sum_j \Delta R^j \leqslant R_\Delta \tag{6.4}$$

On peut interpréter ce résultat en exprimant l'équation précédente dans chaque composante i. En effet, chaque opération $\vec{\Delta}^j \equiv \vec{\Delta}_{\lambda_i}$ correspond à la descente d'un unique multiplicateur critique λ_i dans une unique composante i donnée. Le principe des chemins convexes fait donc que la séquence d'opérations élémentaires $\{\vec{\Delta}^1, \ldots, \vec{\Delta}^j, \ldots\}$ correspond à la descente d'un certain nombre de multiplicateurs critiques qui se succèdent dans leurs composantes respectives. Si tous les $\vec{\Delta}^j$ se trouve être associées à des multiplicateurs critiques de la même composante i, on obtient que les $\sum_j \Delta R^j$ et $\sum_j \Delta D^j$ s'expriment comme suit :

$$\sum_j \Delta R^j = R_i(k_i') - R_i(k_i) \qquad \sum_j \Delta D^j = D_i(k_i) - D_i(k_i') \tag{6.5}$$

où k_i et k_i' sont les points entre lesquels résident les multiplicateurs critiques dans la composante i.

D'où le résultat :

Lemme 6.2.1
Soit $\underline{k}^{SG}$ l'allocation de la solution de Shoham-Gersho pour le budget R_b. Toute séquence d'opérations élémentaires $\{\vec{\Delta}_{k_i}, \vec{\Delta}_{k_i+1}, \ldots, \vec{\Delta}_{(k_i+j-1)}\}$ appliquée à $\underline{k}^{SG}$ vérifiant :

$$R(k_i + j) - R(k_i) > R_\Delta \tag{6.6}$$

ne donnera pas un point dans le triangle ∇_b^h.

Par définition d'un chemin convexe, un $\vec{\Delta}_{k_i}$ ne peut être appliqué dans un chemin convexe avant un $\vec{\Delta}_{k_{(i-1)}}$ pour une composante i donnée. Par conséquent, dans chaque composante, au-delà d'un certain indice déduit de l'allocation $\underline{k}^{\mathrm{SG}}$ et du budget R_b, on n'a plus besoin de

considérer les points (R_i, D_i). En effet, d'après le lemme 6.2.1 les multiplicateurs critiques associés à ces points ne pourront générer par chemins convexes des points cachés dans le triangle ∇_b^h. En découle le théorème suivant :

Théorème 6.2.2
Soit $\underline{k}^{SG}$ l'allocation de la solution de Shoham-Gersho pour le budget R_b. Un chemin convexe issu de $\underline{k}^{SG}$ ne contiendra un point dans le triangle ∇_b^h que si on y trouve des opérations $\vec{\Delta}_{k_i}$ telles que :

$$R(k_i + 1) - R(k_i^{SG}) \leqslant R_\Delta \tag{6.7}$$

En pratique cela signifie que l'on peut omettre l'ensemble des points d'une composante donnée à partir d'un indice particulier lorsqu'on descend des pentes par opération élémentaire $\vec{\Delta}$. Cet indice est déterminé grâce à $\underline{k}^{\mathrm{SG}}$ et ∇_b^h suivant l'équation (6.7). Cet indice se révèle être très proche de celui de $\underline{k}^{\mathrm{SG}}$.

De même, par définition d'un chemin convexe, on peut omettre l'ensemble des points d'une composante i donnée avant l'indice k_i^{SG} lorsqu'on descend des pentes par opération élémentaire $\vec{\Delta}$ depuis le point $\underline{k}^{\mathrm{SG}}$. Par convexité, les multiplicateurs critiques correspondant aux indices précédant k_i^{SG} sont en effet plus grands que le multiplicateur critique associé au triangle ∇_b^h.

Ainsi, ces constatations nous permettent de diminuer notablement le nombre de points à considérer dans chaque composante pour la descente de pente par opération élémentaire $\vec{\Delta}$. Le nombre exact dépend de la composante même mais ne dépasse pas les deux ou trois points. Par ce procédé la complexité chute visiblement.

6.3 Description d'un algorithme sous-optimal de recherche par chemins convexes

À partir des considérations précédentes, on peut adapter efficacement l'algorithme J. Le principe de mise à jour de la zone de recherche $\mathcal{R}$ et de δD est toujours applicable et sera donc employé de la même manière que pour l'algorithme de recherche optimal. On emploiera aussi préalablement l'algorithme RDPE sur chacune des composantes pour permettre à notre méthode de s'appliquer à tout type de fonctions. On obtient au final la méthode décrite par l'algorithme N.

Cet algorithme parcourt certains chemins convexes issus de la solution de Shoham-Gersho et peut s'apparenter à une procédure avare améliorée. Nous jetterons d'ailleurs un œil intéressé sur les performances relatives de ces deux algorithmes dans le paragraphe 6.4. On verra alors qu'on n'a effectivement pas de garantie d'optimalité avec cet algorithme, mais que les résultats obtenus sont assez probants pour être soulignés.

Algorithme N **Recherche sous-optimale par chemins convexes**

Cet algorithme retourne une solution au problème d'allocation de ressources pour un budget R_b .

1. *[Initialisation]*
   Appliquer l'algorithme RDPE à chaque composante.
   Initialiser $\mathcal{P}$ et (R^*, D^*) avec la solution de l'algorithme D.
   Initialiser δD avec $\delta D_{\min}$ donné par (3.10).
2. *[Boucle]*
   Remplacer chaque point dans $\mathcal{P}$ par ses successeurs de taux $R \leqslant R_b$ vérifiant le théorème 6.2.2.
   Aller à l'étape 4 si $\mathcal{P}$ est vide.
3. *[Mise à jour]*
   Soit (R^h, D^h) le point de plus faible distorsion dans $\mathcal{P}$.
   Si $D^h < D^*$, actualiser $\delta D \longmapsto \delta D - (D^* - D^h)$ et $(R^*, D^*) \longmapsto (R^h, D^h)$.
   Retirer les points de $\mathcal{P}$ satisfaisant à (3.11).
4. Retourner (R^*, D^*) si $\mathcal{P}$ est vide.
   Revenir à l'étape 2 sinon.

6.4 Performances

Nous avons effectué des simulations pratiques de l'algorithme N sur les différents ensembles de test suivants :
– CUT
– CNU
– NCU
– NCNU

Nous avons pris $N = 16$ composantes pour l'ensemble des tests et $M_i = 25$ valeurs par composantes pour tous les ensembles de fonctions hormis CUT où $M_i = 13$. Les tests ont été réalisés pour une suite de 1000 budgets répartis sur l'ensemble de la dynamique de taux. Certains des algorithmes mentionnés dans le chapitre 1, et dont les performances ont déjà été testées sur les même fonctions et séquence de budgets, sont rappelés ici à titre de comparaison. Nous avons aussi mentionné les performances de l'algorithme M du chapitre 5 pour les ensembles de fonctions appropriés. Les résultats expérimentaux sont ainsi récapitulés dans les tableaux 6.1- 6.4, rangés par ensembles de fonctions.

En premier lieu, aux vues des performances en temps de calcul moyen, l'algorithme N s'approche des méthodes les plus rapides, quel que soit le type des fonctions employées. Il est en particulier plus rapide (de l'ordre de 30%) que l'algorithme M.

TAB. 6.1: Performances des algorithmes pour l'ensemble CUT.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.03	51.8	2.48
G	3.8	100	0
$H_{(\Theta = 2)}$	0.6	36.8	4.40
$H_{(\Theta = 5)}$	2.0	95.5	0.27
I	0.03	72.5	0.56
M	0.09	79.4	0.44
N	0.07	73.2	0.53

TAB. 6.2: Performances des algorithmes pour l'ensemble CNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	13.3	5.1
G	112	100	0
$H_{(\Theta = 2)}$	2.5	7.6	14.0
$H_{(\Theta = 5)}$	17.5	50.0	6.0
I	0.07	23.3	3.1
M	0.19	32.7	2.6
N	0.13	26.8	2.9

TAB. 6.3: Performances des algorithmes pour l'ensemble NCU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	49.9	2.84
G	7.7	100	0
$H_{(\Theta = 2)}$	0.36	3.5	11.8
$H_{(\Theta = 5)}$	1.52	56.4	2.69
I	0.07	7.7	8.6
N	0.08	65	0.82

TAB. 6.4: Performances des algorithmes pour l'ensemble NCNU.

Algorithme	Temps de calcul	Pourcentage optimaux	Perte en distorsion
D	0.04	0.3	6.89
G	71.8	100	0
$H_{(\Theta = 2)}$	1.9	1.3	19.7
$H_{(\Theta = 5)}$	11.6	33.3	8.84
I	0.07	3.7	13.7
N	0.09	24.2	2.80

Dans le cas de fonctions convexes (type CUT et CNU), les algorithmes N et I sont très similaires en terme de points optimaux et de perte en distortion (l'algorithme de recherche par omission étant, lui, légèrement meilleur). Ce fait est lié au principe même de construction de l'algorithme N qui évoque certaines similitudes avec la procédure avare.

Par contre, si les fonctions $D_i(R_i)$ sont non-convexes (type NCU et NCNU), nous remarquons alors une augmentation importante (environ 8 fois plus) du nombre de points optimaux trouvés par l'algorithme N comparativement à l'algorithme I. La perte en distortion en est aussi fortement influencée, avec des résultats bien supérieurs pour l'algorithme N. Notons que ce résultat est d'ailleurs surprenant, alors même que l'exploitation des fonctions est lacunaire (de par le passage à des courbes convexes avec l'algorithme RDPE préalable). Le comportement de l'algorithme N est donc très bon, même pour des ensembles de fonctions non-convexes, et nous trouvons des points très proches des solutions optimales (perte en distortion moyenne faible).

En synthèse, l'algorithme N offre un temps de calcul stable et faible quelque soit le caractère des fonctions étudiées. Cela en fait une méthode extrêmement intéressante pour une implémentation. Le gain en performance est substantiel par rapport aux méthodes actuelles.

6.5 Conclusions

Dans ce dernier chapitre, nous avons adapté l'algorithme de recherche par chemins convexes présenté au chapitre 3. Cette méthode a été formulée afin de répondre de manière générale (pour tout type de fonctions $D_i(R_i)$) à notre problème d'allocation de ressources. Bien que n'atteignant pas l'optimalité, elle propose une solution au problème d'allocation de ressources avec une complexité proche des méthodes les plus rapides existantes et ce pour des performances supérieures.

Pour cela, nous avons limité les chemins convexes étudiés et poussé encore plus loin les critères d'élimination des chemins non admissibles. Nous avons aussi adapté la méthode pour qu'elle admette des fonctions $D_i(R_i)$ non convexes.

A titre d'exemple, qui à valeur de référence dans le domaine, nous avoisinons les valeurs de complexité de l'algorithme de Shoham et Gersho pour une perte en distortion trouvée bien inférieure à celle de l'algorithme D. Même sur des ensembles de fonctions $D_i(R_i)$ non convexes, le résultat est particulièrement probant.

Conclusions et perspectives

Conclusions

Cette thèse a permis d'aborder la problématique de l'allocation de ressources en communications numériques. Problème récurrent aussi bien en codage de source qu'en codage de canal, ses multiples ramifications et leurs portées en font un domaine d'étude actuel et complexe. Ce problème méritait donc une étude poussée et unifiée, ce qui n'avait pas encore été fait jusque là d'une manière aussi formelle.

Tout d'abord présenté de manière très générale, nous avons peu à peu détaillé les différents cas du problème motivant notre étude. Nous avons adopté un esprit de cohérence globale pour une approche unifiée des différentes méthodes et algorithmes développés. Nous avons ainsi pu effectuer un large tour d'ensemble du problème général et des techniques existantes pour le résoudre. Nous avons proposé des moyens de quantifier les performances relatives de ces différentes techniques. Les critères mis en avant dans ce manuscrit, si ils ne sont pas les seuls existants, nous ont paru les plus pertinents et adaptés, adéquats au niveau de discours que nous traitions. Enfin, les performances de ces techniques ont été exhaustivement qualifiées afin d'apporter une vision simple des avantages et inconvénients de chacune d'elles.

De cette étude préalable, nous avons dégagé des objectifs clairs d'amélioration potentielle. Ces objectifs visaient deux points découlant des divers critères présentés : d'une part, l'optimalité de la solution et d'autre part, l'efficacité de l'implémentation au regard de la proximité à la solution optimale.

Tout d'abord, nous avons souhaité étudier des techniques d'allocation de ressources *optimales*. Ces méthodes devaient permettre d'atteindre la solution optimale pour tout problème, quelles que soient les conditions. Les algorithmes de référence en codage de source étant soit à forte complexité soit pas optimaux, il semblait indispensable d'apporter de nouvelles méthodes capables d'offrir une alternative à ceux-ci.

Ensuite, dans le but d'une implémentation exploitable, nous avons abordé des techniques d'allocation de ressources sous-optimales *efficaces*. Ces méthodes devaient offrir des performances supérieures aux algorithmes actuels pour une complexité comparable, le gain se situant dans

la recherche de points sous-optimaux particuliers.

Le premier objectif a donné lieu à deux algorithmes optimaux. Ces algorithmes proposent une approche selon des principes novateurs. De plus, ceux-ci atteignent non seulement des performances optimales mais les obtiennent à une complexité inférieure aux différentes techniques optimales existantes. Le deuxième algorithme offre en particulier des performances très intéressantes et une approche très différente des algorithmes jusqu'ici publiés.

Le second objectif a de même occasionné le développement de deux nouveaux algorithmes. Cette fois-ci, les algorithmes ne sont plus optimaux, mais restent de performances supérieures aux autres méthodes efficaces. La complexité de ces algorithmes est elle tout à fait raisonnable et permet une implementation *in fine*.

Nous avons atteint les deux objectifs que nous nous étions fixés. Nous avons proposé quatre nouveaux algorithmes, prouvé leur intérêt et amorcé une vision neuve d'un problème ancien quelque peu délaissé. À ce titre, cette thèse est une réussite et pose la base de travaux ultérieurs d'intérêt. Cette thèse permet d'ouvrir de multiples perspectives dont certaines font l'objet du paragraphe suivant.

Perspectives

Dans un premier temps, il pourra être intéressant de travailler sur certains points non traités dans cette thèse.

Tout d'abord, nous avons parlé d'une version de la méthode de recherche par omission d'index pour laquelle nous pouvions itérer le principe d'omission sur les nouvelles enveloppes ainsi mises en évidence. Nous n'avions toutefois pas apporté d'éléments supplémentaires au sujet des performances de cette adaptation. Des pistes d'investigation pourraient être envisagées et apporter des réponses plus précises sur la profondeur d'itération adaptée en terme de compromis "complexité"/"perte en distortion" et les performances relatives qui en découlent.

D'autre part, les simulations réalisées dans cette thèse permettent de juger de l'efficacité des méthodes décrites et de valider les idées fondatrices, mais n'offrent pas un développement suffisamment poussé pour quantifier les performances de manière plus précise. Des travaux complémentaires pourraient donner des informations utiles sur des détails particuliers de chaque algorithme. Par exemple, le temps de calcul de l'algorithme optimal de recherche par ordonnancement est parfois extrêmement élevé, mais cela intervient dans des situations que nous ne pouvons actuellement identifier. Une meilleure compréhension de l'origine de cette conséquence pourrait permettre de mieux appréhender ce problème, voire de le résoudre.

Ensuite, après le développement et l'analyse de ces différents algorithmes, nous avons pu voir l'impact majeur du type des fonctions $D_i(R_i)$ employées. Or, nous avons pu remarquer au tra-

vers de certaines simulations que la position des points de non-convexité dans les différentes fonctions $D_i(R_i)$ avait peu d'influence sur le nombre de points cachés et que le paramètre quasiment unique déterminant ce nombre est la quantité de points de non-convexité dans les fonctions $D_i(R_i)$. Pourtant, aucune prédiction ou moyen ne permet aujourd'hui de calculer préalablement les performances des méthodes dites lagrangiennes (comme l'algorithme de Shoham-Gersho). En effet, étant des procédures de référence, il aurait pu être intéressant de pouvoir prédire le nombre de points optimaux auxquels ce type d'algorithme permet d'accéder ou de prévoir le nombre de points optimaux cachés. De tels résultats permettraient en particulier de juger de l'opportunité d'utiliser une méthode plus complexe. Si cette thèse ne s'est pas attardée sur ce sujet, il apparaît réellement important d'avoir une meilleure connaissance a priori du nuage global, chose encore difficile aujourd'hui.

Parmi les différentes méthodes et approches proposées, il pourra être intéressant de reprendre certains problèmes du domaine du codage numérique et d'adapter leur formulation à celle décrite dans ce manuscrit. En effet, la formulation, volontairement voulue généraliste, de ce manuscrit permet ainsi d'envisager de multiples applications aussi bien en codage de source et en codage de canal que dans d'autres domaines. Par exemple, dans le cas des algorithmes optimaux, des applications pourraient être faites dans le domaine de la recherche opérationnelle.

Enfin, il est envisageable d'adapter les méthodes développées dans ce manuscrit pour différentes applications indirectes, dont on citera en particulier les problèmes multi dimensions. Dans le cas de 2 dimensions, le problème se pose alors sous la forme :

$$\min_{\sum_{i=1}^{N_i} R_i \leqslant R^I, \sum_{j=1}^{N_j} R_j \leqslant R^J} \left(\sum_{i=1}^{N_i} \sum_{j=1}^{N_j} D_{i,j} \right)$$

Cette fois, il faut optimiser selon 2 contraintes. Une reformulation partielle de ce problème pourrait permettre d'appliquer des techniques en 1 dimension.

Quoi qu'il en soit, la diversité des problèmes se rapprochant de notre problème initial laisse un bon espoir de trouver d'autres applications et dérivations aux algorithmes développés dans cette thèse. De nombreux travaux sont encore envisageables sur les bases de ceux commencés ici.

Annexe A

Equivalence des courbes optimales réciproques

Le problème de la courbe optimale R-D et le problème de la courbe optimale D-R sont équivalents. En voici la preuve.

Soit un point optimal (R^*, D^*). Ce point est solution du problème $\min\limits_{R \leqslant R^*} D$

$$\min_{\sum_{i=1}^{N} R_i \leqslant R^*} \left(\sum_{i=1}^{N} D_i \right) = D^* \iff \begin{cases} \min\limits_{\sum_{i=1}^{N} R_i = R^*} \left(\sum\limits_{i=1}^{N} D_i \right) = D^* \\ \min\limits_{\sum_{i=1}^{N} R_i < R^*} \left(\sum\limits_{i=1}^{N} D_i \right) > D^* \end{cases} \tag{A.1}$$

$$\iff \begin{cases} \sum\limits_{i=1}^{N} R_i = R^* \Rightarrow \sum\limits_{i=1}^{N} D_i \geqslant D^* \\ \sum\limits_{i=1}^{N} R_i < R^* \Rightarrow \sum\limits_{i=1}^{N} D_i > D^* \end{cases} \tag{A.2}$$

$$\iff \begin{cases} \sum\limits_{i=1}^{N} D_i = D^* \Rightarrow \sum\limits_{i=1}^{N} R_i \geqslant R^* \\ \sum\limits_{i=1}^{N} D_i < D^* \Rightarrow \sum\limits_{i=1}^{N} R_i > R^* \end{cases} \tag{A.3}$$

$$\iff \begin{cases} \min\limits_{\sum_{i=1}^{N} D_i = D^*} \left(\sum\limits_{i=1}^{N} R_i \right) = R^* \\ \min\limits_{\sum_{i=1}^{N} D_i < D^*} \left(\sum\limits_{i=1}^{N} R_i \right) < R^* \end{cases} \tag{A.4}$$

$$\min_{\sum_{i=1}^{N} R_i \leqslant R^*} \left(\sum_{i=1}^{N} D_i \right) = D^* \iff \min_{\sum_{i=1}^{N} D_i \leqslant D^*} \left(\sum_{i=1}^{N} R_i \right) = R^* \tag{A.5}$$

Le point (R^*, D^*) est donc aussi solution du problème $\min_{D \leqslant D^*} R$. Les courbes optimales R-D et D-R sont confondues.

Annexe B

Corpus de test

Nous présentons ici des exemples des courbes que nous avons créées pour le corpus de test. Les exemples de nuages de points du plan global sont calculés pour $N = 8$ composantes d'un type donné. Nous avons différencié les types suivant les caractères :

– de convexité des fonctions $D_i(R_i)$,
– de régularité des taux R_i.

Nous considérerons donc les cinq types de fonctions :

– Convexe à taux Uniformément répartis (CU),
– Convexe à taux Uniformément répartis avec trous (CUT),
– Non-Convexe à taux Uniformément répartis (NCU),
– Convexe à taux Non-Uniformément répartis (CNU),
– Non-Convexe à taux Non-Uniformément répartis (NCU).

Pour les exemples de courbes $D_i(R_i)$, les taux R_i sont représentés par des lignes mixtes verticales. On peut ainsi voir la régularité d'espacement des taux.

Pour les exemples de nuage du plan global, la courbe optimale R-D est décrite en ligne solide, et les points d'enveloppe sont représentés par des cercles.

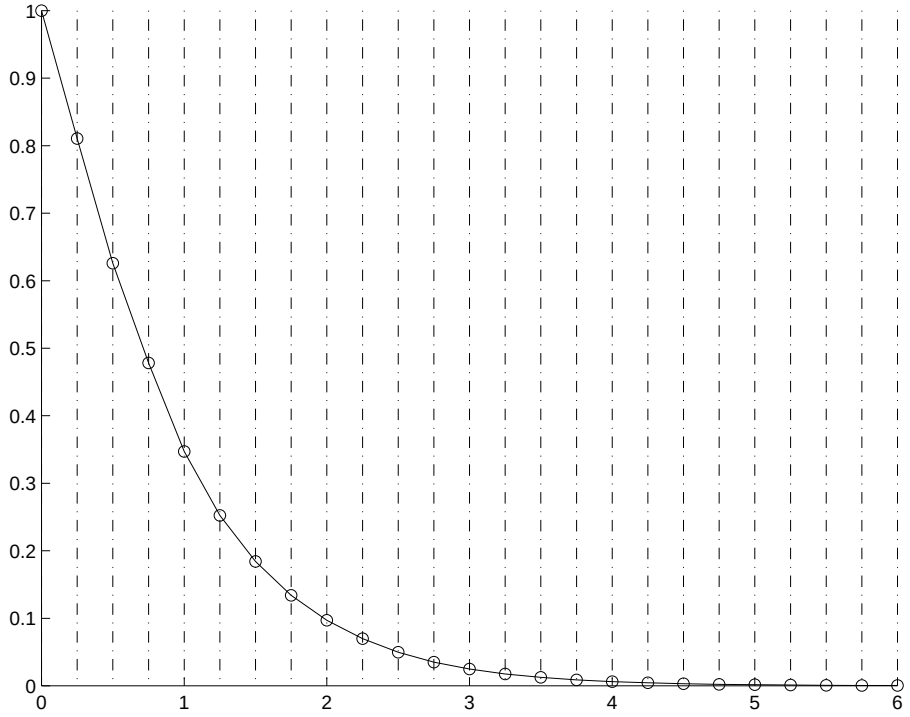


FIG. B.1: Exemple de $D_i(R_i)$ de type **CU**

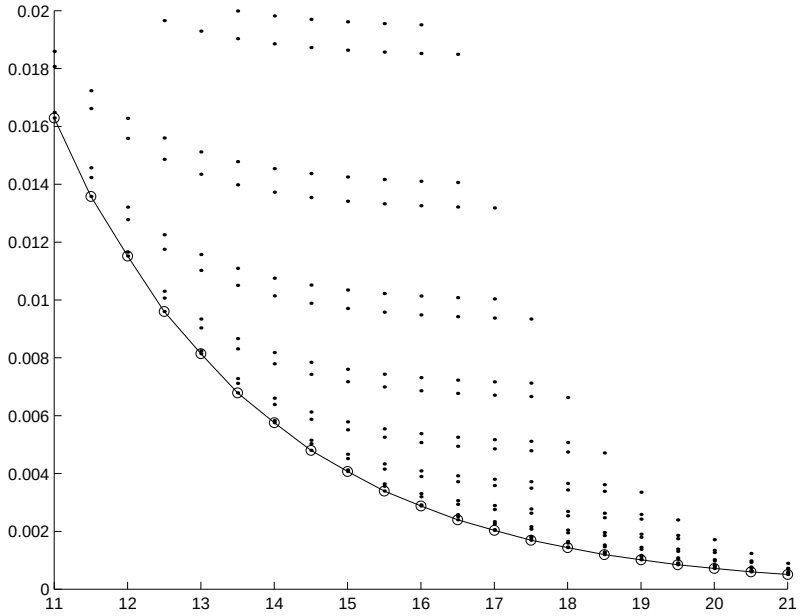


FIG. B.2: Une portion de l'ensemble (R,D) global pour des composantes **CU**

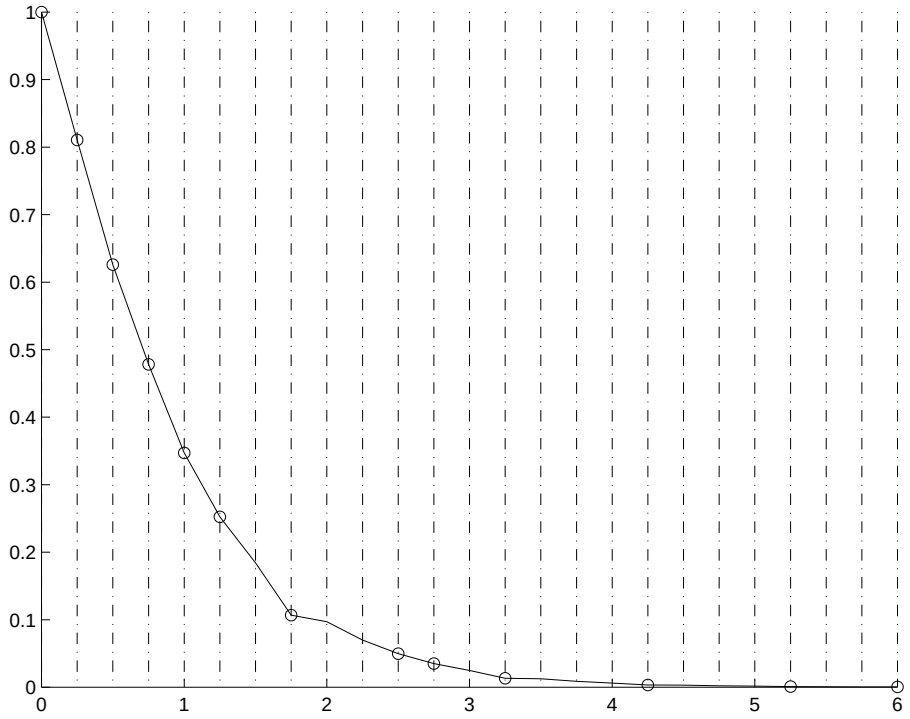


Fig. B.3: Exemple de $D_i(R_i)$ de type **NCU** : solide et de type **CUT** : cerclé

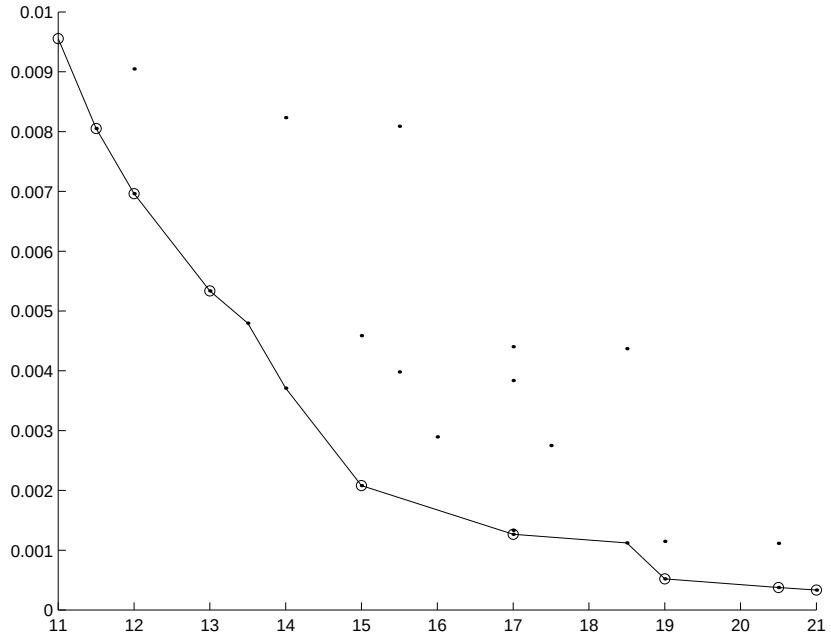


Fig. B.4: Une portion de l'ensemble (R,D) global pour des composantes **CUT**

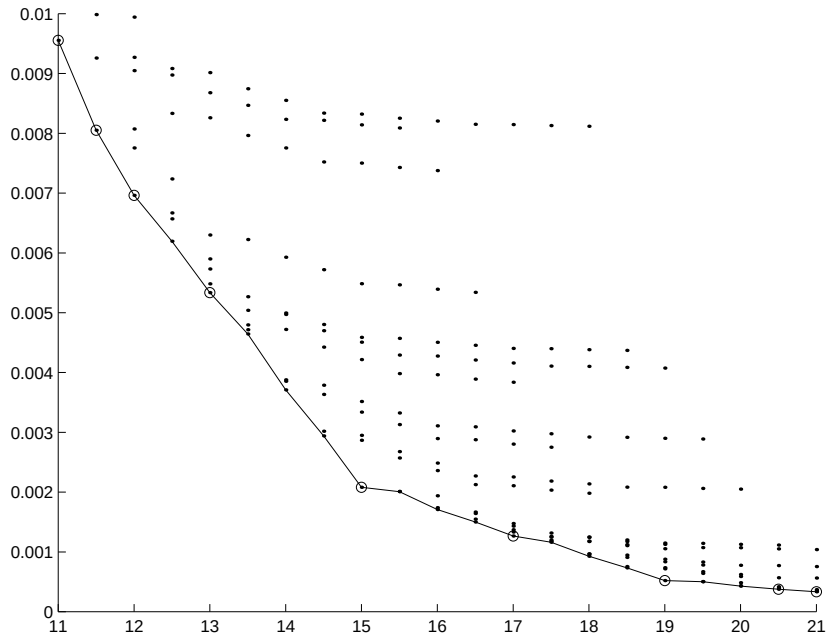


Fig. B.5: Une portion de l'ensemble (R,D) global pour des composantes **NCU**

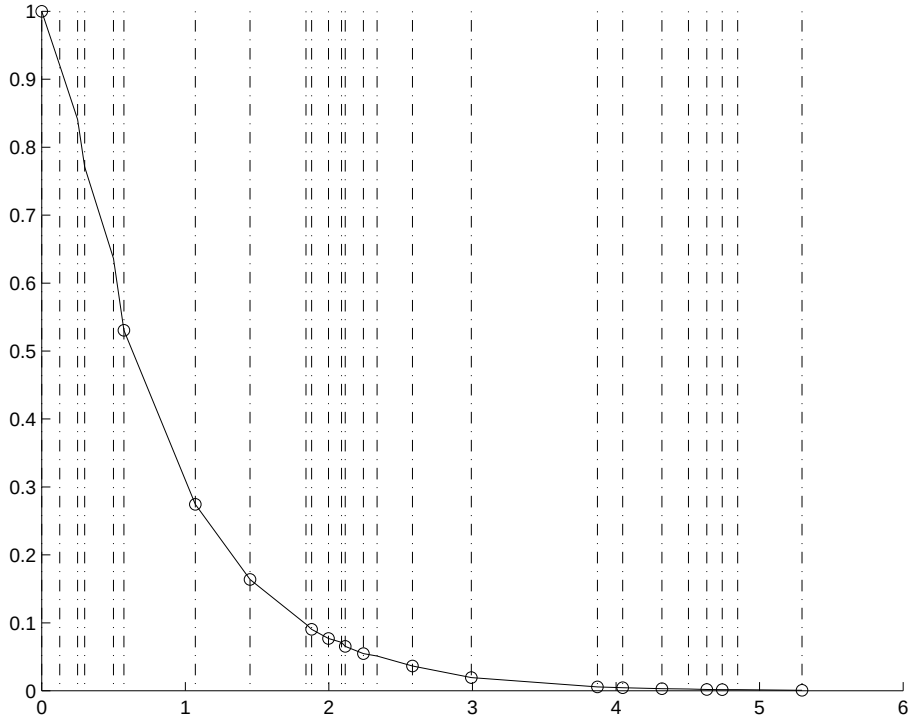


Fig. B.6: Exemple de $D_i(R_i)$ de type **NCNU** : solide et de type **CNU** : cerclé

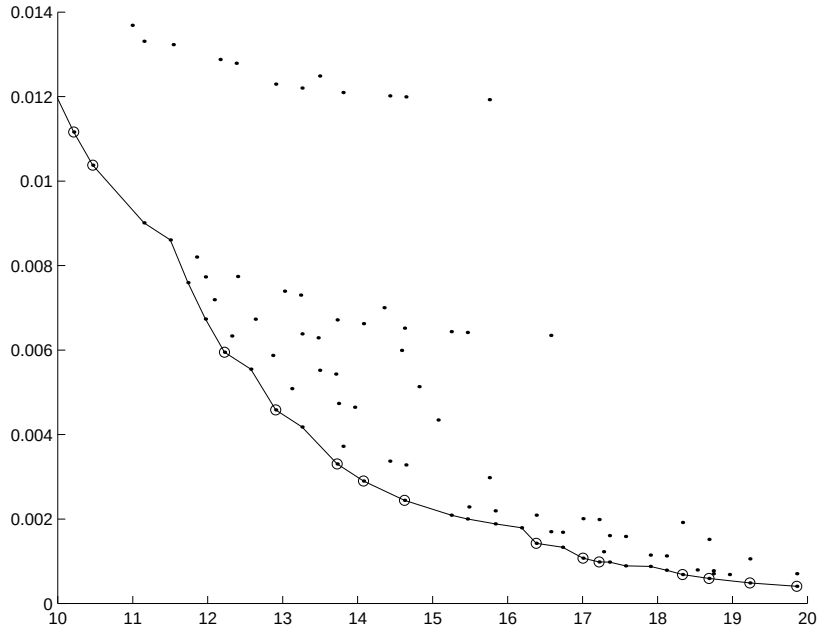


Fig. B.7: Une portion de l'ensemble (R,D) global pour des composantes **CNU**

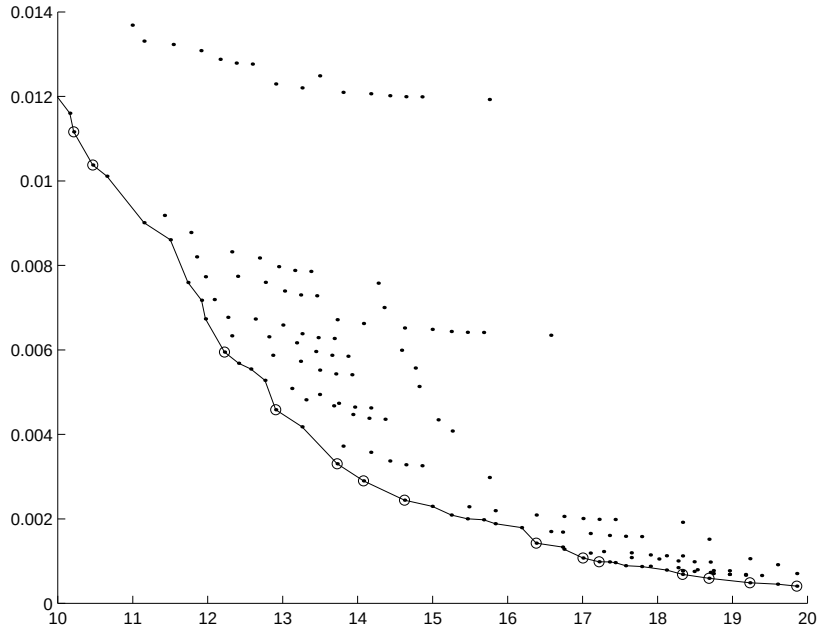


Fig. B.8: Une portion de l'ensemble (R,D) global pour des composantes **NCNU**

Annexe C

Calculs de complexité et de coût mémoire d'algorithmes

Afin de simplifier les calculs de complexité et de coût mémoire des différents algorithmes, nous prendrons $M_i = M$ pour tout i. Ainsi nous avons N composantes de M points chacun, considérés comme optimaux. Ceci n'enlève rien à la généralité de notre exposé.

Pour le coût mémoire, nous étudierons la taille mémoire maximale utilisée pour conserver les points calculés (sachant entre autre qu'un point du plan global est un vecteur d'allocation de taille N).

Pour le calcul de la complexité, nous tiendrons compte des additions, des multiplications et des comparaisons, que nous regrouperons toutes sous le terme d'*opérations*. Quand cela est possible, nous donnerons la complexité en nombre d'opérations effectuées par solution calculée.

C.1 Algorithme A

L'algorithme RDPO n'est pas un algorithme pour résoudre le problème d'allocation de ressource à proprement parler, mais celui-ci intervient dans certains algorithmes. Nous allons donc calculer le nombre d'opérations nécessaires à son exécution sur un nuage de points de cardinal C. Nous ne calculerons pas la taille mémoire nécessaire à l'algorithme car celui-ci correspond à la taille mémoire nécessaire aux données initiales, c'est à dire C.

Nous faisons un tri initial par ordre croissant sur les C points, cela demande donc $Clog(C)$ comparaisons. Ensuite, pour trouver les points optimaux, nous faisons au total C comparaisons pour trouver le point succédant le point courant.

Au final, à partir d'un nuage de C points, nous avons réalisé la recherche des points optimaux

en $O(Clog(C))$ opérations. Nous avons donc un nombre d'opérations en $O(log(C))$ opérations par points du nuage.

C.2 Algorithme B

A l'instar de l'algorithme RDPO, l'algorithme RDPE n'est pas non plus un algorithme pour résoudre le problème d'allocation de ressource, mais il est réutilisé dans plusieurs algorithmes. Nous allons donc calculer le nombre d'opérations nécessaires à son execution sur un nuage de points de cardinal C. Nous ne calculerons pas la taille mémoire nécessaire à l'algorithme car celui-ci correspond à la taille mémoire nécessaire aux données initiales, c'est à dire C.

Nous faisons un tri initial par ordre croissant sur les C points, cela demande donc $Clog(C)$ comparaisons. Ensuite, à une itération donnée où il reste P points à traiter, il faut effectuer $3P$ opérations pour calculer les pentes et faire P comparaisons pour trouver le point d'enveloppe suivant. Nous obtenons donc au total $\sum_{P=1}^{C-1} 4P = O(C^2)$ opérations au maximum.

Au final, à partir d'un nuage de C points, nous avons réalisé la recherche des points d'enveloppe en $O(C^2)$ opérations. Nous avons donc un nombre d'opérations en $O(C)$ opérations par points du nuage.

C.3 Algorithme D

L'algorithme de Shoham-Gersho commence par appliquer l'algorithme RDPE à chacune des N composantes de M points. Nous avons donc $O(NM^2)$ opérations pour cette étape. Ensuite nous effectuons le calcul et le tri de $N(M-1)$ pentes en $O(NMlog(NM))$ opérations. Puisque qu'il y a $N(M-1)+1$ points d'enveloppe globaux, nous avons au final $O(NM)$ opérations par solution trouvée.

Concernant la mémoire, nous avons les NM points de composantes auxquels s'ajoutent les $N(M-1)$ pentes calculées. Nous obtenons donc un total équivalent à $O(M)$ points globaux conservés en mémoire.

C.4 Algorithme E

A l'inverse de l'algorithme de Shoham-Gersho, les points des différentes composantes sont déjà nécessairement des points d'enveloppe. La méthode se résume donc au calcul et au tri de $N(M-1)$ pentes en $O(NMlog(NM))$ opérations. Puisque qu'il y a $N(M-1)+1$ points

d'enveloppe globaux, nous avons au final $O(log(NM))$ opérations par solution trouvée.

Concernant la mémoire, au même tire que l'algorithme D, nous avons les NM points de composantes auxquels s'ajoutent les $N(M-1)$ pentes calculées. Nous obtenons donc un total équivalent à $O(M)$ points globaux conservés en mémoire.

C.5 Algorithme G

L'algorithme G, d'une étape n à l'étape $n+1$, fusionne un nuage $\mathcal{O}_n$ de points optimaux (vecteurs de taille n) avec une composante de M points (vecteur de taille 1). Le nuage $\mathcal{O}_n$ est composé au minimum des points d'enveloppe au nombre de $n(M-1)+1$. C'est une borne inférieure car il existe des points cachés éventuels. On a donc besoin, à chaque étape n, de temporairement calculer et conserver au moins $[n(M-1)+1]M$ points de taille $n+1$. On effectue ensuite sur ce nuage un algorithme RDPO qui réduit la taille de celui-ci. La mémoire utilisée maximale est donc atteinte pour la dernière étape à la fusion du nuage $\mathcal{O}_{N-1}$ et de la composante N. Cette taille maximale de mémoire est finalement au moins $NM[(N-1)(M-1)+1]$. Nous obtenons alors un total équivalent à $O(NM^2)$ points globaux conservés en mémoire.

D'autre part, à chaque étape n, pour fusionner le nuage $\mathcal{O}_n$ et la composante $n+1$, on fait $Card(\mathcal{O}_n) * M$ additions (minoré par $[n(M-1)+1]M$). On a donc pour l'ensemble des étapes un nombre d'opération d'au moins :

$$\begin{aligned}
\sum_{n=1}^{N-1}[n(M-1)+1]M &= M(M-1)\sum_{n=1}^{N-1} n + M\sum_{n=1}^{N-1} 1 \\
&= M(M-1)\frac{(N-1)N}{2} + M(N-1) \\
&= M(N-1)\left(\frac{(M-1)N}{2}+1\right) \\
&= O(M^2N^2)
\end{aligned}$$

Ensuite, on applique après chaque étape l'algorithme RDPO qui travaille en $O(log(C))$ opérations par point du nuage, soit $O(n^2M^2log(nM))$ opérations à l'étape n. Nous en avons ainsi effectué au minimum $O(N^3M^2log(NM))$ opérations pour la succession d'algorithmes RDPO. Au final, nous avons donc $O(N^3M^2log(NM))$ opérations. Nous ne pouvons ramener ce résultat à une valeur par solution trouvée car le nombre de points optimaux n'est pas prévisible. Toutefois, dans le cas de composantes convexes, le nombre de points optimaux est de $N(M-1)+1$, ce qui ramène alors le nombre d'opérations de l'algorithme G à $O(N^2Mlog(NM))$ opérations par solution trouvée.

C.6 Algorithme H

De manière simple, les calculs concernant l'algorithme G peuvent être adaptés à l'algorithme H en remplaçant M par $2\Theta+1$ où Θ est le paramètre caractéristique de la méthode de fenêtrage.

Nous obtenons donc en coût mémoire un total équivalent à $O(N\Theta^2)$ points globaux conservés en mémoire. Le nombre d'opérations est de l'ordre de $O(N^3\Theta^2 log(N\Theta))$.

Bibliographie

[BB04] E. Baccarelli and M. Biagi. "Optimal integer bit-loading for multicarrier ADSL systems subject to spectral-compatibility limits". *Signal Processing*, 84(4) :729–741, April 2004.

[Bel57] R. Bellman. *"Dynamic Programming"*. Princeton University Press, Princeton, 1957.

[BFB02] E. Baccarelli, A. Fasano, and M. Biagi. "Novel efficient bit-loading algorithms for peak-energy-limited ADSL-type multicarrier systems". *IEEE Trans. Signal Processing*, 50(5) :1237–1247, May 2002.

[Bru87] A.M. Bruckstein. "On soft bit allocation". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(5) :614–617, May 1987.

[Cam98] J. Campello. "Optimal discrete bit loading for multicarrier modulation systems". In *International symposium on information theory*, page 193, Aug. 1998.

[Cam99] J. Campello. "Practical bit loading for DMT". In *IEEE International Conf. on Communications*, pages 801–805, June 1999.

[CCB95] P. S. Chow, J. M. Cioffi, and J. A.C. Bingham. "A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels". In *IEEE Transactions on Communications*, volume 43, pages 773–775, February/March/April 1995.

[CLG89] P. A. Chou, T. Lookabaugh, and R. M. Gray. "Optimal pruning with applications to tree-structured source coding and modeling". In *IEEE Transactions on Information Theory*, volume 35-2, pages 299–315, March 1989.

[Dav72] L.D. Davisson. "Rate-distortion theory and applications". In *Proceedings of the IEEE (Special Issue on Digital Picture Processing*, volume 60-7, pages 800–808, July 1972.

[DvIB04] J. Dulj, Ž. Ilić, and A. Bažant. "Efficient power allocation algorithm for OFDM systems". In *IEEE Mediterranean Electrotechnical Conf.*, volume 2, pages 425–428, May 2004.

[Eve63] H. Everett III. "Generalized Lagrange multiplier method for solving problems of optimum allocation of ressources". *Operations Research*, 11 :399–417, 1963.

[FE91] G.D. Forney and M.V. Eyuboglu. "Combined equalization and coding using precoding". *IEEE Communications Magazine*, 29(12) :25–34, December 1991.

[FG86] A. Federgruen and H. Groenevelt. "The greedy procedure for resource allocation problems : Necessary and sufficient conditions for optimality". *Operation Research*, 34 :909–918, 1986.

[FH96] R. F. H. Fischer and J. Huber. "A new loading algorithm for discrete multitone transmission". In *Proceedings IEEE Global Telecommun. Conf.*, volume 1, pages 724–728, November 1996.

[Fox66] B. Fox. "Discrete optimization via marginal analysis". *Management Science*, 13(3) :210–216, November 1966.

[Gal68] R. G. Gallager. *"Information theory and reliable communication"*. Wiley, New York, 1968.

[GG61] P.C. Gilmore and R.E. Gomory. "A linear programming approach to the cutting stock problem I". *Operations Research*, 9 :849–858, 1961.

[GG63] P.C. Gilmore and R.E. Gomory. "A linear programming approach to the cutting stock problem II". *Operations Research*, 11 :863–888, 1963.

[GG66] P.C. Gilmore and R.E. Gomory. "The theory and computation of knapsack functions". *Operations Research*, 14 :1045–1074, 1966.

[GG91] A. Gersho and R. M. Gray. *"Vector Quantization and Signal Compression"*. Kluwer Academic Publishers, 1991.

[HS63] J.J.Y. Huang and P.M. Schultheiss. "Block quantization of correlated Gaussian random variables". In *IEEE Trans. Commun. Syst.*, volume C-10, pages 289–296, September 1963.

[HS74] E. Horowitz and S. Sahni. "Computing partitions with applications to the knapsack problem". *Journal of ACM*, 21 :277–292, 1974.

[Huga] D. Hughes-Hartogs. "Ensemble modem structure for imperfect transmission media". U.S. patents. 4,679,227 (July 1987).

[Hugb] D. Hughes-Hartogs. "Ensemble modem structure for imperfect transmission media". U.S. patents. 4,731,816 (March 1988).

[Hugc] D. Hughes-Hartogs. "Ensemble modem structure for imperfect transmission media". U.S. patents. 4,833,796 (May 1989).

[Kal89] I. Kalet. "The multitone channel". *IEEE Transactions on Communications*, COM-37 :119–124, February 1989.

[Kol67] P.J. Kolesar. "A branch and bound algorithm for the knapsack problem". *Management Science*, 13 :723–735, 1967.

[KRJ00] B. S. Krongold, K. Ramchandran, and D. L. Jones. "Computationally efficient optimal power allocation algorithms for multicarrier communication systems". In *IEEE Transactions on Communications*, volume 48(2), pages 23–27, January 2000.

[KT51] H.W. Kuhn and A.W. Tucker. "Nonlinear programming". In J. Neyman, editor, *Proc. of the Second Berkeley Symposium on Math. Stat. and Prob.*, pages 481–492. U. of California Press, Berkeley, California, 1951.

[LC97] A. Leke and J. M. Cioffi. "A maximum rate loading algorithm for discrete multitone modulation systems". In *Proceedings IEEE Global Telecommun. Conf.*, volume 3, pages 1514–1518, November 1997.

[LR96] W. Y. Lee and J. B. Ra. "Fast algorithm for optimal bit allocation in a rate distortion sense". *Electronics Letters*, 32(20) :1871–1873, September 1996.

[LSC04] J. Lee, R. J. Sonalkar, and J. M. Cioffi. "Multi-user bit loading for multi-carrier systems". Preliminary draft, 2004.

[Raj04] N. M. Rajpoot. "Model based optimal bit allocation". Technical report, Department of Computer Science, University of Warwick, UK, January 2004.

[Ram82] T.A. Ramstad. "Subband coder with a simple adaptive bit-allocation algorithm". In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 203–207, 1982.

[Ram86] T.A. Ramstad. "Considerations on quantization and dynamic bit allocation in subband coders". In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 841–844, 1986.

[Ris91] E.A. Riskin. "Optimum bit allocation via the generalized BFOS algorithm". In *IEEE Trans. Inform. Theory*, volume 37-2, pages 400–402, March 1991.

[RV93] K. Ramchandran and M. Vetterli. "Best wavelet packet bases in a rate distortion sense". In *IEEE Trans. on Image Processing*, volume 2-2, pages 160–175, April 1993.

[Seg76] A. Segall. "Bit allocation and encoding for vector sources". In *IEEE Trans. Inform. Theory*, volume 22-2, pages 162–169, March 1976.

[SG88] Y. Shoham and A. Gersho. "Efficient bit allocation for an arbitrary set of quantizers". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(9) :1445–1453, September 1988.

[SS00] R. V. Sonalkar and R. R. Shively. "An efficient bit loading algorithm for dmt applications". *IEEE Commun. Lett.*, 4(3) :80–82, Mar. 2000.

[TGZ04] M. Tlich, F. Gauthier, and A. Zeddam. "Optimization process of power distribution for a frequency multiplexing DMT transmission". France Telecom RD RTA/Nice, 2004.

[Tru80] A. V. Trushkin. "Bit number distribution upon quantization of a multivariate random variable". *Problems of Inform. Transmission*, 16(1) :76–79, March 1980. Translated from Russian.

[WBB88] P.H. Westerink, J. Biemond, and D.E. Boekee. "An optimal bit allocation algorithm for sub-band coding". In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 757–760, 1988.

[WLK04] A. M. Wyglinski, F. Labeau, and P. Kabal. "An efficient bit allocation algorithm for multicarrier modulation". In *IEEE Wireless Communications and Networking Conference*, volume 2, pages 1194–1199, March 2004.

[WO86] J. W. Woods and S. D. O'Neil. "Subband coding of images". In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1005–1008, 1986.

[YGS02] C. Yisong, W. Guoping, and D. Shihai. "Further improvment of Lagrange multiplier method for optimal bit allocation". In *IEEE TENCON*, volume 2, pages 877–880, 2002.

[YOR96] Y. Yoo, A. Ortega, and K. Ramchandran. "A novel hybrid technique for discrete rate-distortion optimization with applications to fast codebook search for SVQ". In

Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), volume 4, pages 2040–2043, May 1996.