No Free Lunch et recherche de solutions structurantes en coloration

Jean-Noel Martin

▶ **To cite this version:**

Jean-Noel Martin. No Free Lunch et recherche de solutions structurantes en coloration. Modélisation et simulation. Université de Technologie de Belfort-Montbeliard, 2010. Français. NNT : 2010BELF0147 . tel-00607481

HAL Id: tel-00607481

https://theses.hal.science/tel-00607481

Submitted on 8 Jul 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 147 Année 2010

THESE

pour obtenir le grade de

DOCTEUR de L'UNIVERSITE DE TECHNOLOGIE DE BELFORT-MONTBELIARD

ECOLE DOCTORALE : SCIENCES POUR L'INGENIEUR ET MICROTECHNIQUES

Spécialité : Informatique

Par

Jean-Noël MARTIN

NFL ET RECHERCHE DE SOLUTIONS STRUCTURANTES EN COLORATION

Soutenue le 9 décembre 2010 à Belfort

Jury

Rapporteur	HAO Jin-Kao, Professeur, Université d'Angers
Rapporteur	PECHER Arnaud, Professeur, Université de Bordeaux 1
Examinateur	COLLET Pierre, Professeur, Université de Strasbourg
Examinateur	KHEDDOUCI Hamamache, Professeur, Université de Lyon 1
Examinateur	TOGNI Olivier, Maître de Conférence-HDR, Université de Bourgogne
Directeur de thèse	CAMINADA Alexandre, Professeur, UTBM

Plan de l'ouvrage

A ma mère,

à mon père,

à mon toujours jeune maître

Evariste Galois.

Remerciements

Merci aux rapporteurs de ce travail, Mr J-K. Hao et Mr A. Pêcher, pour la précision de leur lecture et l'intérêt de leurs remarques. Merci aux examinateurs de mon jury, Mr P. Collet, Mr H. Kheddouci et O. Togni, pour avoir accepté cette tâche.

Je tiens à remercier mon directeur de thèse A. Caminada, qui a guidé mon travail de recherche dans la situation objectivement originale d'un thésard atypique ! Au delà de l'ordinaire, je veux saluer la qualité de son écoute; il a su me proposer une thématique qui tienne compte de mes centres d'intérêt profonds et puisse s'intégrer dans ceux de notre équipe.

J'exprime ma profonde gratitude à Ph. Galinier pour le temps qu'il m'a consacré, ses questionnements sans complaisance à l'occasion de diverses rencontres de travail. Merci aussi à JP. Hamiez pour ses suggestions tout au long de la mise en oeuvre de notre projet GDR-RO de 2007.

Merci à plusieurs de mes collègues qui, d'une manière ou d'une autre, m'ont aidé: F. Holweck, I. Devarenne, J. Hu, M. Dib mais spécialement L. Moalic qui en raison de sa proximité a supporté les affres de la création... et m'a apporté plusieurs fois une aide technique précieuse.

Merci enfin à mes proches qui m'ont espéré dans les moments de doute.

INTRODUCTION

Tenter une approche des problèmes d'optimisation en conservant liées, tel un principe actif, la préoccupation de l'algorithme choisi, la prise en compte de la fonction à optimiser et les interactions de ces deux composants intimement noués, mais aussi se risquer à rechercher des méthodes qui apporteraient, en elles-mêmes, une structuration des solutions cherchées fait naître quelques craintes légitimes... devant l'énormité de la tâche ! Aussi nous est il apparu indispensable de réaliser en premier lieu, un état de l'art relatif aux liens profonds qui existent entre méthodes et fonctions d'évaluation dans les problèmes d'optimisation.

Dans le chapitre 1, nous mènerons l'analyse des théorèmes du No Free Lunch (NFL) en nous basant bien évidemment sur l'article initial de DH Wolpert et WG Macready dans leur version très aboutie de 1997, mais en présentant aussi les très intéressantes réactions d'un grand nombre d'équipes de chercheurs du monde entier, suites à ce travail d'une grande nouveauté. L'ensemble, en plus de ses finalités propres, constitue un exemple remarquable de construction scientifique.

Produire un descriptif suffisamment précis du papier inital, se révèle absolument indispensable tant il a été souvent mal compris, et nous n'avons pas manqué d'occasions de nous en convaincre ! En fait, ces incompréhensions sont autant de freins qui occultent l'importance des résultats produits: tel est leur effet négatif majeur. En effet les théorèmes du No Free Lunch ont des conséquences considérables, y compris en termes pratiques. Ils ont de surcroît objectivement permis de forger le langage destiné à traiter les problématiques qui étaient leur objet, et produit, rendu naturel en quelque sorte, un mode de pensée connexe. A titre d'exemple, nous

citerons un seul point tout à fait signifiant : l'interprétation sous forme géométrique de ces théorèmes proposée par les auteurs, met en évidence une symétrie, mieux quasiment une équivalence, au départ impensable, entre algorithmes de résolution et fonctions d'évaluation; et de tels regards changent la vie d'une communauté, doivent la changer certainement.

Mais les travaux des équipes qui ont apporté réponses à ces premiers résultats, ont contribué à développer l'oeuvre, à la perfectionner. Nous avons retenu parmi les très nombreuses contributions offertes à notre choix, celles qui nous sont apparues comme les plus pertinentes, comme les constituants majeurs d'une pensée en cours d'élaboration. Nous avons opté pour une présentation groupée par thèmes, selon les types d'approche et d'apports intellectuels. Nous fournirons d'abord quelques éléments destinés à la vérification expérimentale des théorèmes du No Free Lunch; puis nous nous interrogerons sur ce qu'il est possible d'attendre, mais aussi impossible d'espérer de résultats statistiques, en termes de guides de convergence. Enfin, et nous pensons qu'il s'agit là de l'essentiel rétrospectivement, nous avons centré notre attention sur la présentation des papiers destinés à la détermination des champs de validité des théorèmes du NFL. Pour que deux algorithmes a et a' aient des performances identiques en moyenne, sur un ensemble F de fonctions d'évaluation, il convient que les fonctions de F présentent des propriétés d'invariance par permutation de leurs arguments. L'étude précise des ensembles F qui présentent cette caractéristique, montre qu'ils sont relativement rares; on pourrait en conclure qu'il s'agissait donc d'une grosse frayeur pour rien ! Hélas, la situation est pire. Les champs de validité du NFL, grâce à un théorème du quasi-NFL établi en 2002 par S Droste, T Jansen et I Wegener, sont présents presque partout, à ε près, de façon non exacte peut-être, seulement approchée, mais d'une efficacité redoutable comme fauteurs de troubles en termes applicatifs. Les auteurs montrent en plus comment cet effet ne se cantonne pas à quelques exemples académiques qui feraient sourire

tout le monde, mais porte atteinte à des classes entières de problèmes qui sont le quotidien de la communauté de l'optimisation.

A ce stade de la réflexion mais aussi de la réalité, il devient clair qu'il serait insensé de faire l'impasse sur une étude globale des questions rencontrées, si l'on souhaite du moins approcher quelque peu les causes des phénomènes en jeu. Nous avons alors choisi de tenter de mettre en oeuvre ces préoccupations dans un champ applicatif suffisamment précis et large à la fois. Nous avons retenu le domaine de la coloration des graphes, par intérêt personnel d'abord, mais aussi en raison des thématiques de notre équipe; de surcroît, ce domaine est très fécond comme "traducteur" de nombreuses autres problématiques d'optimisation. Pour qu'il n'y ait pas d'ambiguité, il convient de préciser de suite qu'il n'était pour nous pas question de rechercher une méthode permettant de traiter tels ou tels types d'instances, d'entrer dans une course aux performances... Notre objectif est plutôt d'essayer de comprendre quelque chose globalement à la difficulté de coloration d'un graphe, à la détection des causes de cette difficulté et non de sa probabilité, à la tentative de mise en oeuvre d'une méthode qui structure par elle-même, l'espace de solutions cherché. Fini les rêves de boîtes noires généralistes, dont il nous faudra certainement apprendre à porter le deuil. D'ailleurs si on y pense, il y a objectivement dans cette façon de croire qu'on va trouver une solution sans tout à fait regarder le problème en lui-même, une certaine présomption ! Nous avons pour notre part tenté de nous lancer dans une recherche de solution qui respecte, en quelque sorte, davantage l'adversaire...

Dans le chapitre 2, nous rappelons quelques généralités de la théorie des graphes limitées aux concepts directement liés à l'usage que nous en ferons. Puis nous établissons le théorème d'existence et unicité de la décomposition d'un graphe simple non orienté en cliques maximales. L'idée en est simple; une arête $\{x_s, x_t\}$ est une 2-clique du graphe; elle est maximale pour G si on ne peut pas trouver de sur-clique

de $\{x_s, x_t\}$ "respectée" par G; on itère ensuite la propriété pour définir les cliques maximales. Nous pensons que l'ensemble $\{c_1, ..., c_N\}$ des cliques maximales de G - on pourra parler aussi de primary cliques - constitue l'essentiel de l'information - déjà structurée - fournie par un graphe en termes de coloration. Comme les nombres premiers permettent d'obtenir un entier relatif quelconque par produit, les cliques maximales permettent de reconstruire par étapes, en totalité, le graphe donné par une sorte d'addition. Un aspect important de cette appoche est son caractère purement algébrique, ensembliste; une partie de la combinatoire y est en quelque sorte "encapsulée", ce qui ne peut rester sans effets heureux.

Ensuite nous faisons émerger la notion de suite constructive d'un graphe, à partir de la volonté d'obtenir une "arithmétique" satisfaisante, réellement efficace sur les cliques maximales. Il s'agit de choisir un ordonnancement $(c_1, ..., c_N)$ de celles-ci de telle sorte que pour tout i compris entre 1 et N, les sous-graphes engendrés par les $\{c_1, ..., c_i\}$ constituent une suite croissante le plus souvent strictement, en dépit de quelques passages de "relâches" correspondant à des croissances larges, indispensables parfois, représentant autant d'intermèdes noethériens. Nous établissons l'existence - et constatons la non unicité - de ces suites constructives qui seront élaborées de manière effective par les algorithmes dévolus.

Le principe de base est le suivant. A l'étape i, nous supposons avoir engendré un sous-graphe du graphe total grâce aux cliques maximales $\{c_1, ..., c_i\}$. Alors nous cherchons parmi les cliques non encore prises en compte celles, si toutefois il en existe plusieurs, qui sont les plus fortement en contact avec le sous-graphe considéré mais sont en même temps les plus contraintes par les cliques restantes; nous choisissons alors l'une de celles-ci au sens de critères éventuels complémentaires et itérons le procédé. Autrement dit, à l'étape i, nous choisirons comme clique c_{i+1}, l'une de celles qui sont les plus contraintes par le graphe déjà engendré et qui transmettront le mieux aux cliques restantes le "feu de la coloration"; il s'agit de déterminer les

meilleurs transmetteurs de couleur. Il est troublant mais pas étonnant au fond, que le principe "vital" qui permet de définir cet ordonnancement se révèle très proche des pratiques de nombreuses heuristiques qui privilégient, au moment de définir une coloration propre, l'étude première des zones de plus forts conflits.

L'apport essentiel du chapitre 3 est le descriptif de l'algorithme principal lié aux concepts développés au chapitre 2; en raison de la technicité embarquée, nous ne pourrons évoquer ici que les principes fondateurs. Toutefois, un point mérite d'être de suite précisé; lorsque nous considérons une coloration de G - c'est-à-dire une application de l'ensemble des sommets $\{x_1, ..., x_n\}$ dans l'ensemble des couleurs disponibles $\{y_1, ..., y_k\}$ - nous ne nous intéresserons jamais ou presque au fait que tel sommet soit bleu par exemple; nous lui interdirons si besoin de prendre la couleur (formelle) de tel ou tel sommet déjà coloré à ce stade tout comme on attribue les valeurs prises par les arguments successifs d'une injection. Ainsi, ipso facto, une part de complexité est dans le principe même évacuée; en pratique, nous manipulons le concept plutôt que l'instance.

Dans une première partie, nous caractérisons les colorations propres du graphe G. Une coloration est propre pour le graphe total si et seulement si elle est injective sur chaque clique maximale de sa décomposition. En fait, cette propriété que nous pouvons dire absolue ici, présente une traduction relative: si on considère en effet les colorations des sous-graphes intermédiaires engendrés par les familles $\{c_1, ..., c_i\}$, les propres sont encore celles qui sont injectives sur chacune des cliques de la famille, à condition toutefois que cette famille soit la décomposition complète du graphe associé.

Cette propriété, soigneusement utilisée, permettra de déterminer les liens existants entre les colorations propres de la chaîne croissante des différents sous-graphes associés à la suite constructive choisie. La méthode de contraction-expansion, visitée

à nouveau dans ce cadre, s'invite comme naturellement à ce stade pour en fournir un descriptif ascendant total.

Il convient de remarquer que les propriétés étudiées ne sont pas universelles ou canoniques, en raison du choix prélable d'une suite constructive particulière; et pourtant, bien que portant la mémoire du chemin emprunté par la suite constructive retenue, les colorations propres obtenues in fine, seront bien celles du graphe total G quelle que soit la suite choisie et révéleront ainsi une invariance lourde de signification... Finalement, le choix effectué ne revêt pas l'importance qu'on aurait pu craindre.

Dans une sous-section que le lecteur pressé pourra omettre même si elle paraît cruciale à nos yeux, nous étudions quelques exemples génériques qui ont fait naître la méthode générale.

Puis dans une seconde partie nous décrivons l'algorithme principal. Nous montrons comment, à partir de la suite constructive retenue pour G, nous ordonnons les sommets constitutifs des cliques mises en oeuvre jusqu'alors et déterminons à chaque stade de la reconstruction du graphe, par principe même, une partition de l'ensemble de toutes les colorations propres du sous-graphe considéré - partition dont nous avons la faiblesse de penser sans le prouver, qu'elle pourrait être optimale -. Une fois épuisée la liste des cliques maximales de G, nous obtenons une partition des colorations propres pour le graphe total. Mais à tout stade intermédiaire, la même propriété est atteinte comme serait respecté un oignon, par couches successives. Nous sommes évidemment limités temporellement, par la taille des objets manipulés, mais les résultats sont étonnants. La méthode de contraction-expansion, adaptée à la nouvelle situation traitée, apparaît comme l'outil naturel de génération exhaustive des classes de colorations propres.

Enfin nous étudions deux versions "dégradées" de l'algorithme général. Tout d'abord nous déterminons une sous-partition de l'ensemble des colorations propres

du graphe total, constituées de celles qui utilisent seulement r couleurs. Si $r < \chi$, nombre chromatique du graphe, la partition trouvée est vide évidemment. Sinon, nous obtenons des propriétés agréables liant r et les partitions trouvées. Puis nous étudions une version de l'algorithme principal, en quelque sorte inversée dans ses objectifs. Au lieu de déterminer toutes les classes de colorations propres, nous en cherchons seulement une, classe solution particulière permettant de colorier le graphe considéré. Cette version plus récente nous donne encore quelques soucis même si le principe très proche des deux autres, n'est pas en cause. Pourtant, l'inversion des préoccupations poursuivies a des effets très conséquents sur la mise en oeuvre et apporte son lot de difficultés nouvelles. Le sentiment qui demeure, reste un certain étonnement qu'une même idée résiste à des situations aussi diverses, sous des modifications finalement mineures, même si objectivement, ce fait ne doit pas surprendre.

Nous avons choisi de présenter au chapitre 4, tout à la fois, une synthèse critique des résultats obtenus et les prolongements qui nous semblent des pistes de travail intéressantes, dont nous explorons déjà certaines actuellement. Il ne nous paraît pas possible en effet de présenter les prolongements envisagés hors du contexte précis qui les fait naître, trop loin aussi des soucis qui les accompagnent... Pour les mêmes raisons, nous ne saurions être exhaustifs ici et nous contenterons de signaler les champs abordés, en fournissant les seuls éléments indispensables à leur compréhension.

Nous étudierons d'abord quelques résultats expérimentaux qui nous permettront d'interroger la notion de difficulté de coloration d'un graphe, et de remettre en cause un certain nombre de critères retenus souvent comme outils de mesure sans nous apparaître véritablement fondés. Pour autant nous constaterons que nous n'avons pas réussi à définir un outil imparable et qu'il reste un gros travail pour espérer

cerner de façon précise cette complexité, même si certaines intuitions commencent à s'imposer.

Puis nous utiliserons la production pour un graphe donné de la partition de ses classes de colorations propres, telles qu'elles sont issues de nos algorithmes. Dans un premier temps nous serons en possibilité de déterminer les différentes colorations d'une même classe obtenues par certaines permutations licites, que nous serons en mesure de définir de façon précise. Toutes ces colorations seront équivalentes, de manière standard, au sens de la partition prédéfinie. Dans un second temps, nous serons en état de décompter les colorations propres puisque nous serons capables de déterminer le nombre des éléments d'une même classe, et par suite leur nombre total par sommation de ces résultats élémentaires. Nous obtiendrons par là le polynôme chromatique du graphe considéré par son écriture formelle totalement indépendante du nombre r de couleurs disponibles et signalerons quelques propriétés intéressantes relatives à certains de ses sous-polynômes liés au nombre maximal r des couleurs autorisées. Nous nous interrogerons sur la signification éventuelle du polynôme chromatique, en particulier de ses racines, et exprimerons notre doute quant au fait que ce polynôme, résultat en fait d'une projection assez complexe mais non orthogonale, puisse vraiment signifier quelque chose de profond au sens du réel ainsi projeté. Dans un troisième temps enfin, nous verrons qu'il est possible de prouver certaines propriétés formelles du graphe en termes de colorations propres; ce point abordé seulement sur un exemple générique, nécessiterait toutefois de plus amples développements.

Ensuite, nous nous intéresserons à la représentation des graphes, aspect minoré par notre approche essentiellement algébrique. Nous utilisons les suites constructives pour aider à représenter certains de nos graphes petits évidemment, issus de générations aléatoires, à l'occasion de diverses présentations de nos travaux; nous pensons que ces outils, appelés de façon plus systématique, pourraient donner des résultats

intéressants globalement pour des graphes petits, localement pour des graphes ordinaires. Il ne nous semble pas impossible qu'il y ait intérêt à soumettre le théorème des quatre couleurs à une relecture à l'aide des cliques maximales constitutives, nécessairement d'ordre 3 ou 4, sous les hypothèses ordinaires.

Enfin, nous établissons une correspondance de Galois entre les familles emboîtées de cliques maximales issues d'une suite constructive et les colorations propres des sous-graphes intermédiaires engendrés. Cette correspondance révèle à nos yeux le caractère profondément structuré des objets mis au jour; elle apparaît avec le résultat indiqué mais a opéré dans l'ombre dès le départ, car elle nous semble responsable, au fond, du fait que les algorithmes "tournent".

Pour achever ce travail, nous proposerons un bilan lié à l'esquisse des perspectives essentielles.

Chapitre 1

Les théorèmes du No Free Lunch

1.1 Introduction

Quiconque se préoccupe de questions liées à l'optimisation a été confronté à une situation très ordinaire dans ce domaine: disposer d'une méthode, d'une technique, d'un logiciel , "bricolés" même, fournisseurs de solutions plutôt satisfaisantes, sans parvenir à saisir le pourquoi de cette qualité parfois inattendue. Les plus honnêtes le reconnaissent ! Le scientifique qui sommeille en chacune et chacun s'en émeut, l'image mythique du chercheur est écornée. Au delà de cet ultime aspect, secondaire au fond, il convient d'admettre que cela fait objectivement désordre, mais stimule les découvreurs.

Les problèmes posés étant, par essence, souvent très difficiles, une telle situation peut durer très longtemps, trop en tout cas... Les lignes se figent alors, provoquent la naissance d'écoles dont les rapports se révèlent par instants conflictuels !

Lorsque DH Wolpert et WG Macready font paraître leur premier papier, daté de février 1995 [WM95], que la communauté désignera à leur instar "théorème du No Free Lunch", un climat polémique voit le jour. Pourtant, cette contribution mérite à nos yeux qu'on y porte grande attention; elle nous paraît exemplaire, constitue un corpus très construit.

D'une part, l'article initial est d'une grande richesse. Il a le mérite de proposer un cadre, des notations, d'apporter des résultats précis, de structurer une problématique, d'esquisser les grandes lignes et de baliser le terrain de nombreuses questions connexes.

D'autre part, il a déclenché un grand nombre de suites et de réponses, très diverses, d'un intérêt certain. Sans s'arrêter à l'écume des réactions agacées, les apports multiples qui vont en découler forment une excellente matière scientifique, un exemple remarquable de concepts en train de naître, d'une communauté scientifique à l'oeuvre...

C'est pourquoi, nous commencerons évidemment par la présentation de l'article initial; nous aborderons ensuite les contributions qui en ont découlé lorsqu'elles nous paraîssent spécialement pertinentes, en termes de construction de la pensée.

La règle sera, à ce stade de notre travail, de risquer une lecture synthétique, précise quant à l'enchaînement des idées, évitant de se perdre dans les détails techniques même s'il paraît indispensable et fécond de les appréhender sous l'aspect de leur typologie; nous tenterons aussi de décrire les errements, les espoirs liés, fûssent-ils parfois déçus...

Pour ce qui concerne les preuves, nous renvoyons ordinairement aux articles étudiés; toutefois nous tentons d'en indiquer les principes, les idées et de fournir quelques indications brèves quant à leur difficulté technique.

1.2 L'article initial: DH Wolpert, WG Macready (1997)

DH Wolpert et WG Macready ont proposé plusieurs versions de l'article : *"No Free Lunch Theorems for Optimization"*. Nous choisissons ici la structure de la version parue en avril 1997 dans IEEE Transactions on Evolutionnary Computation [WM97];

elle est très aboutie, plus épurée que la première quant aux idées, mais s'est aussi enrichie.

1.2.1 Introduction

Les auteurs situent leur apport dans le contexte des habitudes de la profession qui recourt fréquemment à des algorithmes d'optimisation, utilisés comme de véritables boîtes noires. Ils affirment la nécessité de mener une analyse formelle du lien entre la qualité des algorithmes mis en oeuvre et le type du problème traité, afin d'entrer dans la compréhension de leurs relations, dans l'étude d'invariants liant méthodes et classes de problèmes, en se libérant le plus possible des contextes particuliers. Ils annoncent la preuve de deux théorèmes du "No Free Lunch" qui, à leurs yeux, rendent caduques un certain nombre de pratiques ordinaires. Enfin ils présentent le plan de leur étude.

1.2.2 Notations

- On considère deux ensembles X et Y de cardinaux respectifs $|X|$ et $|Y|$ finis, éventuellement grands. Y est totalement ordonné.

  Soit f une fonction de coût ($f : X \rightarrow Y$), application de X dans Y, choisie dans l'ensemble des possibles $\mathcal{F} = Y^X$ dont le cardinal est évidemment fini, sous les hypothèses antérieures.

  X désigne l'espace de recherche, ensemble des points x_i pris en compte dans le problème considéré. Y représente l'ensemble des coûts, et $f\,(x_i\,)$ le coût associé à x_i.

  f peut être indépendante du temps en cours d'exécution des algorithmes (voir théorème 1), mais aussi en dépendre (voir théorème 3).

- Un échantillon d_m de taille m ($m \in \mathbb{N}$) désigne une suite finie de couples $(d_m^x(i), d_m^y(i))$ de $X \times Y$, où l'indice i, élément de $\{1, ..., m\}$, désigne le numéro d'ordre de génération par l'algorithme a considéré, des points $d_m^x(i)$ de X, supposés **distincts** [1] (a est alors dit sans répétition). Ainsi on aura $d_0 = \varnothing$ et pour $m > 0$:

$$d_m = ((d_m^x(1), d_m^y(1)), ..., (d_m^x(m), d_m^y(m))) \tag{1.1}$$

où $d_m^x(i)$ représente le $i^{ième}$ point généré, de coût $d_m^y(i)$.

d_m^x [respectivement d_m^y] désigne la suite finie $(d_m^x(1), ..., d_m^x(m))$ [respectivement de leurs coûts $(d_m^y(1), ..., d_m^y(m))$].

- L'ensemble de tous les échantillons de taille m est noté $\mathcal{D}_m$ tandis que l'ensemble $\mathcal{D}$ des échantillons de taille quelconque est défini par :

$$\mathcal{D} = \bigcup_{0 \leq m \leq |X|} \mathcal{D}_m$$

- Un algorithme d'optimisation a est une fonction de $\mathcal{D}$ dans X , qui à l'ensemble, noté d^x, des points déjà visités par l'algorithme, associe un point nouveau de X. Elle peut être définie comme suit :

$$a : d \in \mathcal{D} \mapsto \{s \in X \quad / \quad s \notin d^x\}. \tag{1.2}$$

Une telle définition se limite implicitement aux algorithmes déterministes. On atteint le stochastique, en choisissant pour image de d un élément parmi les valeurs de X non encore visitées, de façon stochastique.

[1] Cette hypothèse est raisonnable au sens où la préoccupation première est de connaître le nombre d'évaluations nécessaires à un résultat, en évitant certains phénomènes périodiques. Elle permettra aussi le dénombrement des fonctions de coût responsables d'un histogramme donné.

- Pour mesurer les performances d'un algorithme a après m itérations, on fera choix d'une fonction Φ qui à tout d_m^y associe un réel; par exemple :

$$\Phi\left(d_m^y\right) = \min_{i\in\{1,\ldots,m\}} \left(d_m^y\left(i\right)\right). \tag{1.3}$$

Après avoir argumenté de façon très précise l'intérêt que présente le fait de se positionner résolument dans le cadre de la théorie des probabilités (généralisation simple des résultats obtenus au cas déterministe, cadre théorique favorable à la conduite des preuves et au traitement des problèmes ordinaires standards), les auteurs posent enfin les notations suivantes.

- Pour tout f de $\mathcal{F}$, on définit la probabilité $p\left(f\right)$ que f représente la fonction de coût prise en compte via :

$$p\left(f\right) = p\left(\left\{\left(f\left(x_1\right), ..., fx_{|X|}\right)\right\}\right). \tag{1.4}$$

  Même si pour un f donné, $p\left(f\right)$ n'est pas totalement connue, ceci permettra de considérer des classes de fonctions de coût sur lesquelles seront étudiés les algorithmes considérés.

- Par suite, la performance d'un algorithme a itéré m fois sous la fonction de coût f , sera obtenue grâce à $p\left(d_m^y|f,m,a\right)$, qui représente la probabilité conditionnelle d'obtenir d_m^y sous les conditions indiquées. On en déduira aisément $\Phi\left(d_m^y\right)$.

Suit une remarque montrant le caractère complémentaire de cette approche probabiliste par rapport aux études de complexité classiques.

1.2.3 Les théorèmes du "No Free Lunch" (NFL)

Le terme de No Free Lunch est lié au fait que les théorèmes proposés montrent que, si un algorithme est performant sur une certaine classe de problèmes, il devra le

"payer" pour les autres classes [2]. La question précise, sous-jacente à ce paragraphe, est la suivante :

"Quels sont les liens du sous-ensemble $\mathcal{F}_1$ de $\mathcal{F}$ sur lequel un algorithme a_1 est meilleur qu'un algorithme a_2 avec le sous-ensemble $\mathcal{F}_2$ de $\mathcal{F}$ sur lequel on a la situation inverse ?"

Fonction de coût indépendante du temps : premier théorème

Théorème 1 *Sous les hypothèses et notations antérieures, pour tout couple d'algorithmes déterministes (a, a'), on a :*

$$\sum_{f \in \mathcal{F}} p\left(d_m^y | f, m, a\right) = \sum_{f \in \mathcal{F}} p\left(d_m^y | f, m, a'\right). \tag{1.5}$$

Preuve. La preuve est très intéressante, relativement simple dans les idées, mais rendue assez technique par les difficultés de notations, essentiellement. Elle sera étudiée avec profit. Elle est basée sur une récurrence sur la taille m de l'échantillon, et représente globalement un beau morceau de mathématiques ! Les différentes fonctions f sont astucieusement considérées via le vecteur des valeurs prises sur X; ceci fait penser à l'apport de Lebesgues dans la définition de l'intégrale du même nom, par rapport à la définition de Riemann, par exemple. Dans le calcul central, le recours au symbole de Kronecker rend la démonstration particulièrement élégante.

■

Corollaire 2 *Pour toute mesure de performance $\Phi\left(d_m^y\right)$, la moyenne (sur l'ensemble des fonctions f) des $p\left(\Phi\left(d_m^y\right) | f, m, a\right)$ est indépendante de l'algorithme a considéré.*

[2] Sans doute pourrait on dire :"On a rien sans rien", passé en langue allemande : "Von nichts kommt nichts". Il semblerait que le terme soit issu de l'histoire de l'Amérique du XIX° siècle et de l'existence de soupes populaires, nommées "free lunchs".

En conséquence, si un algorithme est spécialement performant sur une classe de problèmes, l'invariance de la moyenne impose qu'il ne le soit pas sur les autres problèmes. [3]

Fonction de coût dépendante du temps : second théorème

a) Définitions

On considère donnée, à l'instant du début de la construction de l'échantillon des valeurs de x par l'algorithme d'optimisation, une fonction de coût f_1.

Avant chaque itération suivante, la fonction de coût courante f_i est modifiée en une nouvelle f_{i+1}, via une application T_i, permutation de $\mathcal{F}$ afin d'éviter un biais [4] qui affecterait l'évaluation de la qualité des algorithmes étudiés. Deux types de dépendance du temps pour les fonctions de coût, sont alors proposés.

- Premier type

  A l'étape j la valeur du coût $d_m^y(j)$ prise dans Y correspondant à la valeur $d_m^x(j)$ associée, est fournie par la fonction de coût f_j valide à l'instant de l'obtention de $d_m^x(j)$.

- Second type

  Dans cette variante les auteurs prennent en compte un échantillon D_m^y construit à partir de la fonction de coût valide à l'instant de l'obtention de la dernière valeur de l'échantillon $d_m^x(m)$, c'est-à-dire f_m où $f_m = T_{m-1}(f_{m-1})$.

[3] Ce résultat est important en ce qu'il incite à une certaine prudence et ruine quelques tentatives hasardeuses. En même temps il reste décevant et objectivement modeste; le critère de la "moyenne" est assez grossier au regard de l'attente formulée dans l'introduction de la section en cours...

[4] Le caractère bijectif de T_i est requis afin que les fonctions de coût ne restent pas fixées dans une zone de $\mathcal{F}$ où l'algorithme étudié aurait des performances trop particulières: anormalement bonnes ou anormalement mauvaises. On sent poindre à ce stade, les théorèmes réciproques relatifs aux domaines de validité des théorèmes du "No Free Lunch" tels qu'ils appparaîtront dans certains articles ultérieurs, par exemple dans [SVW01].

- En guise de bilan, si $d_m^x = (d_m^x(1), ..., d_m^x(m))$,

  pour le premier type :

$$d_m^y = (f_1(d_m^x(1)), [T_1(f_1)](d_m^x(2)), ..., [T_{m-1}(f_{m-1})](d_m^x(m))); \quad (1.6)$$

  pour le second :

$$D_m^y = (f_m(d_m^x(1)), f_m(d_m^x(2)) ..., f_m(d_m^x(m))). \quad (1.7)$$

Les auteurs signalent la possibilité d'obtenir des résultats intéressants dans les applications, en comparant les durées de vie respectives de la fonction de coût et celles des composantes de l'échantillon.

b) Généralisation du premier théorème du "No Free Lunch"

Théorème 3 *Sous réserve que $m > 1$, sous les notations précédentes, pour tout d_m^y et D_m^y,*

pour tout couple d'algorithmes déterministes (a, a') et toute fonction de coût initiale f_1 de $\mathcal{F}$, on a :

$$\sum_T p(d_m^y | f_1, T, m, a) = \sum_T p(d_m^y | f_1, T, m, a'); \quad (1.8)$$

$$et \quad \sum_T p(D_m^y | f_1, T, m, a) = \sum_T p(D_m^y | f_1, T, m, a'). \quad (1.9)$$

Preuve. La preuve de ce deuxième résultat, bien que reposant sur un principe proche de la première, est considérablement plus délicate techniquement.

- Une première étape est commune aux deux types de dépendance étudiés, puis la preuve se sépare en fonction de ceux-ci.

- Pour le deuxième type, on ramène la somme dont on cherche à prouver l'invariance par rapport à l'algorithme considéré, à une quantité qui s'interprète comme un dénombrement d'objets, indépendant de l'algorithme a choisi.

- Pour le premier type, la situation est plus complexe encore car elle nécessite des décompositions très subtiles de la somme invariante dont seuls les principes parfois sont exposés très clairement, sans que les calculs ne soient totalement finalisés, le plus souvent.

■

Premières conséquences des théorèmes du NFL

DH Wolpert et WG Macready rappellent l'importance d'éviter un certain nombre de naïvetés au moment de statuer sur la qualité de différents algorithmes. Si l'un d'eux par exemple, se comporte mieux qu'une procédure au hasard pour une certaine classe de fonctions de coût, il se comportera nécessairement moins bien pour les autres fonctions...

Ils en concluent que, rendre compte des qualités d'un algorithme, agissant pour un choix de paramètres particuliers sur un certain nombre de problèmes, présente un intérêt limité à leurs yeux [5].

- Les théorèmes du NFL ne peuvent en général constituer une méthode de comparaison de sous-classes de fonctions de coût $\mathcal{F}_1$ et $\mathcal{F}_2$, ou d'opérateurs d'évolution T_1 et T_2 selon le cadre de l'étude.

- Ils peuvent par contre fournir des éléments de comparaison lorsque les f sont quelconques, mais par exemple équiprobables, c'est-à-dire : $p(f) = 1/\left|Y^X\right|$.

[5] Le message à la communauté est clair... La remarque mérite d'être faite sans doute. La question théorique de fond demeure. Existe-t-il des algorithmes meilleurs que d'autres? Ne sommes-nous pas condamnés à mettre en oeuvre le meilleur outil pour un problème donné? Reste à prouver qu'il l'est effectivement, et à étudier son champ d'excellence.

Dans le cas contraire de non-équiprobabilité, il faudra analyser précisément la somme suivante :

$$p\left(d_m^y|m,a\right) = \sum_f p\left(d_m^y|f,m,a\right).p\left(f\right). \tag{1.10}$$

- DH Wolpert et WG Macready sont amenés alors à s'interroger quant aux ensembles de fonctions de coût, sur lesquels peuvent s'appliquer les théorèmes du NFL. Ce probléme a été résolu dans l'une des réactions à [WM95] via une caractérisation de leurs champs d'application, au théorème 56.

Cas des algorithmes d'optimisation stochastiques

Théorème 4 *Les résultats des théorèmes du NFL restent valides dans le cas d'algorithmes stochastiques, n'autorisant pas la reprise de termes x_i déjà choisis dans d^x.*

Preuve. On se ramène au cas déterministe, en considérant un algorithme stochastique σ conforme à l'hypothèse indiquée, comme convenu au paragraphe 1.2.2. Ainsi dans la définition de σ, intervient une distribution de probabilité dépendant de d^x telle que la probabilité d'atteindre tout x_i de d^x soit nulle; σ apparaît comme un hyper-paramètre définissant $p\left(d_{m+1}^x\left(m+1\right)|d_m,\sigma\right)$ pour tout m et d. On remplace alors dans les preuves des théorèmes 1 et 3 les a par les σ. ■

1.2.4 Interprétation géométrique des théorèmes du "No Free Lunch"

Interprétation de $p\left(d_m^y|m,a\right)$ en termes de produit scalaire

DH Wolpert et WG Macready considèrent la somme $p\left(d_m^y|m,a\right)$ définie par :

$$p\left(d_m^y|m,a\right) = \sum_{f\in\mathcal{F}} p\left(d_m^y|f,m,a\right).p\left(f\right), \tag{1.11}$$

et proposent de l'interpréter comme un produit scalaire sur $\mathcal{F}$.

Nous éprouvons une gêne considérable à propos de cette lecture de $p\left(d_m^y|m,a\right)$. Il est vrai que cette somme ressemble à un produit matriciel, que l'on aimerait voir comme celui de composantes de vecteurs d'un espace vectoriel.

- $\mathcal{F}$ semble être considéré comme le corps de référence; or il n'est pas doté d'une telle structure. Nous ne voyons pas quel plongement l'en doterait.

- Par ailleurs, les matrices colonnes ${}^t\left(p\left(d_m^y|f,m,a\right)\right)_{f\in\mathcal{F}}$ et ${}^t\left(p\left(f\right)\right)_{f\in\mathcal{F}}$ ne paraissent pas pouvoir constituer les composantes de vecteurs $\overrightarrow{v}$ et $\overrightarrow{p}$, dans quel espace vectoriel, pour quel corps, relativement à quelle base orthonormée?

- Les "composantes" proposées ne bénéficient pas des propriétés indispensables de linéarité par rapport à la somme et au produit externe. En particulier on ne voit pas comment : $p\left(\lambda.f\right)=\lambda p(f)$.

En conséquence, les interprétations ultérieures d'alignements, de projections qui toutes deux sont intrinsèquement liées à une base orthogonale sous-jacente, voire à la norme déduite, souffrent considérablement à moins d'envisager les choses autrement.

Une formulation affine

Il est possible d'interpréter la somme $\sum_{f\in\mathcal{F}}p\left(d_m^y|f,m,a\right).p\left(f\right)$ de façon affine, en termes de barycentres. La distribution de probabilité sur $\mathcal{F}$ constitue, dans cette approche, des masses de somme 1, parfaitement adaptées à la situation. On retrouve alors l'interprétation souhaitée, grâce au plongement dans l'espace $\mathbb{R}^{|\mathcal{F}|}$ et la prise en considération des coordonnées du barycentre étudié.

On pose donc :

$$\overrightarrow{v}_{d_m^y,a,m}=\left(p\left(d_m^y|f,m,a\right)\right)_{f\in\mathcal{F}}\quad\text{et}\quad\overrightarrow{p}=\left(p(f)\right)_{f\in\mathcal{F}}. \tag{1.12}$$

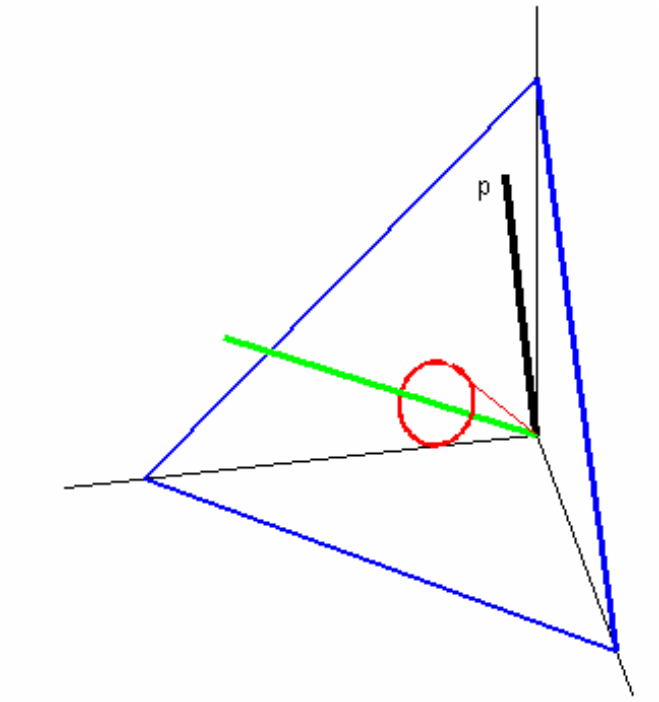

Figure 1.1: Interprétation géométrique en dimension 3

On retrouve alors l'interprétation en termes de produit scalaire, voulue par les auteurs :

$$p\left(d_m^y|m,a\right) = \overrightarrow{v}_{d_{m,a,m}^y}.\overrightarrow{p} \tag{1.13}$$

Eléments d'interprétation

On se propose d'interpréter géométriquement le théorème du NFL; voir figure 1.1.

- La relation 1.13 exprime que la performance d'un algorithme a est mesurée par la taille de la projection de $\overrightarrow{v}_{d_m^y,a,m}$ sur $\overrightarrow{p}$, autrement dit, est directement liée à la manière dont le vecteur $\overrightarrow{v}_{d_m^y,a,m}$ est aligné avec le vecteur $\overrightarrow{p}$ considéré.

  De façon similaire, on voit facilement que les relations antérieures "passent" en termes d'espérance mathématique. En considérant une moyenne sur les d_m^y, l'espérance mathématique $E\left(d_m^y|m,a\right)$ s'écrit comme le produit scalaire suivant:

$$E\left(d_m^y|m,a\right) = \overrightarrow{E\left(d_m^y|f,m,a\right)}.\overrightarrow{p} \tag{1.14}$$

  Ce type de résultat peut s'étendre à toute fonction de mesure de performance $\Phi\left(d_m^y\right)$.

- Les théorèmes du NFL, en affirmant que la somme $\sum_f p\left(d_m^y|m,a\right)$ est indépendante de a, énoncent que, pour tout choix particulier de d_m^y et m, tous les vecteurs $\overrightarrow{v}_{d_m^y,a,m}$ ont la même projection sur le vecteur diagonal $\overrightarrow{1}$. Ainsi pour tout a, le produit scalaire $\overrightarrow{v}_{d_m^y,a,m}.\overrightarrow{1}$ est une constante $c\left(d_m^y,m\right)$, dépendant des seuls d_m^y et m.

- Dans le cas des algorithmes déterministes,

  - la probabilité qu'un algorithme a donne un échantillon d_m^y après m itérations, sera égale à 0 ou 1. Par suite la somme $\sum_f p\left(d_m^y|m,a\right)$ s'écrit:

$$\sum_f p\left(d_m^y|m,a\right) = \sum_f \left[p\left(d_m^y|m,a\right)\right]^2 = c\left(d_m^y,m\right). \tag{1.15}$$

    Ainsi, la norme du vecteur $\overrightarrow{v}_{d_m^y,a,m}$ est indépendante de a [6].

    Par suite des algorithmes différents vont produire des vecteurs $\overrightarrow{v}_{d_m^y,a,m}$ de même longueur et de même projection sur le vecteur $\overrightarrow{1}$. Ils décrivent

[6] ce qui pourra s'interpréter en termes de distance de Hamming par rapport à $\overrightarrow{1}$, en regardant la figure comme un hypercube booléen.

donc un cône d'axe le vecteur $\overrightarrow{1}$; telle est l'expression géométrique des théorèmes du NFL.

– Si de plus, on impose aux algorithmes considérés d'avoir la même probabilité de fournir un d_m^y donné, ils appartiendront à l'intersection de deux cônes: le premier, d'axe le vecteur $\overrightarrow{1}$, pour satisfaire le NFL, le second, d'axe le vecteur $\overrightarrow{p}$, pour exprimer la nouvelle contrainte.

L'extrémité des vecteurs $\overrightarrow{v}_{d_m^y,a,m}$ décrira une variété affine de dimension $|\mathcal{F}|-2$, en général. L'ajout de contraintes s'interprète comme une réduction des degrés de liberté géométriques.

- Comparer deux algorithmes a et a' revient essentiellement à comparer $\overrightarrow{v}_{d_m^y,a,m}.\overrightarrow{p}$ avec $\overrightarrow{v}_{d_m^y,a',m}.\overrightarrow{p}$ plutôt que comparer les vecteurs $\overrightarrow{v}_{d_m^y,a,m}$ et $\overrightarrow{v}_{d_m^y,a',m}$qui définissent a et a' de façon complète certes, mais en partie inutilement, comme le montre la relation 1.13.

  Il sera aussi possible de comparer deux algorithmes en termes de similarités de comportement relativement au vecteur $\overrightarrow{p}$.

- Hélas, si l'interprétation permet d'analyser, de visualiser une situation, elle ne permet pas de définir une stratégie de recherche, en raison des difficultés d'évaluation des grandeurs qui interviennent.

Remarque 5 *L'interprétation géométrique révèle finalement qu'il existe une sorte d'équivalence, au moins partielle, entre fonctions de coût et algorithmes, puisque ces deux entités peuvent être représentées de façon presque similaire. Ce phénomène est assez troublant; il sera confirmé dans le corps de l'article fondateur [WM97], mais aussi dans plusieurs papiers qui ont suivi.*

1.2.5 Applications de nature calculatoire des théorèmes du NFL

Certaines ont un intérêt pratique, d'autres un intérêt théorique. Le principe des preuves des résultats fournis sera toujours le même: obtenir la valeur des quantités étudiées par le choix d'un algorithme qui en facilite le calcul, en raison de l'invariance prouvée dans les théorèmes du NFL.

L'optimisation vue en termes de théorie de l'information

Pour des raisons de simplification de l'exposé mais aussi théoriques, au fond, les auteurs se limitent à la prise en considération pour une fonction de coût f, de l'ensemble des valeurs prises sans tenir compte de leur ordre de génération.

Considérons donc l'histogramme de f noté $\overrightarrow{c}$, défini par:

$$\overrightarrow{c} = \left(c_1, c_2, ..., c_{|Y|}\right), \tag{1.16}$$

sachant que c_i désigne le nombre de fois où la valeur $y_i = f(x_i)$ a été prise dans l'échantillon d_m^y.

La question est de savoir quelle part de l'ensemble des fonctions de coût a fourni un histogramme $\overrightarrow{c}$ donné, après production d'un échantillon de m valeurs lors de l'exécution d'une instance d'un algorithme évolutionnaire, par exemple.

En dépit du caractère apparemment insoluble de la question, les théorèmes du NFL fournissent une réponse; il suffit, en raison du comportement similaire des différents algorithmes, d'en choisir un qui permette le calcul proposé.

Théorème 6 *Pour tout algorithme, la part des fonctions de coût qui mènent à un histogramme particulier $\overrightarrow{c} = m\overrightarrow{\alpha}$ est définie par:*

$$\rho_f\left(\overrightarrow{\alpha}\right) = \frac{\binom{m}{c_1 c_2 ... c_{|Y|}} |Y|^{|X|-m}}{|Y|^{|X|}} = \frac{\binom{m}{c_1 c_2 ... c_{|Y|}}}{|Y|^m}, \tag{1.17}$$

où $\binom{m}{c_1 c_2 ... c_{|Y|}}$ désigne le coefficient du binôme généralisé.

De plus, pour m suffisamment grand, on obtient une approximation de $\rho_f(\overrightarrow{\alpha})$ donnée comme suit:

$$\rho_f(\overrightarrow{\alpha}) \simeq C(m,|Y|)\frac{\exp\left[mS(\overrightarrow{\alpha})\right]}{\prod_{i=1}^{|Y|}\sqrt{\alpha_i}} \tag{1.18}$$

où $C(m,|Y|)$ est une constante indépendante de $\overrightarrow{\alpha}$ et $S(\overrightarrow{\alpha})$ l'entropie de $\overrightarrow{\alpha}$.

Preuve. Seules les étapes cruciales sont indiquées ci-dessous.

- L'algorithme choisi pour le calcul consiste à visiter les points de X dans l'ordre x_1, x_2, ..., x_m.
- Le premier résultat du théorème revient à solutionner un problème assez classique de dénombrement.
- Pour obtenir la deuxième partie du théorème, c'est-à-dire une approximation de $\rho_f(\overrightarrow{\alpha})$, pour m assez grand, on applique la formule de Stirling, pourvu que tous les c_i soient grands. Il vient:

$$\ln\left[\binom{m}{c_1 c_2 ... c_{|Y|}}\right] \simeq m\ln(m) - \sum_{i=1}^{|Y|} c_i \ln(c_i) + \frac{1}{2}\left[\ln(m) - \sum_{i=1}^{|Y|}\ln(c_i)\right]$$

soit finalement:

$$\ln\left[\binom{m}{c_1 c_2 ... c_{|Y|}}\right] \simeq mS(\overrightarrow{\alpha}) + \frac{1}{2}\left[(1-|Y|)\ln(m) - \sum_{i=1}^{|Y|}\ln(\alpha_i)\right]. \tag{1.19}$$

Le résultat en découle par exponentiation [7].

■

Remarque 7 *Si certains α_i sont nuls, l'approximation précédente demeure, pourvu que l'on supprime de l'ensemble Y les y_i pour lesquels les α_i sont nuls.*

[7] La constante $C(m,|Y|)$ peut être calculée à partir de $\overrightarrow{\alpha}$ bien choisis.

On note $\overrightarrow{N}$ le vecteur $\left(N_1, N_2, ..., N_{|Y|}\right)$ où N_i désigne le cardinal $\left|f^{-1}\left(\{y_i\}\right)\right|$, puis on définit $\overrightarrow{\beta}$ via $\overrightarrow{N} = |X|\,\overrightarrow{\beta}$. Alors on a le résultat suivant.

Théorème 8 *Pour une fonction de coût donnée f dont l'histogramme vérifie $\overrightarrow{N} = |X|\,\overrightarrow{\beta}$, la part d'algorithmes qui conduisent à l'histogramme $\overrightarrow{c} = m\overrightarrow{\alpha}$ est donnée par:*

$$\rho_{alg}\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right) = \frac{\prod_{i=1}^{|Y|} \binom{N_i}{c_i}}{\binom{|X|}{m}}. \tag{1.20}$$

De plus, pour m suffisamment grand, on obtient une approximation de $\rho_{alg}\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right)$ donnée comme suit:

$$\rho_{alg}\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right) \simeq C\left(m, |X|, |Y|\right) \frac{\exp\left(-mD_{KL}\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right)\right)}{\prod_{i=1}^{|Y|} \sqrt{\alpha_i}}, \tag{1.21}$$

sachant que $D_{KL}\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right)$ désigne la distance de Kullback-Liebler [8] *entre $\overrightarrow{\alpha}$ et $\overrightarrow{\beta}$,*

et $C\left(m, |X|, |Y|\right)$ une constante indépendante de $\overrightarrow{\alpha}$, $\overrightarrow{\beta}$.

Mesures de performances

DH Wolpert et WG Macready utilisent ensuite les théorèmes du NFL pour fournir quelques repères en termes de performances, afin de situer par rapport à d'autres tel outil d'optimisation ou de comparer entre eux, différents algorithmes. Plus que des résultats particuliers, les théorèmes fournis ci-dessous constituent véritablement trois approches, trois méthodes de mesure de performances d'un algorithme donné, classiquement utilisées dans les applications.

[8] *Pour cette notion, voir [CT91].*

Pour toute cette section 1.2.5, sans affecter la généralité, les auteurs se placent dans le cas de la recherche de minima et supposent, de surcroît, que les valeurs de coût appartiennent à l'ensemble $\{1, ..., |Y|\}$.

Ils se proposent donc d'étudier la dépendance relativement à ε, de la quantité $p\left(\left(\min\left(\overrightarrow{c}\right) > \varepsilon\right)|f, m, a\right)$, probabilité que le coût minimum d'un algorithme a pour résoudre un problème au sens de f en m évaluations, dépasse la valeur ε.

Remarque 9 *Signification de ε*

Ici ε désigne un entier naturel donné, vu comme le nombre de fois où une valeur est prise par la fonction f.

Il convient de noter que l'évènement $E : \left(\min\left(\overrightarrow{c}\right) > \varepsilon\right)$ admet pour complémentaire $\overline{E} : \left(\min\left(\overrightarrow{c}\right) \leq \varepsilon\right)$.

a) Résultat relatif à l'ensemble des fonctions de coût

Les fonctions de coût sont ici supposées équiprobables.

Théorème 10 *Sous les hypothèses et notations précédentes, on a:*

$$\sum_{f} p\left(\min\left(\overrightarrow{c}\right) > \varepsilon | f, m\right) = \left[\omega\left(\varepsilon\right)\right]^{m}, \tag{1.22}$$

où $\omega\left(\varepsilon\right) = 1 - \varepsilon / |Y|$ représente la part des coûts supérieurs à ε.

De plus, lorsque $|Y|$ tend vers l'infini, on a l'équivalence suivante, sachant que E désigne l'espérance mathématique de la variable aléatoire considérée:

$$\sum_{f} E\left(\min\left(\overrightarrow{c}\right)|f, m\right) \underset{|Y| \to +\infty}{\sim} \frac{|Y|}{m+1}. \tag{1.23}$$

b) Résultat relatif à la comparaison avec une recherche aléatoire

Afin de permettre la comparaison d'un algorithme donné avec une procédure choisissant les points au hasard, il est intéressant de disposer d'un résultat pour ce type d'algorithme.

On pose à nouveau $\overrightarrow{N} = |X| \overrightarrow{\beta}$, notation dont le sens est, rappelons le, précisé suite à la remarque 7.

Théorème 11 *Pour l'algorithme de recherche aléatoire, noté $\widetilde{a}$, on a:*

$$p\left(\min\left(\overrightarrow{c}\right) \geq \varepsilon | f, m, \widetilde{a}\right) = \prod_{i=0}^{m-1} \frac{\Omega\left(\varepsilon\right) - \frac{i}{|X|}}{1 - \frac{i}{|X|}}, \tag{1.24}$$

sachant que $\Omega\left(\varepsilon\right) = \sum_{i=\varepsilon}^{|Y|} \frac{N_i}{|X|}$ désigne la part des points de X pour lesquels: $f\left(x\right) \geq \varepsilon$.

De plus, on a l'équivalent suivant:

$$p\left(\min\left(\overrightarrow{c}\right) > \varepsilon | f, m, \widetilde{a}\right) \underset{|X| \to +\infty}{\sim} \Omega^m\left(\varepsilon\right) \left[1 - \frac{m\left(m-1\right)\left(1 - \Omega\left(\varepsilon\right)\right)}{2\Omega\left(\varepsilon\right)} \frac{1}{|X|}\right]. \tag{1.25}$$

Ce résultat permet la détermination d'autres propriétés utiles dans le domaine des mesures de performances, comme par exemple:

$$E\left(\min\left(\overrightarrow{c}\right) | f, m, \widetilde{a}\right) = \sum_{\varepsilon=1}^{|Y|} \varepsilon \left[p\left(\min\left(\overrightarrow{c}\right) \geq \varepsilon | f, m, \widetilde{a}\right) - p\left(\min\left(\overrightarrow{c}\right) \geq \varepsilon + 1 | f, m, \widetilde{a}\right)\right]. \tag{1.26}$$

Dans de nombreuses applications, les valeurs de coût sont approximativement réparties selon une loi gaussienne. On obtient par là des résultats intéressants en termes théoriques et pratiques, en prenant en compte cette hypothèse; cela permet de faciliter et expliciter certains calculs.

c) Résultat relatif à l'ensemble des algorithmes

Après un travail d'approche technique conséquent, DH Wolpert et WG Macready établissent le théorème suivant.

Théorème 12 *Pour f et m donnés, la part des algorithmes qui aboutissent à un histogramme $\overrightarrow{c}$ dont le minimum dépasse ε, est donnée par les seconds membres des relations 1.24 et 1.25 précédentes.*

d) Des preuves complètes mais délicates !

Les détails des preuves de ces théorèmes sont fournis dans les annexes du papier [WM97]. Elles ne sont pas simples, résolument techniques et difficiles, mais précises. Elles font appel à des résultats classiques de dénombrement, à des développements limités au voisinage de l'infini utilisant, par exemple la règle de L'Hôpital mais aussi l'expression d'une quantité reconnue comme somme de Riemann, et par suite évaluée via un calcul intégral...

e) Prolongements

DH Wolpert et WG Macready montrent dans une conclusion assez détaillée et pratique, comment ce groupe de résultats permettra des études comparatives et fines de modalités de convergence.

1.2.6 Différences de type minimax entre algorithmes

L'apport des théorèmes du NFL n'est pas direct à ce sujet comme le reconnaissent les auteurs et comme le laissent prévoir leurs énoncés eux-mêmes; ils expriment en effet un résultat en moyenne sur l'ensemble des f. Il peut en effet très bien exister des fonctions de coût f telles que l'histogramme issu d'un algorithme a_1 soit nettement meilleur, au sens d'une mesure de performances déterminée, que celui d'un algorithme a_2 sans qu'on connaisse, si elles existent, des fonctions de coût pour lesquelles l'inverse est vrai. Il peut très bien exister beaucoup plus de fonctions f pour lesquelles l'algorithme a_2 est meilleur que a_1, ou l'inverse, mais ceci peut n'être vrai que pour certaines fonctions f. Dans une telle situation, a_1 a un meilleur "comportement minimax" que a_2, en "tête-à-tête", si l'on peut dire. Il peut exister des fonctions f pour lesquelles a_1 fait moins bien que a_2, mais pas de f pour lesquels a_1 est nettement plus mauvais que a_2.

Définition 13 *On dit que les algorithmes a_1 et a_2 présentent des différences de type minimax, "en tête-à-tête" , s'il existe un réel non nul k pour lequel on peut trouver au moins une fonction de coût f telle que:*

$$E\left(\overrightarrow{c}|f,m,a_1\right) - E\left(\overrightarrow{c}|f,m,a_2\right) = k \ ,$$

sans qu'il existe de fonction de coût g pour laquelle:

$$E\left(\overrightarrow{c}|g,m,a_2\right) - E\left(\overrightarrow{c}|g,m,a_1\right) = k.$$

Analyser des algorithmes au sens des différences de type minimax, en tête-à-tête, apparaît autrement difficile que comparer leur comportement en moyenne, comme le font les théorèmes du NFL, spécialement en ce qui concerne les algorithmes stochastiques. On peut toutefois établir le résultat suivant.

Considérons la quantité:

$$\sum_f p_{d^y_{m,1},d^y_{m,2}}\left(z_1,z_2|f,m,a_1,a_2\right), \tag{1.27}$$

pour des algorithmes déterministes a_1 et a_2, sachant qu'on convient que $p_V(v)$ désigne la probabilité qu'une variable aléatoire V prenne la valeur v. Pour des algorithmes déterministes, cette quantité représente le nombre de fonctions f pour lesquelles on a, à la fois, le fait que a_1 produit un échantillon de vecteur de coût z_1 et a_2 un échantillon de vecteur de coût z_2.

Théorème 14 *En général, il existe une disymétrie entre les algorithmes a_1, a_2 utilisés et les vecteurs de coût produits z_1 et z_2:*

$$\sum_f p_{d^y_{m,1},d^y_{m,2}}\left(z_1,z_2|f,m,a_1,a_2\right) \neq \sum_f p_{d^y_{m,1},d^y_{m,2}}\left(z_2,z_1|f,m,a_1,a_2\right). \tag{1.28}$$

Preuve. Les auteurs exhibent un exemple, pour lequel $|X| = |Y| = 3$, qui met en évidence la différence entre les deux sommes proposées. La preuve est très fine; elle repose sur le fait qu'aucune des quatre conditions qui assureraient l'égalité attendue, n'est possible. ■

Remarque 15 *En conséquence, s'il y a disymétrie, on doit pouvoir obtenir des renseignements sur les algorithmes utilisés à partir des populations produites.*

On peut s'intéresser non plus aux d_m^y , mais plutôt aux histogrammes des fonctions de coût f, c'est-à-dire aux sommes

$$\sum_f p_{\overrightarrow{c_1},\overrightarrow{c_2}}\left(z_1, z_2 | f, m, a_1, a_2\right). \tag{1.29}$$

Il semble pertinent d'étudier les liens entre, le caractère disymétrique des sommes antérieures lors de la permutation de z_1 et z_2, avec l'existence de différences en termes de minimax, en "tête-à-tête". DH Wolpert et WG Macready fournissent des exemples et contre-exemples relatifs à ces liens, sans faire apparaître pourtant de propriétés vraiment générales. Le fait que les échantillons $d_{m,1}^x, d_{m,2}^x$ partagent ou non des points, se "recoupent ou non", paraît revêtir une importance cruciale. Ici encore on sent affleurer les résultats déjà évoqués plus haut, relatifs à la stabilité ou non par permutations, établis après la parution du papier initial [WM95].

Théorème 16 *S'il n'existe pas de recoupements entre $d_{m,1}^x$ et $d_{m,2}^x$, alors:*

$$\sum_f p_{d_{m,1}^y, d_{m,2}^y}\left(z_1, z_2 | f, m, a_1, a_2\right) = \sum_f p_{d_{m,1}^y, d_{m,2}^y}\left(z_2, z_1 | f, m, a_1, a_2\right). \tag{1.30}$$

Preuve. Elle consiste à se rapporter à la preuve générale des théorèmes du No Free Lunch. ■

1.2.7 Résultats indépendants de $p(f)$

Une approche nouvelle

Contrairement à ce qui a été fait précédemment et concernait les propriétés des algorithmes sur l'ensemble des fonctions de coût f, l'objectif est ici de s'intéresser aux propriétés communes de différents algorithmes pour la même fonction f. D'autres résultats que ceux évoqués ci-dessous, sont fournis à ce propos dans [MW96].

On considère deux algorithmes de recherche a et a'. On appelle "procédure de choix" le fait d'énoncer une règle qui, basée sur l'examen des échantillons d_m et d'_m produits par a et a' respectivement, permet de décider de poursuivre la recherche en utilisant a ou a'. Dans ces conditions, la fonction de coût doit prendre en compte l'échantillon $d_U = d_m \cup d'_m$. On désigne par $d_{>m}$ l'échantillon des valeurs de la fonction de coût, apparues après la procédure de choix; de même $c_{>m}$ désigne l'histogramme associé. On suppose encore, sans perte de généralité, que l'algorithme retenu par cette procédure de choix ne renvoie aucune des valeurs de d_U [9].

Les résultats

Le théorème qui suit, établi pour des raisons de simplicité sur les seuls algorithmes déterministes, montre qu'il n'existe pas de justification a priori, d'utiliser une procédure de choix particulière, plutôt qu'une autre, en moyenne !

Théorème 17 *Soit d_m et d'_m des échantillons produits par les algorithmes a et a' respectivement, relatifs à une fonction de coût quelconque f.*

Soit A et B deux procédures de choix quelconques. On note k le nombre d'éléments de $d_{>m}$. Alors:

$$\sum_{a,a'} p\left(c_{>m}|f,d,d',k,a,a',A\right) = \sum_{a,a'} p\left(c_{>m}|f,d,d',k,a,a',B\right). \tag{1.31}$$

Il est implicite que la sommation ne prend pas en compte ceux des algorithmes a et a' qui ne produiraient pas leurs résultats relatifs à f, dans d et d'.

Preuve. Les détails sont fournis dans l'annexe G de [WM97]. L'essentiel est d'interpréter la somme proposée grâce à l'analyse de la forme des échantillons à retenir parmi les possibles; ils doivent contenir les éléments de d, d'. Les échantillons convenables sont donc construits à partir de ces éléments premiers, puis complétés au

[9] Il est possible et même nécessaire d'affiner légèrement cette condition.

sens d'une certaine orthogonalité qui permet de simplifier les écritures et de parvenir au résultat. ■

Théorème 18 *Sous les hypothèses et notations du théorème 17,*

$$\sum_{a,a'} p\left(c_{>m}|f,m,k,a,a',A\right) = \sum_{a,a'} p\left(c_{>m}|f,m,k,a,a',B\right). \tag{1.32}$$

Preuve. La preuve repose sur la définition des probabilités conditionnelles; elles permettent de faire apparaître un terme multiplicatif, qui dans le cadre de l'emploi d'algorithmes déterministes, se réduit à 0 ou 1. On remarque alors que cette dernière valeur est celle prise pour les seuls algorithmes qui doivent intervenir dans la somme; d'où le résultat. ■

En conclusion

DH Wolpert et WG Macready font remarquer que les égalités obtenues ne dépendent pas de $p\left(f\right)$ pour la fonction de coût considérée, ce qui peut paraître étrange, en première lecture. Ils analysent, de façon très intéressante, ce fait en détail et terminent en citant quelques questions connexes et une référence présentant des résultats liés.

1.2.8 Bref bilan et présentation des réactions à l'article initial

Nous avons signalé en introduction la richesse du papier de DH Wolpert et WG Macready; l'ensemble est effectivement impressionnant: la liste des sections décrites ci-dessus en atteste. Il convient de noter que la version première [WM95] était un peu plus limitée.

En conséquence, il n'est pas étonnant que cette contribution majeure ait déclenché des réactions très diverses, elles aussi profondes, venues étayer et prolonger la construction scientifique initiée.

Nous allons présenter maintenant les contributions essentielles, à nos yeux, qui ont suivi ce premier article. Nous avons retenu et présenterons successivement:

- Une confirmation expérimentale, à la section 1.3, grâce au papier de S Droste, T Jansen, I Wegener [DJW99].

  Elle sera l'occasion d'une confrontation avec la pratique: un ensemble de fonctions de coût sera découpé en classes liées à leur difficulté d'évaluation.

- Une analyse des liens existant entre algorithmes de recherche, statistiques descriptives, théorèmes du NFL, à la section 1.4, grâce au papier de D Whitley [W00].

  Nous tenterons de porter un regard transversal sur l'ensemble des problématiques puisque tel est un des rôles de cet outil dans la pratique des chercheurs: contrôler "extérieurement", de façon implacable les performances d'une méthode. Nous y traiterons de la question suivante: "Qu'attendre et ne pas attendre des statistiques ?"

  Le remarquable outil que constitue l'analyse de Walsh, y sera partiellement présenté dans une tentative de formalisation simplifiée et standard.

- Une étude des champs de validité des théorèmes du No Free Lunch, à la section 1.5.

  Nous y verrons l'émergence du lien avec les permutations de l'espace de recherche X, pressenti d'ailleurs dès l'article initial, mais approfondi, finalisé sous la forme d'une propriété caractéristique. Dans cette marche vers la caractérisation des champs de validité, s'effectue en même temps une simplification des définitions, un recentrage sur l'essentiel. Ces différents aspects seront abordés grâce aux articles de:

  - C Schumacher, MD Vose, D Whitley [SVW01] à la sous-section 1.5.1: mise en place des notations, définitions et résultats fondateurs.
  - Igel et Toussaint [IT01] à la sous-section 1.5.2: les ensembles de fonctions clos par permutation sont rares, les voisinages dans X ne sont en général pas invariants par permutation.
  - S Droste, T Jansen et I Wegener [DJW02] à la sous-section 1.5.3: simplification des définitions, des preuves, des concepts même. Sans doute, la forme la plus aboutie.

  On pourra croire à ce stade, en raison de la "petite taille" en général, des zones de validité des théorèmes du No Free Lunch que la peur qu'ils avaient fait naître était inutile, voire artificielle. Hélas, ce serait ignorer la présence d'effets "presque No Free Lunch". Les craintes n'étaient pas vaines: le quasi-NFL agit partout !

- Une approche de la "version locale" des théorèmes du No Free Lunch, à la section 1.6, sous le regard du papier de S Droste, T Jansen et I Wegener [DJW02].

  - Nous y découvrirons qu'autour d'une fonction de coût donnée f, existent un grand nombre de fonctions "voisines", sur l'ensemble desquelles le No Free Lunch est quasi-valide.
  - Les auteurs feront le lien avec des questions bien réalistes, relevant du quotidien des praticiens de l'optimisation. Ils exhibent en particulier, une classe de fonctions de coût qui mettront irrémédiablement en difficulté, algorithmes de recuit simulé et évolutionnaires.

1.3 Confirmation expérimentale des théorèmes du NFL

1.3.1 Introduction

La prise en compte de l'aspect pratique et expérimental nous semble crucial, tout comme sa confrontation avec les théorèmes généraux. Cette double préoccupation constituera l'ossature du propos, en deux occasions, dans cet état de l'art.

A la fin de ce document d'une part, dans la section 1.6 consacrée au quasi-NFL, véritable version locale du No Free Lunch. Nous ne voulons pas détailler ici cet apport, en dépit d'un lien naturel, pour ne pas le retirer de son contexte essentiel. Nous tenons par contre à annoncer ce fait, dès maintenant.

Ici même d'autre part, pour confirmer expérimentalement les premiers théorèmes du NFL. En 1999, paraît *"Perhaps Not a Free Lunch But At Least a Free Appetizer"* [DJW99]. S Droste, T Jansen et I Wegener, interrogés par les théorèmes de DH Wolpert et WG Macready, mais aussi par l'excellence promise des algorithmes évolutionnaires, y développent une réflexion sur ces questions. Les auteurs disent regretter le côté étriqué de leurs expérimentations; bien que limités, leurs résultats nous paraissent très importants.

1.3.2 Le cadre expérimental

S Droste, T Jansen et I Wegener considèrent l'ensemble $\mathcal{F} = Y^X$ où $X = \{0,1\}^3$ et $Y = \{0,1\}^2$. Par suite $|\mathcal{F}| = 4^8 = 65536$. Les couples de Y représentent l'écriture binaire d'entiers, qui ordonnent Y.

Les auteurs définissent dans $\mathcal{F}$ des sous-ensembles, grâce au concept de OBBD (Ordered Binary Decision Diagrams) introduit par Bryant en 1986. Cette notion est destinée à mesurer la difficulté d'évaluation d'une fonction de $\mathcal{F}$, via celle de son écriture. Nous ne fournissons pas ici de définition générale mais nous limitons à un exemple proposé dans [DJW99]; on pourra se reporter à [B86].

Considérons la fonction f de $\mathcal{F}$ définie comme suit:

$$(x_1, x_2, x_3) \in \{0,1\}^3 \mapsto (f_1(x_1, x_2, x_3), f_2(x_1, x_2, x_3)) \in \{0,1\}^2 \tag{1.33}$$

sachant que:

$$[f_1(x_1, x_2, x_3) = 1] \Leftrightarrow [(x_1, x_2, x_3) \in \{(0,0,1), (0,1,1), (1,0,0), (1,1,1)\}] \tag{1.34}$$

et

$$[f_2(x_1, x_2, x_3) = 1] \Leftrightarrow [(x_1, x_2, x_3) \in \{(0,0,1), (1,0,1), (1,1,0), (1,1,1)\}]. \tag{1.35}$$

f peut alors être représentée par le diagramme de la figure 1.2.

Ce diagramme est lié au triplet (x_2, x_3, x_1) des variables, puisque c'est dans cet ordre que s'effectue, ici, l'évaluation de la fonction f en un point de $\{0,1\}^3$. Le diagramme contient six noeuds internes (cerclés); on dit qu'il est de taille 6.

On appelle alors taille de f (en termes de OBBD) le minimum des tailles des diagrammes OBBD représentant f parmi les $\left(|3|!, \text{puisque } X = \{0,1\}^3\right)$ permutations des variables possibles; ce minimum existe et on peut le déterminer.

On note alors $(F_i)_{0 \leq i \leq 8}$ la famille des parties F_i de $\mathcal{F}$, telles que les fonctions de F_i aient une taille d'OBBD majorée par i. $(F_i)_{0 \leq i \leq 8}$ n'est évidemment pas une partition de $\mathcal{F}$ mais $\mathcal{F} = F_8$.

1.3.3 Objectifs poursuivis, méthode

S Droste, T Jansen et I Wegener se proposent de comparer les performances, sur les différents sous-ensembles F_i, de trois types d'algorithmes d'optimisation des fonctions f de $\mathcal{F}$. Les algorithmes s'arrêtent dès que l'optimum de la fonction f est trouvé; on compte le nombre d'arguments en lequel f a été évaluée avant d'atteindre l'optimum.

La mise en oeuvre expérimentale est très soignée; en particulier les algorithmes objets de l'étude, opèrent sur une classe de fonctions autre que les F_i, en vue de

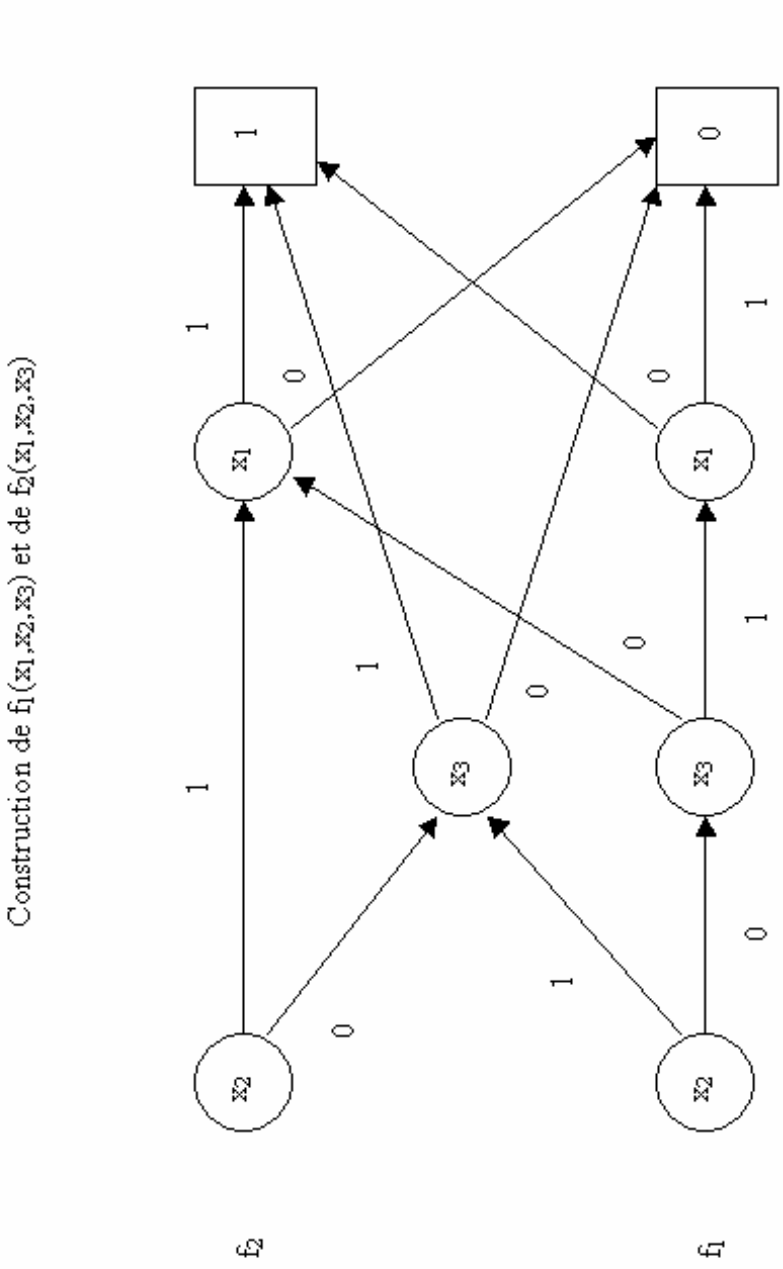

Figure 1.2: Diagramme OBBD associé à f pour (x_2, x_3, x_1)

mettre en évidence un biais éventuel relatif au choix de la représentation; nous n'en donnons pas les résultats ici. D'autres précisions sont fournies dans la bibliographie liée à [DJW99] quant aux modalités des calculs menés, garantes de leur qualité.

- Le premier algorithme étudié est un non-adaptatif. Vu qu'un tel algorithme est intrinsèquement lié à une permutation des éléments de l'espace de recherche, l'étude est menée sur l'ensemble des permutations de X, qui sont au nombre de $8! = 40320$.

- En second lieu, les auteurs mettent en oeuvre deux algorithmes évolutionnaires nommés $EA_{\geq}$ et $EA_{>}$, variantes d'algorithmes étudiés par Rudolph en 1997, retenues en raison de leur simplicité [R97]. Les nombres de points des échantillons utilisés ont été déduits par calculs, après étude des chaînes de Markov associées aux algorithmes considérés.

- Enfin, sont étudiés deux algorithmes "classiques et spécifiques opérant sur les OBBD", nommés $MaxOpt$ et $MinMax$.

1.3.4 Résultats obtenus

Les têtes de colonnes portent l'indication du nom des ensembles (set) étudiés, de leur cardinal ($|set|$), des algorithmes étudiés $EA_{\geq}$, $EA_{>}$, $MaxOpt$, $MinMax$. Pour ce qui concerne le premier type d'algorithmes, les résultats sont identifiés grâce au préfixe na; pour l'ensemble de ceux-ci, sont déterminés la moyenne $na.moy$ des coûts de chacun, la moyenne $na.min$ de leurs plus faibles coûts, la moyenne $na.max$ des plus forts.

Les lignes comportent les noms des ensembles étudiés de F_0 jusqu'à F_8.

A chaque intersection de la table se trouvent les valeurs moyennes arrondies à la cinquième décimale, obtenues pour chaque ensemble, lors de la mise en oeuvre de l'algorithme défini en tête de colonne. Voir figure 1.3.

Résultats obtenus

set	|set|	EA>=	EA>	na.moy	na.min	na.max	MaxOpt	MinMax
F_0	4	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000
F_1	34	1,79288	1,79594	1,70588	1,44118	2,02941	1,44118	1,44118
F_2	280	2,29274	2,29443	2,24408	1,96071	2,52143	1,94643	2,00714
F_3	2166	2,50852	2,50887	2,48673	2,26362	2,69575	2,24331	2,23223
F_4	8908	2,87136	2,87085	2,86225	2,69185	3,00707	2,64111	2,64042
F_5	25694	2,98338	2,98287	2,97920	2,86974	3,05943	2,83518	2,84600
F_6	51520	3,02621	3,02609	3,02528	2,98453	3,04926	2,96555	2,97977
F_7	65144	3,06392	3,06387	3,06392	3,06126	3,06499	3,05928	3,06219
F_8	65536	3,06369	3,06369	3,06369	3,06369	3,06369	3,06369	3,06369

Figure 1.3: Confirmation expérimentale du NFL

1.3.5 Bilan relatif à l'expérimentation proposée

- On peut noter, en se limitant à la seule optique de notre étude actuelle, que les résultats obtenus par ces algorithmes de nature bien différentes confirment le résultat théorique annoncé par les théorèmes du No Free Lunch, en raison de l'uniformité des moyennes obtenues.
- On peut établir quelques comparaisons plus fines entre les différents algorithmes, au vu du tableau de résultats, dans ce cas restreint.
- L'exemple proposé est certes limité mais il a le mérite d'exister; il est traité avec un grand soin.

Remarque 19 *En raison de la taille de l'ensemble étudié, l'uniformité des résultats pour F_8 à la cinquième décimale, indique une similitude des coûts globaux tout à fait remarquable !*

1.4 Algorithmes de recherche et statistiques descriptives

Dans Parallel Problem Solving from Nature - PPSN VI, pages 169-178, Berlin, 2000. Springer, paraît l'article de D. Whitley *"Functions as Permutations: Implications for No Free Lunch, Walsh Analysis and Statistics"* [W00]. Utilisant l'excellent outil que représente l'analyse de Walsh, il obtient des résultats fort intéressants quant au rapport entre moments statistiques et conduite des algorithmes de recherche.

1.4.1 Les fonctions de coût comme permutations

Introduction et notations

Soit X un ensemble de n points distincts et Y son image par une fonction de coût, supposée bijective [10].

[10] Cette situation correspond au cadre théorique adopté en général, de fait.

Soit Π l'ensemble des permutations de Y; Π représente donc l'ensemble de toutes les fonctions de coût qui peuvent être construites sur Y.

On peut aussi utiliser Π pour représenter tous les algorithmes de recherche qu'on peut appliquer sur X relativement à l'ensemble des fonctions de coût; en effet un algorithme de recherche sur X peut être vu comme l'ordre dans lequel les valeurs de Y sont visitées [11]. Si différentes procédures de recherche parcourent les points de Y dans le même ordre, elles font partie d'une classe d'équivalence.

Sous ces hypothèses, algorithmes et fonctions de coût sont considérés comme des permutations de Y. Sur leur ensemble, on sait que les théorèmes du NFL sont valides (voir la section 1.5). Par suite, en moyenne, aucun algorithme n'est meilleur que le tirage au hasard pour localiser un optimum !

Proposition 20 *La numérotation des $n!$ éléments de Π nécessite en moyenne $\mathcal{O}(n \ln(n))$ bits, où $n = |X|$, désigne le nombre de points de l'espace de recherche.*

Proposition 21 *Tout élément σ de Π, où σ désigne une permutation de $\{1, ..., n\}$, nécessite $\mathcal{O}(n \ln(n))$ bits, pour être décrite.*

Preuve. Les deux preuves sont classiques. ■

D Whitley en déduit donc qu'une méthode de type NP-complet ne peut être utilisée pour générer l'ensemble des $n!$ fonctions considérées. Sont par exemple concernés les $nk-$Landscapes et $MAXSAT$.

Il rappelle le questionnement qui entoure les théorèmes du NFL, depuis leur intrusion dans le paysage... Sont-ils des spéculations gratuites, peuvent-ils apporter quelque chose en pratique, pour ceux qui travaillent en optimisation? Les preuves sont là, mais les ornières de la pensée toujours profondes, et les précautions sans doute indispensables !

[11] On retrouve ici la similitude entre fonctions de coût et algorithmes de recherche notée lors de l'interprétation géométrique des théorèmes du NFL dans [WM97]. Voir la remarque 5.

Nous proposons ci-dessous une version "standardisée", de la seule partie de l'analyse de Walsh qui nous est utile. Au lieu de besogner pour prouver que certaines quantités sont nulles, il nous paraît préférable de mettre en évidence qu'un produit scalaire est à l'oeuvre; les résultats gagnent alors en lisibilité et interprétation.

On considère l'ensemble $\mathcal{L}$ des fonctions définies sur l'ensemble $\{0,1\}^L$ des $L-$uplets de nombres binaires ($L \in \mathbb{N}^*$), à valeurs réelles. $\mathcal{L}$ est un espace vectoriel réel de dimension 2^L, isomorphe à $\mathbb{R}^{2^L}$.

Définition 22 *On appelle fonctions de Walsh, les éléments de $\mathcal{L}$ définis pour tout j de $\left\{0,...,2^L-1\right\}$ par:*

$$\begin{array}{rccc} \psi_j: & \{0,1\}^L & \rightarrow & \{-1,1\} \\ & x & \mapsto & \psi_j(x) = (-1)^{j\left({}^t x\right)} \end{array}, \tag{1.36}$$

où j est confondu dans le produit matriciel $j\,({}^t x)$, avec la matrice ligne de sa traduction binaire [12].

Remarque 23 *La fonction ψ_j permet de visualiser et étudier l'effet des seules composantes de x correspondant à la présence des 1 dans l'écriture de j.*

Proposition 24 *Propriétés des ψ_j*

- *Pour tout j de $\left\{0,...,2^L-1\right\}$ les ψ_j sont des mophismes du groupe $\left(\left(\mathbb{Z}/2\mathbb{Z}\right)^L,+\right)$ dans $\left(\{-1,1\},\times\right)$ et l'image réciproque H_j de 1 par ψ_j est un sous-groupe de $\left(\left(\mathbb{Z}/2\mathbb{Z}\right)^L,+\right)$.*
- *En particulier:*
  - *Si $j=0$, pour tout x de $\{0,1\}^L$, $\psi_0(x)=1$;*

[12] *Cette ambivalence est ordinairement utilisée dans la suite, sans être signalée désormais!*

– *Si* $j \neq 0$, $|H_j| = \frac{1}{2}\left|\{0,1\}^L\right|$, *autrement dit, il y a autant de* $L-$*uplets d'image* 1 *par* ψ_j *que de* $L-$*uplets d'image* -1.

Théorème 25 *Transformée de Walsh*

Sous les notations précédentes, on a les résultats suivants.

1. *Sur* $\mathcal{L} \times \mathcal{L}$ *on définit la fonction* φ *à valeurs réelles par:*

$$\varphi\left[(f,g)\right] = \langle f, g\rangle = \sum_{x=0}^{2^L-1} f(x)g(x); \tag{1.37}$$

alors φ *est un produit scalaire.*

2. *Dans l'espace euclidien* $(\mathcal{L}, \varphi)$, *la famille* $\left(\psi_j\right)_{0\leq j\leq 2^L-1}$ *est orthogonale, mais non normée car*

$$\forall j \in \left\{0, ..., 2^L - 1\right\} \qquad \left\langle \psi_j, \psi_j \right\rangle = \left\|\psi_j\right\|^2 = 2^L. \tag{1.38}$$

En conséquence $\left(\psi_j\right)_{0\leq j\leq 2^L-1}$ *est une base de* $\mathcal{L}$, *dite base de Walsh.*

3. *On peut écrire pour tout* x *de* $\{0,1\}^L$ *et toute fonction* f *de* $\mathcal{L}$:

$$f(x) = \sum_{j=0}^{2^L-1} w_j \psi_j(x); \tag{1.39}$$

de plus les 2^L *réels* w_j, *dits coefficients de Walsh de* f, *sont obtenus grâce à la transformation de Walsh, c'est-à-dire donnés par:*

$$w_j = \frac{1}{2^L} \sum_{x=0}^{2^L-1} \psi_j(x) f(x). \tag{1.40}$$

Preuve. On pourra consulter [S98] mais aussi les nombreux liens renvoyant à l'utilisation de ces fonctions dans le domaine des algorithmes génétiques, par exemple. Toutefois, la preuve des résultats précédents, très algébrique, n'est en général pas fournie; nous en fournissons une ci-dessous, en annexe de ce chapitre, à la section 1.8. ■

Proposition 26 *Détermination pratique des coefficients de Walsh*

On note F [respectivement W] les matrices colonnes définies par:

$$F =^t \left(f(0)\ \ f(1)...f\left(2^L-1\right)\right) \quad \textit{[respectivement } W =^t (w_0\ w_1...w_{2^L-1})\textit{]},$$

et U la matrice carrée de côté 2^L définie par:

$$U = (U_{ij}) \quad \textit{avec } U_{ij} = \psi_i(j).$$

Alors les coefficients de Walsh sont donnés par:

$$W = \frac{1}{2^L} U \times F. \tag{1.41}$$

Exemple 27 *L'auteur propose l'étude du cas particulier $L = 3$; il détermine explicitement la matrice U, carrée de côté 8.*

1.4.2 Statistiques descriptives pour instances de problèmes

L'analyse de Walsh peut être utilisée dans le cadre de problèmes d'optimisation discrets. D'abord Goldberg et Rudnick ont utilisé les coefficents de Walsh pour le calcul de variance des coûts. Pour les détails, voir [GR91].

Puis Heckendorn, Rana and Whitley étendent les premiers résultats grâce à une relation permettant d'exprimer le moment d'ordre r d'une distribution à partir des coefficients de Walsh [HRW99(1)]. Ils établissent aussi que pour certains types de fonctions, les coefficients de Walsh et les divers moments statistiques peuvent être calculés en un temps polynômial.

Expression d'un moment d'ordre r en fonction des coefficients de Walsh

Le principe de la preuve du résultat est fort bien présenté dans [W00]. Nous ne décrivons ici que les grandes lignes du raisonnement.

On considère un problème relatif à 2^L entrées, équiprobables; pour tout x de $\{0,1\}^L$, on a donc $p(x) = 1/2^L$. Pour tout x de $\{0,1\}^L$, $\psi_0(x) = 1$ donc la moyenne μ de la distribution des coûts vérifie:

$$\mu = \frac{1}{2^L} \langle f, \psi_0 \rangle = w_0.$$

Considérons le moment μ_r d'ordre r de la distribution des $f(x)$ relative à une fonction de coût f. Il est défini par:

$$\mu_r = E\left[(f-\mu)^r\right] = \sum_{x=0}^{2^L-1} (f(x)-\mu)^r \, p(x). \tag{1.42}$$

Proposition 28 *Expression du moment μ_r*

Sous les hypothèses et notations antérieures, on a:

$$\mu_r = \sum_{a_1 \oplus a_2 \oplus \ldots \oplus a_r = 0 \ et \ a_i \neq 0} w_{a_1} w_{a_2} \ldots w_{a_r}. \tag{1.43}$$

Plus précisément, la sommation de ces produits de coefficients de Walsh (non nuls !) est étendue à l'ensemble de tous les indices dont le ou-exclusif est nul.

Preuve. Les étapes cruciales sont les suivantes:

Vu la définition de μ_r:

$$\mu_r = \frac{1}{2^L} \sum_{x=0}^{2^L-1} \left(\left(\sum_{j=0}^{2^L-1} w_j \psi_j(x) \right) - w_0 \right)^r, \tag{1.44}$$

soit encore

$$\mu_r = \frac{1}{2^L} \sum_{x=0}^{2^L-1} \left(\sum_{j=1}^{2^L-1} w_j \psi_j(x) \right)^r,$$

$$\text{et} \quad \mu_r = \frac{1}{2^L} \sum_{x=0}^{2^L-1} \left[\left(\sum_{a_1=1}^{2^L-1} w_{a_1} \psi_{a_1}(x) \right) \ldots \left(\sum_{a_r=1}^{2^L-1} w_{a_r} \psi_{a_r}(x) \right) \right] \tag{1.45}$$

qui prend la forme annoncée en raison de l'orthogonalité des fonctions $\left(\psi_j\right)_{0 \leq j \leq 2^L-1}$ et de leur norme. ■

Exemple 29 *Dans le cas de* $r = 2$*, pour toute fonction* f*, la variance* μ_2 *sera donnée par:*

$$\mu_2 = \sum_{i=1}^{2^L-1} w_i w_i,$$

puisque $a_1 \oplus a_2 = 0$ *si et seulement si* $a_1 = a_2$*.*

Remarque 30 *Validité du résultat*

Dans [HRW99(2)], les auteurs montrent que dans le cadre des Embedded Landscapes:

- *il existe seulement un nombre polynômial de coefficients de Walsh non nuls;*
- *on peut les identifier et les calculer en un temps polynômial.*

De plus $MAX3SAT$*, les* $nk-$*Landscapes sont des instances d'un Embedded Landscape. Ainsi pour ces classes de problèmes, les divers indicateurs de statistiques descriptives peuvent être calculés en un temps polynômial en raison de la proposition 28.*

Remarque 31 *En guise de bilan*

Il est donc possible de calculer les indicateurs de statistiques desciptives relatifs à des problèmes NP- complets en un temps polynômial. Peut-on en conséquence extraire des statistiques descriptives de tels problèmes, des éléments permettant de guider une recherche ?

Si P≠NP, ceci n'est pas possible. On pourra s'affranchir de cette hypothèse par utilisation des théorèmes du No Free Lunch.

Sur les statistiques descriptives et les algorithmes de recherche

On considère ici l'ensemble de toutes les permutations Π des éléments de Y.

Proposition 32 *L'ensemble des indicateurs de statistiques descriptives n'apporte aucune information utilisable dans la sélection des algorithmes de recherche ou pour guider un algorithme spécifique, lorsqu'on considère une fonction de coût particulière construite sur Y.*

Preuve. Tous les indicateurs de statistiques descriptives sont identiques sur Y. Or les théorèmes du No Free Lunch sont valides sur Π, qui permute les éléments de Y. Donc ces indicateurs ne peuvent apporter les éléments attendus pour une fonction particulière, puisqu'ils sont identiques sur tout ensemble où s'appliquent ces théorèmes. [13] ■

Remarque 33 *Un certain nombre de résultats ne peuvent pas non plus être attendus des coefficients de Walsh puisque qu'ils sont eux-mêmes sous contraintes des indicateurs statistiques !*

Statistiques, voisinages et coefficients de Walsh

Une intéressante analyse de D. Whitley sur les coefficients de Walsh met en évidence leur écriture récursive. Sur des exemples précis, l'auteur montre, à partir du code réfléchi (de Gray), comment différentes représentations d'une même fonction peuvent avoir une même structure de voisinage, et comment ceci se reflète dans l'invariance des coefficients de Walsh associés.

1.4.3 Bilan relatif au lien "algorithmes de recherche et statistiques"

- D. Whitley établit qu'un certain nombre de résultats ne peuvent provenir des statistiques descriptives calculées dans le cadre de fonctions de coût, présentant

[13] Ici encore on sent venir très nettement le théorème de caractérisation 56...

des propriétés d'invariance par permutations. Les statistiques ne peuvent pas, en général, guider une recherche, dans un tel cadre.

- Ce résultat négatif est capital. Des exemples similaires existent dans de nombreux domaines des mathématiques; ils apportent beaucoup en termes de méthode et bousculent parfois complètement la manière d'aborder certaines questions !

1.5 Champs de validité des théorèmes du NFL

La question de savoir, sur quelle partie de l'espace des fonctions de coût les théorèmes du No Free Lunch sont valides, est absolument centrale. Pressentie dès le papier [WM95], elle fait l'objet d'une lente maturation, en strates successives, acquises grâce à l'intervention des multiples équipes qui traitent ce problème. Nous abordons successivement les contributions majeures à cette construction, l'affinement qui en découle, mais aussi les désillusions et la mise en cause de certaines idées simples.

1.5.1 L'apport de C Schumacher, MD Vose, D Whitley 2001 [SVW01]

En 2001 paraît le papier intitulé *"The No Free Lunch and Problem Description Length"*, [SVW01]. Il s'agit d'une contribution essentielle qui collecte une certain nombre de tentatives partielles antérieures, les généralise, précise la dualité liant algorithmes d'optimisation et fonctions optimisées; le tout aboutit à la caractérisation du champ de validité des théorèmes du NFL. Cette question, à laquelle nous nous limiterons ici, est cruciale quant à l'efficience pratique de ces théorèmes, en particulier.

On se propose d'identifier les notions centrales, véritables invariants de la connaissance des algorithmes de recherche déterministes, sans répétition. Pour des raisons de simplicité nous adoptons les notations fournies en amont sous la section 1.2.2.

a) Définitions: rappels et compléments

Soit X et Y deux ensembles finis, Y totalement ordonné et une fonction de coût $f : X \to Y$.

On considère l'ensemble $\mathcal{D}_m$ des échantillons d_m de taille m ($m \in \mathbb{N}$, $m \leq |X|$) définis par:

$$d_0 = \varnothing \text{ et } d_m = \left(\left(d_m^x(1), d_m^y(1)\right), ..., \left(d_m^x(m), d_m^y(m)\right)\right). \tag{1.46}$$

On définit l'opérateur de concaténation $\|$ sur les échantillons comme suit:

$$d_m \|(x, y) = \left(d_m(1), d_m(2), ..., d_m(m), (x, y)\right). \tag{1.47}$$

Comme l'échantillon d_m est sans répétition: $\left(d_m^x[i] = d_m^x[j]\right) \Rightarrow (i = j)$.

L'échantillon d_m est dit complet s'il recouvre le domaine de recherche, autrement dit:

$$\forall z \in X \quad \exists i \in \{1, ..., m\} \quad z = d_m^x(i). \tag{1.48}$$

Vu qu'un échantillon est un $m-$uplet, il correspond bijectivement à une fonction d'une partie de X dans Y; si l'échantillon est complet, cette fonction est f.

Définition 34 *Algorithmes de recherche*

Un algorithme boîte-noire déterministe de recherche optimisant f, est appelé simplement algorithme et noté a_f. Il est lié à un opérateur g de recherche, qui associe à l'échantillon précédent (au début d_0), le point suivant de l'espace de recherche à évaluer.

Après m pas, a_f admet comme argument d_m; il retourne l'échantillon:

$$d_{m+1} = a_f(d_m) = d_m \| (g(d_m), f \circ g(d_m)) \ . \tag{1.49}$$

Par exemple les deux premiers pas permettent d'écrire:

$$\begin{aligned} d_1 &= a_f(d_0) = d_0 \| (g(d_0), f \circ g(d_0) \ ; \\ d_2 &= a_f(d_1) = a_f^2(d_0) = d_1 \| (g(d_1), f \circ g(d_1)) \ . \end{aligned} \tag{1.50}$$

Définition 35 *Vecteurs de performances*

- *On appelle vecteur de performance de longueur m tout suite de m valeurs de Y.*
- *Le vecteur performance déduit de l'échantillon d_m, est d_m^y; inversement, une fonction et un algorithme vont générer un vecteur performance à partir de d_0.*

Notation 36 *L'échantillon de longueur m généré par a et f, sera désigné par $d_m(a, f)$, tandis que $V_m(a, f)$ représente le vecteur performance déduit, dont la taille m sera indiquée seulement si nécessaire.*

Définition 37 *Deux algorithmes a et a' sont dits identiques si et seulement si les deux génèrent le même échantillon complet pour toute fonction f, autrement dit:*

$$\forall f \in Y^X \quad d_{|X|}(a, f) = d_{|X|}(a', f) \ .$$

Définition 38 *On appelle mesure globale sur un ensemble de fonctions $F \subset Y^X$ d'un algorithme a, une application qui fait correspondre un nombre réel à l'ensemble des vecteurs performances générés par a et les f de F.*

N.B. Une telle mesure globale sera utilisée à des fins de comparaison entre algorithmes; en cas d'égalité de mesure sur F, on dira que les deux algorithmes présentent les mêmes performances sur F.

Exemple 39 *Si l'on note M une fonction qui associe à un vecteur performance un réel, on peut donner comme exemple de mesure globale sur un ensemble de fonctions F le réel défini par*:

$$\frac{1}{|F|}\sum_{f\in F} M\left(V\left(a,f\right)\right). \tag{1.51}$$

b) Résultats du No Free Lunch

Définition 40 *On appellera résultat du No Free Lunch sur F, toute situation dans laquelle deux algorithmes quelconques ont des mesures globales égales sur F. On dira aussi que F est un champ d'application du NFL.*

De plus, si l'ensemble F des fonctions n'est pas précisé, il s'agit implicitement de $F = Y^X$.

Théorème 41 *Pour toute mesure globale, les performances de tous les algorithmes sont les mêmes.*

Théorème 42 *Etant donné deux algorithmes a et a', alors pour toute fonction f, il existe une fonction g telle que $V\left(a,f\right) = V\left(a',g\right)$.*

N.B. Ce résultat est le point central de la preuve du théorème 41 fournie par Radcliffe et Surry [RS95].

Théorème 43 *Tout algorithme génère la même collection* [14] *de vecteurs performances lorsque toutes les fonctions sont considérées.*

Remarque 44 *Ce résultat est la base de la preuve du théorème 41, fournie par Schumacher dans [S00].*

Schumacher a établi de surcroît que $([V\left(a,f\right) = V\left(a,g\right)] \Rightarrow [f = g])$, c'est-à-dire que les collections évoquées au théorème 43 sont alors des ensembles. Ce résultat est crucial pour établir l'équivalence des quatre théorèmes: théorème 41 , 42, 43 et 46.

[14] *Pour préciser ce terme voir [S00].*

Remarque 45 *On a vu dans l'exemple 39 un cas très simple de mesure globale.*

On peut aussi définir une mesure pondérée globale sur un ensemble de fonctions F: elle est obtenue en sommant sur F des éléments $W(f)M(V(A,f))$, où $W(f)$ désigne le poids accordé à la fonction f.

En général, une mesure pondérée globale ne met pas facilement en évidence "un champ d'application" des théorèmes du No Free Lunch, hormis le cas où les fonctions f sont affectées d'un poids identique; nous dirons d'une telle mesure qu'elle est pondérée globale uniforme.

Théorème 46 *Pour toute mesure pondérée globale uniforme, tous les algorithmes ont même performance.*

Remarque 47 *Ce résultat est celui établi dans [WM95]. L'ensemble total Y^X, clos par permutation, fut le premier découvert comme étant un lieu de validité du NFL.*

Affinement du No Free Lunch

a) Vocabulaire et notations

Définition 48 *Soit f une fonction de coût de X dans Y. Soit σ une permutation de X. On appelle permutée de f par σ, la fonction notée σf, de X dans Y, définie par:*

$$\forall x \in X \qquad \sigma f(x) = f\left[\sigma^{-1}(x)\right].$$

Définition 49 *Un ensemble F de fonctions est dit clos (ou stable ou invariant) par permutation si et seulement si pour toute fonction f de F, toute permutée de f est dans F.*

Définition 50 *Soit a un algorithme, d'opérateur de recherche associé g; soit σ une permutation de X.*

On appelle algorithme permuté de a, l'algorithme noté σa dont l'opérateur de recherche σg est défini par:

$$\sigma g\left(\varphi\right)=\sigma^{-1}\left(g\left(\sigma_{x}\left(\varphi\right)\right)\right),$$

où $\sigma_{x}\left(\varphi\right)$ opère sur les x-composantes d'un échantillon φ en appliquant σ à chacune d'elles tandis que les y-composantes restent inchangées.

b) Résultats centraux

Théorème 51 *Pour tout n de $\mathbb{N}$, si on note*

$$d_{n}\left(a,\sigma f\right)=\left(\left(x_{1},y_{1}\right),\left(x_{2},y_{2}\right),...,\left(x_{n},y_{n}\right)\right)$$

alors

$$d_{n}\left(\sigma a,f\right)=\left(\left(\sigma^{-1}\left(x_{1}\right),y_{1}\right),...,\left(\sigma^{-1}\left(x_{n}\right),y_{n}\right)\right). \tag{1.52}$$

Preuve. La preuve se mène par récurrence sur la longueur n de l'échantillon. Elle n'est pas très technique; elle consiste essentiellement à s'approprier les notations.

■

Corollaire 52 *(dit de dualité)*

Sous les mêmes notations, on a:

$$V\left(a,\sigma f\right)=V\left(\sigma a,f\right). \tag{1.53}$$

Remarque 53 *La preuve provient des notations posées. Ce corollaire met en évidence une correspondance entre une permutation d'algorithme et une permutation de fonction, phénomène central, déjà rencontré plusieurs fois, mais prouvé ici de façon formelle.*

Lemme 54 *Si l'ensemble $F \subset Y^X$ est clos par permutation, alors F est un champ d'application du NFL.*

Preuve. Elle provient de l'application du théorème 42. ■

Lemme 55 *Si un ensemble de fonctions $F \subset Y^X$ est un champ d'application du NFL, alors F est clos par permutation.*

Preuve. La preuve est assez fine; elle appelle le théorème 43 et repose sur un raisonnement par l'absurde. ■

Théorème 56 *Caractérisation des champs de validité du NFL*

Un ensemble de fonctions $F \subset Y^X$ est un champ d'application du No Free Lunch si et seulement si F est clos par permutation.

No Free Lunch et clôture par permutation

Les preuves des différentes propositions suivantes sont en général très simples donc non évoquées ici.

a) Définitions, propriétés premières

Définition 57 *Soit F une partie de Y^X. On appelle clôture par permutation de F la partie $C(F)$ de Y^X (contenant F) définie par:*

$$C(F) = \{\sigma f \quad / \quad f \in F \text{ et } \sigma \text{ est une permutation de } X\}. \tag{1.54}$$

Proposition 58 *Pour toute partie F de Y^X, $C(F)$ est clos par permutation.*

Proposition 59 *Pour toutes parties F et F', on a*

$$C(F \cup F') = C(F) \cup C(F'),$$

Cette propriété s'étend à toute réunion finie de parties de Y^X.

Définition 60 *Equivalence de fonctions et algorithmes*

1. *Deux fonctions f et g de Y^X sont dites équivalentes si et seulement si il existe une permutation σ de X telle que: $f = \sigma g$. On note alors $f \equiv g$.*

2. *Deux algorithmes a et a' sont dits équivalents si et seulement si il existe une permutation σ de X pour laquelle: $a = \sigma a'$. On note alors $a \equiv a'$.*

Proposition 61 *Les deux relations définies ci-dessus sont d'équivalence.*

On notera $\overline{f}$ et $\overline{a}$ les classes d'équivalence respectives de f et a.

Définition 62 *Soit f une fonction de coût et a un algorithme donné.*

Soit F un ensemble de fonctions de coût [respectivement $\mathcal{A}$ un ensemble d'algorithmes].

On définit $V(a, F)$ et $V(\mathcal{A}, f)$ par:

$$\begin{aligned} V(a, F) &= \{V(a, f) \quad / \quad f \in F\}, \\ V(\mathcal{A}, f) &= \{V(a, f) \quad / \quad a \in \mathcal{A}\}. \end{aligned} \tag{1.55}$$

Proposition 63 *Soit f une fonction de coût donnée. Alors*

- *$\overline{f}$ est le plus petit ensemble clos par permutation contenant f.*

- *Tout ensemble clos par permutation est une réunion disjointe de classes d'équivalence.*

Proposition 64 *Etant donné deux algorithmes a et a', et f une fonction de coût donnée, on a:*

- $\overline{f} = C(\{f\})$.

- $V\left(a, \overline{f}\right) = V\left(a', \overline{f}\right)$.

b) Propriétés essentielles

Proposition 65 *Etant donné un algorithme a et une fonction de coût f, on a:*

$$V(\overline{a}, f) = V\left(a, \overline{f}\right). \tag{1.56}$$

Proposition 66 *Pour un algorithme a et une fonction de coût f donnés, les grandeurs suivantes sont identiques:*

- *La moyenne des performances sur tous les algorithmes utilisant la fonction f;*
- *la moyenne des performances sur une classe d'équivalence quelconque d'algorithmes utilisant la fonction f;*
- *la moyenne des performances sur toutes les fonctions d'une classe d'équivalence utilisant l'algorithme a.*

C Schumacher, D Vose et D Whitley fournissent quelques exemples de classes d'équivalence et étudient le lien à la difficulté de représentation des fonctions; nous n'abordons pas ces questions ici, en dépit de leur intérêt.

1.5.2 L'apport de C Igel et M Toussaint 2001 [IT01]

Dans [IT01], C. Igel et M. Toussaint commencent par rappeler les résultats du très important papier [SVW01] présenté ci-dessus en les réinterprétant. Ils montrent que les ensembles de fonctions clos par permutation sont rares, très rares. On mesure rapidement l'importance d'un tel résultat et le questionnement rugueux qui en découle: "Se serait-on inquiété d'un phénomène trop rare, autant dire inexistant, en pratique...?"

Enfin C Igel et M Toussaint exhibent des ensembles de fonctions non clos par permutation, liés au concept de voisinages sur X.

PART DE SOUS-ENSEMBLES CLOS PAR PERMUTATION

a) Etude théorique

Proposition 67 *Il existe* $2^{|Y|^{|X|}} - 1$ *parties non vides de* $\mathcal{F} = Y^X$.

Preuve. Elle est évidente, via la bijection entre les parties d'un ensemble et celui des fonctions caractéristiques. ■

Théorème 68 *Part de sous-ensembles clos par permutation*

1. *Le nombre* N *de sous-ensembles non vides de* Y^X *qui sont clos par permutation est donné par:*
$$N = 2^{\binom{|X|+|Y|-1}{|X|}} - 1. \tag{1.57}$$

2. *Par suite, la part* t *de ces sous-ensembles parmi les sous-ensembles non vides de* $\mathcal{F} = Y^X$ *est défini par:*
$$t = \frac{2^{\binom{|X|+|Y|-1}{|X|}} - 1}{2^{|Y|^{|X|}} - 1}. \tag{1.58}$$

Preuve. Ici, bien sûr, $\binom{n}{p}$ désigne le nombre de combinaisons de p éléments pris parmi n, souvent noté C_n^p en France. La preuve est très technique, délicate mais fort intéressante comme enchaînement d'idées; elle fait appel à des méthodes de dénombrements originales, voire savantes.

Elle repose sur un lemme très instructif proposant une démarche pour calculer le nombre des parties cherchées grâce à l'étude d'une relation d'équivalence sur l'ensemble des fonctions et la prise en considération des orbites sous cette relation.

Elle fait appel à un ouvrage de théorie des probabilités pour fournir un résultat intermédiaire de dénombrement, non trivial et crucial pour la preuve de la première partie du lemme, fondé sur une application du théorème d'inclusion-exclusion. ■

Remarque 69 *On peut étudier expérimentalement les variations de t en fonction de $|X|$ et $|Y|$. En particulier, sous le théorème précédent, lorsque $|X|$ et $|Y|$ sont petits, cette part est infime. A titre d'exemple, les auteurs signalent que pour des fonctions booléennes de trois variables (c'est-à-dire $X = \{0,1\}^3$ et $Y = \{0,1\}$), on trouve:*

$$t \ll 10^{-170}. \tag{1.59}$$

b) Résultats expérimentaux

Le graphique proposé par C Igel et M Toussaint en figure 1.4 fournit en ordonnées la part des sous-ensembles clos par permutation sur une échelle logarithmique, en fonction du cardinal de l'espace de recherche X; plusieurs courbes sont proposées pour différentes valeurs du cardinal de Y. La question qui demeure est de connaître l'évolution de cette part lorsque $|X|$ et $|Y|$ deviennent grands, ce qui correspond à la situation réelle dans le cadre des applications ordinaires.

Cas d'espaces de recherche dotés de relations de voisinage

L'objet de cette sous-section est d'étudier des situations conformes à la réalité des applications les plus fréquentes. Les auteurs supposent donc:

- que l'espace de recherche présente une certaine structure,
- et que les fonctions de coût prises en considération, ont des caractéristiques liées à cette structure.

De façon plus précise, il existe une relation de voisinage sur X liée à ces caractéristiques. Par exemple, on peut s'intéresser à des contraintes exprimées en termes de rugosité ou d'optima locaux, en lien avec la notion de voisinage retenue.

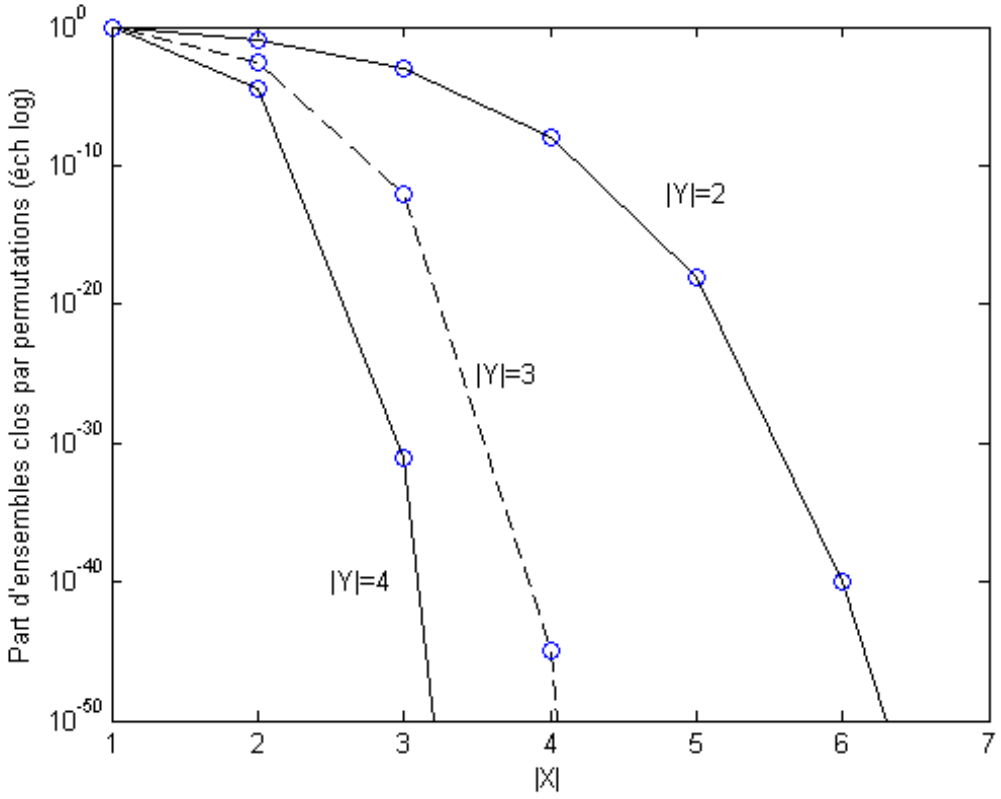

Figure 1.4: Part de sous-ensembles clos par permutation

a) Voisinages et clôture par permutation

Définition 70 *Notion de voisinage* [15]

1. *On appelle relation de voisinage une fonction symétrique v définie sur $X \times X$ à valeurs dans $\{0,1\}$.*
2. *Deux éléments x_i et x_j sont dits voisins (au sens de v) ssi $v(x_i, x_j) = 1$.*
3. *De plus le voisinage v est dit non trivial ssi il existe (x_i, x_j) et (x_k, x_l) de X^2 tels que:*

$$[(x_i \neq x_j) \wedge v(x_i, x_j) = 1] \wedge [(x_k \neq x_l) \wedge v(x_k, x_l) = 0].$$

Théorème 71 *Un voisinage non trivial n'est pas invariant par permutation sur X.*

Preuve. Elle est très simple. ■

Remarque 72 *On suppose, ce qui correspond à une situation fréquente en pratique, que X est un produit cartésien $X_1 \times ... \times X_l$ et que sur chaque composante X_i du produit existe un voisinage non trivial v_i. La notion de voisinage sur les composantes, en induit une sur X: deux points de X sont dits voisins ssi leur $i^{ème}$ composante l'est au sens de v_i. Ainsi les contraintes complexes sur X pourront se traduire par des contraintes simples sur chacune des composantes.*

Remarque 73 *La notion de voisinage définie ci-dessus est en général non canonique puisqu'elle dépend du mode de représentation des objets de X. Elle est donc basée sur des éléments de phénotype.*

[15] *Sans doute conviendrait-il de vérifier un certain nombre de propriétés complémentaires comme on le pratique en topologie, pour valider cette définition de voisinage et obtenir les standards ordinaires connexes, comme les voisinages produits et autres concepts liés.*

b) Exemples de sous-ensembles non clos par permutation

On suppose donc X doté d'une relation de voisinage v et on munit Y d'une distance d_Y, qui peut être la distance euclidienne.

Premier type de contrainte

- Nous pouvons définir une mesure de gradient maximal d'une fonction f, liée à la notion de voisinage sur X, comme suit:

$$g^{\max}(f) = \max_{(x_i,x_j)\in X^2 \wedge v(x_i,x_j)=1} d_Y\left[\left(f(x_i), f(x_j)\right)\right].$$

- Puis on définit le diamètre de l'image d'une fonction f par:

$$d^{\max}(f) = \max_{(x_i,x_j)\in X^2} d_Y\left[\left(f(x_i), f(x_j)\right)\right].$$

On en déduit le corollaire suivant.

Corollaire 74 *Soit F une partie non vide de Y^X. Si le gradient maximum $g^{\max}(f)$ de toute fonction f prise dans F est strictement inférieur à $\max_{f\in F} d^{\max}(f)$,*

alors F n'est pas clos par permutation.

Preuve. Elle est assez simple, par l'absurde: on exhibe une permutation qui provoque le non-respect de la contrainte. ■

Deuxième type de contrainte

Les auteurs considèrent alors le nombre de minima locaux pour une fonction f, souvent utilisé comme mesure de complexité d'un problème d'optimisation.

- Pour une fonction f donnée, un point x de X est un minimum local ssi $f(x) < f(x_i)$ pour tout voisin x_i de x.

- On donne alors une relation de voisinage sur X et on définit l'entier $l^{\max}(f)$ comme le nombre maximal de minima que peuvent présenter des fonctions de même histogramme que f.

Corollaire 75 *Soit F une partie non vide de Y^X. Si le nombre de minima locaux de toute fonction f prise dans F est strictement inférieur à $\max_{f \in F} l^{\max}(f)$,*

alors F n'est pas clos par permutation.

Preuve. La preuve repose sur des lemmes techniques présentés en appendice du papier de C Igel et M Toussaint. En particulier ils établissent que deux fonctions f et g ont le même histogramme ssi il existe une permutation σ de X telle que $g = f \circ \sigma$. ■

Bilan

Si une partie F de l'ensemble Y^X des fonctions de coût est choisie uniformément, et si les cardinaux $|X|$ et $|Y|$ sont de taille raisonnable, la probabilité que le théorème du NFL soit valide sur F est très faible.

Si dans un domaine d'application, les fonctions de coût utilisées ne présentent pas le nombre maximal de minima locaux ou le maximum de gradient, alors il est probable que le théorème du NFL n'y soit pas valide.

1.5.3 L'apport de S Droste, T Jansen, I Wegener 2002 [DJW02]: champs de validité du NFL, une version optimale ?

Dans l'article [DJW02] intitulé "*Optimization with Randomized Search Heuristics - The (A)NFL Theorem, Realistic Scenarios, and Difficult Functions*", les auteurs font, dans l'introduction, un point sans complaisance sur le manque cruel de formalisation d'un bon nombre de notions fréquemment évoquées dans le monde de l'optimisation. Après l'exposé du plan du papier, ils commencent par établir une

généralisation du théorème de No Free Lunch et atteignent, semble-t-il, une version "définitive" en raison de sa simplicité.

Le fil directeur du papier est d'aborder tour à tour, les scenarii ordinaires auxquels sont confrontés les chercheurs en optimisation. Parmi les grands classiques, se présente une situation de base pour laquelle le théorème du No Free Lunch est valide. Il est ainsi astucieux de mettre en perspective ce résultat avec l'ensemble des préoccupations du métier en évitant qu'il ne cristallise les craintes d'étrangeté.

Définition 76 *Scenario* 1*, dit du No Free Lunch*

Les ensembles X et Y sont finis, et Y muni d'un ordre total. On note $\mathcal{F} = Y^X$ l'ensemble de toutes les fonctions $f : X \to Y$ qu'on cherche à maximiser. La fonction f est choisie uniformément dans $\mathcal{F}$.

Les auteurs font d'abord remarquer qu'on attend d'un algorithme d'optimisation qu'il s'arrête dès qu'il a trouvé un point x de X optimal, ce qui implique que l'algorithme mis en oeuvre a prouvé que x était optimal; ceci n'est pas apporté par les boîtes noires d'optimisation, en général.

Pour établir que x est optimal il suffit de prouver que $f(x)$ est maximum dans Y. Si tel n'est pas le cas, il convient d'apporter une preuve complémentaire. En conséquence, il est naturel de mesurer la qualité de l'algorithme utilisé pour optimiser la fonction f, par la donnée du nombre de valeurs x différentes de X pour lesquelles on aura évalué $f(x)$ avant de trouver un x^* optimal.

Les auteurs se proposent alors d'établir une généralisation du théorème du NFL, valide dans le seul scenario précisé sous les hypothèses de la définition 76.

Définition 77 *Une partie F de Y^X est dite close par permutation si elle contient avec une fonction f toutes les fonctions f_σ définies par:*

$$\forall x \in X \quad f_\sigma(x) = f\left[\sigma\left(x\right)\right], \tag{1.60}$$

sachant que σ désigne une permutation de X.

N.B. En particulier, évidemment, Y^X est clos par permutation.

Théorème 78 *Généralisation du théorème du NFL*

Soit a une heuristique quelconque de recherche (déterministe ou stochastique) destinée à optimiser les fonctions f d'un ensemble $F \subset Y^X$, où F est clos par permutation. On note $r(a)$ la moyenne, relative à la distribution uniforme sur F, des temps d'exécution attendus de a sur F.

Alors $r(a)$ est une valeur indépendante de a.

Preuve. Nous précisons ici le plan de cette preuve, en raison de son aspect minimal et donc très fécond [16].

1. Lemmes techniques

   Soit x_0 de X et $Y'_{x_0} = \{y \in Y / f(x_0) = y \text{ avec } f \in F\}$. Soit y_0 un élément de Y'_{x_0}.

   On note $F_{x_0\,,\,y_0}$ l'ensemble des fonctions g de $X - \{x_0\}$ dans Y dont le prolongement f à X donné par:

   $$[\forall x \in X - \{x_0\} \quad f(x) = g(x)] \quad \text{et} \quad [f(x_0) = y_0]\,. \tag{1.61}$$

   est dans F.

   Sous ces notations, on prouve les résultats suivants.

   Lemme 79 *$F_{x_0\,,\,y_0}$ est clos par permutation.*

[16] On remarquera ici que le lemme central concerne une transposition dont on sait que l'ensemble génère les permutations. En ce sens, la preuve est fondée sur le coeur "causal" de la problématique étudiée.

Lemme 80 *$F_{x_0\ ,\ y_0}$ et $F_{x'_0\ ,\ y_0}$ sont isomorphes pour tout $x_0 \neq x'_0$ au sens défini ci-dessous.*

Soit g un élément de $F_{x_0\ ,\ y_0}$.

On lui associe g' défini comme suit:

$$[\forall x \in X - \{x_0, x'_0\} \quad g'(x) = g(x)] \quad et \quad [g'(x_0) = g(x'_0)]. \tag{1.62}$$

Alors g' est élément de $F_{x'_0\ ,\ y_0}$.

Preuve. Les preuves de ces deux lemmes constituent des exercices de mise en oeuvre des notations utilisées [17] et au delà, la maîtrise d'un cas de permutation primitif. ■

2. La preuve du théorème est alors donnée pour les heuristiques déterministes grâce à une démonstration par récurrence sur le cardinal de X, utilisant les lemmes antérieurs.

3. Les auteurs donnent ensuite la preuve pour les heuristiques stochastiques qui se ramènent au cas déterministe puisqu'on est amené à déterminer une moyenne pondérée de coûts identiques.

■

Remarque 81 *S Droste, T Jansen, I Wegener notent que ce résultat n'est pas surprenant, si l'on y réfléchit quelque peu. Si la classe des fonctions considérées est invariante par permutation, on ne peut pas espérer utiliser une quelconque structuration de l'espace de recherche.*

[17] La notion d'isomorphisme proposée gagnerait sans doute à être approfondie et la structure sous-jacente explicitée...

1.5.4 Bilan relatif aux champs de validité du NFL

- Grâce à la contribution de plusieurs équipes, les parties de l'ensemble des fonctions de coût sur lesquels les théorèmes du NFL sont valides sont précisées, et même davantage, caractérisées.

- Un ensemble F de fonctions est un champ d'application du No Free Lunch si et seulement si F est clos (ou stable, ou invariant) par permutation. Voir théorème 56.

- La part des sous ensembles F de $\mathcal{F} = Y^X$ clos par permutation est faible, et même très faible. Voir sous-section 1.5.2.

- Des exemples de parties de $\mathcal{F} = Y^X$, classiquement utilisées et liées au concept de voisinage sur X, ne sont pas clos par permutation. Voir sous-section 1.5.2.

- Dans la démarche menant à la caractérisation des champs de validité du NFL, les définitions se simplifient, se dépouillent, se réduisent au strict minimum. Voir sous-section 1.5.3.

1.6 Le quasi No Free Lunch: aspect local du NFL

1.6.1 Introduction

Dans [DJW02], S Droste, T Jansen et I Wegener ont commencé par simplifier les notations et l'expression du théorème du NFL pour le généraliser, comme rappelé ci-dessus à la sous-section 1.5.3, dont nous reprenons les notations désormais.

L'objectif des auteurs, dans cette première partie, était de prendre en compte une situation, dite scénario du No Free Lunch, correspondant à un ensemble de fonctions de coût sur lequel le théorème du No Free Lunch généralisé est valide. Ce cas de figure est évidemment possible, mais il demeure peu réaliste, et ne constitue

pas l'ordinaire des praticiens de l'optimisation. C'est pourquoi dans la suite de leur papier, ils accordent leur attention à des scenarii plus ordinaires.

Après avoir défini le cadre de leur étude, ils établissent un théorème de Quasi No Free Lunch grâce à un travail préparatoire très soigné sur une situation pratique; le détail de cette approche nous paraît d'un grand intérêt. Il convient ici d'être technique et précis, car les généralités ne suffisent pas... Ils fournissent alors quelques applications, nommées par eux "académiques", de ce théorème ANFL, pour Almost No Free Lunch.

Puis ils proposent une réflexion sur les difficultés d'optimiser un certain nombre de fonctions de coût rencontrées ordinairement. Ils donnent des exemples de fonctions faciles... mais surtout exhibent un exemple de fonction non-artificielle difficile. Ils montrent que les méthodes classiquement utilisées (recuit simulé, algorithmes évolutionnaires) vérifient des conditions suffisantes qui prouvent qu'elles seront en très grande difficulté sur cette fonction, voire en situation désespérée !

En passant, les auteurs fournissent une synthèse de grande qualité sur le recuit simulé et les algorithmes évolutionnaires.

1.6.2 Scenarii plus réalistes pour boîtes noires d'optimisation

Soit X et Y deux ensembles finis; Y est totalement ordonné. Y^X désigne l'ensemble des fonctions de X dans Y.

Définition 82 *scenario 2 (à un coup !)*

Une fonction donnée $f : X \to Y$ doit être maximisée.

On ne peut développer de théorie adaptée à cette situation singulière, si ce n'est en précisant un type de fonctions concernées.

Définition 83 *scenario 3 (relatif à un type de fonction donné)*

f est choisie parmi une classe de fonctions présentant des propriétés données.

Si les fonctions présentent des propriétés données, il existe fréquemment des méthodes d'optimisation spécifiques et aussi des heuristiques adaptées à ces situations. (Voir [PS98] et [H96]).

Définition 84 *scenario 4 (Boîtes noires d'optimisation restreintes)*

Le scenario est le même que celui du No Free Lunch, hormis le fait que Y^X est remplacé par un certain sous-ensemble $F \subset Y^X$ de fonctions dont la complexité, en un sens à préciser, est restreinte.

1. *Scenario 4.1 (restriction temporelle)*

   La restriction est exprimée en termes de majoration de temps d'exécution relatif au nombre de pas nécessaires à l'évaluation de la fonction f.

2. *Scenario 4.2 (restriction spatiale)*

   La restriction est exprimée en termes de majoration de la taille de la représentation de f, par exemple par un OBBD. Voir la section 1.3.

3. *Scenario 4.3 (restriction en complexité de Kolmogorov)*

   La restriction est exprimée en termes de majoration de la complexité de Kolmogorov de f. Pour cette notion, voir par exemple [P94].

Les auteurs concluent en indiquant leur conjecture relative aux liens entre restrictions sur f et NFL. Pour S Droste, T Jansen, I Wegener, un scenario dans lequel les restrictions ne sont pas triviales, fait perdre la validité du No Free Lunch. Le problème est de le prouver... Ceci ne paraît faisable que dans le cas du parcours exhaustif des possibles, ce qui impose des ensembles petits mais par là certainement, la naissance d'autres difficultés...

1.6.3 Un quasi No Free Lunch

Introduction

Pour S Droste, T Jansen, I Wegener, connaître une restriction relative à la complexité des fonctions f considérées, brise la "symétrie" de l'espace des fonctions. Ceci provoque un "libre appétit", c'est-à-dire que les algorithmes qui utilisent l'information donnée risquent d'être meilleurs que les autres. Pour autant, savoir qu'une fonction peut être évaluée en temps linéaire ou quadratique peut ne pas apporter d'amélioration de la recherche. Des fonctions très simples à évaluer sont de véritables bêtes noires pour les boîtes de même couleur, en optimisation... parce que liées à des problèmes très difficiles.

Les auteurs conjecturent que la considération de boîtes noires restreintes n'apportent guère plus qu'un "libre appétit". Eu égards à la difficulté de la formalisation des problèmes, ils annoncent espérer des résultats modestes !

Théorème du quasi No Free Lunch (Almost NFL)

a) Démarche menant au théorème

Soit f une fonction de coût définie sur $X = \{0,1\}^n$ à valeurs dans $Y = \{0,1,2,...,N-1\}$. Soit a une heuristique stochastique travaillant sur f , supposée efficiente [18]. On étudie les $2^{n/3}$ premiers pas d'exécution de a sachant qu'en termes temporels, on impose une majoration de leur temps d'exécution.

- On note $x = (x_1, ..., x_n)$ un élément donné de $\{0,1\}^n$ et $q(x)$ la probabilité que l'algorithme a évalue $f(x)$ durant les premiers $2^{n/3}$ pas. Vu la contrainte

[18] Une heuristique stochastique ne peut pas en général reconnaître si elle atteint un optimum. On utilise donc pour stopper l'exécution un critère externe. En conséquence on considère, a priori, une exécution en n pas de a ($n \in \mathbb{N}$).

temporelle, il vient:

$$\sum_{x\in\{0,1\}^n} q(x) \leq 2^{n/3}. \tag{1.63}$$

- On partitionne alors $\{0,1\}^n$ en $2^{2n/3}$ sous-espaces disjoints S_z, où z est élément de $\{0,1\}^{2n/3}$, tels que S_z contient tous les éléments x de $\{0,1\}^n$ vérifiant $x_i = z_i$ pour tout i compris entre 1 et $2n/3$.

- Notons $q^*(z)$ la probabilité que a évalue au moins un $f(x)$ avec x dans S_z, durant ses premiers $2^{n/3}$ pas. Comme

$$q^*(z) \leq \sum_{x\in S_z} q(x),$$

via le principe des cages de pigeons, il existe au moins un z^* tel que [19]:

$$q^*(z^*) \leq \frac{2^{n/3}}{2^{2n/3}} = 2^{-n/3}. \tag{1.64}$$

- On définit alors un ensemble d'au moins $N^{2^{n/3}-1}$ fonctions f^* comme suit:

$$\begin{array}{ll} \text{Si } x \notin S_{z^*} & f^*(x) = f(x); \\ \text{si } x \in S_{z^*} & \left\{ \begin{array}{l} f^* \text{ quelconque, hormis qu'il existe} \\ x^* \text{ tel que } f^*(x^*) = N. \end{array} \right. \end{array} \tag{1.65}$$

- Une heuristique, pour distinguer deux fonctions f et f^*, doit les évaluer en un x pour lequel $f^*(x) \neq f(x)$. Ainsi, avant de trouver un tel point x, a travaille sur f comme sur f^*. En conséquence vu les hypothèses antérieures, avec une probabilité d'au moins $1 - 2^{-n/3}$, l'heuristique a confrontée aux f^*, ne peut évaluer en un x de S_{z^*} durant ses premiers $2^{n/3}$ pas. Par suite, sa probabilité de succès est majorée par $2^{-n/3}$.

[19] Les auteurs étendent ce résultat par inégalité de Markov. Ceci n'est pas indispensable à la preuve en cours.

- S Droste, T Jansen, I Wegener terminent cette analyse en montrant comment les fonctions f^* sont coûteuses à évaluer a priori, mais en pratique relativement guère plus coûteuses que f, puisqu'on peut établir que le surcoût est en $\mathcal{O}(n)$.

b) Théorème du quasi No Free Lunch

Théorème 85 *ANFL ou quasi No Free Lunch*

Soit a un algorithme de recherche stochastique et f une fonction de coût avec $f : \{0,1\}^n \to \{0,1,...,N-1\}$.

Alors il existe au moins $N^{2^{n/3}-1}$ fonctions f^*:$\{0,1\}^n \to \{0,1,...,N\}$, ne diffèrant de f que sur au plus $2^{n/3}$ arguments, telles que a trouvera l'optimum de f^* en $2^{n/3}$ pas avec une probabilité majorée par $2^{-n/3}$. Un nombre exponentiel de ces fonctions ont la propriété supplémentaire que leur temps d'évaluation, leur taille de représentation en circuits, et leur complexité de Kolmogorov dominent les complexités correspondantes de f seulement d'un terme en $\mathcal{O}(n)$.

Remarque 86 *Le théorème du quasi No Free Lunch, montre qu'un algorithme de recherche doit "payer" pour son succès relativement à certaines fonctions f par un mauvais comportement sur de nombreuses autres fonctions, proches, guère plus complexes que les premières. Ainsi chaque stratégie de recherche est entravée par une certain présupposé lié à la "mémoire" des fonctions qui lui conviennent pour l'optimisation.*

c) Applications du ANFL dans quelques cas "académiques"

Les auteurs rappellent d'abord quelques fonctions classiques et le niveau de difficulté de leur traitement.

- La fonction nulle sur $\{0,1\}^n$ est facile pour toute heuristique.

- La fonction, notée $abf_{n,z}$ pour "aiguille dans une botte de foin", définie par:

$$abf_{n,z}: \{0,1\}^n \rightarrow \{0,1\}$$
$$x \mapsto \begin{cases} \text{Si } x = z \quad abf_{n,z}(x) = 1; \\ \text{si } x \neq z \quad abf_{n,z}(x) = 0. \end{cases}$$

  porte bien son nom pour l'ensemble des heuristiques ! La preuve du théorème ANFL montre que toute heuristique échoue sur beaucoup de ces fonctions $abf_{n,z}$. Une stratégie de recherche qui ne démarre pas en cherchant des points totalement au hasard peut être efficiente pour certaines fonctions $abf_{n,z}$.

- Chacun s'attend à ce que toute heuristique trouve effectivement [20] l'optimum de la fonction $OneMax_n$ définie par:

$$OneMax_n(x) = \|x\|_1 = x_1 + ... + x_n.$$

- Pourtant la fonction $Trap_n$, identique à $OneMax_n$ hormis en 0^n pour lequel $Trap_n(0^n) = n + 1$, est réputée difficile pour les heuristiques !

  On peut généraliser $Trap_n$ en définissant $Trap_{n,z}$ comme suit:

$$Trap_{n,z}(z) = n + 1 \quad \text{et} \quad [\text{Si } x \neq z, \text{ alors } Trap_{n,z}(x) = OneMax_n(x)\,].$$

  Ici encore, la preuve du théorème ANFL montre que toute heuristique est en échec pour un grand nombre de $Trap_{n,z}$. Une modification proposée par Whitley en 1997, pour rendre efficiente les heuristiques sur $Trap_n$ ne résiste pas à la généralisation.

En conclusion, certains résultats proposés par le ANFL peuvent certes être obtenus par d'autres voies, mais le théorème apporte plus de clarté, une vision moins anecdotique.

[20] ce qui a été établi pour les plus classiquement utilisées.

1.6.4 Un exemple de fonction simple et non-artificielle difficile

Les exemples évoqués en sous-section 1.6.3 ne sont pas réalistes dans les problèmes d'optimisation. L'intérêt de ces classes de fonctions est de produire des cas souvent difficiles pour les heuristiques de maximisation.

L'enjeu actuel est donc d'exhiber une fonction classiquement présente dans les problèmes réels, pressentie comme difficile pour toute heuristique ordinairement utilisée. Ce fait sera prouvé pour les algorithmes de recuit simulé et évolutionnaires.

Outils d'étude

Il convient de développer d'abord un certain nombre de résultats, utilisés ensuite comme des lemmes techniques pour conclure.

a) Fonctions symétriques décroissantes

On s'attend, pour ce type de fonctions, à ce que toute heuristique "raisonnable" trouve rapidement leur maximum, d'où leur intérêt.

Définition 87 *Une fonction f de $\{0,1\}^n$ dans $\mathbb{N}$ est dite symétrique ssi deux x de $\{0,1\}^n$ de même norme $\| \ \|_1$ ont même image.*

Définition 88 *Une fonction symétrique est dite décroissante si:*

$$\forall (x, x') \in [\{0,1\}^n]^2 \quad [(\|x\|_1 < \|x'\|_1) \Rightarrow (f(x) > f(x'))] .$$

b) Condition suffisante sur heuristiques

Hypothèse 1: Soit a une heuristique "raisonnable". Pour tout $\varepsilon > 0$ et toute suite de fonctions symétriques décroissantes (f_n), il existe des réels α et β strictement positifs tels que la probabilité que a teste parmi les premiers $\exp(o(n^\alpha))$ points, un point x pour lequel $\|x\|_1 \geq (1/2 + \varepsilon) n$, est majorée par $\exp\left(-\Omega\left(n^\beta\right)\right)$.

Remarque 89 *Cette hypothèse 1, exprimée ci-dessus techniquement, peut être interprétée de façon plus informelle. Une fonction symétrique décroissante est efficace pour des éléments x de faible norme $\|x\|_1$. Via l'inégalité de Chernoff (voir [HR89] [21]), on montre que la part de points x telle que $\|x\|_1 \geq (1/2+\varepsilon)\, n$ est majorée par $\exp(-\varepsilon^2 n/3)$. Par suite une recherche aléatoire exécutée sur $\exp(\varepsilon^2 n/6)$ pas, a une probabillité d'au plus $\exp(-\varepsilon^2 n/6)$ de tester un tel point.*

Le terme raisonnable ne peut être défini précisément. Il désigne les heuristiques qui n'ont pas de préférence quant au bassin de recherche, qui basent leur recherche sur des points de fort fitness et privilégient les voisins, au sens de la distance de Hamming.

Prouver que cette hypothèse 1 est valide pour le recuit simulé est assez simple mais nettement plus difficile pour les algorithmes évolutionnaires.

c) Cas du recuit simulé

Théorème 90 *L'hypothèse 1 précédente est valide pour les heuristiques de recuit simulé relatives à des fonctions symétriques décroissantes; les paramètres associés sont alors donnés par:*

$$\alpha = 1 \quad et \quad \beta = 1. \tag{1.66}$$

Preuve. La preuve est très technique, savante en termes de théorie des probabilités classiques. Elle est l'occasion d'un exposé très clair du modèle de recuit simulé dans la structuration même de la démonstration. ■

d) Cas des algorithmes évolutionnaires

Les auteurs rappellent de façon sobre et précise, les étapes cruciales mises en oeuvre dans ce type d'algorithmes.

[21] *On pourra aussi consulter Wikipedia, très claire et précise sur ce sujet.*

Théorème 91 *L'hypothèse 1 précédente est valide pour les heuristiques évolutionnaires relatives à des fonctions symétriques décroissantes; les paramètres associés sont alors donnés par:*

$$\alpha = \beta = \frac{1}{2}. \tag{1.67}$$

Preuve. L'idée de la preuve est de construire un processus de Markov descripteur de la méthode, qui permette d'en déduire le résultat de majoration attendu.

Ici encore les auteurs utilisent la majoration de Chernoff, mais aussi la notion d'historique utilisée pour la première fois dans [RRS95]. Ils signalent très clairement les hypothèses cruciales utilisées. ■

e) Conséquence générale

Issu directement des deux théorèmes précédents, on obtient le corollaire suivant.

Corollaire 92 *On considère f de $\{0,1\}^n$ dans $\mathbb{N}$, égale à une fonction symétrique décroissante pour tout argument x telle que $\|x\|_1 < (1/2+\varepsilon)\,n$ et qui, de plus, présente la propriété que $\|x^*\|_1 \geq (1/2+\varepsilon)\,n$ pour tous les points optimaux x^*.*

Alors la probabilité qu'une heuristique de recuit simulé ou évolutionnaire trouve l'optimum de f en $\exp\left(o\left(n^{1/2}\right)\right)$ pas, est majorée par $\exp\left(-\Omega\left(n^{1/2}\right)\right)$.

Preuve. La preuve est rapide en raison des lemmes disponibles. ■

L'EXEMPLE PROPOSÉ

Il s'agit d'une instance du problème d'optimisation NP-complet $MAXSAT$. Il vérifie les hypothèses du corollaire 92 pour $\varepsilon = 1/6$.

a) Rappel des notions de base

Un littéral est une variable booléenne v_i ou $\overline{v_i}$. Un littéral est satisfait pour l'entrée $x = (x_1, ..., x_n)$ si la variable booléenne associée prend la valeur 1. Une clause est

une disjonction de littéraux; ainsi une clause est satisfaite pour l'entrée x ssi l'un au moins des littéraux correspondants est satisfait.

Une instance du problème $MAXSAT$ est définie par une suite de clauses $c_1, c_2, ..., c_m$ sur les variables $v_1, ..., v_n$ et la question est de trouver une entrée satisfaisant simultanément un nombre maximal de clauses, comme son nom l'indique.

b) Instance proposée par Papadimitriou (1994)

On pourra consulter [P94].

L'instance proposée est constituée de n clauses de longueur 1 et de $n(n-1)(n-2)$ clauses de longueur 3. Plus précisément:

- $\forall i \in \{1, ..., n\} \quad v_i$;
- $\forall\, (i, j, k) \in \{1, ..., n\}^3$, pour tous i, j, k distincts $\quad v_i \vee \overline{v_j} \vee \overline{v_k}$.

Les diverses clauses, contenant chacune exactement un littéral positif, sont nommées clauses de Horn et sont utilisées dans des questions de base de données. Cette instance est intéressante au sens où elle est née d'un problème réel, relativement simple à formuler. Tout le monde peut voir que 1^n est la seule valeur permettant de satisfaire toutes les clauses. De plus, il existe une heuristique stochastique spécifique à $MAXSAT$, due à Papadimitriou, qui opère en temps exponentiel (voir à nouveau [P94]); mêmes performances de la meilleure variante connue pour $MAXSAT$, due à Schöning (voir [S99]).

c) Transformation du problème initial

Les auteurs transforment alors l'instance de $MAXSAT$ en un problème de décompte polynômial basé sur la fonction $Count_n$ définie sur $\{0,1\}^n$, à valeurs dans $\mathbb{N}$, qui détermine le nombre de clauses satisfaites. On montre alors le résultat suivant.

Proposition 93 *Sous les notations précédentes:*

- $Count_n(x)$ *est donnée par:*

$$Count_n(x) = \sum_{1 \leq i \leq n} x_i + \sum_{1 \leq i \leq n} \left(\sum_{\substack{1 \leq j \leq n \\ j \neq i}} \left(\sum_{\substack{1 \leq k \leq n \\ k \neq i, k \neq j}} \left[1 - (1 - x_i) x_j x_k \right] \right) \right). \quad (1.68)$$

- *La fonction* $Count_n$ *est symétrique.*

  De façon plus précise, elle peut s'exprimer en posant $s = \|x\|_1$*, à partir de la fonction* $Count_n^*$ *définie de* $\{0, 1, ..., n\}$ *dans* $\mathbb{N}$ *par:*

$$Count_n^*(s) = s^3 - (n+1)s^2 + (n+1)s + n(n-1)(n-2). \quad (1.69)$$

  D'où il vient que:

$$\forall x \in \{0,1\}^n \quad Count_n(x) = Count_n^*(\|x\|_1). \quad (1.70)$$

- *La fonction* $Count_n^*$ *est décroissante pour tout* s *vérifiant:* $0 \leq s < (2/3)\, n$*; donc le corollaire 92 est applicable pour* $\varepsilon = 1/6$*. On en tire le résultat final suivant.*

Corollaire 94 *Sous les notations antérieures, la probabilité qu'une heuristique de recuit simulé ou évolutionnaire trouve l'optimum de* $Count_n$ *en* $\exp\left(o\left(n^{1/2}\right)\right)$ *pas, est majorée par* $\exp\left(-\Omega\left(n^{1/2}\right)\right)$.

Ainsi la fonction $Count_n$ ne peut être optimisée de façon efficiente par une quelconque heuristique vérifiant l'hypothèse 1 proposée. Par suite, les auteurs de [DJW02] conjecturent qu'aucune heuristique raisonnable n'est efficiente pour $Count_n$.

1.6.5 Bilan relatif au théorème du Quasi No Free Lunch

- Dans la deuxième partie du papier [DJW02], présentée ci-dessus dans la sous-section 1.6.3, S Droste, T Jansen, I Wegener montrent comment on peut trouver un grand nombre de fonctions f^* proches et guère plus complexes qu'une fonction de coût $f : \{0,1\}^n \rightarrow \{0,1,...,N-1\}$ donnée, pour lesquelles un algorithme de recherche stochastique a, est en difficulté, même s'il a réussi à optimiser f dans des conditions satisfaisantes.

- Le fait que a soit adapté pour f, se "paie" en quelque sorte autour de f !

- Enfin, les auteurs mettent en évidence dans la sous-section 1.6.4 une fonction réaliste mais non triviale, qui s'avère très difficile, pour l'ensemble des algorithmes classiquement utilisés en optimisation, de quoi désespérer d'un coup l'ensemble de la profession...

1.7 Conclusion générale sur le NFL

- Dans la première section de ce chapitre, nous rappelons les résultats issus de l'article de DH Wolpert et WG Macready dans sa version d'avril 1997 [WM97].

  - Après l'exposé des notations indispensables, nous rappelons le résultat central, appelé théorème du No Free Lunch (NFL).

    Etant donnés deux ensembles finis X (l'espace de recherche) et Y totalement ordonné (l'espace des coûts), on considère l'ensemble $\mathcal{F} = Y^X$ des fonctions de coût f, applications de X dans Y. Soit deux algorithmes a et a', déterministes ou stochastiques, destinés à optimiser f sur X.

    Alors en moyenne sur $\mathcal{F} = Y^X$, les deux algorithmes sont également performants !

  - Puis nous développons les premières conséquences suivies d'une interprétation géométrique des théorèmes du NFL. Pour cette dernière nous apportons quelques précisions, à notre avis indispensables, à la manière de voir des auteurs.
  - Nous évoquons ensuite les applications, de nature calculatoire, des résultats antérieurs ainsi qu'une série de corollaires qui mettent en évidence la richesse de cette contribution.

- Dans un seconde temps, nous fournissons un descriptif des réactions, les plus importantes à nos yeux, qui ont suivi le papier initial.

  - Tout d'abord, section 1.3, nous présentons une confirmation expérimentale des théorèmes précités, menée par S Droste, T Jansen et I Wegener en 1999 [DJW99]. Ils analysent, sur divers sous-ensembles de $\mathcal{F} = Y^X$ liés à la complexité de représentation des fonctions considérées via les OBBD, le comportement d'algorithmes nettement différents. Nous annonçons un exemple réaliste de fonction de coût (voir sous-section 1.6.4) qui s'annonce très difficile pour l'ensemble des méthodes ordinairement mises en oeuvre en optimisation.
  - Puis nous invitons à une vision transversale dans la section 1.4, grâce à l'apport de D Whitley [W00], qui fournit quelques indications importantes sur ce qu'on est en droit d'attendre ou non des statistiques. L'analyse de Walsh, cruciale, sous-tend les résultats; elle conduit à la preuve de l'impossibilité pour les statistiques descriptives de constituer un guide déterminant dans l'exécution d'un algorithme de recherche travaillant sur un problème NP.

- Nous tentons ensuite de cerner l'émergence de la caractérisation des zones de validité des théorèmes du NFL parmi l'ensemble des fonctions de coût.
  - Production des définitions, notations, propriétés premières dans la sous-section 1.5.1.
  - Mise en évidence de la rareté des zones de validité des théorèmes, dans la sous-section 1.5.2.
  - Présentation d'une version épurée, simplifiée, allée à l'essentiel, de la caractérisation des champs de validité dans la sous-section 1.5.3.
- Enfin dans la section 1.6 nous parachevons l'étude de validité au niveau local. Nous montrons grâce à [DJW02], comment le théorème du NFL ne reste pas cantonné à une zone réduite et "collectée" de l'ensemble des fonctions de coût. Il se diffuse littéralement localement, sous forme d'un théorème de Quasi No Free Lunch, à tout l'espace $\mathcal{F} = Y^X$; il entoure alors même les fonctions pour lesquelles un algorithme s'est avéré satisfaisant d'une zone de quasi No Free Lunch et fait "payer" cette réussite.

• A ce stade de notre travail, il nous paraît important de voir que l'approche globale des questions, que la prise en considération de l'ensemble des fonctions de coût, sont indispensables. Il n'est plus possible de rester dans ce jeu de miroir, narcissique et souvent mortel, d'une fonction de coût et de "son algorithme dévolu". Des pistes s'ouvrent résolument:

- recherche systématique de propriétés générales liant fonctions de coût et algorithmes;
- pour une même fonction de coût f, recherche de similitudes entre certains jeux de données;

- étude fine des modalités de convergence des algorithmes lorsqu'ils fournissent des résultats satisfaisants;
- mais aussi étude approfondie de la manière dont les algorithmes se perdent, par instants !
- La présence enfin d'une confrontation régulière à l'expérience nous paraît indispensable à tous les niveaux de cette recherche, mais dans un cadre choisi pour être fécond. Nous élisons la coloration de graphe comme exemple référence. Le rôle essentiel de modèle, joué par ce problème, sa capacité à représenter d'autres algorithmes dans le cadre d'une sorte de morphisme entre problèmes, nous semblent suffisants pour justifier ce choix.

1.8 Annexe: analyse de Walsh

Nous fournissons le plan des preuves de la proposition 24 et du théorème 25.

1.8.1 Preuve de la proposition 24

- Pour tout j de $\left\{0, ..., 2^L - 1\right\}$
  - par la définition même de $\psi_j\left(x + x'\right)$, il vient:

$$\psi_j\left(x + x'\right) = (-1)^{j^t(x+x')} = (-1)^{j^t(x)}\,(-1)^{j^t(x')}.$$

  - L'image réciproque H_j de 1est le noyau de ψ_j, donc un sous-groupe de $\left(\left(\mathbb{Z}/2\mathbb{Z}\right)^L, +\right)$.
- Selon j:
  - Si $j = 0$, $\psi_j = \psi_0$ vérifie pour tout x de $\{0,1\}^L$: $\psi_0(x) = (-1)^0 = 1$.

– Si non, $j \neq 0$, admet au moins une composante égale à 1. L'élément de $\{0,1\}^L$ constitué de 0 hormis cette composante a pour image -1 par ψ_j. L'image réciproque de -1 par ψ_j est une classe d'équivalence modulo le sous-groupe H_j, donc par théorème de Lagrange est en bijection avec H_j. D'où l'égalité des cardinaux indiquée et la conclusion proposée.

1.8.2 Preuve du théorème 25

1. Produit scalaire sur $\mathcal{L}$

   La forme φ définie sur $\mathcal{L} \times \mathcal{L}$ par $\langle f, g \rangle = \sum_{j=0}^{2^L-1} f(j)\, g(j)$ est symétrique, bilinéaire, définie, positive.

   - La symétrie provient de la commutativité du produit réel; la bilinéarité se réduit donc à une linéarité monolatère.
   - La linéarité est évidente.
   - Les caractères positif et défini, proviennent de ce que les valeurs prises par f sont réelles.

   N.B. φ est la version discrète du produit scalaire bien connu, défini pour des fonctions continues par une intégrale sur des compacts.

2. La famille $\left(\psi_j\right)_{0 \leq j \leq 2^L-1}$ est orthogonale. En effet:

   - Si $j \neq k$, alors:

$$\left\langle \psi_j, \psi_k \right\rangle = \sum_{x=0}^{2^L-1} \psi_j(x)\, \psi_k(x). \tag{1.71}$$

$$\text{D'où } \left\langle \psi_j, \psi_k \right\rangle = \sum_{x=0}^{2^L-1} (-1)^{j^t(x)} (-1)^{k^t(x)} = \sum_{x=0}^{2^L-1} (-1)^{(j+k)^t(x)}.$$

Mais $j+k \neq 0$ car dans $(\mathbb{Z}/2\mathbb{Z})^L$, $j+k=0$ ssi $j=k$. D'après la proposition précédente, le nombre de 1 est le même que celui de -1, donc la somme est nulle; d'où l'orthogonalité de ψ_j et ψ_k.

- Si $j=k$, par le même calcul que précédemment, il vient pour tout j:

$$\left\|\psi_j\right\|^2 = \left\langle \psi_j, \psi_j \right\rangle = \sum_{x=0}^{2^L-1} (-1)^{(j+j)^t(x)} = \sum_{x=0}^{2^L-1} (-1)^{(0)^t(x)} = 2^L. \quad (1.72)$$

En conséquence, cette famille orthogonale de vecteurs non nuls de $\mathcal{L}$, de cardinal la dimension de $\mathcal{L}$ est une base de $\mathcal{L}$ non normée puisque L est non nul.

3. On déduit de ce qui précède l'existence des coefficients de Walsh, qui sont les composantes d'une fonction quelconque f de $\mathcal{L}$ dans la base de Walsh. Pour tout f de $\mathcal{L}$, il existe des réels w_j tels que pour tout x de $\{0,1\}^L$, on ait:

$$f = \sum_{j=0}^{2^L-1} w_j.\psi_j \quad (1.73)$$

Vu le caractère orthogonal de la base de Walsh, il vient de façon totalement standard, pour tout i:

$$\begin{aligned} \langle f, \psi_i \rangle &= w_i \langle \psi_i, \psi_i \rangle \quad (1.74) \\ \text{d'où } w_i &= \frac{1}{2^L} \langle f, \psi_i \rangle = \frac{1}{2^L} \sum_{j=0}^{2^L-1} \psi_i(j) f(j). \end{aligned}$$

1.8.3 Preuve de la proposition 26

On confond toute fonction f de $\mathcal{L}$ avec le 2^L-uplet $\left(f(0), f(1), ..., f\left(2^L-1\right)\right)$ de $\mathbb{R}^{2^L}$ rapporté à sa base canonique $\mathcal{B} = (e_j)_{0 \leq j \leq 2^L-1}$; par suite f admet pour matrice colonne de composantes dans cette base la matrice F. On note W la matrice colonne des composantes de f dans la base de Walsh $\mathcal{B}' = \left(\psi_j\right)_{0 \leq j \leq 2^L-1}$.

1. Version 1 de preuve

   Ecrire pour tout i de $\left\{0,...,2^L-1\right\}$ que $w_i = \frac{1}{2^L}\sum_{j=0}^{2^L-1}\psi_i(j)f(j)$ c'est transcrire l'égalité matricielle:

$$w_i = \frac{1}{2^L}\left(\psi_i(0)\ \ \psi_i(1)\ \ ...\ \psi_i\left(2^L-1\right)\right) \times^t \left(f(0)\ f(1)...\ f\left(2^L-1\right)\right)$$

   soit finalement:

$$w_i = \frac{1}{2^L}\left(\psi_i(0)\ \ \psi_i(1)\ \ ...\ \psi_i\left(2^L-1\right)\right) \times F. \tag{1.75}$$

   On en déduit l'égalité matricielle proposée:

$$W = \frac{1}{2^L}UF \tag{1.76}$$

   où U est la transposée de la matrice de passage de $\mathcal{B}$ à $\mathcal{B}'$, c'est-à-dire la matrice dont la ligne i est donnée par:

$$\begin{aligned} U_i &= \left(\psi_i(0)\ \ \psi_i(1)\ \ ...\ \psi_i\left(2^L-1\right)\right), \qquad (1.77)\\ \text{ou encore } U_{ij} &= \psi_i(j). \end{aligned}$$

2. Version 2 de preuve

- Considérons l'application u définie par:

$$\begin{array}{rccc} u: & \left(\mathbb{R}^{2^L},(e_j)_{0\le j\le 2^L-1}\right) & \to & \left(\mathcal{L},(\psi_j)_{0\le j\le 2^L-1}\right) \\ & \left(f(0),f(1),...,f\left(2^L-1\right)\right) & \mapsto & u(f)=f \end{array}. \tag{1.78}$$

  u est évidemment linéaire (il s'agit en fait de l'identité).

- Par suite, on en déduit l'égalité matricielle: $W = VF$ (sous notation française), où V désigne la matrice de cette application linéaire relativement aux bases $(e_j)_{0\le j\le 2^L-1}$ et $(\psi_j)_{0\le j\le 2^L-1}$, c'est-à-dire la matrice de passage de $\mathcal{B}'$ à $\mathcal{B}$. Déterminons son inverse $V^{-1} = P = P_{\mathcal{B}}^{\mathcal{B}'}$ qui est plus facilement connue dans notre contexte.

- La colonne j de P est constituée des composantes de ψ_j dans la base $\mathcal{B}$, c'est-à-dire ${}^t\left(\psi_j(0)\ \ \psi_j(1)\ \ ...\ \psi_j\left(2^L-1\right)\right)$. Comme les vecteurs ψ_j sont de norme $2^{L/2}$, la matrice $\frac{1}{2^{L/2}}P$ est orthogonale comme matrice d'isométrie de $\mathcal{L}$, donc son inverse est sa transposée. Par suite:

$$\begin{aligned} P &= 2^{L/2}\left(\frac{1}{2^{L/2}}P\right) \\ \text{donc } V &= P^{-1} = \frac{1}{2^{L/2}} \times \left[{}^t\left(\frac{1}{2^{L/2}}P\right)\right], \\ \text{soit encore} &: V = \frac{1}{2^{L/2}}\left[\frac{1}{2^{L/2}}\left({}^tP\right)\right] = \frac{1}{2^L}\left({}^tP\right) = \frac{1}{2^L}U, \end{aligned} \tag{1.79}$$

avec $U=(U_{ij})$ et donc $U_{ij}=\psi_i(j)$. Finalement: $W=\frac{1}{2^L}UF$.

Chapitre 2

Vers une approche globale d'un graphe: décomposition en cliques maximales, suites constructives

2.1 Introduction

Comme l'ont montré les théorèmes du No Free Lunch, le fait de considérer les problèmes dans leur globalité est garant de résultats d'une part, de promesse d'éléments de compréhension des phénomènes d'autre part. Dans ce chapitre, nous tentons d'aborder dans cet esprit la question de la coloration des graphes finis, non orientés et simples.

Nous avons retenu ce champ d'étude en raison de sa fécondité propre, qui n'est plus à prouver dans la communauté de l'optimisation. Chacun(e) connaît de surcroît, son efficience majeure comme outil de transcription d'algorithmes de natures parfois bien différentes. Sans doute, une notion d'homomorphisme d'algorithmes opère-t-elle dans l'ombre...

Notre étude sera menée comme suit:

- Nous commençons par rappeler un certain nombre de définitions générales relatives aux graphes, adoptons quelques conventions d'écriture et présentons la notion centrale de décomposition en cliques maximales ou premières (primary clique) pour un graphe donné.

- Puis nous étudions les "liens" d'une clique de la décomposition avec le graphe donné G. Par là, nous définissons une suite constructive d'un graphe connu par l'ensemble de ses cliques maximales.

- Nous annonçons dès lors la finalité de ces considérations parfois assez techniques... Nous utiliserons dans le chapitre suivant cette reconstruction de G pour déterminer les colorations propres du graphe G, et retrouverons en particulier leur nombre, donc le polynôme chromatique du graphe étudié, mais surtout les regrouperons en une partition de classes.

Un certain nombre de résultats d'expérimentations ont accompagné l'émergence des outils généraux, de façon systématique. Menées de façon indépendante, elles ont permis d'affiner les notions de base, de formuler certaines conjectures et de les vérifier localement avant l'indispensable passage à la preuve... chaque fois que cela fut possible !

2.2 Graphes

2.2.1 Conventions d'écriture et de notations

Nous notons $|E|$ le cardinal d'un ensemble E.

Tous les graphes considérés sont supposés finis, non orientés et simples; ces qualificatifs seront désormais en général omis mais implicites, sauf indications contraires. Les graphes considérés ne contiennent pas en général de sommets isolés; ce cas, trivial en termes de coloration, ramène facilement au traitement du cas générique auquel nous nous limitons a priori, comme étant le seul ... sérieux.

Les ensembles de sommets sont non vides et finis; celui des arêtes est fini par suite, et supposé non vide en général.

2.2.2 Définitions de base

Définition 95 *Graphe non orienté et simple*

Un graphe non orienté et simple G est un couple (X, E), où X désigne un ensemble fini de n sommets $\{x_1, ..., x_n\}$ et E l'ensemble des arêtes, paires non ordonnées $\{x_i, x_j\}$ d'éléments distincts de X; la paire $\{x_i, x_j\}$ est dite d'extrémités x_i et x_j.

Définition 96 *Matrice associée au graphe G*

On appelle matrice associée au graphe G la matrice carrée de $\mathcal{M}_n(\{0,1\})$ [1], notée $\mathcal{G} = (g_{ij})$, définie par:

$$\forall (i,j) \in \{1, ..., n\}^2 \qquad \{[g_{ij} = 1] \Leftrightarrow [(i < j) \ \ et \ \ \{x_i, x_j\} \in E]\}. \tag{2.1}$$

N.B. On remarque que la contrainte $(i < j)$ impose que $\mathcal{G}$ est triangulaire supérieure, de diagonale nulle; cette forme suffit évidemment à représenter tout graphe vérifiant les hypothèses antérieures. On aurait pu effectuer d'autres choix. La matrice associée $\mathcal{G}$ est donc une "demi-matrice" d'adjacence du graphe.

Définition 97 *Matrice d'adjacence de G*

Sous les notations précédentes on appelle matrice d'adjacence de G, la matrice carrée de $\mathcal{M}_n(\{0,1\})$, notée $\mathcal{A} = (a_{ij})$, avec:

$$\forall (i,j) \in \{1, ..., n\}^2 \qquad \{[a_{ij} = 1] \Leftrightarrow [\{x_i, x_j\} \in E]\}.$$

Ainsi $\mathcal{A} = \mathcal{G} + ({}^t\mathcal{G})$.

N.B. En cas d'ambiguïté, $\mathcal{A}$ sera notée $\mathcal{A}_G$.

Définition 98 *Graphe complémentaire du graphe G*

Etant donné le graphe $G = (X, E)$, on appelle graphe complémentaire de G le graphe, noté $\overline{G}$, défini par: $\overline{G} = (X, \mathcal{P}_2(X) - E)$, où $\mathcal{P}_2(X)$ désigne l'ensemble des parties de X de cardinal 2.

[1] $\{0,1\}$ *est ici doté de sa structure d'algèbre de Boole standard.*

Définition 99 *Sous graphe induit par $G = (X, E)$ sur X'*

Soit $X' \subset X$. $G' = (X', E')$ est dit sous-graphe induit de G sur X' ssi:

$$E' = E \cap \mathcal{P}(X').$$

Définition 100 *Cliques de G*

- *Une partie C_r de X, de cardinal r ($r \in \mathbb{N}^*$), est dite une clique d'ordre r (ou de taille r, ou une r-clique) de G si et seulement si le sous-graphe de G induit sur C_r est complet. Autrement dit, la clique C_r représente le sous-graphe $(C_r, E(C_r))$ de G, tel que:*

$$\forall (x_i, x_j) \in C_r^2 \quad [(x_i \neq x_j) \Rightarrow (\{x_i, x_j\} \in E(C_r))] \tag{2.2}$$

Ainsi:

$$E(C_r) = \{\{x_i, x_j\} \in E \quad / \quad x_i \in C_r \text{ et } x_j \in C_r\}.$$

- *Une telle clique C_r est dite maximale dans G s'il n'existe pas de clique de G d'ordre strictement supérieur à r contenant C_r. Il s'agit de la maximalité ordinaire, au sens de l'inclusion.*

Définition 101 *Stables de G*

Une partie S de X est dite un stable du graphe $G = (X, E)$ si et seulement si le sous-graphe de G induit par S est sans arêtes. Autrement dit un stable S de G est le sous-graphe $(S, \varnothing)$ de G.

Remarque 102 *Les notions de clique et de stable de G sont intimement liées puisqu'elles s'échangent si l'on considère le graphe complémentaire.*

2.3 Décomposition d'un graphe en cliques maximales

2.3.1 Généralités

Proposition 103 *Soit G un graphe. Tout sommet de G appartient à au moins une clique maximale de G.*

Preuve. Elle est très simple et classique.

Le point de départ est qu'un singleton constitue une clique de G. La finitude de l'ensemble des sommets entraîne l'impossibilité de construire une suite infinie de cliques, strictement croissante de premier terme un singleton donné. Il s'agit d'une propriété noethérienne ordinaire.

N.B. Il convient de remarquer que dans un graphe G, il n'y a pas nécessairement unicité pour une clique maximale contenant un sommet donné, comme les exemples de la section 2.3.4 le montreront. ■

Théorème 104 *et définition*

Etant donné un graphe G, il existe un ensemble unique de cliques maximales pour G.

Cet ensemble est appelé la décomposition de G en cliques maximales; il sera noté $\mathcal{D}(G)$.

Preuve. La preuve est immédiate. Nous tenons toutefois à identifier nettement ce résultat, au sens où il annonce la problématique de sa réciproque qui sera abordée dans le chapitre 4. A quelles conditions un ensemble de parties de X représente-t-il une décomposition d'un sous-graphe de $G = (X, E)$? ■

Notation 105 *La décomposition $\mathcal{D}(G)$ du graphe G sera notée comme suit:*

$$\mathcal{D}(G) = \bigcup_{i=1}^{p} S_i,$$

sachant que pour tout i de $\{1, ..., p\}$, S_i est l'ensemble $\{C_{r_{i,1}}, ..., C_{r_{i,n_i}}\}$ des n_i cliques maximales de G d'ordre r_i, avec pour tout i de $\{1, ..., p\}$, $n_i \geq 1$. Sauf indication contraire, $(r_i)_{1 \leq i \leq p}$ désigne la suite décroissante des ordres des cliques maximales présentes dans la décomposition de G. Nous noterons désormais N le nombre total de cliques maximales présentes dans la décomposition de G. Ainsi:

$$N = \sum_{i=1}^{p} n_i.$$

N.B. On regroupe donc les cliques maximales par tailles décroissantes; il peut exister plusieurs cliques maximales de G de même ordre; certains ordres peuvent évidemment être "absents".

Proposition 106 *Liens entre cliques maximales de G_n et de G_{n+1}*

Considérons un graphe G_{n+1} comportant $n+1$ sommets $\{x_1, ..., x_{n+1}\}$. Soit G_n le sous graphe de G_{n+1} induit par $\{x_1, ..., x_n\}$.

1. *Si x_{n+1} n'est adjacent à aucun élément de $\{x_1, ..., x_n\}$, la clique $\{x_{n+1}\}$ est maximale dans G_{n+1}.*

2. *Soit $M_{\mathcal{C}}$ une clique maximale de G_n.*

   (a) *Si x_{n+1} est adjacent à tous les sommets de $M_{\mathcal{C}}$,*

   alors $M_{\mathcal{C}} \cup \{x_{n+1}\}$ est une clique maximale de G_{n+1} d'ordre $|M_{\mathcal{C}}| + 1$. Par suite $M_{\mathcal{C}}$ n'est plus maximale pour G_{n+1}. Il devient inutile d'étudier $P \cup \{x_{n+1}\}$, où P désigne une partie propre de $M_{\mathcal{C}}$ puisqu'elle ne sera pas maximale pour G_{n+1}.

   (b) *Si x_{n+1} n'est pas adjacent à tous les éléments de $M_{\mathcal{C}}$ mais seulement à tous ceux d'une de ses parties propres, P, non vide,*

   alors $P \cup \{x_{n+1}\}$ est potentiellement maximale pour G_{n+1}; elle pourra être absorbée par une autre clique la contenant, de taille supérieure, issue d'une autre clique maximale $M'_{\mathcal{C}}$ de G_n.

Preuve. Elle permettra la construction des cliques maximales de G_{n+1} à partir de celles de G_n.

1. Ce point a été vu plus haut.

2. Soit $M_{\mathcal{C}}$ une clique maximale de G_n.

   (a) S'il existe une clique de G_{n+1} contenant $M_{\mathcal{C}} \cup \{x_{n+1}\}$, nécessairement elle est égale puisque $M_{\mathcal{C}}$ est maximale dans G_n; d'où le résultat annoncé.

   (b) Un contre-exemple permet de mettre en évidence le caractère non nécessairement maximal de $P \cup \{x_{n+1}\}$, d'où le résultat.

■

Remarque 107 *On notera que l'ensemble des cliques maximales défini au théorème 104 contient la totalité de l'information sur le graphe initial G.*

- *Connaître G c'est connaître les ensembles $S_1, ..., S_p$ des cliques maximales de G d'ordre respectifs $r_1, ..., r_p$.*

- *La réciproque doit être vraie sous conditions: la donnée d'un ensemble de cliques maximales fait naître un graphe non orienté et simple, unique, sous réserve du respect de propriétés, de compatibilté en particulier, qui assurent que l'ensemble des cliques données en représente la décomposition en cliques maximales. Cet aspect sera précisé avec la notion de suite constructive d'un graphe.*

- *Nous disposons ici de la promesse d'un outil puissant, une sorte d'équivalent des nombres premiers (on pourrait d'ailleurs les nommer cliques premières au sens de "primary clique") dans un anneau factoriel, ou des monômes premiers d'une fonction booléenne. Au demeurant, la question n'est pas tellement de*

connaître chacun des graphes possibles et ses propriétés particulières, mais d'établir des résultats qui les concernent possiblement tous, tout comme il ne présente plus aucun intérêt de résoudre toutes les équations de degré donné à coefficients dans un corps... même si, parfois, cette question est incontournable pour certaines applications exactes ou approchées.

- *Il convient ici de remarquer le lien intime existant entre ce concept de clique maximale d'un graphe donné et les problématiques de coloration (voir section 3.2). Ces deux notions sont faites "pour s'entendre", comme nous le verrons par la suite; cette "évidence" est d'ailleurs immédiatement sensible ! Elle a été fréquemment notée, mais souvent limitée à la recherche de "la clique maximale" du graphe; on pourra voir [JT96]. Par ailleurs, le concept de degré, intimement lié à l'appartenance d'un sommet à plusieurs cliques maximales, est un indicateur de multi-appartenance "probable" mais joue parfois un rôle de leurre, en termes de difficulté de coloration, comme peuvent le montrer des exemples simples.*

2.3.2 Graphe engendré par une famille de cliques

Définition

Définition 108 *Soit un ensemble $X = \{x_1, ..., x_n\}$ de n sommets.*

On considère un ensemble $\mathcal{D} = \{c_1, ..., c_N\}$ de parties non vides de X. Pour tout i de $\{1, ..., N\}$, on note E_i la partie de $\mathcal{P}(X)$ définie par:

$$E_i = \left\{ \{x_j, x_l\} \ / \ (l,j) \in \{1, ..., n\}^2, \ l \neq j \ et \ (x_j, x_l) \in (c_i)^2 \right\}.$$

On appelle graphe engendré par $\mathcal{D}$ le graphe $G_{\mathcal{D}} = (X_{\mathcal{D}}, E_{\mathcal{D}})$ avec $X_{\mathcal{D}} = \cup_{i=1}^{N} c_i$ et $E_{\mathcal{D}} = \cup_{i=1}^{N} E_i$.

Ainsi $\mathcal{D}$ est un ensemble de cliques de $G_{\mathcal{D}}$.

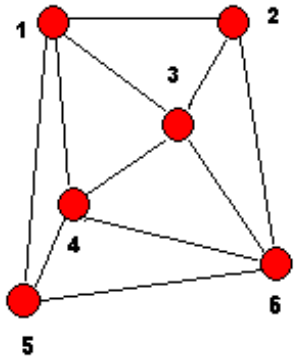

Figure 2.1: Graphe $G_{\mathcal{D}}$

Remarque 109 *Une famille $\mathcal{D} = \{c_1, ..., c_N\}$ ne constitue pas nécessairement l'ensemble des cliques maximales de $G_{\mathcal{D}}$, même si les parties c_i vérifient des propriétés d'intersections ... sympathiques.*

- *Par exemple pour le graphe $G_{\mathcal{D}}$ représenté en figure 2.1, l'ensemble $\mathcal{D}$ donné par:*

$$\mathcal{D} = \{\{3,4,6\}, \{1,2,3\}, \{2,3,6\}, \{1,4,5\}, \{4,5,6\}\}$$

  suffit à apporter la totalité des arêtes de $G_{\mathcal{D}}$ mais n'en constitue pas pour autant l'ensemble des cliques maximales puisque $\{1,3,4\}$ fait défaut.

- *Définir les ensembles de cliques, parties non vides de $X = \{x_1, ..., x_n\}$ qui représentent effectivement des décompositions en cliques maximales d'un graphe donné est une problématique entière et délicate qui fournirait une réciproque du théorème de décomposition. Elle sera abordée dès la section 2.5*

consacrée aux suites constructives de graphes, mais non totalement solutionnée; on la retrouvera au chapitre 4.

- *On voit assez vite quelques conditions nécessaires pour qu'un ensemble de cliques donné constitue un ensemble de cliques maximales; elles ne doivent pas être incluses l'une dans l'autre, par exemple. Par contre, il n'est pas du tout évident de voir qu'il ne manque pas de clique maximale pour constituer une décomposition totale.*

  - *Il peut manquer des cliques "au-dessus" de cliques données: par exemple, on peut fournir quatre 3-cliques, sans se rendre compte que la 4-clique les contenant, les remplacerait avantageusement...*
  - *On peut aussi disposer de toutes les arêtes d'un graphe engendré par un ensemble de cliques donné, sans que ce dernier représente sa décomposition en cliques maximales, comme dans l'exemple proposé figure 2.1.*

Lemme de Gauss sur les cliques maximales d'un graphe

On considère un graphe $G = (X, E)$ avec $X = \{x_1, ..., x_n\}$ et sa décomposition en cliques maximales $\mathcal{D}(G) = \{c_1, ..., c_N\}$.

Proposition 110 *Pour tout $k \in \{1, ..., N\}$, on note G_k le graphe engendré par $\{c_1, ..., c_k\}$. Alors les cliques $c_1, ..., c_k$ sont maximales pour le graphe G_k.*

Preuve. Si l'une des c_i pour i élément de $\{1, ..., k\}$ n'est pas maximale pour G_k, alors il existe une sur-clique stricte de c_i dans G_k, qui est aussi une sur-clique stricte de c_i dans G; ainsi c_i n'est pas maximale dans G. C'est exclu. ■

Lemme 111 *de Gauss*

Soit c_1 et c_2 deux cliques maximales distinctes de $\mathcal{D}(G)$. On note G_2 le graphe engendré par $\{c_1, c_2\}$. Si c désigne une clique maximale de G_2, alors nécessairement $c = c_1$ ou $c = c_2$.

Preuve. Le résultat est valide pour toutes les cliques maximales c_1, c_2, même si elles sont d'ordre 1; ce cas présente par contre, peu d'intérêt. Elles paraissent être l'équivalent des éléments inversibles dans un anneau, en termes de décomposition en facteurs premiers.

- Premier cas : l'une au moins de c_1 et c_2 est d'ordre 1.

  Sans perdre de généralité on peut supposer $c_2 = \{s_2\}$. Désignons par c une clique maximale de G_2. Si $c = c_2$ le résultat est vrai; sinon c est différente de c_2. Or c est incluse dans $c_1 \cup c_2$; par suite c est incluse dans c_1. Comme ce sont deux cliques maximales de G_2, il vient $c = c_1$.

- Deuxième cas : c_1 et c_2 sont d'ordre 2 au moins.

  Soit c une clique maximale de G_2; nécessairement c est d'ordre au moins 2, sinon elle serait incluse strictement dans c_1 ou dans c_2 donc non maximale.

  Si c est à la fois distincte de c_1 et de c_2, il existe dans c au moins un sommet s_1 de $c_1 - c_2$ et un sommet s_2 de $c_2 - c_1$ tel que $\{s_1, s_2\}$ est une arête du graphe engendré par $\{c_1, c_2\}$. C'est impossible, en raison de la nature des arêtes de G_2; leurs extrémités doivent être toutes deux dans c_1 ou dans c_2.

■

Corollaire 112 *Sous les notations du lemme précédent, $\{c_1, c_2\}$ représente la décomposition en cliques maximales du graphe G_2.*

Preuve. En effet, c_1 et c_2 sont des cliques maximales de G_2 et il n'en existe pas d'autres. ■

Remarque 113 *Nous nous proposons maintenant de généraliser le résultat précédent. Pour ce faire, l'ensemble* $\{c_1, ..., c_N\}$ *des cliques considérées doit vérifier certaines contraintes, qui interviendront ensuite. Par conséquent nous formalisons ces contraintes par le biais d'une définition.*

Définition 114 *Suite constructive d'un graphe*

Soit un graphe G *et sa décomposition en* N *cliques maximales* $\mathcal{D}(G)$*; on écrit* $\mathcal{D}(G) = \{c_1, ..., c_N\}$.

- *Notations*

  Pour tout $j \in \{1, ..., N\}$*, soit* G_j *le graphe engendré par* $\{c_1, ..., c_j\}$*; donc* G_j *est un sous-graphe de* G_{j+k} *pour tout* k *élément de* $\mathbb{N}$ *vérifiant* $1 \leq k \leq N - j$*. On se propose de comparer les graphes* G_{j+k} *et* G_j*.*

  - *Soit il n'existe pas de* $k > 0$ *tel que* $G_{j+k} = G_j$*.*

    Ceci signifie que la clique maximale c_{j+1} *"apporte" des sommets et/ou des arêtes non présents dans* G_j *et par suite que* G_j *est distinct de* G_{j+1}*.*

  - *Soit il existe un* $k > 0$ *tel que* $G_{j+k} = G_j$*. Il existe alors un plus grand entier* $k_j \leq N - j$ *vérifiant* $G_{j+k_j} = G_j$*.*

    Cela signifie que les cliques c_{j+1}*, ...,* c_{j+k_j} *n'apportent ni sommets nouveaux, ni arêtes nouvelles au graphe* G_j*. Elles constituent seulement en quelque sorte des "cliques clés de voûte"; les arêtes concernées sont déjà existantes apportées par des cliques antérieures. Voir la figure 2.2: la clique maximale* $\{2, 5, 6\}$*, par exemple, n'apporte ni sommets ni arêtes nouvelles, une fois les sept autres prises en compte.*

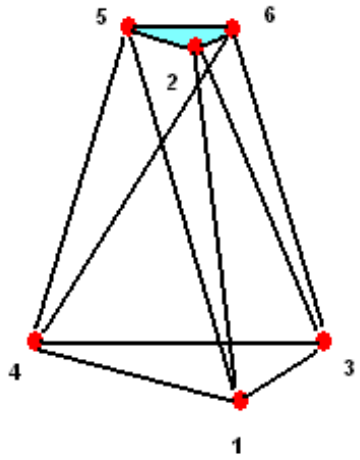

Figure 2.2: Exemple de clique clé de voûte

– *Les deux cas précédents se regroupent, en notant k_j l'entier naturel k maximal, éventuellement nul, tel que $G_{j+k_j} = G_j$.*

- *Définition proprement dite*

  Sous les notations précédentes, on dit que la suite $(c_1, ..., c_N)$ est constructive pour le graphe G si et seulement si elle vérifie la condition $(\mathcal{C})$ suivante:

  $(\mathcal{C})$: Pour tout j élément de $\{1, ..., N\}$, on considère le graphe G_j engendré par $\{c_1, ..., c_j\}$, dont on sait par l'étude antérieure qu'il est identique à G_{j+k_j}. Alors aucune clique c_t de la suite $(c_1, ..., c_N)$, avec $t > j + k_j$ (si toutefois il en existe !), n'est une clique maximale de G_j.

  N.B. 1: Dire qu'une suite de cliques vérifie $(\mathcal{C})$, signifie que les cliques qui "viennent après" c_{j+k_j} n'ont pas été "oubliées" en cours de construction de G, que G_{j+k_j} présente une certaine complétude, une sorte de totale décomposition

en cliques maximales. Le fait de contenir telle ou telle clique maximale apparaît ici pour un graphe comme l'équivalent d'une divisibilité d'un entier par un nombre premier dans $\mathbb{Z}$.

N.B. 2: Une suite constructive permet donc de fournir une suite croissante de graphes (G_j) *qui conduit au graphe total; en général la croissance est stricte, hormis quelques "replats" qui constituent autant de phénomènes noethériens.*

N.B. 3: L'existence d'une telle suite sera prouvée plus loin, au moment de l'étude de l'algorithme dévolu qui permettra d'en construire une et pas seulement d'en affirmer l'existence formelle.

Lemme 115 *Lemme de Gauss généralisé*

Soit G *un graphe et sa décomposition* $\{c_1, ..., c_N\}$ *en* N *cliques maximales; on suppose que la suite* $(c_1, ..., c_N)$ *est constructive pour le graphe* G.

Soit j *un élément de* $\{1, ..., N\}$, G_j *le graphe engendré par* $\{c_1, ..., c_j\}$; k_j *désigne l'entier naturel mis en évidence sous la définition 114.*

Alors:

1. *Si* c *est une clique maximale de* G_j, *il existe* i *de* $\{1, ..., j, ..., j + k_j\}$ *tel que* $c = c_i$.

2. *Autrement dit,* G_j *se décompose en* $j + k_j$ *cliques maximales distinctes; sa décomposition est donnée par:*

$$\mathcal{D}(G_j) = \left\{c_1, ..., c_j, ..., c_{j+k_j}\right\}.$$

Preuve. On prouve ce lemme généralisé par récurrence sur le nombre de cliques maximales présentes dans la décomposition de G. Soit donc $\mathcal{P}(N)$ la propriété

suivante:

$\mathcal{P}(N)$: Soit $(c_1, ..., c_N)$ une suite constructive de G.

Pour tout r vérifiant $1 \leq r \leq N$:

(1) Si c est une clique maximale du graphe G_r engendré par $\{c_1, ..., c_r\}$, alors c est l'une des cliques $c_1, ..., c_r, ..., c_{r+k_r}$.

(2) La décomposition de G_r comporte $r + k_r$ cliques maximales;

ainsi $\mathcal{D}(G_r) = \{c_1, ..., c_r, ..., c_{r+k_r}\}$.

- Base: $\mathcal{P}(1)$ est évidemment vraie mais d'intérêt purement formel. $\mathcal{P}(2)$ correspond au lemme de Gauss: en effet, les cliques c_1 et c_2 sont distinctes et maximales dans G_2; il n'en existe pas d'autres dans la décomposition de G_2.

- Pas de récurrence

  Montrons que pour tout entier naturel N supérieur à 2, on a:

  $$\mathcal{P}(N) \Rightarrow \mathcal{P}(N+1).$$

  Supposons donc $\mathcal{P}(N)$ vraie et prouvons $\mathcal{P}(N+1)$.

  On considère un entier r vérifiant $1 \leq r \leq N+1$. Considérons la clique c_{N+1} On note comme d'ordinaire G_j le graphe engendré par $\{c_1, ..., cj\}$. De deux choses l'une:

  - Premier cas: $G_N \neq G$, sachant que $G = G_{N+1}$.

    Dans ces conditions $(c_1, ..., c_N)$ est une suite constructive pour G_N. Le seul point qui mérite vérification est le fait que $\{c_1, ..., c_N\}$ constitue bien la décomposition de G_N en cliques maximales; or s'il manquait une clique maximale, vu que ce ne peut pas être c_{N+1}, $(c_1, ..., c_{N+1})$ ne serait pas elle-même une suite constructive pour le graphe total G. Alors la propriété $\mathcal{P}(N+1)$ vient directement de $\mathcal{P}(N)$.

– Deuxième cas: $G_N = G$.

Dans ces conditions il existe un entier j vérifiant $1 < j \leq N$ minimal tel que: $G_j = G$. Ainsi $(c_1, ..., c_{j-1})$ est une suite constructive pour G_{j-1}; ici encore $\{c_1, ..., c_{j-1}\}$ est la décomposition en cliques maximales de G_{j-1}, car si une faisait défaut, $\{c_1, ..., c_{N+1}\}$ ne serait pas la décomposition de G, donc $(c_1, ..., c_{N+1})$ pas une suite constructive pour G.

On utilise alors l'hypothèse de récurrence pour $j - 1 < N$, ce qui permet de conclure puisque $N + 1 = j + k_j$.

■

Proposition 116 *Réciproque*

Sous les notations précédentes, une suite $(c_1, ..., c_N)$ telle que les propriétés 1 et 2 du lemme sont vérifiées, c'est-à-dire telle que le lemme soit valide pour G, est constructive.

Preuve. Pour tout j, si le lemme est vérifié pour la suite considérée, toute clique c_t avec $t > j + k_j$ n'est pas une clique de G_j, sinon k_j ne serait pas le maximum annoncé. Donc $(c_1, ..., c_N)$ est constructive pour G. ■

Nous allons définir les suites de cliques de décomposition de G vérifiant l'hypothèse $(\mathcal{C})$, c'est-à-dire constructives pour G. Un algorithme dévolu les fournira effectivement, pour tout graphe G.

2.3.3 Lemmes techniques

Sous les notations antérieures, soit $\mathcal{A}$ la matrice d'adjacence du graphe G; pour tout i élément de $\{1, ..., n\}$ on désignera par $\mathcal{A}_i$ la ligne de numéro i de $\mathcal{A}$.

Lemme 117 *Soit i, j, k des éléments distincts de $\{1, 2, ..., n\}$.*

Soit $\{x_i, x_j\} \subset X$ une clique de G.

Alors $\{x_i, x_j, x_k\}$ *est une clique de* X *ssi* k *est un indice de colonne tel que le vecteur somme* s *des lignes* $\mathcal{A}_i + \mathcal{A}_j$ *vérifie* $s(k) = |\{x_i, x_j\}|$*, c'est-à-dire* 2 *dans ce cas particulier.*

Preuve. Elle est évidente.

Ce résultat permet une traduction matricielle, donc généraliste, de la propriété, sous Matlab en particulier. Il se généralise évidemment aux cliques d'ordres supérieurs, qui mettent en oeuvre alors des sommes de trois, quatre ou plus, lignes de la matrice $\mathcal{A}$.

On peut penser ici à un certain nombre de résultats proches mais plus développés, liés à des préoccupations initiales différentes, comme ceux fournis dans [JTH04] et [JH06] par exemple. ■

Proposition 118 *Soit* $G = (X, E)$ *un graphe et* $\mathcal{D}(G)$ *sa décomposition en cliques maximales.*

Soit $c = \{x_{i_1}, ..., x_{i_r}\}$ *une* r*-clique de* $\mathcal{D}(G)$*.*

On considère le vecteur $s_r(c)$*, somme des* r *lignes de la matrice* $\mathcal{A}$*, de numéros* $i_1, ..., i_r$*, ceux des sommets éléments de la clique* c*.*

Alors $s_r(c)$*, définie par:*

$$s_r(c) = \sum_{j \in \{1,...,r\}} \mathcal{A}_{i_j}$$

vérifie:

$$s_r(c).1 = r(r-1) + contr.ext(G, c)$$

où $r(r-1)$ *représente le double du nombre d'arêtes internes à la clique* c

et $contr.ext(G, c)$ *le nombre d'arêtes, dans le graphe* G*, liant* c *aux sommets de* X *ne faisant pas partie de* c*.*

Preuve. Lorsqu'on fait la somme des r-lignes de la matrice $\mathcal{A}$ considérées:

- on compte deux fois les arêtes internes à la r-clique maximale c (d'où le terme $r(r-1)$), en raison du caractère symétrique de la matrice $\mathcal{A}$;

- mais on décompte une seule fois les arêtes liant les sommets de c aux sommets extérieurs à cette clique.

■

2.3.4 Algorithme de décomposition d'un graphe en cliques maximales

Introduction: pour une méthode généraliste

- Il convient de rappeler d'abord, pour éviter toute illusion, qu'en général le nombre de cliques maximales d'un graphe est exponentiel; de plus le problème consistant à déterminer la taille maximale d'une clique est à lui seul NP-complet.

- Il existe évidemment plusieurs méthodes pour déterminer l'ensemble des cliques maximales figurant dans la décomposition d'un graphe. En général, et pour cause, les auteurs ne se préoccupent pas de déterminer l'ensemble des cliques maximales au sens où nous le pensons; pourtant les méthodes proposées peuvent souvent être détournées vers notre objectif. On pourra voir par exemple [GM85] et [GJ79].

  Les méthodes décrites peuvent dépendre plus ou moins de la configuration du graphe traité, du fait que l'on dispose ou non d'informations précisant ou limitant les types de décomposition présents. Ceci renvoie précisément d'ailleurs aux théorèmes du NFL... Nous ne disposerons pas d'un algorithme optimal pour tous les graphes existants; nous y reviendrons plus tard, au chapitre 4 lors d'une discussion générale critique des propriétés et résultats obtenus.

- Nous privilégierons ici une approche généraliste, qui ne tient pas compte de ces particularités. En effet, quiconque a saisi les liens dialectiques existant entre un graphe donné et sa décomposition en cliques maximales, sera bien persuadé qu'il existe toujours un graphe qui échappe à une typologie prédéfinie, dès qu'elle est précisée... En conséquence, nous ne nous préoccupons pas à ce stade des performances obtenues; nous tenons seulement à disposer de la décomposition du graphe traité en cliques maximales.

  Il serait sans doute intéressant, de particulariser ultérieurement la méthode proposée à telle ou telle sous-classe de graphes. Tel n'est pas notre objectif actuel !

- Nous avons conduit bon nombre d'expérimentations sous Matlab d'une part, d'autre part nous fournissons quelques éléments pour une analyse de complexité qui devrait, sans être exhaustive, faire sentir comment telle classe de graphe est en mesure de "piéger" ou non, la méthode proposée, et donc la mettre en difficulté. Nous évoquerons au chapitre 4 quelques problématiques connexes.

Algorithme de décomposition en cliques maximales

Nous décrivons dans cette section un algorithme $rech.cliques.max\ (\mathcal{G} \rightarrow tailles, cliques)$ qui fournit la décomposition du graphe G, en cliques maximales conformément à la théorie développée plus haut.

a) Données et résultats

- On donne un graphe non orienté et simple, $G = (X, E)$ avec $X = \{x_1, ..., x_n\}$, par sa matrice $\mathcal{G}$ triangulaire supérieure, de diagonale nulle, définie dans la section 2.2.2.

- L'algorithme proposé produit:
  - un vecteur d'entiers, $tailles$, de longueur p contenant les tailles des cliques maximales effectivement présentes dans la décomposition;
  - une suite $cliques$ de p tables $(cliques\,\{i\})_{1\leq i\leq p}$ dont les colonnes ordonnées représentent les cliques maximales de tailles $tailles(i)$.

b) Algorithme de décomposition en cliques maximales

Algorithme 119 **rech.cliques.max** $(\mathcal{G} \rightarrow tailles, cliques)$

- ***Entrée:*** *$\mathcal{G}$ matrice associée au graphe G définie sous la définition 96.*
- ***Sortie:***
  - *$tailles$: vecteur de p entiers, tailles des cliques maximales de G.*
  - *Pour tout $j \in \{1, ..., p\}$, la table non vide $cliques\,\{j\}$ admet pour colonnes les $tailles(j)$-cliques maximales de G.*
- ***Corps d'algorithme***

  Début

  1. *Initialisations*
     - *$j = 1$; S'il existe des 1-cliques dans G:*

       affectation de $cliques\,\{1\}$; $tailles(j) = 1$; $j = j + 1$.
     - *Initialisation de la table à deux lignes atraiter par les arêtes du graphe G qui constituent ses colonnes.*
     - *Initialisation par $i \leftarrow 2$ de l'ordre potentiel i des prochaines cliques maximales.*

2. ***Tant que*** *(atraiter* $\neq \varnothing$*)*

   (a) *Pour toute i-clique* c_i*, colonne de atraiter, on détermine si elle est incluse dans une clique* c_{i+1} *d'ordre* $i+1$*:*

   * *si oui : on place* c_{i+1} *dans le prochain atraiter;*

   * *si non : on extrait* c_i *de atraiter et on la place dans la table cliques* $\{j\}$ *sachant que* $taille(j) = i$*.*

   (b) *Préparation de l'étape suivante*

   * *Si nécessaire: mise à jour du vecteur tailles et* $j = j + 1$*;*

   * *Incrémentation de i via:* $i \leftarrow i + 1$*; traitement spécifique de fin.*

Fin de Tant que

Fin de *rech.cliques.max* ( )

N.B. Le traitement dévolu à chaque clique de *atraiter*, décrit dans le corps de boucle **Tant que** ci-dessus, représentera l'unité temporelle de base pour les évaluations ultérieures. Ce temps d'exécution "unitaire" sera pour la suite noté u; on pourra considérer u comme une valeur moyenne en première analyse; puis dans un second temps, si besoin, on pourra en affiner l'expression en fonction de la taille $tailles\,(j)$ des cliques en cours de traitement.

Remarque 120 *La fonction rech.cliques.max* ( ) *sera écrite de façon soignée, afin de faciliter les recherches en utilisant les cliques déjà existantes. Le seul fait d'ordonner leurs éléments évitera les permutations d'écritures.*

EXEMPLE

On se propose de rechercher la décomposition en cliques maximales du graphe Γ donné en figure 2.3 grâce à l'algorithme *rech.cliques.max* ( ). Cet exemple sera réutilisé par la suite.

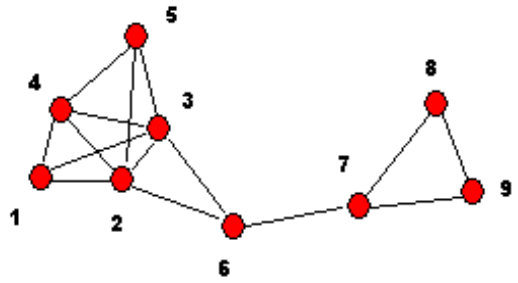

Figure 2.3: Graphe initial Γ

1. Initialisation

   - Pas de 1-cliques; $j = 1; i = 2;$
   - $atraiter = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 6 & 7 & 7 & 8 \\ 2 & 3 & 4 & 3 & 4 & 5 & 6 & 4 & 5 & 6 & 5 & 7 & 8 & 9 & 9 \end{pmatrix}.$

2. Fin de boucle 1

   $atraiter = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 7 \\ 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 8 \\ 3 & 4 & 4 & 4 & 5 & 6 & 5 & 5 & 9 \end{pmatrix}$; $cliques\,\{1\} = \begin{pmatrix} 6 \\ 7 \end{pmatrix}$;

   $tailles(1) = 2; j = 2; i = 3;$

3. Fin de boucle 2

   $atraiter = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \end{pmatrix}$; $cliques\,\{2\} = \begin{pmatrix} 2 & 7 \\ 3 & 8 \\ 6 & 9 \end{pmatrix}$;

$tailles(2) = 3; j = 3; i = 4;$

4. Fin de boucle 3

$$atraiter = \varnothing;\ cliques\,\{3\} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \end{pmatrix};\ tailles(3) = 4.$$

5. Bilan

La décomposition en cliques maximales de Γ est donnée par:

$$\mathcal{D}(\Gamma) = \underbrace{\{C_{4.1}, C_{4.2}\}}_{S_1} \cup \underbrace{\{C_{3.1}, C_{3.2}\}}_{S_2} \cup \underbrace{\{C_{2.1}\}}_{S_3}$$

sachant que les S_i, connus par la définition 105, représentent les ensembles de cliques maximales de mêmes tailles. On a de plus les informations suivantes:

Ordre et nombre de cliques maximales	Cliques maximales
$r_1 = 4; \quad n_1 = 2.$	$\begin{cases} C_{4.1} = \{1,2,3,4\} \\ C_{4.2} = \{2,3,4,5\} \end{cases}$
$r_2 = 3 \quad n_2 = 2.$	$\begin{cases} C_{3.1} = \{2,3,6\} \\ C_{3.2} = \{7,8,9\} \end{cases}$
$r_3 = 1 \quad n_3 = 1.$	$C_{2.1} = \{6,7\}.$

ELÉMENTS D'ÉTUDE DE COMPLEXITÉ DE $rech.cliques.max\,(\mathcal{G} \rightarrow tailles, cliques)$

a) Position du problème - Questions connexes

- On considère un graphe $G = (X, E)$ comportant $n = |X|$ sommets et $|E|$ arêtes, avec évidemment $|E| \leq n(n-1)/2$. On suppose savoir qu'il admet pour décomposition en cliques maximales, celle décrite par la table suivante:

i : ordre des cliques	1	2	...	k	...	n
Nombre N_i des cliques maximales d'ordre i	N_1	N_2	...	N_k	...	N_n

- Cette répartition sera connue en fin d'exécution de $rech.cliques.max(\ )$ par exemple, pour un graphe donné, mais elle permettra aussi de constituer des classes de graphes sur lesquelles comparer les performances d'outils présentés ultérieurement. De manière générale, la notion de décomposition en cliques maximales va permettre une classification efficace des graphes et fondera des comparaisons pertinentes.

- Il convient de noter que toutes les suites $(N_i)_{1\leq i\leq n}$ ne représentent pas nécessairement des décompositions possibles ou licites; un certain nombre de contraintes doivent, par exemple lier en particulier la somme des N_i, pour i élément de $\{1, ..., n\}$. On peut établir des résultats partiels à cet égard, qui certainement en appellent d'autres, relatifs à des conformations classiques de graphes. On peut donner des résultats relatifs aux familles de graphes constitués:

  - d'une seule clique maximale d'ordre maximal;
  - d'une "chaîne" de cliques, uniques de leur ordre, et d'intersection avec leurs voisines, optimale en divers sens à préciser (maximale ou minimale au contraire);
  - d'un "réseau" de cliques d'ordres 3 s'appuyant les unes sur les autres et partageant entre elles, au plus, une clique d'ordre 2 à la façon des triangulations de Delaunay;
  - d'un "réseau" de cliques d'ordre 4 constituant l'équivalent avec un ordre supplémentaire, du phénomène décrit au point antérieur;
  - de diverses autres formes souvent présentes dans les applications.

Proposition 121 *Pour tout graphe G*

1. *On a $N_n \leq 1$.*

   Si $N_n = 1$, cette information a des répercussions sur d'autres paramètres et spécialement les autres cliques présentes; plus précisément:

$$|E| = \frac{n\,(n-1)}{2} \text{ et } \forall j \in \{1, ..., n-1\} \quad N_j = 0.$$

2. *De même, si $N_1 = n$, alors:*

$$\forall j \in \{2, ..., n\} \quad N_j = 0.$$

3. *Cliques maximales d'ordre k*

   - *Toute clique d'ordre k (avec $1 < k < n$) représente parmi les A arêtes supposées exister, $C_k^2 = k(k-1)/2$ arêtes du graphe. Deux cliques distinctes d'ordre k, peuvent partager au plus $k-1$ sommets sinon elles seraient égales mais elles peuvent aussi être disjointes.*
   - *Ainsi deux cliques d'ordre k représentent au moins $k(k-1)/2+(k-1) = (k-1)(k/2+1)$ arêtes distinctes et au plus $(k-1)k$ arêtes, si elles sont disjointes.*

Preuve. Elle est très simple. Les résultats énoncés sont modestes certes, mais ils devraient permettre de sentir la difficulté de caractériser le fait qu'un ensemble de parties de X est véritablement une décomposition de graphe en cliques maximales.

■

Remarque 122 *Il serait certainement intéressant d'étudier de façon systématique ces conditions de compatibilité. Elles conduiraient sans doute au descriptif des formes optimales de "cohabitation" de k-cliques, problématique proche de l'étude*

des triangulations dans le plan ou de celle de la construction de "diamants" dans l'espace. On sent affleurer la question des simplexes d'un espace affine de dimension donnée, sans toutefois nous trouver "ligotés" par une quelconque représentation puisque les problématiques sont de nature algébrique uniquement... L'intrusion de notions affines interprétatives peut présenter un avantage en termes de représentation certes, mais elle risque d'avoir surtout un effet limitant, voire productif d'idées fausses...

On nous a fait remarquer que ces questions étaient liées au complexe simplicial abstrait formé par l'ensemble des cliques du graphe étudié. Nous n'avons pas été en mesure de prendre en compte cette nouvelle vision de la situation, en raison du caractère tardif de cette information.

Nous supposons pour la suite, que la table fournie en début de section 2.3.4 représente une décomposition en cliques maximales licite, donc possible.

b) Etude de $rech.cliques.max\,(\ )$

Soit un graphe $G = (X, E)$, avec $n = |X|$. Soit encore le vecteur d'entiers naturels $(N_i)_{1 \leq i \leq n}$, où N_i désigne le nombre de cliques maximales d'ordre i figurant dans la décomposition de G; on note $N = \sum_{i=1}^{n} N_i$ le nombre total des cliques de cette décomposition.

Nommons k la taille maximale des cliques figurant dans la décomposition de G. Par suite k vérifie:

$$k = \max\left(\{j \in \{1, ..., n\ \} \quad / \quad N_j \neq 0\}\right).$$

Proposition 123 *Si k est supérieur à 2, on note $T(n, (N_i))$ le temps d'exécution de $rech.cliques.max\,(\ )$, exprimé en unités u correspondant au temps de traitement d'une clique, défini dans l'algorithme 119.*

1. *Alors* $T(n,(N_i))$ *est défini comme suit:*

$$\begin{aligned} T(n,(N_i)) &= [a_0 + a_1 n + a_2 |E|] + (k-1)\, a_3 \\ &\quad + u\left[\alpha_1 + \alpha_2 + \alpha_3 + ... + \alpha_{k-2}\right], \end{aligned}$$

où a_0, a_1, a_2, a_3, α_i *sont des constantes réelles indépendantes de* u *et les* α_i *des entiers vérifiant* [2]*:*

$$\forall i \in \{1, ..., k-2\} \quad \alpha_i = N_{i+1} + \sum_{j=i+2}^{k} C_j^{i+2} N_j.$$

2. *Autre expression de* $T(n,(N_i))$

On peut écrire le dernier terme de $T(n,(N_i))$ *sous la forme* $u\left(\sum_{i=1}^{k-2} \alpha_i\right)$ *avec:*

$$\begin{aligned} \left(\sum_{i=1}^{k-2} \alpha_i\right) &= N_2 + N_3\left(1 + C_3^3\right) + N_4\left(1 + C_4^3 + C_4^4\right) \\ &\quad + ... + N_{k-1}\left(1 + C_{k-1}^3 + ... + C_{k-1}^{k-1}\right) + N_k\left(C_k^3 + ... + C_k^k\right). \end{aligned}$$

Preuve. Il convient d'analyser l'algorithme étudié. Le deuxième encadrement est un résultat de technique mathématique dont nous signalons seulement le principe.

- Le premier terme $[a_0 + a_1 n + a_2 |E|]$ correspond à la phase d'initialisation, tandis que $(k-1)\, a_3$ représente le cumul des deuxièmes parties de la boucle **tant que**.

- On peut fournir une expression de la somme $u\left[\alpha_1 + \alpha_2 + \alpha_3 + ... + \alpha_{k-2}\right]$ à partir de ses constituants grâce à l'étude de la taille de la table *atraiter*, en cours d'exécution.

  - Lors de la première entrée en boucle, α_1 2-cliques de *atraiter* seront étudiées; nous en déterminons le détail ici.

[2] *Lors de l'étape* i *de la boucle, l'algorithme extrait les* $(i+1)$*-cliques maximales; lors de la dernière étape* $(k-2)$ *on extrait à la fois les* $(k-1)$ *et* k*-cliques maximales.*

* On aura N_2 2-cliques maximales.

* Chacune des N_3 3-cliques maximales sera présente une fois, c'est-à-dire C_3^3 fois.

* Chacune des N_4 4-cliques maximales sera présente par ses C_4^3 sous cliques d'ordre 3.

* Et ainsi de suite jusqu'aux N_k k-cliques maximales présentes par leurs C_k^3 sous-cliques distinctes d'ordre 3.

Par suite α_1 est donné par:

$$\alpha_1 = N_2 + \sum_{j=3}^{k} C_j^3 N_j.$$

– Lors de la seconde entrée en boucle, α_2 3-cliques de *atraiter* seront étudiées.

* On aura N_3 3-cliques maximales.

* Chacune des N_4 4-cliques maximales sera présente une fois, c'est-à-dire C_4^4 fois.

* Chacune des N_5 5-cliques maximales sera présente par ses C_5^4 sous cliques d'ordre 4.

* Et ainsi de suite jusqu'aux N_k k-cliques maximales présentes par leurs C_k^4 sous cliques distinctes d'ordre 4.

Par suite α_2 est donné par:

$$\alpha_2 = N_3 + \sum_{j=4}^{k} C_j^4 N_j.$$

– Le même raisonnement est appliqué aux ordres supérieurs, jusqu'à obtenir α_{k-2} lors de la $(k-2)^{ième}$ entrée en boucle qui permettra de déterminer à la fois les $(k-1)$ cliques et celles d'ordre k, comme restantes.

- On additionne les différents termes obtenus dans l'étude précédente et on factorise les $(N_i)_{2\leq i\leq k}$.

■

Remarque 124 *Limite des résultats obtenus !*

- *Si pour un graphe G, il existe une seule clique maximale, d'ordre (maximal !) n, l'algorithme aura pourtant été contraint de visiter tous les ordres inférieurs de cliques possibles.*

- *L'expression de $T(n,(N_i))$ a le mérite d'exister; toutefois, elle est décevante en pratique. Son intérêt essentiel est de faire sentir la complexité latente, et de convaincre qu'il est possible de faire choix de graphes qui dégradent singulièrement les performances attendues, en moyenne.*

  - *La première expression privilégie la visualisation de la dynamique de l'algorithme de décomposition, via l'écriture du poids de chacune des itérations constitutives de la boucle* ***Tant que****.*

  - *La seconde expression, au contraire, met en évidence les effets séparés des facteurs $(N_i)_{2\leq i\leq k}$ et fait apparaître qu'une clique de forte taille aura coûté cher pour être identifiée.*

  - *Une étude plus fine des "assemblages de cliques maximales" serait sans doute intéressante; la première difficulté sera de déterminer les vecteurs $(N_i)_{2\leq i\leq k}$ valides pour n et $|E|$ donnés.* [3]

[3] *La question semble assez délicate en termes algébriques... A titre d'exemple en effet, un ensemble de neuf arêtes liant cinq sommets, peut permettre de constituer deux 4-cliques, formant deux pyramides collées par une face triangulaire mais aussi de très nombreuses autres configurations.*

2.4 Contraintes d'un graphe sur une de ses cliques maximales

On adopte les notations des sections 2.2.2 et 2.3.

2.4.1 Définitions

On propose ci-dessous deux outils destinés à exprimer les liens d'une clique maximale d'un graphe donné avec son "voisinage". La préoccupation n'est pas nouvelle bien qu'envisagée ici de façon différente. Les méthodes développées par de nombreux utilisateurs d'heuristiques fondées sur les paysages, reconnaissent une place effective conséquente à la notion de voisinage, exprimée de manières diverses; on pourra voir par exemple [DH98], [HH04], [BH00], [BH99] mais aussi [HW87] et [C92].

- Un premier outil mesure les contraintes de l'ensemble de toutes les cliques maximales de la décomposition $\mathcal{D}(G)$ de G sur la clique maximale choisie c; nous le notons $contr(\mathcal{D}(G), c)$.

- Un second outil généralise la situation antérieure; il traduit en quelque sorte la "pression d'un ensemble $\{c_1, ..., c_r\}$ de cliques maximales" du graphe initial donc d'une partie de G considéré, sur une clique c donnée; nous le notons $contr([\{c_1, ..., c_r\} \cup \{c\}], c)$.

Il serait possible de fournir une seule définition pour traiter de façon simultanée ces deux situations; toutefois, pour des raisons de clarté, nous préférons différencier les deux cas.

Définition 125 *Contraintes de G sur une de ses cliques maximales*

On considère un graphe G et sa décomposition $\mathcal{D}(G)$ en N cliques maximales; soit c l'un des éléments de $\mathcal{D}(G)$.

On appelle contrainte de G (ou de $\mathcal{D}(G)$) sur la clique maximale c l'entier naturel, noté $contr(\mathcal{D}(G), c)$, défini comme le nombre des arêtes du graphe G

"pesant" sur la clique maximale c. En utilisant la proposition 118, il vient:

$$contr(\mathcal{D}(G), c) = \left(\sum_{v \in c} d(v)\right) - \frac{|c|(|c|-1)}{2},$$

où $d(v)$ désigne le degré du sommet v de c, au sens du graphe G.

N.B. En raison de la proposition citée ci-dessus, $contr(\mathcal{D}(G), c)$ apparaît comme la somme:

- *des contraintes internes à la clique c, à savoir $|c|(|c|-1)/2$ comptées une seule fois,*
- *et des contraintes extérieures à c provenant des autres cliques de G, définies comme $contr.ext(G, c)$ dans la proposition 118.*

N.B. $contr(\mathcal{D}(G), c)$ est un entier naturel, supérieur ou égal au nombre des arêtes internes à c.

Définition 126 *Contraintes d'une partie de G sur une de ses cliques maximales c*

On considère un graphe G et sa décomposition $\mathcal{D}(G)$ en N cliques maximales, dont c l'un des éléments de $\mathcal{D}(G)$. Soit $\{c_1, ..., c_r\}$ une partie éventuellement vide de $\mathcal{D}(G)$ ne contenant pas c. On note G' le sous graphe de G engendré par $\{c\} \cup \{c_1, ..., c_r\}$.

On appelle contrainte de G' sur la clique maximale c - en effet c reste maximale dans G' puisqu'elle l'est dans G - l'entier naturel, noté $contr([\{c_1, ..., c_r\} \cup \{c\}], c)$, défini comme le nombre des arêtes du graphe G' "pesant" sur la clique maximale c. Il vient:

$$contr([\{c_1, ..., c_r\} \cup \{c\}], c) = \left(\sum_{v \in c} d'(v)\right) - \frac{|c|(|c|-1)}{2},$$

où $d'(v)$ désigne le degré du sommet v de c, au sens du seul sous-graphe G'.

2.4.2 Propriétés - Exemples

Dès que les cliques constituantes de la décomposition $\mathcal{D}(G)$ d'un graphe G sont connues, on est en mesure de déterminer $contr(\mathcal{D}(G), c)$, pour tout élément c de $\mathcal{D}(G)$. En particulier, on a le résultat suivant.

Proposition 127 *Sous les notations antérieures relatives à $G = (X, E)$:*

- *Si un sommet x_i est isolé, le singleton $\{x_i\}$ est une clique maximale de la décomposition de G. Alors $contr(\mathcal{D}(G), \{x_i\}) = 0$.*
- *Si une clique c est disjointe des autres cliques, $contr(\mathcal{D}(G), c)$ représente les seules contraintes internes à la clique maximale c. Par suite:*
$$contr(\mathcal{D}(G), c) = \frac{|c|(|c|-1)}{2}.$$
- *Les seules cliques maximales de $\mathcal{D}(G)$ agissantes sur $contr(\mathcal{D}(G), c)$ sont celles d'intersection non vides avec c.*

Preuve. La preuve des deux premiers points est évidente, le second contenant le premier. Pour le dernier point, si un sommet extérieur à c agit sur c, il fait partie d'une clique maximale autre que c, d'intersection non vide avec c. ■

Exemple 128 *L'exemple proposé à la section 2.3.4, représenté figure 2.4, permet de remplir le tableau suivant, sachant que les cliques maximales sont étudiées dans l'ordre indiqué par l'écriture choisie pour $\mathcal{D}(\Gamma) = \{C_{4.1}, C_{4.2}, C_{3.1}, C_{3.2}, C_{2.1}\}$.*

Cliques maximales c	*Valeurs de $contr(\mathcal{D}(\Gamma), c)$*
$C_{4.1} = \{1, 2, 3, 4\}$	$contr(\mathcal{D}(\Gamma), C_{4.1}) = 11$
$C_{4.2} = \{2, 3, 4, 5\}$	$contr(\mathcal{D}(\Gamma), C_{4.2}) = 11$
$C_{3.1} = \{2, 3, 6\}$	$contr(\mathcal{D}(\Gamma), C_{3.1}) = 10$
$C_{3.2} = \{7, 8, 9\}$	$contr(\mathcal{D}(\Gamma), C_{3.2}) = 4$
$C_{2.1} = \{6, 7\}$	$contr(\mathcal{D}(\Gamma), C_{2.1}) = 5$

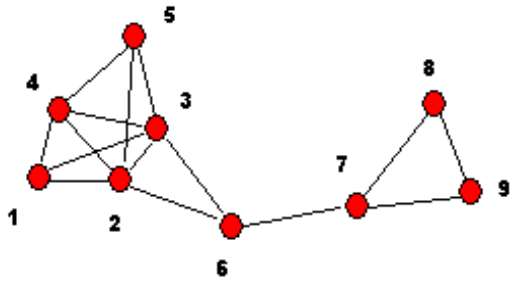

Figure 2.4: Exemple 1 *graphe* Γ

Exemple 129 *On considère le graphe* Γ' *représenté à la figure 2.5. Sa décomposition en cliques maximales est donnée par:* $\mathcal{D}(\Gamma') = \{C_{3.1}, C_{3.2}, C_{3.3}, C_{3.4}, C_{3.5}, C_{3.6}\}$. *Ceci mène à la table suivante:*

Cliques maximales c	*Valeurs de contr* (Γ', c)
$C_{3.1} = \{1, 2, 3\}$	$contr(\mathcal{D}(\Gamma'), C_{3.1}) = 8$
$C_{3.2} = \{1, 3, 4\}$	$contr(\mathcal{D}(\Gamma'), C_{3.2}) = 9$
$C_{3.3} = \{1, 4, 5\}$	$contr(\mathcal{D}(\Gamma'), C_{3.3}) = 8$
$C_{3.4} = \{2, 3, 6\}$	$contr(\mathcal{D}(\Gamma'), C_{3.4}) = 8$
$C_{3.5} = \{3, 4, 6\}$	$contr(\mathcal{D}(\Gamma'), C_{3.5}) = 9$
$C_{3.6} = \{4, 5, 6\}$	$contr(\mathcal{D}(\Gamma'), C_{3.6}) = 8$

Remarque 130 *On notera que les cliques constitutives du graphe* Γ' *paraissent se "valoir"; elles semblent toutes identiquement "liées" et même "ligotées" par les cliques voisines en termes de contact puisque chacune est incluse dans la réunion*

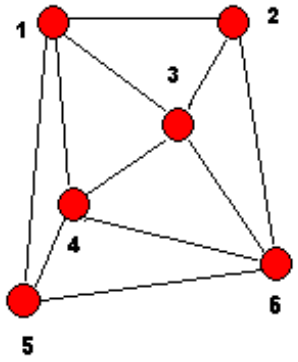

Figure 2.5: Exemple 2 graphe Γ'

des autres. Cette sensation manque de précision et ne résiste pas à un deuxième regard ! En effet, intuitivement, certaines sont nettement plus légères que d'autres plus "enfouies" dans Γ'*. Parmi les moins libres en un sens objectivé par la valeur de contr* (Γ', c)*, on confirme l'intuition que* $\{1, 3, 4\}$ *et* $\{3, 4, 6\}$ *sont davantage "liées"; elles sont l'expression de contraintes plus lourdes dans le graphe total. Par contre, certaines cliques sont "frontières de la zone liée" et par là, plus légères... même si elles sont totalement "tenues", en termes de contact, par le reste du graphe. Il en est ainsi dans notre exemple pour* $\{1, 4, 5\}$*,* $\{4, 5, 6\}$*,* $\{1, 2, 3\}$ *ou* $\{2, 3, 6\}$*.*

Remarque 131 *On pourra effectuer la décomposition en cliques maximales du graphe roue* $W7$ *fourni à la figure 2.6 et mettre en oeuvre sa généralisation pour* Wp*.*

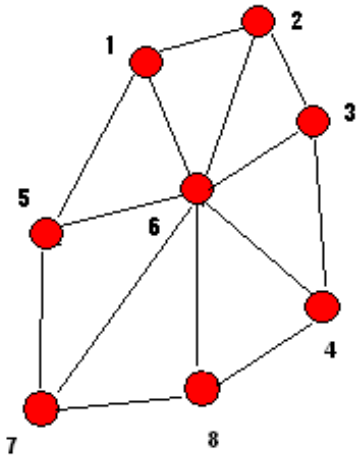

Figure 2.6: Graphe roue W7

2.5 Détermination de suites constructives d'un graphe

Tout graphe admet un ensemble unique de cliques maximales. Mais il existe plusieurs ordres possibles pour décrire cet ensemble. Une première vision assez "naturelle" nous amène à les considérer par ordre de taille, croissant ou non. Nous verrons qu'un autre ordre, plus subtil, présente une réelle fécondité dans les applications relatives aux colorations de graphes; nous devons retrouver l'ordre défini dans la définition 114 en raison de ses qualités propres, tout en l'améliorant pour favoriser le caractère "compact" du graphe en cours de reconstruction.

2.5.1 Suites constructives de G

L'objectif est maintenant de privilégier un ordonnancement sur les cliques maximales qui permette, in fine, de prendre en compte les sommets du graphe menant

des plus fortes contraintes vers la plus grande "légéreté" afin de reconstruire G. En pratique, bon nombre d'heuristiques de coloration usent de cette intuition, et spécialement les méthodes tabous, en traitant en premier lieu les zones de forts conflits. Voir par exemple [DMC06] et [HH04].

Une question moins simple qu'il n'y paraît...

Trouver un bon critère de mesure des liens existants présente un certain nombre d'écueils... d'où une réelle difficulté pour trouver un outil d'évaluation structurant.

- Le concept de degré nous paraît un indicateur de "probabilité" de difficulté de coloration, en dépit de diverses manières de l'affiner, fort astucieuses d'ailleurs, proposées par différents auteurs. Ici comme pour traiter la délinquance, confondre probabilité et réalité se révèle hautement préjudiciable...

  Certes, notre paramètre $contr(\mathcal{D}(G), c)$ est totalement lié au degré mais il sert à déterminer seulement l'une des cliques c les plus "enfouies" dans le graphe G, première prise en compte. Par après, sera décisif un "degré relatif" des cliques suivantes par rapport au graphe en cours de génération; la notion de "degré absolu" s'estompe. Mais surtout, la notion de degré individualise les sommets, sans les structurer par leur appartenance aux cliques dont ils font partie. A titre d'exemple, considérons le graphe représenté en figure 2.7.

  On peut en augmentant le nombre de paire de rayons faire croître le degré des deux sommets constitutifs du moyeu, et pourtant ceci n'est pas un indicateur d'une difficulté réelle de coloration, seulement d'un risque de difficulté. Ceci montre que le degré n'est pas nécessairement un marqueur caractéristique de liens complexes pénalisant la coloration.

- Une autre idée disponible serait de compter le nombre de sommets partagés par les diverses cliques maximales. Pourtant selon leur taille, ce nombre n'a

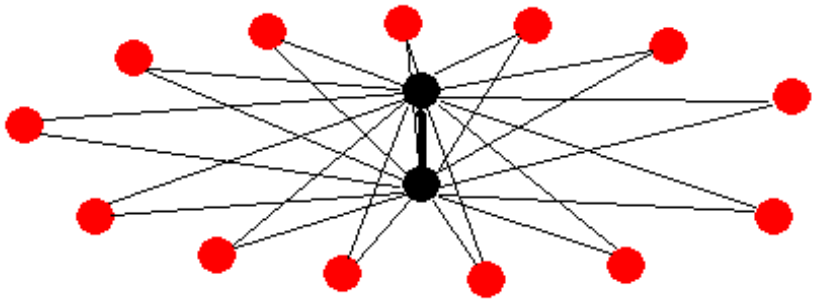

Figure 2.7: Graphe de type Rayons

pas la même signification, n'exprime pas la même puissance de contraintes. Il convient donc d'être plus précis, plus objectif, de faire choix d'un critère plus caractéristique et facile à évaluer.

Extraction des cliques de G: principe

On adopte les notations de la définition 105. Soit $\mathcal{D}(G)$ la décomposition du graphe G supposé contenir N cliques maximales. Nous noterons $(\mathcal{E}_n)_{0\leq n\leq N}$ la suite constructive à "extraire", initialisée par: $\mathcal{E}_0 = \varnothing$.

- Etape 1

  - Etape 1.1

    En raison du fait qu'au début du processus $\mathcal{E}_0 = \varnothing$, nous verrons dans le descriptif des étapes suivantes, que toutes les cliques maximales de la décomposition $\mathcal{D}(G)$ sont retenues pour subir l'étape 1.2. Ainsi nous posons: $\mathcal{L}_1 = \mathcal{D}(G)$.

  - Etape 1.2

    Pour toutes les cliques c de $\mathcal{L}_1$, nous déterminons $contr(\mathcal{D}(G), c)$ et nous en calculons le maximum M_1. Ainsi:

    $$M_1 = \max_{c \in \mathcal{D}(G)} (contr(\mathcal{D}(G), c)).$$

    Par suite il existe au moins une clique maximale e_1 de $\mathcal{D}(G)$ telle que: $M_1 = contr(\mathcal{D}(G), e_1)$. On pose $\mathcal{E}_1 = (e_1)$.

  - Etape 1.3

    Il n'y a pas nécessairement unicité de la clique e_1. Nous effectuons un choix, en cas de multiplicité des cliques solutions selon différents critères d'amélioration, modifiables d'ailleurs; ils seront précisés ultérieurement, à l'étape 2, générique. Le choix effectué orientera puissamment le processus de reconstruction du graphe G. La clique retenue sera l'une des plus "lourdes de contraintes" parmi les cliques de la décomposition de G.

- Etape 2

  - Etape 2.1: optimisation côté cliques déjà extraites

    Parmi les cliques restantes, non encore extraites (ici $\mathcal{D}(G) - \{e_1\}$), nous déterminons d'abord l'ensemble $\mathcal{L}_{2.1}$ des cliques de $\mathcal{D}(G) - \{e_1\}$ qui ont

l'intersection la plus grande en termes de cardinal (c'est-à-dire le nombre maximal de sommets déjà colorés, une fois connue la clique e_1) avec la réunion U_1 des cliques déjà extraites. $\mathcal{L}_{2.1}$ est l'ensemble des cliques de $\mathcal{D}(G) - \{e_1\}$, d'intersection de cardinal maximal avec $U_i = \cup_{j=1}^{i} e_j$, pour $i = 1$.

Parmi les éléments de $\mathcal{L}_{2.1}$, nous déterminons l'ensemble $\mathcal{L}_{2.2}$ de celles qui sont les plus fortement contraintes par la famille, ici $\{e_1\}$, des cliques déjà extraites. Ainsi $\mathcal{L}_{2.2}$ constitue l'ensemble des meilleurs "transmetteurs" de contraintes des cliques déjà extraites. En pratique, à contact égal, cette deuxième opération favorise les cliques de taille maximale.

– Etape 2.2: optimisation côté cliques restant à extraire

On discrimine les (éventuellement nombreux) éléments de $\mathcal{L}_{2.2}$ par une série de critères qui se révèleront facilitateurs des traitements ultérieurs.

∗ Parmi les éléments de $\mathcal{L}_{2.2}$, on choisit les cliques qui ont le meilleur pouvoir absorbant des cliques restantes.

Définition 132 *Etant donné la suite des cliques déjà extraites* $(e_1, ..., e_i)$*, on considère leur réunion notée* $U_i = \cup_{j=1}^{i} e_i$*. On dit qu'une clique restante - ou non encore extraite - c absorbe une clique restante* c'*, autre que c, ssi* $c' \subset U_i \cup c$*. Autrement dit, si la clique c est colorée,* c' *le sera aussi.*

Définition 133 *Sous les notations précédentes, on désigne par* $sat(c)$ *l'ensemble des cliques* c' *absorbées par c;* $sat(c)$ *est appelé l'ensemble des satellites de c.*

On appelle pouvoir absorbant de c le cardinal de $sat(c)$*.*

Remarque 134 *Cette contrainte revient encore à privilégier dans le choix des cliques, celles dont les sommets sont "collectivement"*

les plus contraints. Il serait possible d'exprimer ce critère d'autre manière.

On note alors $\mathcal{L}_{2.3}$ la partie de $\mathcal{L}_{2.2}$, constituée des cliques qui absorbent le plus grand nombre de cliques restantes, quelle que soit leur taille. Pour chaque clique c de $\mathcal{L}_{2.3}$, on conserve l'ensemble de ses cliques satellites $sat(c)$, car elles seront extraites en même temps que c, si c est choisie. La prise en compte de ce critère permet de "compactifier" l'ensemble des cliques extraites.

N.B. On aura compris que lors de l'étape 1, il ne peut exister de satellites pour e_1, puisque les cliques sont maximales.

* On calcule le maximum $m_{2.3}$, parmi les éléments de $\mathcal{L}_{2.3}$, des quantités $contr\left(\left[\{e_1\} \cup \{c\}\right], c\right)$. Ainsi:

$$m_{2.3} = \max_{c \in \mathcal{L}_{2.3}} \left(contr\left(\left[\{e_1\} \cup \{c\}\right], c\right)\right).$$

Les éléments retenus seront les meilleurs transmetteurs des contraintes des cliques déjà extraites, ici $\{e_1\}$ vers $\mathcal{D}(G) - \{e_1\}$, à cette étape; on note leur ensemble $\mathcal{L}_{2.4}$: les cliques les plus "noyées" parmi les restantes sont retenues.

* Parmi les éléments de $\mathcal{L}_{2.4}$, on retient l'ensemble $\mathcal{L}_{2.5}$, de celles qui ont un contact le plus récent possible avec les dernières cliques extraites. Ce critère a surtout une finalité esthétique, afin d'éviter une apparence de chaos.

De façon plus précise sous les notations précédentes, pour tout élément c de $\mathcal{L}_{2.4}$, on détermine le plus grand indice j élément de $\{1, ..., i\}$ tel que $c \cap e_j \neq \varnothing$. $\mathcal{L}_{2.5}$ est alors constitué des cliques c optimales au sens indiqué.

∗ Il existe au moins une clique dans $\mathcal{L}_{2.5}$; on en choisit une, selon un critère quelconque, notée e_2, qui est effectivement extraite avec ses satellites $sat\,(e_2)$. On note alors $\mathcal{E}_2 = (e_1, e_2, sat(e_2))$.

N.B. L'ordre des éléments de $sat(e_2)$ importera peu en termes de coloration car ils seront traités comme un tout. Par contre, ils seront ordonnés de fait dans l'algorithme de définition des suites constructives de façon à ce que chaque clique satellite nouvelle apporte des arêtes nouvelles par rapport aux précédentes retenues. Ainsi $\mathcal{E}_2$ pourra s'écrire:

$$\mathcal{E}_2 = \left(e_1, e_2, ..., e_{2+(|sat(e_2)|)}\right).$$

- On itère la procédure jusqu'à l'étape finale qui extrait en général l'avant-dernière clique donc la dernière simultanément, ou un groupe plus conséquent constitué d'une clique et de ses satellites.

  N.B. En fin de processus, on aura extrait toutes les cliques de la décomposition; certaines seront arrivées une par une, d'autres par paquets "indivisibles" formés d'une clique maximale avec ses satellites. L'algorithme est certainement convergent.

Remarque 135 *Dès qu'une clique maximale est extraite, toutes les autres "s'en souviennent" pour la suite du processus puisque les cliques extraites précédemment interviennent à chaque choix; elles orientent ainsi puissamment les extractions ultérieures.*

A chaque étape de la procédure, on fait choix d'une des cliques maximales restant à extraire les plus en contact avec celles déjà extraites, qui soit aussi la plus "noyée" parmi les cliques maximales restantes. On détermine en fait à chaque étape la meilleure clique "transmetteuse" des contraintes venues des cliques déjà extraites...

SUITE CONSTRUCTIVE DE G

Nous prouvons ci-dessous l'existence d'une suite constructive pour tout graphe; l'extraction des cliques décrite ci-dessus nous permet d'en construire une effectivement.

Proposition 136 *Sous les notations antérieures, la procédure d'extraction des cliques maximales décrite ci-dessus fournit une suite constructive de G.*

N.B. On remarquera sur des exemples simples qu'il n'y a pas unicité en général, pour les cliques constructives d'un graphe donné.

Preuve. L'extraction des cliques maximales s'opère soit une à une, soit par paquets obtenus en adjoignant à une clique extraite, ses satellites. Cette preuve doit permettre de préciser l'ordonnancement des éléments des paquets éventuels, comme annoncé ci-dessus où ce point n'était pas traité.

- Lors de la première étape, on extrait une clique e_1. Toute clique de la décomposition autre que e_1 ne peut pas être une clique maximale du graphe engendré par e_1, puisqu'elle en est distincte.

- Supposons valide le caractère de suite constructive des graphes intermédiaires, pour l'ensemble des cliques extraites jusqu'à l'étape j. A l'étape suivante $j+1$, on extrait une clique e_s avec l'ensemble éventuellement non vide de ses satellites $sat(e_s)$. On note G_s le graphe engendré par les cliques extraites jusqu'à e_s.

  - Si $sat(e_s)$ est vide, aucune des cliques restantes de la décomposition ne peut être clique maximale du graphe G_s, sinon elle serait satellite de e_s ou contredirait l'hypothèse relative à l'étape j.

- Sinon $sat(e_s)$ est non vide. S'il en existe, on extrait après e_s, en les numérotant toutes les cliques de $sat(e_s)$ qui n'apportent pas d'arêtes nouvelles pour G_s. De deux choses l'une, après cette opération: il en reste d'autres ou non dans $sat(e_s)$.

  S'il n'en reste pas d'autres, la totalité du paquet à traiter l'est.

  S'il en reste, on en extrait une $e_{s'}$ parmi les plus contraintes par le graphe engendré par les cliques déjà traitées et on lui adjoint celles qui n'apportent pas d'arêtes nouvelles par rapport à $e_{s'}$. Et on itère le procédé, jusqu'à vider $sat(e_s)$.

  Dans tous les cas, une fois extraites une clique et, s'il en existe, celles qui n'apportent pas d'arêtes nouvelles - et donc engendrent le même graphe -, il est certain que toute clique de la décomposition, restante, n'est pas une clique maximale du graphe déjà engendré.

- Ce qui achève la preuve.

■

Exemple 137 *Suite constructive du graphe de l'exemple 128*

1. *Extraction de la première clique*

   On utilise la table des contraintes étudiées dans l'exemple 128. Par suite de l'égalité des contraintes $contr\left(\mathcal{D}\left(\Gamma\right), C_{4.1}\right)$ *et* $contr\left(\mathcal{D}\left(\Gamma\right), C_{4.2}\right)$*, on peut choisir d'extraire d'abord* $e_1 = C_{4.2}$*. A ce stade,* $\mathcal{E}_1 = \left(C_{4.2}\right)$.

2. *Extraction de la deuxième clique*

   (a) *On détermine la table des cardinalités des intersections des cliques restantes avec* $C_{4.2}$*; on en choisit le maximum.*

   Par suite $\mathcal{L}_2 = \{C_{4.1}\} = \{\{1, 2, 3, 4\}\}$.

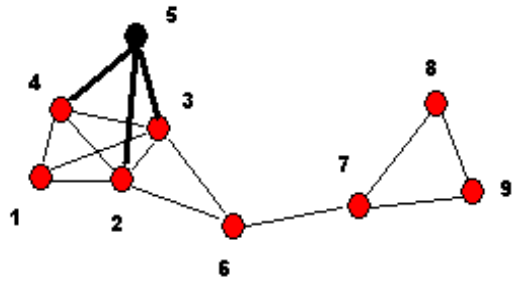

Figure 2.8: Extraction de la clique $C_{4.2} = \{2, 3, 4, 5\}$

(b) *Comme* $\mathcal{L}_2$ *contient une seule clique maximale, toute étude d'amélioration ultérieure est inutile; on extrait* $e_2 = C_{4.1}$. *Ainsi* $\mathcal{E}_2 = (C_{4.2}, C_{4.1})$.

3. *Extraction de la troisième clique*

(a) *On montre facilement que* $\mathcal{L}_3 = \{C_{3.1}\} = \{\{2, 3, 6\}\}$.

(b) *Ici encore,* $\mathcal{L}_3$ *contient une seule clique maximale. On extrait* $e_3 = C_{3.1}$. *Ainsi* $\mathcal{E}_3 = (C_{4.2}, C_{4.1}, C_{3.1})$.

4. *Extraction de la quatrième clique*

(a) *On montre facilement que* $\mathcal{L}_4 = \{C_{2.1}\} = \{\{6, 7\}\}$.

(b) *Ici encore,* $\mathcal{L}_4$ *contient une seule clique maximale. On extrait* $e_4 = C_{2.1}$. *Ainsi* $\mathcal{E}_4 = (C_{4.2}, C_{4.1}, C_{3.1}, C_{2.1})$.

5. *Extraction de la cinquième et dernière clique*

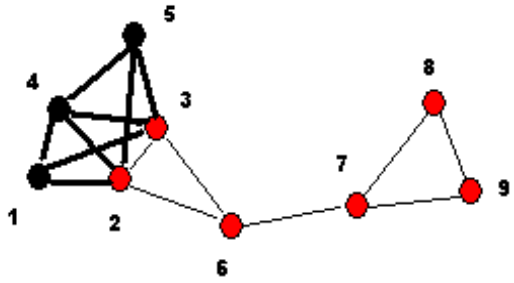

Figure 2.9: Extraction de la clique $C_{4.1} = \{1, 2, 3, 4\}$

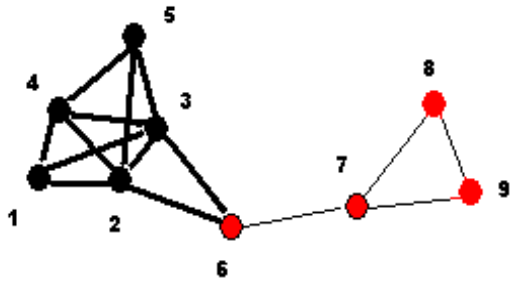

Figure 2.10: Extraction de la clique $C_{3.1} = \{2, 3, 6\}$

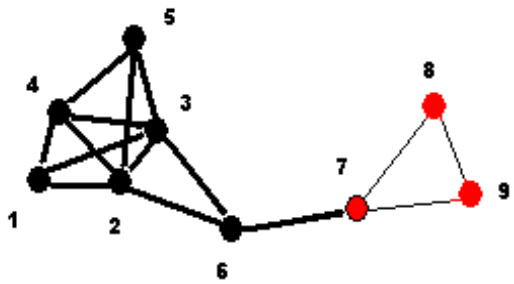

Figure 2.11: Extraction de la clique $C_{2.1} = \{6, 7\}$

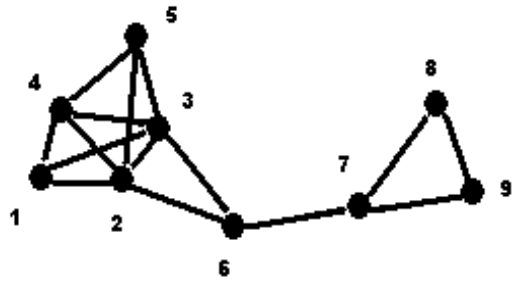

Figure 2.12: Dernière extraction $C_{3.2}$

On ne fait rien ! Il vient $\mathcal{E}_5 = (C_{4.2}, C_{4.1}, C_{3.1}, C_{2.1}, C_{3.2})$.

6. *En guise de bilan, une suite constructive du graphe* Γ *est donnée par* $(e_i)_{1 \leq i \leq 5}$ *sachant que:*

$$\begin{aligned} e_1 &= \{2,3,4,5\}\,; \quad e_2 = \{1,2,3,4\}\,; \quad e_3 = \{2,3,6\}\,; \\ e_4 &= \{6,7\}\,; \quad e_5 = \{6,7,8\}\,. \end{aligned}$$

Remarque 138 *Deux constatations s'imposent.*

- *Il n'y a pas unicité des suites constructives de G, puisqu'à chaque pas, celle-ci n'est pas garantie; ceci avait été annoncé. En ce sens les propriétés d'invariance montrées plus loin n'en seront que plus profondes.*

- *Par son élaboration même, une suite constructive du graphe G,* $(e_i)_{1 \leq i \leq K}$*, par l'apport successif des cliques maximales prises en compte, enrichit le graphe en gestation pour le conduire vers le graphe total, d'une manière très particulière.*

En effet, lorsqu'une clique suit l'ensemble des précédentes, elle a été choisie comme l'une des plus contraintes par les cliques maximales restantes, parmi celles qui sont les plus en contact avec les cliques déjà extraites.

D'où le résultat suivant, pour conclure.

Théorème 139 *Existence d'une suite constructive d'un graphe donné*

On considère un graphe G dont l'ensemble de décomposition en N cliques maximales $\mathcal{D}(G)$ est supposé connu.

Alors il existe une suite constructive $(e_i)_{1\leq i\leq N}$ pour G.

Remarque 140 *On comprend qu'il serait intéressant de disposer d'un résultat réciproque permettant d'affirmer qu'une suite de cliques constitue d'abord l'ensemble de décomposition d'un graphe en cliques maximales (ce n'est pas trivial), et que cette suite est effectivement constructive pour le graphe ainsi donné. L'intérêt est très grand en pratique: vu que cette étape de détermination des cliques maximales, puis d'obtention d'une suite constructive est coûteuse, cela permettrait de définir à moindre frais une telle suite constructive sans être obligé de vérifier qu'elle l'est !*

Remarque 141 *La notion de suite constructive permettra aussi fréquemment de fournir une représentation satisfaisante d'un graphe donné aléatoirement par exemple; on se limitera à des dimensions faibles bien sûr ou bien à des portions d'un graphe donné. Ce concept permet de rassembler "naturellement" les cliques qui ont des raisons objectives de l'être, donc les zones du graphe qui sont fortement liées, en donnant du sens et en visualisant le sens ainsi donné. Nous y reviendrons dans le chapitre 4, au moment d'aborder les prolongements de ce travail.*

2.5.2 Algorithme de détermination d'une suite constructive de G

Nous définissons d'abord quelques modules élémentaires, puis les intégrons pour obtenir une description formelle de l'algorithme annoncé.

On suppose connue la décomposition $\mathcal{D}(G)$ de G en N cliques maximales; $\mathcal{F} = \{c_1, ..., c_r\}$ désigne une partie éventuellement vide de $\mathcal{D}(G)$.

FONCTION $contr.c = contrainte(\mathcal{F}, c)$

- Détermine l'entier naturel $contr.c = contr(([\mathcal{F} \cup \{c\}], c)$ défini ci-dessus à la section 2.4; cet entier représente le nombre de contraintes que fait peser la partie $\mathcal{F}$ de $\mathcal{D}(G)$ sur la clique maximale c, fourni dans la définition 126 dont les notations sont adoptées pour la suite.

- Entrées, Sorties:

  - $\mathcal{F} = \{c_1, ..., c_r\}$ est défini comme indiqué ci-dessus;
  - c est une clique maximale de la décomposition de G, élément de $\mathcal{D}(G) - \mathcal{F}$.
  - $contr.c$ est le nombre de contraintes que fait peser la famille $\mathcal{F} = \{c_1, ..., c_r\}$ sur la clique maximale c.

- Temps d'exécution: u'_1

  N.B. On note u'_1 le temps d'exécution de cette fonction; il est supposé indépendant de c. Ceci constitue une convention conforme à la réalité de l'implémentation. Une étude plus fine imposerait de revoir cette hypothèse simplificatrice.

FONCTION $[max_1.T, Sol_1] = max.contrainte(\mathcal{F}, T)$

- Détermine le maximum des entiers $contr(([\mathcal{F} \cup \{c\}], c)$ pour tous les c éléments de l'ensemble $T \subset \mathcal{D}(G) - \mathcal{F}$, ainsi que l'ensemble des cliques qui ont atteint ce maximum.

- Entrées, Sorties:
  - $\mathcal{F}$ donné par la définition 126;
  - T est une partie de $\mathcal{D}(G) - \mathcal{F}$;
  - $max_1.T$ désigne le maximum des entiers $contr(([\{c\} \cup \mathcal{F}], c)$ pour tous les c éléments de l'ensemble $T \subset \mathcal{D}(G) - \mathcal{F}$.
  - Sol_1 désigne l'ensemble des cliques de T qui ont atteint le maximum $max_1.T$.
- Temps d'exécution: $|T| u'_1$.

FONCTION $[max_2.T, Sol_2] = max.contact(\mathcal{F}, T)$

- Détermine le maximum des cardinaux $|(\cup_{c\in\mathcal{F}}) \cap c|$ pour tous les c éléments de l'ensemble $T \subset \mathcal{D}(G) - \mathcal{F}$, ainsi que l'ensemble des cliques qui ont atteint ce maximum.
- Entrées, Sorties:
  - $\mathcal{F}$ donné par la définition 126;
  - T est une partie de $\mathcal{D}(G) - \mathcal{F}$;
  - $max_2.T$ désigne le maximum des cardinaux des intersections $((\cup_{c\in\mathcal{F}}) \cap c)$, pour tous les c éléments de l'ensemble $T \subset \mathcal{D}(G) - \mathcal{F}$.
  - Sol_2 désigne l'ensemble des cliques de T qui ont atteint le maximum $max_2.T$.
- Temps d'exécution: $|T| u'_2$,

  où u'_2 représente le temps d'exécution unitaire de cette recherche, supposé indépendant de c.

FONCTION $[max_3.T, Sol_3] = max.satellite\,(\mathcal{F}, T)$

- Détermine le maximum du nombre des satellites des cliques c, relativement à $\mathcal{F}$, pour tous les c éléments de l'ensemble $T \subset \mathcal{D}\,(G) - \mathcal{F}$, ainsi que l'ensemble des cliques qui ont atteint ce maximum.

- Entrées, Sorties:

  - $\mathcal{F}$ donné par la définition 126;
  - T est une partie de $\mathcal{D}\,(G) - \mathcal{F}$;
  - $max_3.T$ désigne le maximum du nombre des satellites des cliques c, relativement à $\mathcal{F}$, pour tous les éléments c de l'ensemble $T \subset \mathcal{D}\,(G) - \mathcal{F}$.
  - Sol_3 désigne l'ensemble des cliques sol de T qui ont atteint le maximum $max_3.T$; de plus on conserve l'ensemble $sat(sol)$ des satellites de chaque clique sol.

- Temps d'exécution: $|T|\,u'_3$,

  où u'_3 représente le temps d'exécution unitaire de cette recherche, supposé indépendant de c.

FONCTION $ord.sat = ordonnancement\,(\mathcal{F}, e_i, sat(e_i))$

- Ordonne les éventuels éléments de $sat(e_i)$, satellites de la clique e_i pour que chaque clique maximale de $sat(e_i)$, lorsque $|sat(e_i)| > 1$, "apporte un plus" aux cliques précédemment choisies, en termes de graphes engendrés.

- Entrées, Sorties:

  - $\mathcal{F}$ donné par la définition 126;
  - e_i clique maximale considérée;

- $sat(e_i)$ ensemble des cliques satellites de e_i relativement à $\mathcal{F}$, défini sous la fonction $max.satellite\,(\ )$;
- $ord.sat$ représente l'ensemble $sat(e_i)$ ordonné au sens défini ci-dessus.

- Temps d'exécution: $|T|\,u'_4$,

  où u'_4 représente le temps d'exécution unitaire de l'extraction d'un élément de $sat(e_i)$, supposé indépendant de la clique traitée. Il ne s'agit pas à proprement parler d'un tri des éléments de $sat(e_i)$ mais d'une extraction de chacun d'eux tour à tour, après recherche de maximum.

Algorithme de détermination d'une suite constructive de G

On considère un graphe G et sa décomposition $\mathcal{D}(G)$ en N cliques maximales est supposée connue.

Algorithme 142 **suite.constructive** $\left(\mathcal{D}(G) \rightarrow (e_i)_{1\leq i\leq N}\right)$

- ***Entrée:*** *$\mathcal{D}(G)$ fournie par la définition 105.*
- ***Sortie:*** *$(e_i)_{1\leq i\leq N}$ une suite constructive de G.*
- ***Corps d'algorithme***

  Début

  1. *Initialisations*

     $\mathcal{E} \leftarrow (\ )$; $T \leftarrow \mathcal{D}(G)$; $i \leftarrow 1$.

2. ***Tant que*** $i < N$

   (a) ***Selon*** i

   **** Cas:*** $i = 1$

   * $[max_1.T, Sol_1] = max.contrainte\,(\mathcal{D}\,(G)\,, T)$;

   * *Choisir un* e_i *dans* Sol_1 [4]; $sat(e_i) = (\ )$

   **** Cas:*** $i > 1$

   * $[max_1.T, Sol_1] = max.contact\,(\mathcal{E}, T)$;

   * $[max_2.T, Sol_2] = max.contrainte\,(\mathcal{E}, Sol_1)$;

   * $[max_3.T, Sol_3] = max.satellite\,(\mathcal{E}, Sol_2)$;

   * $[max_4.T, Sol_4] = max.contrainte\,(T, Sol_3)$;

   * *Choix d'une clique* e_i *dans* Sol_4 *et ordonnancement des éléments éventuels de* $sat(e_i)$ *pour que chaque clique satellite choisie dans* $sat(e_i)$ *"apporte" un plus à celles retenues auparavant.*

   $sat(e_i) = ordonnancement\,(\mathcal{E}, sat(e_i))$.

   Fin de Selon

   (b) ***Mises à jour de*** $\mathcal{E}$ ***,*** T ***et*** i

   * $\mathcal{E} \leftarrow (\mathcal{E}\ , e_i, sat(e_i))$;

   * $T \leftarrow T - \{e_i, sat(e_i)\}$;

   * $i \leftarrow i + |sat(e_i)| + 1$.

   Fin de Tant que

Fin de **suite.constructive** $(\ )$

[4] *Il pourra être choisi optimal en un sens à préciser; par exemple, comme le plus enfoui parmi les enfouis.*

ELÉMENTS D'ÉTUDE DE **suite.constructive**( )

Le paramètre devenu central ici, est le nombre N, des cliques maximales, présentes dans la décomposition du graphe G étudié. En effet, le nombre de sommets et la densité des arcs s'estompent même s'ils oeuvrent en arrière plan. Ces deux données se sont exprimées dans l'algorithme de décomposition mais ne semblent plus devoir être déterminantes désormais, en dépit de leur présence sous-jacente...

Proposition 143 *Complexité de* **suite.constructive**( )

Sous les notations antérieures, le temps d'exécution $T(N)$ de l'algorithme **suite.constructive**( ) *mis en oeuvre sur un graphe G dont la décomposition comporte N cliques maximales vérifie:*

$$b_0 + b_1 N \leq T(N) \leq b_2 + b_3 N + b_4 \left(\sum_{i=2}^{N-1} i \right),$$

$$\text{soit encore } \; b_0 + b_1 N \leq T(N) \leq b_2 + b_3 N + b_4 \frac{N(N-1)}{2},$$

où les b_0, b_1, .., b_4 sont des constantes positives réelles. En particulier le terme b_4 est somme des durées élémentaires associées à l'étude des différents critères mis en oeuvre pour l'extraction des cliques maximales u'_1, u'_2,...: leur nombre est celui des tests d'amélioration pratiqués sous le bloc ***Selon*** *pour $i > 2$, dans l'algorithme 142.*

Preuve. Elle relève de l'analyse de l'algorithme proposé. De façon plus précise:

- Pour la minoration, apparaît certainement un terme linéaire en N, en raison du premier bloc de **Selon**, mais aussi du second, lorsque le premier critère a retenu une seule solution optimale.

- Pour la majoration, $b_2 + b_3 N$ résulte de l'initialisation, du premier bloc de **Selon** et du bloc **Mises à jour**. Le terme $b_4 \frac{N(N-1)}{2}$ est issu du second bloc de **Selon**.

■

2.5.3 Degré de liberté d'une clique maximale dans une suite constructive - Clique isthme

On considère un graphe G dont l'ensemble de décomposition en N cliques maximales $\mathcal{D}(G)$ est supposé connu; soit $(e_i)_{1\leq i\leq N}$ une suite constructive de G.

Définition 144 *Degré de liberté de la clique e_j dans la suite constructive $(e_i)_{1\leq i\leq N}$.*

On appelle degré de liberté de e_j dans la suite constructive $(e_i)_{1\leq i\leq N}$. l'entier naturel, noté $deg\left(e_j,(e_i)_{1\leq i\leq N}\right)$ défini par:

$$deg\left(e_j,(e_i)_{1\leq i\leq N}\right)=|e_j|-\left|(e_j)\cap\left(\bigcup_{i=1}^{j-1}e_i\right)\right|.$$

N.B. Si $j=1$, le cardinal de l'intersection de la clique e_j avec la réunion des cliques "antérieures" de la suite constructive, sera évidemment nul puisque la réunion est vide; ainsi $deg\left(e_1,(e_i)_{1\leq i\leq N}\right)=|e_1|$. L'aspect général de l'écriture résiste.

Définition 145 *Graphes liés, ensembles de cliques liés*

- *Sous les notations antérieures, si pour tout j de $\{1,...,N\}$, tous les entiers $deg\left(e_j,(e_i)_{1\leq i\leq N}\right)$ sont strictement positifs, on dit que le graphe G (ou l'ensemble de ses cliques maximales) n'est pas lié relativement à la suite constructive $(e_i)_{1\leq i\leq N}$.*

- *S'il existe un j de $\{1,...,N\}$ tel que $deg\left(e_j,(e_i)_{1\leq i\leq N}\right)$ est nul, le graphe G (ou l'ensemble de ses cliques maximales) est lié relativement à la suite constructive $(e_i)_{1\leq i\leq N}$.*

  En particulier, si un graphe est tel que toute clique de sa décomposition est incluse dans la réunion des autres, alors il est nécessairement lié. On le verra dans les exemples suivants.

Exemple 146 *Les graphes représentés par les figures 2.6 et 2.5 sont liés.*

Le fait qu'un graphe soit lié, est promesse de conflits, de complexité dans la détermination des colorations propres.

Définition 147 *Sous les notations antérieures, on considère une clique* e_j *de la suite constructive* $(e_i)_{1\leq i\leq N}$ *de* G*, avec* $1 < j < N$*.*

N.B. les cas "pathologiques" du type $j = 2$ *ou* $j = N - 2$ *et similaires, seront évidemment dépourvus d'intérêt.*

On dit que la clique e_j *est un isthme pour* G *relativement à la suite constructive* $(e_i)_{1\leq i\leq N}$ *si et seulement si elle vérifie les propriétés suivantes:*

- $\left(\bigcup_{i=1}^{j-1} e_i\right)$ *et* $\left(\bigcup_{i=j+1}^{N} e_i\right)$ *sont disjoints;*
- $\left(\bigcup_{i=1}^{j-1} e_i\right) \cap e_j$ *est non vide et inclus dans l'une des cliques maximales* e_1*,* e_2*, ...,* e_{j-1}*;*
- $\left(\bigcup_{i=j+1}^{N} e_i\right) \cap e_j$ *est non vide.*

Remarque 148 *L'existence d'une clique isthme permettra de scinder l'étude de la coloration de* G *en celle de deux ou plusieurs sous-graphes, en récursivant. La définition proposée généralise évidemment le concept ordinaire d'isthme, qui ne pouvait être autre en l'absence de décomposition en cliques maximales, pour obtenir une condition suffisante de séparabilité de graphes en sous-graphes en vue de leur coloration!*

2.6 Conclusion

Dans ce chapitre, nous avons analysé un graphe G donné en faisant apparaître d'abord ses composants fondamentaux, appelés cliques maximales ou primary cliques, véritables équivalents pour un graphe des nombres premiers pour un entier quelconque.

Les cliques maximales représentent en effet à nos yeux les éléments cruciaux, structurants pour la problématique de coloration comme nous tenterons de le montrer ultérieurement. Pour nous, l'ensemble des cliques maximales contient toute l'information relative à la coloration du graphe, avec sa complexité ou au contraire sa simplicité, exprimée par les liens existant entre les différents éléments de sa décomposition.

La problématique est à notre avis essentiellement algébrique, ensembliste; partiellement au moins, la combinatoire est encapsulée dans les cliques elles-mêmes. Les questions de représentations deviennent secondaires, même si quelques grands classiques demeurent incontournables !

Nous découpons alors le graphe initial en briques élémentaires et grâce à la notion de suite constructive d'un graphe, nous reconstruisons celui-ci en prenant en compte d'abord les contraintes les plus lourdes, pour aller vers les plus légères, comme le font d'ailleurs de nombreuses heuristiques. Cette méthode, ce choix d'ordonnancement, agissent comme un "principe vital" menant au graphe total vu dans sa globalité et non sa singularité.

A partir de la clique ou d'une clique, très profondément "noyée" dans le graphe G étudié, nous complétons ce noyau initial par couches successives de cliques ajoutées jusqu'à obtenir le graphe total.

A chaque étape, le coeur en cours d'élaboration transmet sa "chaleur", plus tard sa coloration, aux autres composants en préférant les plus "proches" de ce coeur mais aussi les meilleurs "transmetteurs de chaleur" vers les éléments restants afin de favoriser les constructions compactes. De cette façon, à chaque étape, aucune clique maximale intermédiaire n'est oubliée de telle sorte que le graphe en construction admet pour décomposition précisément l'ensemble des cliques maximales déjà prises en compte, au stade considéré.

Nous allons maintenant étudier les liens mais surtout les effets de ces outils sur la définition de colorations propres du graphe donné G.

Chapitre 3

Utilisation des suites constructives pour la coloration: algorithmes généraux

3.1 Introduction

Nous disposons depuis le chapitre précédent, de la décomposition d'un graphe non orienté simple G en cliques maximales, ainsi que d'un ordonnancement de celles-ci, appelé suite constructive de G qui permettra de reconstruire le graphe G dans sa globalité à partir de "briques élémentaires", en même temps que nous déterminerons l'ensemble de ses colorations propres.

- Nous rappelons d'abord les définitions de base relatives aux colorations propres du graphe étudié, et établissons une caractérisation de celles-ci, liée aux suites constructives de G.

- Nous développons ensuite l'étude de quelques exemples destinés à faire émerger une méthode générale d'obtention des colorations propres du graphe total, basée sur le principe d'expansion - contraction, mais revisité afin de ne pas altérer le graphe G, pour ne pas en modifier les cliques maximales constituantes.

- Nous présentons alors la version principale de notre algorithme de construction des classes de colorations propres du graphe G.

- Nous montrons comment ces colorations "arrivent" sous forme d'une partition de leur ensemble; comme corollaire, nous retrouvons leur dénombrement en particulier, c'est-à-dire, le polynôme chromatique de G obtenu de manière formelle, sans prise en considération du nombre des couleurs disponibles. Puis nous évoquons deux versions "dégradées" de cet algorithme principal, très proches de la version fondamentale.
- D'abord, nous produisons une partition des colorations propres limitées à un nombre r de couleurs avec r supérieur ou égal au nombre chromatique χ de G. Nous produisons ainsi un sous-polynôme chromatique, lié au choix du nombre r de couleurs disponibles.
- Puis nous exposons une deuxième version "dégradée" de l'algorithme principal destiné à produire une classe solution de colorations propres particulière, sans se préoccuper cette fois de leur ensemble. Le caractère plus récent de cette dernière version a pour effet objectif, qu'il nous donne encore quelques soucis !

Par contre nous renverrons au chapitre suivant l'analyse des propriétés générales issues de ces différents outils, l'exposé de la discussion des défauts constatés, des liens avec l'expérience issue de certaines heuristiques qu'il conviendrait sans doute de développer pour les améliorer, des prolongements multiples envisageables, de la correspondance entre graphes et colorations propres seulement esquissée, mais généralisable à coup sûr...

3.2 Colorations propres d'un graphe

On pourra se reporter à [B83] pour les généralités relatives aux colorations d'un graphe.

3.2.1 Définitions premières

Définition 149 *k-coloration d'un graphe*

Soit G un graphe non orienté, simple, défini par $G = (X, E)$ où $X = \{x_1, ..., x_n\}$ désigne l'ensemble des sommets et E celui des arêtes.

On appelle k-coloration du graphe G toute application f de $\{x_1, ..., x_n\}$ dans $Y = \{y_1, ..., y_k\}$ où k, entier naturel non nul, désigne le cardinal $|Y|$.

$Y = \{y_1, ..., y_k\}$ représente l'ensemble des couleurs disponibles pour f.

N.B. Si k est implicite, on parlera simplement de coloration de G.

Remarque 150 *Pour l'instant, on fixe $\{x_1, ..., x_n\}$ et $\{y_1, ..., y_k\}$. Le fait de considérer l'ensemble des graphes sur un ensemble de sommets donnés et l'ensemble des colorations possibles revient à considérer toutes les instances du problème donc l'ensemble des fonctions d'évaluation au sens du No Free Lunch.(voir chapitre 1).*

Définition 151 *Coloration propre d'un graphe*

Une coloration f est dite propre pour le graphe G si et seulement si deux sommets extrémités d'une arête de G sont de couleurs distinctes. Ainsi:

$$\forall (i,j) \in \{1, ..., n\}^2 \quad [(\{x_i, x_j\} \in E) \Rightarrow (f(x_i) \neq f(x_j))]$$

Inversement, s'il existe une arête $\{x_i, x_j\}$ de E pour laquelle une coloration f vérifie $f(x_i) = f(x_j)$, on dit que f présente une violation de contrainte pour $\{x_i, x_j\}$ et éventuellement p violations de contrainte ($p \geq 1$) pour le graphe G.

En conséquence, une coloration propre de G présente 0 violation de contrainte pour G.

Définition 152 *Nombre chromatique*

Il existe un plus petit entier naturel k, noté $\chi(G)$ - ou plus simplement χ en absence d'ambiguïté - tel qu'il existe des χ-colorations propres pour G.

χ est appelé le nombre chromatique de G.

Définition 153 *Clique de cardinal maximal*

Il existe une clique de cardinal maximal dans G; on note $\omega(G)$ son cardinal.

N.B. 1: On en déduit que $\omega(G) \leq \chi(G)$ grâce au théorème 155 par exemple.

N.B. 2: De nombreux travaux sont consacrés aux graphes parfaits H définis par le fait que tout sous-graphe induit de H admet un nombre chromatique égal à la taille de la plus grande clique de ce sous-graphe induit. On pourra par exemple voir [PZ10].

Définition 154 *Polynôme chromatique d'un graphe*

Le nombre des k-colorations propres du graphe G est une fonction polynôme de la variable k. Cette fonction polynôme est appelée le polynôme chromatique de G.

3.2.2 Caractérisation des colorations propres de G

Théorème 155 *Caractérisation des colorations propres de G*

Soit f une coloration de G. Alors les deux propriétés suivantes sont équivalentes:

1. *f est propre pour G.*
2. *Pour toute clique maximale c élément de la décomposition $\mathcal{D}(G)$ de G, l'application f_c, réduite de f à c, est injective sur la clique c.*

   Ainsi:

$$f_c : \begin{array}{ccc} c & \rightarrow & Y \\ x & \mapsto & f_c(x) = f(x) \end{array} \quad \text{est injective.}$$

Preuve. La preuve est élémentaire; on démontre une double implication.

N.B. Ainsi pour prouver que f est propre pour G, on pourra montrer que pour toute clique c de sa décomposition, le cardinal de l'image $f(c)$ est le même que celui de c, ce qui évitera $|c|(|c|-1)/2$ comparaisons. ■

Remarque 156 *Nous nous limiterons de manière générale pour les questions de dénombrement de colorations propres au cas de graphes sans sommets isolés, ce que nous supposons désormais. Le cas contraire s'en déduit aisément; il fait apparaître un facteur mulptiplicatif assez factice, préjudiciable à la compréhension du phénomène central.*

Proposition 157 *Soit G un graphe dont une suite constructive est désignée par $(c_1, ..., c_N)$ sous les notations de la définition 114 du chapitre 2. On considère un entier j tel que: $1 < j < N$ et $\mathcal{F}$ la partie propre de la décomposition $\mathcal{D}(G)$ définie par: $\mathcal{F} = \{c_1, ..., c_{j-1}\}$. Alors on a les propriétés suivantes:*

- *Si g est une coloration propre du graphe engendré par $\mathcal{F} \cup \{c_j\}$, alors g est une coloration propre du graphe engendré par $\mathcal{F}$.*
- *Si f est une coloration propre du graphe engendré par $\mathcal{F}$,*

  pour que f soit une coloration propre du graphe engendré par $\mathcal{F} \cup \{c_j\}$, il faut et il suffit que f soit injective sur $c_j, ..., c_{j+k_j}$.

  N.B. On rappelle que k_j désigne le plus grand entier naturel (éventuellement nul) tel que les graphes engendrés par $\{c_1, ..., c_j\}$, ..., $\{c_1, ..., c_j, ..., c_{j+k_j}\}$ soient égaux.

Preuve. Elle est évidente via le théorème de caractérisation et la définition de suite constructive d'un graphe. ■

Remarque 158 *On dispose ainsi d'une procédure de construction ascendante des colorations propres d'un graphe G, grâce aux suites constructives de ce graphe.*

3.2.3 Emergence d'une méthode de génération et de dénombrement des colorations propres de G

Cette section est destinée à montrer comment est née notre procédure de construction des colorations propres d'un graphe à partir de celles de sous-graphes. Elle pourra être omise par le lecteur pressé... , mais nous paraît cruciale pour l'enchaînement des idées ! Les résultats établis ne sont pas nouveaux (voir par exemple pour les roues et les trous [B74]) mais ils sont décrits ici de façon à annoncer l'algorithme principal.

Notations

Soit G et sa décomposition en N cliques maximales $\mathcal{D}(G)$; nous considérons une suite constructive $(c_j)_{1\leq j\leq N}$ du graphe G.

L'étude de la génération et du dénombrement des colorations propres de G sera menée en deux temps.

- D'abord nous traiterons le cas, dit élémentaire, de graphes dont une suite constructive vérifie une hypothèse simplificatrice $(\mathcal{H})$ précisée ci-dessous.
- Puis nous étudierons quelques exemples du cas contraire; la démarche mise en oeuvre, les problèmes rencontrés conduiront à l'algorithme général.

Cas élémentaire

Considérons une suite constructive $(c_j)_{1\leq j\leq N}$ du graphe G, vérifiant l'hypothèse $(\mathcal{H})$ simplificatrice suivante:

$(\mathcal{H})$: Pour tout j de $\{1,..,N-1\}$, il existe r de $\{1,..,j\}$ tel que:
$(c_{j+1}) \cap \left(\bigcup_{i=1}^{j} c_i\right) \subset c_r$.

Autrement dit, l'hypothèse $(\mathcal{H})$ signifie que la suite constructive $(c_j)_{1\leq j\leq N}$ du graphe G est telle que toute clique de la suite, si elle est en contact avec les précédentes, l'est à l'intérieur d'une seule clique antérieure et non avec plusieurs cliques disjointes. Nous conviendrons que si l'intersection de c_{j+1} avec $\left(\bigcup_{i=1}^{j} c_i\right)$ est vide, $(\mathcal{H})$ est vérifiée.

Proposition 159 *Si la suite constructive $(c_j)_{1\leq j\leq N}$ du graphe G vérifie l'hypothèse $(\mathcal{H})$ alors pour tout i de $\{1,..,N-1\}$, le nombre $deg\left(c_{i+1},(c_j)_{1\leq j\leq i}\right)$ de degrés de liberté de la clique c_{i+1} par rapport à celles qui la précèdent, donné dans la définition 144 par:*

$$deg\left(c_{i+1},(c_j)_{1\leq j\leq i}\right)=|c_{i+1}|-\left|(c_{i+1})\cap\left(\bigcup_{j=1}^{i}c_j\right)\right|$$

est non nul.

Preuve. Si non il existe un i de $\{1,..,N-1\}$ tel que $deg\left(c_{i+1},(c_j)_{1\leq j\leq i}\right)$ est nul. Par suite $c_{i+1}\subset\left(\bigcup_{j=1}^{i}c_j\right)$. Par conséquent $(c_{i+1})\cap\left(\bigcup_{j=1}^{i}c_j\right)=c_{i+1}$ et par suite $c_{i+1}\subset c_r$. Or ceci est impossible en raison du caractère maximal des cliques de la décomposition, toutes distinctes. ■

a) Résultat dans le cas élémentaire: variante 1

Théorème 160 *Construction des k-colorations propres de G*

Soit une suite constructive $(c_j)_{1\leq j\leq N}$ de G vérifiant l'hypothèse $(\mathcal{H})$.

Alors les k-colorations propres du graphe G, sont obtenues comme décrit ci-dessous et leur nombre est donné par le terme $P_N(k)$ d'une suite $(P_j(k))_{1\leq j\leq N}$ de fonctions polynômes en k définie par la relation de récurrence suivante:

- *Donnée de $P_1(k)$*

$$P_1(k)=\frac{k!}{(k-|c_1|)!}\quad \text{où } |c_1| \text{ est le cardinal de la clique } c_1.$$

En effet, $P_1(k)$ est le nombre d'injections de c_1 dans $Y=\{y_1,...,y_k\}$.

- *Pour tout j élément de $\{1, ..., N-1\}$ on a:*

$$P_{j+1}(k) = P_j(k) \frac{\left[k - \left(|c_{j+1}| - deg\left(c_{j+1}, (c_i)_{1 \leq i \leq j}\right)\right)\right]!}{[k - |c_{j+1}|]!}.$$

Remarque 161 *Interprétation des grandeurs intervenantes*

- *L'entier naturel $\left(|c_{j+1}| - deg\left(c_{j+1}, (c_i)_{1 \leq i \leq j}\right)\right)$ représente le nombre des contraintes pesant sur les sommets de la clique c_{j+1}, venues des cliques précédentes dans la suite constructive du graphe G, sous l'hypothèse $(\mathcal{H})$.*

- *$\left[k - \left(|c_{j+1}| - deg\left(c_{j+1}, (c_i)_{1 \leq i \leq j}\right)\right)\right]!$ / $[k - |c_{j+1}|]!$ représente le nombre des injections de l'ensemble des sommets "libres" de c_{j+1} dans l'ensemble des couleurs autorisées restantes.*

- *Finalement, $P_{j+1}(k)$ représente le nombre des colorations propres du sous-graphe G_{j+1} de G de suite constructive $(c_1, c_2, ..., c_{j+1})$, d'après la définition et les propriétés des suites constructives.*

- *$P_N(k)$ désigne le nombre de k-colorations du graphe G présentant 0 violation de contraintes, donc le polynôme chromatique du graphe.*

Preuve. Elle est conduite en plusieurs étapes.

- L'initialisation de la suite par la donnée de $P_1(k)$ revient à dénombrer, pour une clique de cardinal $|c_1|$, les colorations propres lorsqu'on dispose de k couleurs. Ceci revient à compter les injections de $\{1, 2, ..., |c_1|\}$ dans $\{y_1, ..., y_k\}$, d'où le résultat bien connu.

- Soit j un élément de $\{1, ..., N-1\}$. Supposons connaître pour le graphe G_j de suite constructive $(c_1, ..., c_j)$ le nombre de ses colorations propres, noté $P_j(k)$. Pour obtenir une coloration propre de G_{j+1} de suite constructive $(c_1, ..., c_j, c_{j+1})$

on complète chacune des colorations propres de G_j en colorant successivement les sommets libres de la dernière clique c_{j+1} "ajoutée" aux précédentes pour passer de G_j à G_{j+1}, en raison de l'interprétation en termes d'injections sur les cliques de la décomposition de G_{j+1}.

Or on dispose, pour colorer c_{j+1}, d'un nombre de variantes distinctes égal au nombre d'injections de l'ensemble des sommets "libres" de c_{j+1} dans l'ensemble des couleurs disponibles pour ces sommets. Les couleurs disponibles pour ces sommets libres de c_{j+1} sont au nombre de k, moins, s'il en existe, le nombre de celles fixées par les cliques précédemment prises en compte, en raison de l'hypothèse $(\mathcal{H})$ qui assure que les sommets éventuels de $c_{j+1} \cap \left(\cup_{i=1}^{j} c_i\right)$ sont assurément de couleurs distinctes puisqu'éléments d'une clique c_r avec r élément de $\{1, ..., j\}$. D'où le résultat.

■

Corollaire 162 *On en déduit aisément le nombre $P_{imp}(k)$ des k-colorations de G impropres, c'est-à-dire ne vérifiant pas l'ensemble des contraintes:*

$$P_{imp}(k) = |\{y_1, ..., y_k\}|^n - P_N(k) = k^n - P_N(k).$$

Preuve. Une k-coloration est une application de $X = \{x_1, ..., x_n\}$ dans $Y = \{y_1, ..., y_k\}$; d'où le résultat. ■

Remarque 163 *Il existe donc des classes de colorations propres déduites les unes des autres par permutations, et finalement "identiques". Ne pas en tenir compte dès leur recherche conduirait à éprouver "the futility of a blind research". Voir [C99].*

Corollaire 164 *Si le polynôme chromatique d'un graphe est connu, on peut déterminer facilement le nombre chromatique χ de ce graphe en évaluant $P_N(k)$, pour l'instant dans les seuls cas relevant de l'hypothèse $(\mathcal{H})$. Ce qui hélas ne résout pas le problème dans sa généralité...*

b) Résultat dans le cas élémentaire: variante 2

Nous pouvons fournir une autre expression du résultat proposé dans le théorème 160. L'élément fondateur demeure la suite constructive $(c_j)_{1\leq j\leq N}$ retenue pour le graphe G. Cette nouvelle expression annonce la démarche générale adoptée dans l'algorithme principal décrit plus loin.

- Considérons la clique c_1; elle est constituée des sommets que nous noterons $x_{1.1}, x_{1.2}, ..., x_{1.|c_1|}$. Pour tout i compris entre 1 et $|c_1|$, à chacun des sommets $x_{1.i}$, nous faisons correspondre une fonction polynôme en k, appelée $fact\,(x_{1.i})$, donnée par $fact\,(x_{1.i}) = (k-(i-1))$ de telle sorte que l'on ait:

$$P_1\,(k) = \prod_{i=1}^{|c_1|} fact\,(x_{1.i}) = ((k+1)-1)\,((k+1)-2)\,...\,((k+1)-|c_1|)\,.$$

  N.B. Il convient de remarquer que le choix effectué pour $fact\,(x_{1.i})$ aurait pu subir une permutation des $|c_1|$ facteurs possibles, sans modifier le produit $P_1\,(k)$, invariant dans cette permutation des facteurs constituants.

- Puis nous considérons la clique c_2, et plus spécialement ses sommets libres par rapport à c_1, c'est-à-dire autres que ceux de c_1 que nous noterons $x_{2.1}, ..., x_{2.deg(c_2,(c_1))}$. A chacun d'eux, pour tout i compris entre 1 et $deg\,(c_2,(c_1))$, nous associons le facteur polynômial $fact\,(x_{2.i})$ défini par:

$$fact\,(x_{2.i}) = \left[k+1-\left(|c_j| - deg\left(c_j,(c_i)_{1\leq i\leq j-1}\right)\right)-i\right], \text{ pour } j=2.$$

  de telle sorte que:

$$P_2\,(k) = P_1\,(k) \times \prod_{i=1}^{deg(c_2,(c_1))} fact\,(x_{2.i})\,.$$

  N.B. Ici encore, on aurait pu permuter dans l'écriture de $P_2\,(k)$ les facteurs du deuxième bloc, sans pour autant le modifier.

- On itère le procédé jusqu'à l'obtention de $P_N(k)$ comme suit:

$$P_N(k) = P_1(k) \times \left[\prod_{i=1}^{deg(c_2,(c_1))} fact(x_{2.i}) \right] \times ... \times \left[\prod_{i=1}^{deg(c_N,(c_1,c_2,...,c_{N-1}))} fact(x_{N,i}) \right].$$

Les différents facteurs polynômiaux attribués aux sommets du graphe, peuvent "supporter" des permutations dans les conditions précisées plus haut; mais ces permutations laisseront invariants chacun des produits constitutifs de $P_N(k)$. On peut noter que l'écriture de $P_1(k)$ peut être considérée comme analogue aux autres, en convenant que:

$$deg\left(c_1, (c_i)_{1 \leq i \leq 0}\right) = |c_1|.$$

Théorème 165 *Sous les notations précédentes, pour toute clique c_j (avec $1 \leq j \leq N$) de la suite constructive $(c_j)_{1 \leq j \leq N}$ du graphe G, on associe à une permutation près, un facteur polynômial noté $fact(x_{j.i})$ à chacun des sommets $x_{j.i}$ de c_j libres par rapport à la suite $(c_1, ..., c_{j-1})$ (vide si $j = 1$), défini par:*

$$fact(x_{j.i}) = \left[k + 1 - \left(|c_j| - deg\left(c_j, (c_i)_{1 \leq i \leq j-1}\right)\right) - i \right]$$
$$et\ i \in \{1, ..., deg(c_j, (c_1, c_2, ..., c_{j-1}))\}.$$

Alors le nombre $P_N(k)$ des k-colorations propres du graphe G est donné par:

$$P_N(k) = \prod_{j=1}^{N} \left(\prod_{i=1}^{deg(c_j,(c_1,c_2,...,c_{j-1}))} \left[k + 1 - \left(|c_j| - deg\left(c_j, (c_i)_{1 \leq i \leq j-1}\right)\right) - i \right] \right).$$

N.B. L'affectation à chaque sommet d'un facteur polynômial est choisie, une fois pour toutes, mais supporte une permutation dans les conditions précisées plus haut, qui rendrait invariant globalement chacun des produits intérieurs écrits ci-dessus.

Remarque 166 *La fonction polynôme $P_N(k)$ semblait a priori dépendre de la suite constructive $(cj)_{1 \leq j \leq N}$; elle en est invariante, en tant que polynôme chromatique.*

c) Exemple

Nous reprenons l'étude du graphe Γ proposé dans l'exemple 128 du chapitre 2 et allons déterminer son polynôme chromatique sous la deuxième variante.

Clique concernée	$deg\,(C_{4.2}, \varnothing)$	Sommet(s)	Facteur(s) associé(s)
$C_{4.2} = \{2,3,4,5\}$	4	2	$(k - (\lvert C_{4.2}\rvert - 4))$
		3	$k-1$
		4	$k-2$
		5	$k-3$

Clique concernée	$deg\,(C_{4.1}, (C_{4.2}))$	Sommet(s)	Facteur(s) associé(s)
$C_{4.1} = \{1,2,3,4\}$	1	1	$(k - (\lvert C_{4.1}\rvert - 1))$

Clique concernée	$deg\,(C_{3.1}, (C_{4.2}, C_{4.1}))$	Sommet(s)	Facteur(s) associé(s)
$C_{3.1} = \{2,3,6\}$	1	6	$(k - (\lvert C_{3.1}\rvert - 1))$

Clique concernée	$deg\,(C_{2.1}, (C_{4.2}, C_{4.1}, C_{3.1}))$	Sommet(s)	Facteur(s) associé(s)
$C_{2.1} = \{6,7\}$	1	7	$(k - (\lvert C_{2.1}\rvert - 1))$

Clique concernée	$deg\,(C_{3.2}, (C_{4.2}, C_{4.1}, C_{3.1}, C_{2.1}))$	Sommet(s)	Facteur(s) associé(s)
$C_{3.2} = \{7,8,9\}$	2	8	$(k - (\lvert C_{3.2}\rvert - 2))$
		9	$(k-2)$

On en déduit la table bilan des facteurs polynômiaux des différents sommets:

Sommet	2	3	4	5	1	6	7	8	9
Facteur associé	k	$k-1$	$k-2$	$k-3$	$k-3$	$k-2$	$k-1$	$k-1$	$k-2$

d'où il vient, par produit, le polynôme chromatique du graphe Γ:

$$P_5(k) = [k(k-1)(k-2)(k-3)]\,[(k-3)]\,[(k-2)]\,[(k-1)]\,[(k-1)\,(k-2)]\,,$$

soit encore:

$$P_5(k) = k\,(k-1)^3\,(k-2)^3\,(k-3)^2\,.$$

N.B. Il n'existe évidemment pas de 3-coloration propre puisque $P_5(3) = 0$; par contre on peut dénombrer $P_5(4) = 864$ 4-colorations propres pour le graphe Γ, ce qui est confirmé expérimentalement.

Suites constructives de G ne vérifiant pas l'hypothèse $(\mathcal{H})$: exemples

Nous rappelons que la notion de graphe lié, d'ensemble de cliques maximales lié a été donnée dans la définition 145.

a) *R*oues Wp

Définition 167 *Soit un entier p vérifiant $p \geq 4$. On considère p 3-cliques contigües partageant le même sommet central. Un tel graphe G est une p-roue Wp.*

N.B. Si $p = 3$, on se trouve en présence de quatre 3-cliques non maximales puisqu'incluses dans une 4-clique. Les cas $p = 2$ et $p = 1$ ne présentent aucun intérêt non plus.

Exemple 168 *Nous fournissons ici un exemple de roue $W5$, à la figure 3.1.*

Proposition 169 *Tout graphe G de type roue Wp est constitué de 3-cliques maximales liées; une suite constructive de ce graphe est donnée par toute suite de 3-cliques constitutives, en contact par une arête avec la précédente.*

Preuve. Elle est évidente.

Dans le cas de l'exemple 168, une suite constructive sera donnée par $(c_j)_{1\leq j\leq 5}$ avec:

$$c_1 = \{1,2,6\}\,;\, c_2 = \{1,5,6\}\,;\, c_3 = \{4,5,6\}\,;\, c_4 = \{3,4,6\}\,;\, c_5 = \{2,3,6\}\,.$$

■

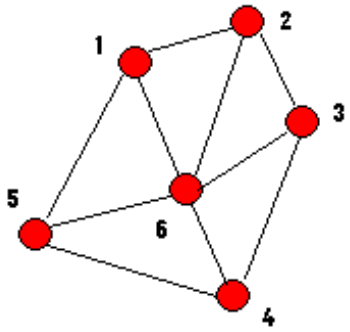

Figure 3.1: Graphe roue $W5$

Proposition 170 *Soit $k \geq 3$ et $p \geq 4$.*

Le nombre $F_1(k,p)$ de k-colorations propres de la p-roue G est donné par l'expression suivante:

$$F_1(k,p) = N_1\ c(k,p),$$

sachant que:

- $N_1 = k(k-1)(k-2)$,
- *et que le coefficient $c(k,p)$ est donné pour tout entier k supérieur ou égal à 3 par la propriété $\mathcal{Q}(p)$ suivante:*

$$\mathcal{Q}(p): \quad \begin{cases} \textit{Si } p = 4\textit{:} & c(k,4) = \left[(k-2) + (k-3)^2\right]; \\ \textit{si } p = 5\textit{:} & c(k,5) = \left[2(k-2)(k-3) + (k-3)^3\right].; \\ \textit{si } p > 4\textit{:} & c(k,p+1) = (k-3)\,c(k,p) + (k-2)\,c(k,p-1). \end{cases}$$

Preuve. Une suite constructive du graphe G est constituée de c_1 l'une quelconque des 3-cliques, suivie de c_2 l'une des 3-cliques contigües à la précédente et

ainsi de suite jusqu'à c_p. Nous voyons confirmée ici, s'imposer dans le raisonnement la nécessité de la méthode de contraction-expansion. On pourra voir [B83].

1. Montrons que $\mathcal{Q}(4)$ est vraie.

   Nous noterons x_1,...,x_4 les différents sommets non centraux.

   - Pour la première clique c_1, puisqu'elle n'est pas contrainte, on dispose de $N_1 = k(k-1)(k-2)$ colorations propres;
   - Pour la seconde c_2, deux de ses sommets sont contraints par c_1(le centre de roue et le sommet x_2), donc nous devons choisir la couleur du troisième sommet x_3 de c_2 en nous préparant à tenir compte des conflits terminaux dûs au fait que la 4-roue est un ensemble de cliques liées. De deux choses l'une:
     - Si la couleur de x_3 est la même que la couleur de x_1, nous disposons de N_1 colorations propres pour le graphe engendré par (c_1, c_2). Par suite, pour vérifier les contraintes indispensables nous aurons $(k-2)$ choix pour la couleur de x_4. Finalement, il existe $(k-2)N_1$ colorations propres de la 4-roue si les sommets x_1 et x_3 sont de même couleur.
     - Si la couleur de x_3 est différente de la couleur de x_1, la couleur de x_3 sera choisie parmi $(k-3)$ possibles, de même que celle de x_4. Par suite on dispose de $N_1(k-3)^2$ colorations de ce deuxième type.
     - Finalement le nombre total de colorations propres de la 4-roue est donné par:

       $$F_1(k,4) = N_1(k-2) + N_1(k-3)^2 = N_1\left[(k-2) + (k-3)^2\right]$$

       ce qui établit le résultat annoncé.

2. Montrons que $\mathcal{Q}(5)$ est vraie.

Nous noterons x_1,...,x_5 les différents sommets non centraux.

- Pour la première clique c_1, puisqu'elle n'est pas contrainte, on dispose de $N_1 = k\,(k-1)\,(k-2)$ colorations propres;
- Pour la seconde c_2, deux de ses sommets sont contraints. On dispose de $(k-2)$ possibles pour colorer x_3.
- Pour colorer c_3, donc x_4, nous disposons a priori de $(k-2)$ possibles mais nous devons considérer deux cas pour préparer le conflit final relatif à x_5.
  - Si la couleur de x_4 est la même que la couleur de x_1, nous disposerons de $(k-2)$ possibles pour colorer le dernier sommet x_5; par contre le choix effectué sur x_4 rétroagit sur le choix des couleurs possibles sur x_3. En effet, la couleur de x_3 devra être différente de celles de x_2, x_6 et de x_1 qui sont toutes trois distinctes et pré-choisies, en raison de la première clique. Ainsi le nombre de choix pour x_3 se réduit à $(k-3)$ en lieu et place de $(k-2)$; le choix de x_4 est imposé par celui de x_1 déjà effectué. Le nombre T_1 de colorations possibles pour ce premier type de 5-roue est donc donné par:

$$T_1 = \underbrace{N_1}_{\text{choix pour } c_1} \left[\underbrace{(k-3)}_{\text{choix pour } x_3} * \underbrace{1}_{\text{choix pour } x_4} * \underbrace{(k-2)}_{\text{choix pour } x_5} \right].$$

  - Si la couleur de x_4 est différente de la couleur de x_1, celle de x_5 sera choisie parmi $(k-3)$ possibles. Nous aurons pour amener cette situation autant de possibles qu'il n'existe de 4-roue à k couleurs, ce qui correspond au fait que les deux dernières cliques sont "effacées" et remplacées par une seule. Par suite on dispose de T_2 colorations de ce deuxième type, avec T_2 donné par:

$$T_2 = N_1 \left[(k-2) + (k-3)^2 \right] (k-3)\,.$$

– Finalement le nombre total de colorations propres de la 5-roue est donné par:

$$F_1(k,5) = N_1\left[2(k-2)(k-3) + (k-3)^3\right].$$

ce qui établit le résultat annoncé.

N.B. Nous remarquons ici qu'il n'existe pas de 3-colorations pour une 5-roue, en dépit du fait que toutes les cliques maximales sont d'ordre 3. On retrouve le fait bien connu que la taille de la clique maximale la plus importante ne fournit pas nécessairement le nombre minimal de couleurs nécessaires.

3. Montrons que pour tout $p > 4$, on a:

$$c(k,p+1) = (k-3)\,c(k,p) + (k-2)\,c(k,p-1).$$

Considérons une $(p+1)$-roue; elle dispose de $p+2$ sommets. Nous notons le centre de la roue x_{p+2}, et les sommets de la "circonférence" x_1, ..., x_{p+1} de telle sorte qu'une suite constructive de la $(p+1)$-roue choisie est notée $(c_1, c_2, ..., c_{p+1})$ avec:

$$\begin{aligned} c_1 &= \{x_1, x_2, x_{p+2}\}; \quad c_2 = \{x_2, x_3, x_{p+2}\}; ...; \\ c_p &= \{x_p, x_{p+1}, x_{p+2}\}; \quad c_{p+1} = \{x_1, x_{p+1}, x_{p+2}\}. \end{aligned}$$

Parmi les colorations propres de la $(p+1)$-roue, distinguons les colorations du premier type au nombre de T_1 et celles du deuxième type au nombre de T_2.

- Une coloration du premier type est telle que la couleur de x_p est la même que celle de x_1. Par suite la couleur de x_{p+1} sera choisie parmi $(k-2)$ possibles, c'est-à-dire k moins les couleurs de x_1 et x_{p+2}. Le nombre de colorations du premier type est donc $(k-2)$ fois le nombre de colorations

propres d'une $(p-1)$-roue obtenue en gommant les cliques c_p et c_{p+1}, en confondant x_1 et x_p, faisant disparaître x_{p+1}, phénomène de contraction interprété en termes de cliques maximales. Vu l'hypothèse de récurrence:

$$T_1 = (k-2)\, F_1(k, p-1).$$

- Une coloration du deuxième type correspond au cas contraire; elle donne à x_p une couleur différente de celle de x_1. Par suite la couleur de x_{p+1} sera choisie parmi $(k-3)$ possibles, c'est-à-dire k moins les couleurs de x_1, x_p et x_{p+2}. Le nombre de colorations du second type est donc $(k-3)$ fois le nombre de colorations propres d'une p-roue obtenue en gommant la clique c_{p+1}, en faisant disparaître x_{p+1}(expansion). Vu l'hypothèse de récurrence:

$$T_2 = (k-3)\, F_1(k, p).$$

- Par conséquent, il existe $T_1 + T_2$ colorations propres de la $(p+1)$-roue et on a:

$$F_1\,(k, p+1) = (k-2)\, F_1(k, p-1) + (k-3)\, F_1(k, p).$$

Ceci achève la preuve.

■

Corollaire 171 *On peut fournir une écriture explicite de $F_1\,(k, p)$ en fonction de k et p. Le résultat, qui présente un réel intérêt formel, sera traité en exercice; il peut être confirmé informatiquement par dénombrement des colorations propres pour des valeurs de p et k "raisonnables".*

Corollaire 172 *Si la p-roue traitée ci-dessus est un constituant d'un graphe plus vaste, dont elle représente un sous-ensemble de cliques maximales liées, il est possible que la clique c_1 soit contrainte par les cliques précédentes dans la suite constructive*

du graphe total. On pourra selon le nombre de contraintes pesant sur c_1, la première clique de la p-roue, fournir le nombre de colorations propres de celle-ci.en adaptant le résultat de la proposition antérieure, par action sur la seule valeur de $F(k,4)$.

1. *Si la clique c_1 subit une seule contrainte de la part des cliques qui la précèdent dans la suite constructive d'un graphe, sur x_1 par exemple, ce sommet sera déjà coloré par son appartenance à une clique déjà traitée. Donc la clique c_1 perdra un degré de liberté. Ainsi le nombre de ses colorations propres sera donné par:*

$$F_2(k,4) = (k-1)(k-2)\,c(k,p) = N_2 c(k,p),$$

puisque le premier sommet n'est plus libre.

2. *Si la clique c_1 subit deux contraintes de la part des cliques antérieures de la suite constructive, sur x_1 et x_2 par exemple, ces sommets seront déjà colorés par leur appartenance à une ou des cliques déjà traitées; ainsi la clique c_1 perdra deux degrés de liberté relatifs. Par suite le nombre de ses colorations propres sera donné par:*

$$F_3(k,p) = (k-2)\,c(k,p) = N_3\ c(k,p),$$

puisque les deux premiers sommets ne sont plus libres.

N.B. Ces résultats peuvent en partie se généraliser.

Remarque 173 *On aura noté que les résultats antérieurs fournissent des conditions nécessaires sur le nombre de couleurs garant de l'existence de k-colorations propres d'un graphe G du type p-roue.*

- *On retrouve évidemment le fait que k doit dominer le maximum des tailles des cliques maximales présentes dans la décomposition de G.*

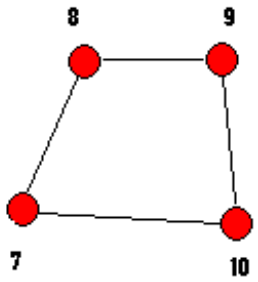

Figure 3.2: Trou G_1

- *On voit l'intérêt de considérer au sens de la coloration, tel ou tel sous-graphe d'un graphe total, ce qui confirme l'aspect modulaire prometteur des résultats obtenus.*

b) Trou

Nous étudions ci-dessous des exemples de situations qui se généralisent avec un peu de soin et produisent des résultats nettement plus vastes. Nous ne fournirons pas de formules généralistes qui donneraient le nombre des k-colorations propres d'un graphe quelconque. L'objectif poursuivi ici est de montrer la faisabilité de la détermination des k-colorations propres pour un graphe de petite taille, grâce à une démarche algorithmique dont les différents modules apparaissent nettement.

b1) Donnée du graphe à traiter

On considère le graphe G_1 fourni par la représentation de la figure 3.2. C'est un cycle sans corde, constitué de 2-cliques maximales qui forment un graphe lié.

b2) Suite constructive de G_1

Une suite constructive du graphe G_1 est donnée par (c_1, c_2, c_3, c_4) où les cliques maximales c_i sont définies par la table suivante:

i	1	2	3	4
c_i	$\{7,8\}$	$\{7,10\}$	$\{9,10\}$	$\{8,9\}$

b3) Résultats relatifs à l'obtention des k-colorations propres du graphe G_1

Proposition 174 *Le nombre de k-colorations propres du graphe G_1, est donné par:*

$$D_1(k,4) = k(k-1)\left[(k-1) + (k-2)^2\right].$$

Preuve. On effectue le suivi des différentes cliques de la suite constructive afin de déterminer une méthode.

- La première clique c_1 est sans contraintes; en conséquence on dispose de $k(k-1)$ colorations propres du sous-graphe associé.

- Pour les colorations de la clique c_2 nous devons préparer l'étude des contraintes dues au fait que l'ensemble de cliques considéré est lié. Les k-colorations propres du graphe G_1 sont encore de deux types.

  - Premier type: Il s'agit des colorations pour lesquelles la couleur du sommet 10 est la même que celle du sommet 8. Ainsi il existe seulement $k(k-1)$ colorations propres du sous-graphe engendré par $\{c_1, c_2\}$. Ainsi le nombre total T_1 de colorations propres de G_1, du premier type, est donné par:

    $$T_1 = k(k-1)\left[(k-1)\right],$$

    puisqu'on dispose de $(k-1)$ choix pour la couleur du sommet 9.

– Deuxième type: Il s'agit des colorations pour lesquelles la couleur du sommet 10 est différente de celle du sommet 8. Ainsi il existe $k\,(k-1)$ colorations du sous-graphe associé à (c_1), multiplié par les $(k-2)$ couleurs disponibles pour le sommet 10; finalement ceci donne donc $k\,(k-1)\,[(k-2)]$ colorations propres pour le sous-graphe engendré par $\{c_1, c_2\}$. Ainsi le nombre total T_2 de colorations de G_1, engendré par $\{c_1, c_2, c_3, c_4\}$ et du second type, est donné par:

$$T_2 = k\,(k-1)\left[(k-2)^2\right],$$

puisque nous avons seulement $(k-2)$ couleurs disponibles pour le sommet 9.

– Ainsi le nombre de k-colorations propres du graphe G_1 est donné par:

$$D_1(k,4) = k\,(k-1)\left[(k-1)+(k-2)^2\right].$$

■

Remarque 175 *Multiples !*

- *On peut facilement généraliser ce résultat à un trou de longueur p ($p \geq 4$) décomposé en 2-cliques formant un ensemble lié. On obtient une relation assez semblable, dans sa nature, à celui obtenu dans le cas des p-roues. On prend ici $p \geq 4$, pour éviter le cas d'une 3-clique qui rendrait non maximales les 2-cliques considérées.*

- *Ici encore si le sommet initial est soumis à une contrainte provenant de cliques antérieures dans une suite constructive d'un graphe plus vaste, seul le facteur initial sera affecté par disparition de son facteur k, puisqu'un degré de liberté est perdu. Il viendra alors $D_2(k,4)$ colorations propres avec:*

$$D_2(k,4) = (k-1)\left[(k-1)+(k-2)^2\right].$$

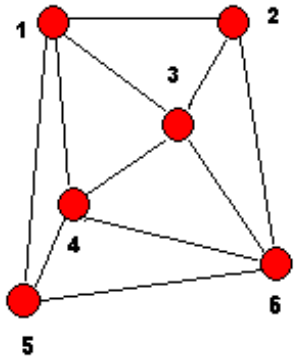

Figure 3.3: 3-cliques liées G_2

c) Pavage de 3-cliques liées

c1) Donnée du graphe à traiter

On considère le graphe G_2 donné par la représentation de la figure 3.3.

Sa décomposition en 3-cliques montre un graphe lié.

c2) Suite constructive de G_2

Une suite constructive du graphe G_2 est donnée par $\{c_1, c_2, c_3, c_4, c_5, c_6\}$ où les cliques maximales c_i sont définies par la table suivante:

i	1	2	3	4	5	6
c_i	$\{1,3,4\}$	$\{3,4,6\}$	$\{2,3,6\}$	$\{1,2,3\}$	$\{4,5,6\}$	$\{1,4,5\}$

.

c3) Nombre de k-colorations propres du graphe G_2

Nous pouvons remarquer ici que l'ensemble de toutes les cliques est lié ; il contient lui-même un sous-ensemble lié, comme il apparaît lors de l'algorithme d'extraction

des cliques maximales: en effet (c_1, c_2, c_3, c_4) est lié, son graphe associé est une 4-roue. Cette présence imposera des contraintes de compatibilité comme tout sous-ensemble lié de cliques maximales.

Proposition 176 *Le nombre $P_1(k,6)$ de k-colorations propres du graphe G_2, ensemble lié de six 3-cliques, est donné par:*

$$P_1(k,6) = k(k-1)(k-2)\left[(k-2)^2 + (k-3)^3\right].$$

Preuve. On adopte ici encore le même principe de preuve.

- Etape 1: étude de (c_1, c_2, c_3, c_4)

  Pour éviter des redondances, nous reconnaissons le graphe engendré par (c_1, c_2, c_3, c_4) comme une 4-roue et adoptons la méthode utilisée dans son étude sous la proposition 170. Par suite il existe $F_1(k,6)$ k-colorations propres de cette 4-roue, avec:

  $$F_1(k,6) = k(k-1)(k-2)\left[(k-2) + (k-3)^2\right].$$

  En généralisant la notion de facteur polynômial mise en évidence à la section 3.2.3, nous pouvons construire les tables suivantes pour visualiser la méthode utilisée:

Sommets	1 3 4	6
Facteurs	k $(k-1)$ $(k-2)$ $\times$	1 ... $(k-3)$
Cliques traitées	$\{1,3,4\}$	$\{3,4,6\}$

Sommets	2	5
Facteurs	$(k-2)$ $\times$ $(k-3)$	$(k-2)$ $\times$ $(k-3)$
Cliques traitées	$\{2,3,6\}$ liées $\{1,2,3\}$	$\{1,4,5\}$ liées $\{4,5,6\}$

Dans la construction conduite dans le cas non lié, le facteur du sommet 6 serait simplement $(k-2)$. La détection des cliques liées $\{2,3,6\}$ et $\{1,2,3\}$ rétro-agit sur le facteur du sommet 6, en nécessitant l'étude de deux cas:

- sommet 1 de la même couleur que le sommet 6, d'où le facteur 1 pour exprimer un seul choix possible;
- sommet 1 de couleur différente de celle du sommet 6, d'où le facteur $(k-3)$.
- On en déduit les facteurs liés au sommet 2, obtenus dans le respect de la distinction des deux cas précédents.
- Les lignes horizontales donnent lieu à des produits qui s'ajoutent, en cas d'existence de lignes parallèles, pour fournir la valeur de $F_1(k,6)$ lors de cette étape 1, puis celle de $P_1(k,6)$ lors de l'étape 2.

  La présence de lignes parallèles dans une table correspond au partitionnement indispensable des colorations propres de sous-graphes. Nous y reviendrons de façon systématique ultérieurement.

- Etape 2: adjonction des cliques c_5 puis c_6

  Comme ci-dessus, la prise en compte de c_5 seule apporterait un facteur $(k-2)$ pour le sommet 5. Mais la présence des cliques liées $\{1,4,5\}$ et $\{4,5,6\}$ rétro-agit sur ce facteur en le modifiant car il nécessite la même distinction sur les

couleurs respectives des sommets 1 et 6, d'où la construction de la colonne du sommet 5, avec respect des mêmes règles de calculs en cas de présence de lignes parallèles. Ainsi $P_1(k,6)$ est donné par:

$$P_1(k,6) = k(k-1)(k-2)\left[(k-2)^2 + (k-3)^3\right].$$

■

Corollaire 177 *Comme plus haut, on peut en déduire facilement le nombre de k-colorations propres de G_2 au cas où la première clique c_1 est soumise à une ou deux contraintes, par des cliques maximales associées à un graphe plus vaste.*

- *Si c_1 est soumise à une contrainte, on disposera de:*

$$P_2(k,6) = (k-1)(k-2)\left[(k-2)^2 + (k-3)^3\right]$$

k-colorations propres de G_2.

- *Si c_1 est soumise à deux contraintes, on disposera de:*

$$P_3(k,6) = (k-2)\left[(k-2)^2 + (k-3)^3\right]$$

k-colorations propres de G_2.

d) Exemple d'assemblage de sous-graphes antérieurs

d1) Donnée du graphe à traiter

On considère le graphe G donné par la représentation de la figure 3.4.

Il est constitué d'un ensemble de cliques maximales qui forment un graphe lié.

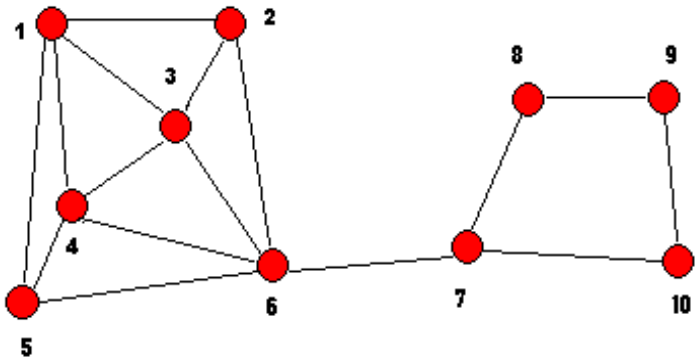

Figure 3.4: G assemblage de graphes liés

d2) Suite constructive de G

On montre aisément qu'une suite constructive de cliques maximales pour G est donnée par:

$$\left(\underbrace{\{3,4,6\},\{1,3,4\},\{2,3,6\},\{1,2,3\},\{4,5,6\},\{1,4,5\}}_{\text{sous-graphe } G_2} \ldots \right.$$
$$\left. \{6,7\}; \underbrace{\{7,8\},\{7,10\},\{9,10\},\{8,9\}}_{\text{sous-graphe } G_1} \right)$$

d3) Résultats relatifs au nombre de k-colorations propres du graphe total G

Proposition 178 *Sous les notations antérieures, le nombre des k-colorations propres du graphe G est donné par:*

$$nb\_col\_val\,(k) = P_1\,(k,6) * (k-1) * D_2(k,4).$$

avec

- $P_1(k,6) = k(k-1)(k-2)\left[(k-2)^2 + (k-3)^3\right]$;
- *et* $D_2(k,4) = (k-1)\left[(k-1) + (k-2)^2\right]$.

Preuve. Il suffit d'utiliser la suite constructive de G et d'intégrer les divers résultats produits en amont. ■

d4) Vérifications numériques

Les résultats théoriques annoncés sont confirmés par l'expérimentation. Nous générons pour ce faire, toutes les k-colorations propres de G avec k élément de $\{4,5,6\}$. On obtient les résultats suivants, où $nb\_col(k)$ désigne le nombre total de colorations, propres ou pas, du graphe G. Nous rappelons que $nb\_col(k)$ désigne le cardinal de $\{y_1, y_2, ..., y_k\}^{\{1,2,...,10\}}$.

k	4	5	6
$nb\_col\_val(k)$	7560	212160	2709000
$nb\_col(k)$	1048576	9765625	60466176

.

Remarque 179 *Il serait fort intéressant d'analyser la décomposition en facteurs premiers des nombres de k-colorations propres du graphe G. Leur étude permet de retrouver les interventions des différents composants k, $(k-1)$ ou $(k-2)$ attendus à partir des résultats théoriques énoncés et d'interpréter les termes complémentaires.*

Bilan relatif à la définition des k-colorations propres des exemples traités

Etant donné un graphe G, nous déterminons d'abord sa décomposition en cliques maximales, puis une suite constructive $(c_1, ..., c_N)$ de G.

Pour construire l'ensemble des colorations propres de G, nous déterminons successivement les injections sur les cliques $\{c_1, ..., c_j\}$ pour j variant de 1 à N. Deux situations structureront la pensée:

- Ou bien, pour tout j de $\{1, ..., N-1\}$, la clique c_{j+1} a une intersection avec la réunion des cliques précédentes incluse dans une clique c_k pour un k de $\{1, ..., j\}$. Dans ce cas, dit élémentaire, la définition d'une coloration propre pour le graphe engendré par $\{c_1, ..., c_{j+1}\}$ s'obtient très aisément à partir d'une coloration propre pour le graphe engendré par $\{c_1, ..., c_j\}$. La coloration de G s'en déduit facilement.

- Ou bien, pour un j de $\{1, ..., N-1\}$, la clique c_{j+1} a une intersection avec la réunion des cliques $\{c_1, ..., c_j\}$ incluse dans aucune des c_1, ..., c_j. Dans ces conditions, le caractère distinct des couleurs des sommets de cette intersection n'est plus assuré. Il convient alors de déterminer une partition des solutions possibles en étudiant:

  - si certains sommets de l'intersection considérée peuvent être de couleurs identiques;
  - mais aussi le cas contraire.

  Pour ces deux cas, il sera nécessaire de transmettre les contraintes d'égalité ou de différence de couleurs aux sommets déjà colorés à ce stade.

Le dénombrement des colorations propres s'en déduira. Mais ce décompte sera l'expression de la "projection" du partitionnement de l'ensemble des colorations propres; des classes fort différentes pourront fournir le même dénombrement. En fait, le décompte est secondaire à l'égard de la détermination et de la structuration de l'ensemble des colorations propres.

3.3 Algorithme principal de détermination des colorations propres de G

Nous présentons dans cette section l'enchaînement des fonctions qui conduit à la détermination d'un partitionnement de l'ensemble des colorations propres du graphe étudié. Nous en identifierons les diverses étapes de façon exhaustive, en tentant d'éviter d'être trop technique; par contre nous essaierons d'être suffisamment précis, en termes d'idées, afin de présenter nettement les difficultés rencontrées et les passages délicats.

3.3.1 Points de départ: décomposition en cliques maximales et suite constructive de G

Nous renvoyons pour les détails des notations et le descriptif complet des fonctions évoquées, au chapitre 2.

Etant donné le graphe G, connu par sa matrice booléenne triangulaire haute $\mathcal{G}$, nous appelons ici les fonctions:

- **rech.cliques.max** $(\mathcal{G} \rightarrow tailles, cliques)$

  qui détermine la décomposition $\mathcal{D}(G)$ en cliques maximales du graphe G;

- suivie de **suite.constructive** $\left(\mathcal{D}(G) \rightarrow (c_j)_{1 \leq j \leq N}\right)$ qui fabrique une suite constructive de G.

Nous devons pour une bonne compréhension ultérieure faire le rappel crucial suivant, relatif à la seconde fonction.

Lors d'un état intermédiaire, la suite constructive en cours d'élaboration contient pour un j donné, les éléments $(c_1, ..., c_j)$. Lors de l'étape suivante, dans les cas simples, arrive seule une clique maximale c_{j+1}. Dans les cas complexes ordinaires, la clique c_{j+1} arrive accompagnée de cliques satellites, "englobées" par c_{j+1}; une

clique c est satellite de c_{j+1}, à ce stade si la clique c est incluse dans la l'ensemble $c_{j+1} \cup \left(\cup_{i=1}^{j} c_i\right)$, autrement dit si la clique c n'apporte pas de sommets nouveaux par rapport à ceux apportés par c_{j+1}; par contre en général bien sûr, c apportera des arêtes nouvelles, non encore amenées par c_{j+1}.

3.3.2 Construction de la charpente du graphe G

Décrivons la fonction **creation.charpente**$((c_j)_{1\leq j\leq N} \rightarrow (som(i))_{1\leq i\leq n}, charpente)$.

Son champ d'entrée est la suite constructive $(c_j)_{1\leq j\leq N}$ issue du module antérieur.

Résultats produits

- Le vecteur *som* est la suite ordonnée des n sommets du graphe, obtenue comme suit. Sous les notations antérieures, si la clique c_{j+1} "arrive seule", ses sommets autres que ceux déjà présents dans $\left(\cup_{i=1}^{j} c_i\right)$ viendront compléter la suite *som* en cours d'élaboration selon un critère quelconque, par exemple l'ordre des entiers naturels sur le numéro des sommets. Si la clique c_{j+1} "arrive accompagnée de cliques satellites", ses sommets autres que ceux déjà présents dans $\left(\cup_{i=1}^{j} c_i\right)$ viendront compléter la suite *som* en cours d'élaboration en commençant par les plus fortement contraints par les sommets de $\left(\cup_{i=1}^{j} c_i\right)$, au sens des arêtes présentes dans c_{j+1} et de l'ensemble de ses satellites.

- *charpente* est un vecteur de cellules au sens de Matlab précisé par la définition suivante.

  Définition 180 *Définition de charpente*

  Pour chaque élément $som(i)$ de som, charpente $\{i\}$ ordonné comme som, contient les interdits du sommet som, c'est-à-dire la partie des sommets antérieurs $som(1)$, $som(2)$, ..., $som(i-1)$, dont il doit différer par la couleur pour fournir

une coloration propre du graphe engendré par c_1, ..., c_j, c_{j+1} et ses satellites éventuels. De surcroît, ces interdits du sommet $som(i)$ sont disposés sur deux étages:

- *Au premier étage on place ceux de ses interdits qui font partie d'une même clique parmi c_1, ..., c_j. On choisira celle-ci pour que la taille de ce premier étage soit maximale; on bénéficie en effet à ce stade de l'assurance que ces éléments sont déjà de couleurs distinctes, puisqu'éléments d'une même clique maximale. Ce niveau peut être vide, si le sommet $som(i)$ est sans contrainte.*
- *Au deuxième étage, on placera les autres sommets dont il doit différer, si toutefois il en existe; dans le cas contraire cet étage sera inexistant puisque vide.*
- *En résumé charpente $\{i\}$ contient les interdits du sommet $som(i)$ et aura en général la forme suivante:*

$$charpente\,\{i\} = \begin{pmatrix} [sommets\ interdits\ éléments\ de\ l'une\ des\ cliques\ c_1, ..., c_j] \\ [autres\ sommets\ interdits,\ éléments\ de\ plusieurs\ autres\ cliques] \end{pmatrix}.$$

Compléments

- Le vecteur *charpente* contient la totalité de l'information de G relative à la coloration, y compris la promesse de la complexité de sa détermination... Nous y reviendrons plus en détail dans le chapitre 4.
- Les résultats issus de la fonction *creation.charpente*( ) ne sont pas canoniques, puisqu'ils dépendent du choix de la suite constructive. Pourtant, ils permettront d'exprimer certains invariants du graphe.

3.3.3 Détermination des classes de colorations propres pour les sommets $(som(i))_{1\leq i\leq n}$

Décrivons la fonction $classes.de.coloration\left((som(i))_{1\leq i\leq n}, charpente \rightarrow classes\right)$; comme d'ordinaire, nous présenterons auparavant les seuls sous-modules indispensables à sa compréhension.

Contexte, objectifs et principe de fonctionnement

La fonction étudiée détermine à partir des classes de colorations propres pour les sommets $som(1)$, $som(2)$, ..., $som(i-1)$, notées $classes$, l'ensemble des classes de coloration propres pour les sommets $som(1)$, $som(2)$, ..., $som(i)$ en leur ajoutant un niveau i. Il sera nécessaire bien sûr d'initialiser $classes$. Par ailleurs, on notera qu'à toute étape, une classe de colorations propres est représentée par une colonne de l'objet $classes$.

On doit saisir absolument que lorsque nous affirmons que $som(1)$ est coloré, nous ne lui affectons aucune couleur particulière ! Pour respecter la structure du graphe en construction depuis la clique c_1 jusqu'à la clique c_N, nous traduisons seulement de manière formelle, par exemple, que le sommet $som(2)$ admet pour interdit $[som(1)]$, tout comme lors de la définition d'une injection, on affirme que le deuxième sommet aura pour image l'une quelconque des possibles sauf celle de $som(1)$, déjà utilisée. A aucun moment nous ne nous préoccupons du nombre de couleurs disponibles, en particulier.

Nous avons vu dans la section précédente, que l'objet $charpente$ contenait toute l'information initiale relative à la caractérisation d'une coloration propre pour le graphe en construction. Ainsi un sommet $som(i)$ devra être interdit de prendre la même couleur qu'un certain nombre de sommets antérieurs. Ces interdits sont, en général, rangés sur deux étages; au premier étage, un ensemble de sommets dont on

sait assurément qu'ils sont de couleurs distinctes. Au deuxième étage, éventuellement vide mais hélas trop rarement..., un ensemble de sommets dont on ne peut pas affirmer qu'ils sont au sens du haut de la colonne traitée, de couleur distincte de ceux du premier étage. La fonction *classes.de.coloration* (.) va gérer la transformation de tous les interdits encore présents sous forme de cellules à deux étages, en interdits représentés par des vecteurs seulement, constitués de sommets de couleurs distinctes, donc d'interdits limités à un étage unique.

Pour ce faire, la fonction va scruter chaque colonne de *classes* et détecter la première cellule - première au sens de l'ordre de coloration -, à deux niveaux dans la colonne traitée, s'il en existe. Pour cette cellule à deux niveaux, la fonction traite successivement chaque élément s_2 de l'étage 2, et le transfère dans le premier étage en étudiant tous les possibles.

- D'abord, elle va détecter s'il existe des sommets s_1 de l'étage 1, dont s_2 peut avoir la couleur sans que la coloration cesse d'être propre. La fonction traite chacun de ces possibles en gérant les conséquences de ces égalités sur la colonne en cours d'étude; en particulier, si s_2 a même couleur que s_1, les interdits du second coloré auront un effet rétroactif sur le premier coloré des deux. Ce transfert d'interdits pourra faire naître des cellules à deux étages pour des éléments amont qui en étaient libérés.

- Ensuite, elle va aussi traiter le cas où la couleur de s_2 sera distincte de celles de tous les éléments existants de l'étage 1. La fonction gère, dans ce cas encore, les nouveaux interdits nés de cette nécessaire différence au prix éventuel de nouvelles cellules à deux niveaux créées au-dessus de la cellule initiale traitée dans la colonne en cours d'étude.

- Ainsi, l'élément s_2 disparaîtra de l'étage 2 dans la colonne étudiée; cette colonne sera remplacée par zéro, une ou plusieurs colonnes dans lesquelles

de nouveaux interdits pourront être "remontés" vers les sommets supérieurs comme autant de contraintes transférées pour assurer la cohérence d'ensemble de chaque sous-classe née lors du traitement.

Ainsi, chaque colonne sera débarrassée de ses contraintes exprimées dans les deuxièmes étages et remplacées par des colonnes contenant des contraintes transférées sur les sommets antérieurement colorés, et qui toutes finiront évacuées par le haut.

Il suffira de reprendre le processus sur les colonnes restantes tant qu'elles n'ont pas la forme requise. Il s'agit d'une procédure de contraction-expansion (ou de contraction-reliement au sens de [B83]) dont le détail est ici suivi et finalisé, rendant le processus constructif. Il convient de noter que le traitement de l'ensemble des colonnes pour un niveau i donné est totalement parallélisable, sans aucun véritable partage de ressources.

Initialisation de *classes*

Nous déterminons pour commencer l'unique classe de colorations propres, relative à la seule clique maximale c_1, dont les sommets seront notés dans l'ordre de leur coloration $som(1)$, $som(2)$, ..., $som(|c_1|)$; nous fournissons ci-dessous la colonne des sommets colorés dans la première clique, avec leur ordre de coloration, ainsi que la classe initiale des colorations propres associées, relatives à cette seule clique. Ici [] désigne l'ensemble vide; en effet le sommet $som(1)$ n'est soumis à aucun interdit pour assurer le fait qu'une coloration de la clique c_1 soit injective sur c_1.

Sommets		classe unique de coloration propre pour c_1
$som(1)$		$[]$
$som(2)$	et	$[som(1)]$
$som(3)$		$[som(1), som(2)]$
$\vdots$		$\vdots$
$som(\|c_1\|)$		$[som(1), som(2), ..., som(\|c_1\| - 1)]$

,

ETAT DE *classes* AVANT LA COLORATION DU SOMMET $som(i)$

Pour que nous soyons en état de faire intervenir le nouveau sommet $som(i)$ et donc de le colorer, *classes* sera constituée d'une ou plusieurs colonnes représentant une partition de l'ensemble des colorations propres liées aux seuls sommets $som(1)$, $som(2)$, ..., $som(i-1)$; aucune colonne de *classes* ne comportera, au début, de cellules à deux étages. Dans chaque colonne seront présents seulement des vecteurs d'un ou plusieurs sommets déjà colorés, éventuellement vides, ou de la forme $[-som(t)]$ pour signifier que le sommet correspondant a la même couleur que $som(t)$. Dans ce dernier cas, on affectera la valeur opposée du premier sommet coloré au cas où il en existe plusieurs identiques; par exemple si $som(8)$ est de même couleur que $som(2)$ et $som(5)$, nous affecterons comme interdits aux niveaux des sommets $som(5)$ et $som(8)$, le même vecteur $[-som(2)]$.

Notons qu'en fin d'initialisation de *classes*, nous disposons d'une colonne unique qui se trouve effectivement dans l'état décrit ci-dessus.

PRÉTRAITEMENT DE CHACUNE DES CLASSES AU NIVEAU i

Nous décrivons ici la fonction $pretraitement(som(i), charpente, classes \rightarrow classes)$

Nous ajoutons au niveau i de toutes les classes existantes qui ne comportaient encore que $i-1$ lignes, une ligne constituée de la cellule calculée en amont, et notée $charpente\,\{i\}$; elle représente les interdits initiaux du sommet $som(i)$ en cours de traitement, au seul vu de la "géométrie algébrique" du graphe traité.

Pour chaque colonne de *classes*, nous faisons agir la partie supérieure sur la dernière ligne introduite. Sans être complet dans la définition formelle, nous fournissons une liste exhaustive des traitements à effectuer:

- les occurences des sommets antérieurs qui ont changé de couleur sont remplacées par leurs nouvelles définitions;
- des contrôles de cohérence sont effectués;
- des contraintes nouvelles venues de l'identité de couleur de sommets auparavant distincts, seront transmises vers les sommets supérieurs;

Ainsi lors de l'exécution de cette fonction, certaines classes pourront disparaître pour cause d'incohérence, mais surtout les anciennes écritures constituées seulement de vecteurs, devenir à nouveaux des cellules à deux étages lorsqu'un interdit nouveau sera transféré à un sommet précédemment coloré.

Traitement de chacune des classes au niveau i

Nous résumons ici l'effet de la fonction $traitement(som(i), charpente, classes \rightarrow classes)$.

Il s'agit de la partie centrale de la mise en forme des classes de colorations propres; elle consiste pour l'essentiel à transformer chacune des colonnes de *classes* de l'état de classes représentées par des cellules à deux niveaux, en une réunion disjointe de classes contenant des vecteurs seulement, donc réduites à un seul niveau.

- Le principe actif est le suivant. Considérons une colonne *col* quelconque de *classes*. Nous détectons, s'il existe un plus petit niveau j pour lequel $col(j)$ est une cellule à deux niveaux, c'est-à-dire:

$$col(j) = \begin{pmatrix} [\text{ensemble de sommets de couleurs déjà distinctes }] \\ [\text{autres interdits de } som(j)] \end{pmatrix}.$$

  Pour chaque élément *autre* de l'ensemble des "autres interdits de $som(j)$", étage 2 de $col(j)$, nous mettons en oeuvre une procédure de contraction - expansion afin de partitionner la classe représentée par *col* en cours d'étude, et la remplacer dans la nouvelle écriture de *classes* sous forme d'une partition plus fine.

  - Nous détectons les éléments de l'étage 1 qui peuvent être égaux à *autre* sans provoquer d'incohérence, et en déduisons les sous-classes associées.
  - Puis nous faisons naître la sous-classe associée au fait que *autre* est supposé distinct de tous les éléments de l'étage 1 de $col(j)$.
  - Dans les deux étapes précédentes, des relations de cohérence seront transférées vers les niveaux supérieurs de *col* sous forme de nouvelles cellules créées, éventuellement à deux étages.
  - Après ces deux étapes, nous pouvons faire "descendre" *autre* de l'étage 2 vers l'étage 1. L'ancienne colonne *col* est alors remplacée par celle(s) qu'elle a fait naître. Ceci assurera la convergence de l'algorithme. En effet, petit à petit, les cellules à étages doubles vont s'évacuer vers le haut pour disparaître de toutes les colonnes créées en remplacement des colonnes intermédiaires disparues.

- Le traitement précédent est mis en oeuvre tant qu'il existe une colonne de *classes* contenant encore un niveau écrit sous forme de cellule à deux étages.

En guise de bilan, nous obtenons le résultat suivant.

Algorithme 181 classes.de.coloration $\left((som(i))_{1\leq i\leq n}, charpente \rightarrow classes\right)$

- ***Entrée:***
  - $(som(i))_{1\leq i\leq n}$ *vecteurs des sommets de* G*, fournis dans l'ordre de leur coloration.*
  - *charpente définie en amont.*
- ***Sortie:*** *classes désigne une partition de l'ensemble des colorations propres de* G*.*
- ***Corps d'algorithme***

  Début

  1. *Initialisation de classes*
  2. ***Faire pour*** $i = 1$ ***à*** n
     (a) $pretraitement(som(i), charpente, classes \rightarrow classes)$
     (b) $traitement(som(i), charpente, classes \rightarrow classes)$

     Fin de Faire

  Fin de *classes.de.coloration* ( )

Proposition 182 *Par le principe même du fonctionnement de l'algorithme précédent, classes représente une partition de l'ensemble de toutes les colorations propres du graphe* G *étudié; cette partition est connue grâce au suivi des effets sur chaque sommet coloré d'une procédure de contraction-expansion.*

3.3.4 Algorithme principal de détermination de l'ensemble des classes de colorations propres d'un graphe

Algorithme 183 *Fonction principale*

algorithme.principal $\left(\mathcal{G} \rightarrow (c_j)_{1\leq j\leq N}\ ,\ charpente\ ,\ (som(i))_{1\leq i\leq n}\ ,\ classes\right)$

- ***Entrée:*** *$\mathcal{G}$ matrice triangulaire supérieure définissant le graphe étudié G.*
- ***Sortie:*** *Les différents champs ont été définis en amont.*
  - *$(c_j)_{1\leq j\leq N}$ est une suite constructive du graphe.*
  - *charpente désigne la charpente du graphe G; elle représente les liens relatifs des diverses cliques maximales.*
  - *$(som(i))_{1\leq i\leq n}$ est le vecteur des sommets du graphe, dans l'ordre de leur coloration.*
  - *classes désigne une partition de l'ensemble des colorations propres de G.*
- ***Corps d'algorithme***

  Début

  1. **rech.cliques.max** $(\mathcal{G} \rightarrow tailles, cliques)$ *d'où la décomposition en cliques maximales* $\mathcal{D}(G)$.
  2. **suite.constructive** $\left(\mathcal{D}(G) \rightarrow (c_j)_{1\leq j\leq N}\right)$
  3. **creation.charpente**$((c_j)_{1\leq j\leq N} \rightarrow (som(i))_{1\leq i\leq n}, charpente)$
  4. **classes.de.coloration** $\left((som(i))_{1\leq i\leq n} \rightarrow classes\right)$

  Fin de **algorithme.principal** ( )

3.3.5 Exemples de mise en oeuvre de l'algorithme principal

Nous exposons dans cette section la production des classes de coloration pour trois graphes réduits à 7 sommets. Nous présenterons dans le premier cas une situation très simple, dans laquelle il n'existe qu'une classe de colorations propres, en raison du fait que chaque clique maximale du graphe considéré coupe le graphe engendré par les précédentes à l'intérieur seulement de l'une de ces cliques; en conséquence, nous fournissons uniquement les résultats en laissant au lecteur le soin de les retrouver. Par contre nous présentons deux autres cas de natures assez différentes, pour lesquels le procédé de contraction-expansion sera mis en oeuvre et fournirons quelques détails de la genèse des étapes intermédiaires de la construction des classes de coloration.

Nous précisons quelques notations valides pour tous les exemples traités dans cette section.

- i, entier compris entre 1 et 7, désigne le numéro d'ordre de coloration des sommets $som(i)$ des différents graphes;
- E désigne l'ensemble des arêtes du graphe considéré, tandis que $ch\{i\}$ représente la charpente de numéro i;
- $int\{i, j\}$ les interdits du sommet $som(i)$ de la j° classe de coloration propre;
- et $sc\{l\}$ la l° clique maximale de la suite constructive utilisée.

Etude du graphe $grn7.5$

On considère le graphe, noté $grn7.5$, dont une représentation est donnée par la figure 3.5.

- Suite constructive de $grn7.5$

$$sc\{1\} = \{1, 2, 3, 4\}\,; sc\{2\} = \{2, 3, 4, 5\}\,; sc\{3\} = \{3, 4, 6\}\,; sc\{4\} = \{6, 7\}\,.$$

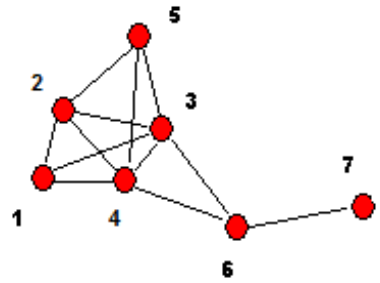

Figure 3.5: Graphe $grn7.5 \quad |E| = 12$

- *charpente* de $grn7.5$

$som(i)$	1	2	3	4
$ch\{i\}$	[]	[1]	[2, 1]	[3, 2, 1]
Graphe engendré par	$\{1, 2, 3, 4\}$	$\{1, 2, 3, 4\}$	$\{1, 2, 3, 4\}$	$\{1, 2, 3, 4\}$

$som(i)$	5	6	7
$ch\{i\}$	[4, 3, 2]	[4, 3]	[6]
Graphe engendré par	$\{1, 2, 3, 4\}$ $\{2, 3, 4, 5\}$	$\{1, 2, 3, 4\}$ $\{2, 3, 4, 5\}$ $\{3, 4, 6\}$	$\{1, 2, 3, 4\}$ $\{2, 3, 4, 5\}$ $\{3, 4, 6\}$ $\{6, 7\}$

- Il existe une seule classe de colorations propres, qui se réduit à la charpente du graphe, évidemment. Ici de façon étonnante, l'ordre de coloration des sommets se réduit au numéro du sommet, ce qui est une situation peu ordinaire.

$som(i)$	$int\{i,1\}$
1	[]
2	[1]
3	[2, 1]
4	[3, 2, 1]
5	[4, 3, 2]
6	[4, 3]
7	[6]

ETUDE DU GRAPHE $grn7.0$

On considère le graphe, noté $grn7.0$, dont une représentation est donnée par la figure 3.6:

- Suite constructive de $grn7.0$

$$sc\{1\} = \{1,3,4\}\,; sc\{2\} = \{1,2,3\}\,; sc\{3\} = \{2,3,6\}\,; sc\{4\} = \{3,4,6\}\,;$$
$$sc\{5\} = \{1,4,5\}\,; sc\{6\} = \{4,5,6\}\,; sc\{7\} = \{1,5,7\}\,; sc\{8\} = \{1,2,7\}\,.$$

- *charpente* de $grn7.0$

$som(i)$	1	3	4	2
$ch\{i\}$	[]	[1]	[3, 1]	[3, 1]
Graphe engendré par	$\{1,3,4\}$	$\{1,3,4\}$	$\{1,3,4\}$	$\{1,3,4\}$ $\{1,2,3\}$

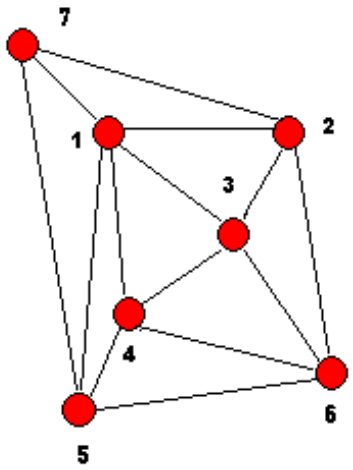

Figure 3.6: Graphe $grn7.0 \quad |E| = 14$.

$som(i)$	6	5	7
$ch\{i\}$	$\begin{pmatrix} [2,3] \\ [4] \end{pmatrix}$	$\begin{pmatrix} [6,4] \\ [1] \end{pmatrix}$	$\begin{pmatrix} [5,1] \\ [2] \end{pmatrix}$
Graphe engendré par	$\{1,3,4\}$ $\{1,2,3\}$ $\{2,3,6\}$ $\{3,4,6\}$	$\{1,3,4\}$ $\{1,2,3\}$ $\{2,3,6\}$ $\{3,4,6\}$ $\{1,4,5\}$ $\{4,5,6\}$	$\{1,3,4\}$ $\{1,2,3\}$ $\{2,3,6\}$ $\{3,4,6\}$ $\{1,4,5\}$ $\{4,5,6\}$ $\{1,5,7\}$ $\{1,2,7\}$

- Génération des différentes classes de colorations propres

– Jusqu'à l'introduction du sommet $2 = som(4)$ il existe une seule classe de colorations propres relative aux deux premières cliques maximales de la suite constructive correspondant respectivement aux trois premières lignes puis à la totalité de la table suivante:

$som(i)$	$int\{i, 1\}$
1	[]
3	[1]
4	[3, 1]
2	[3, 1]

– Lors de l'introduction du sommet $6 = som(5)$, vu les interdits portant sur le sommet 4, celui-ci ne peut être de même couleur que le sommet 3; par contre pour obtenir l'ensemble des colorations propres, on doit distinguer deux cas possibles qui s'excluent: le cas où le sommet 4 sera de la couleur du sommet 2, noté $int\{4, 2\} = [-4]$ et le cas où le sommet 4 sera de couleur différente du sommet 2 qui sera noté $int\{4, 1\} = [4, 3, 1]$ en remplacement de l'ancien $[3, 1]$.

Lors de cette opération de contraction-expansion, il faut être conscient qu'on doit faire rétroagir les contraintes nouvelles exprimées; ici, ce type de prise en charge est transparente en raison du fait qu'elles opèrent déjà, sont déjà exprimées,"naturellement", au sens du graphe engendré par les deux premières cliques maximales. En effet, le fait que le sommet 2 soit de même couleur que le sommet 4 n'impose rien de plus que l'existant puisque le sommet 4 devait déjà être différent de $[3, 1]$ en raison de l'existence de la clique $\{1, 3, 4\}$. De même, le fait que 4 vienne s'adjoindre à $[3, 1]$ repose sur le fait que "naturellement", au sens du graphe engendré par les deux premières cliques maximales, 4 est déjà distinct de 3 (vu la clique $\{1, 3, 4\}$)

et 4 est distinct de 1, en raison de la même clique. S'il en avait été autrement il eût fallu transcrire ces interdits par une cellule à deux étages portant sur le sommet adéquat, et l'effacement de ce deuxième étage aurait été pris en charge ultérieurement par l'algorithme pour obtenir,"in fine" une table sans cellules à double niveau avant de pouvoir passer au traitement du sommet suivant, comme vu dans le descriptif de la fonction **classes.de.coloration** ( ).

Ainsi lors de l'introduction du sommet 6, il existe deux classes de colorations propres, représentées comme suit:

$som(i)$	$int\{i,1\}$	$int\{i,2\}$
1	[]	[]
3	[1]	[1]
4	[3, 1]	[3, 1]
2	[4, 3, 1]	[−4]
6	[2, 4, 3]	[4, 3]

– On itère le processus présenté ci-dessus, en introduisant le sommet $5 = som(6)$. A ce stade, on constate l'existence de quatre classes de colorations propres correspondant au graphe engendré par les six premières cliques maximales. Elles sont données par:

$som(i)$	$int\{i,1\}$	$int\{i,2\}$	$int\{i,3\}$	$int\{i,4\}$
1	[]	[]	[]	[]
3	[1]	[1]	[1]	[1]
4	[3, 1]	[3, 1]	[3, 1]	[3, 1]
2	[4, 3, 1]	[4, 3, 1]	[−4]	[−4]
6	[2, 4, 3, 1]	[−1]	[4, 3, 1]	[−1]
5	[6, 4, 1]	[4, 1]	[6, 4, 1]	[4, 1]

– La poursuite du même traitement conduit à la dernière table résultat comportant six classes de colorations propres pour le graphe $grn7.0$, alors reconstruit en totalité, par la prise en compte des huit cliques maximales de la suite constructive.

On sera attentif au traitement des cas d'égalité de couleurs entre sommets; si par exemple en colonne j, le sommet $som(i)$ est de couleur identique à celle d'un sommet précédent, on affectera $int\{i,j\}$ de la valeur opposée du premier sommet coloré de cette manière, d'où la disparition de certaines occurrences. En cas d'égalité de couleurs multiples, tous les sommets "inférieurs" de couleur identique donneront lieu à des $int\{i,j\}$ affectés de l'opposé de la "tête de colonne".

$som(i)$	$int\{i,1\}$	$int\{i,2\}$	$int\{i,3\}$	$int\{i,4\}$	$int\{i,5\}$	$int\{i,6\}$
1	[]	[]	[]	[]	[]	[]
3	[1]	[1]	[1]	[1]	[1]	[1]
4	[3, 1]	[3, 1]	[3, 1]	[3, 1]	[3, 1]	[3, 1]
2	[4, 3, 1]	[4, 3, 1]	[4, 3, 1]	[4, 3, 1]	[−4]	[−4]
6	[2, 4, 3, 1]	[2, 4, 3, 1]	[−1]	[−1]	[4, 3, 1]	[−1]
5	[6, 2, 4, 1]	[−2]	[2, 4, 1]	[−2]	[6, 4, 1]	[4, 1]
7	[5, 2, 1]	[2, 1]	[5, 2, 1]	[2, 1]	[5, 4, 1]	[5, 4, 1]

– On notera qu'une colonne d'une table antérieure ne produit pas toujours le même nombre de sous-partitions au niveau suivant. Certaines colonnes peuvent disparaître car elles donnent lieu à une incohérence au sens du graphe en cours de construction, ou bien se subdiviser de nombreuses fois, ou bien donner naissance à une seule classe. Ceci s'observe bien en dynamique lorsqu'on demande de visualiser le nombre de colonnes de

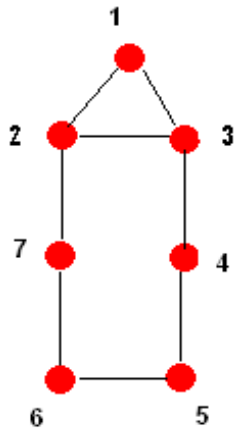

Figure 3.7: Graphe $grn7.4$: $|E| = 8$

la table en cours de génération; on peut passer pour les graphes traités, relativement modestes par exemple de 40 ou 50 sommets, par des tables de 2000 colonnes qui se réduisent à 300 par exemple, lors de passages par de véritables goulots d'étranglement dûs à la "géométrie algébrique" du graphe.

Etude du graphe $grn7.4$

On considère le graphe, noté $grn7.4$, dont une représentation est donnée en figure 3.7.

- Suite constructive de $grn7.4$

$$\begin{aligned} sc\{1\} &= \{1,2,3\}\,; sc\{2\} = \{3,4\}\,; sc\{3\} = \{2,7\}\,; \\ sc\{4\} &= \{6,7\}\,; sc\{5\} = \{5,6\}\,; sc\{6\} = \{4,5\}\,. \end{aligned}$$

- *charpente* de $grn7.4$

$som(i)$	1	2	3	4
$ch\{i\}$	[]	[1]	[2, 1]	[3]
Graphe engendré par	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$ $\{3,4\}$

$som(i)$	7	6	5
$ch\{i\}$	[2]	[7]	$\begin{pmatrix} [6] \\ [4] \end{pmatrix}$
Graphe engendré par	$\{1,2,3\}$ $\{3,4\}$ $\{2,7\}$	$\{1,2,3\}$ $\{3,4\}$ $\{2,7\}$ $\{6,7\}$	$\{1,2,3\}$ $\{3,4\}$ $\{2,7\}$ $\{6,7\}$ $\{5,6\}$ $\{4,5\}$

- Jusqu'au sommet $6 = som(6)$, la structure du graphe en cours de construction est telle qu'il existe une seule classe de colorations propres, constituée des trois premières, des quatre premières et enfin, des six lignes de la table suivante:

$som(i)$	$int\{i,1\}$
1	[]
2	[1]
3	[2, 1]
4	[3]
7	[2]
6	[7]

- Par contre, en raison de l'assemblage algébrique des diverses cliques maximales de la suite constructive du graphe, la prise en considération du sommet 5, fait naître les conflits qui vont provoquer des effets multiples rétroactifs sur la table antérieure, provoquer des sous-partitions de classes qui se fractionneront à nouveau avant que les contraintes exprimées par des cellules à deux étages ne soient traduites par des vecteurs simples de sommets de couleurs, certainement différentes. En conséquence il existera cinq classes de colorations fournies ci-dessous.

$som(i)$	$int\{i,1\}$	$int\{i,2\}$	$int\{i,3\}$	$int\{i,4\}$	$int\{i,5\}$
1	[]	[]	[]	[]	[]
2	[1]	[1]	[1]	[1]	[1]
3	[2, 1]	[2, 1]	[2, 1]	[2, 1]	[2, 1]
4	[3, 2]	[3, 2]	[−2]	[3, 2]	[−2]
7	[−4]	[4, 2]	[2]	[4, 2]	[2]
6	[4]	[−4]	[−2]	[7, 4]	[7, 2]
5	[6, 4]	[4]	[2]	[6, 4]	[6, 2]

On pourra à titre d'exercice, étudier le détail du passage de l'unique classe précédente à sa subdivision en les cinq classes finales.

3.3.6 Bilan relatif à l'algorithme principal

Théorème 184 *Des classes de colorations propres*

Etant donné un graphe G, non orienté et simple, on peut déterminer une partition de ses colorations propres en classes, dont les éléments peuvent être définis explicitement et se déduisent les uns des autres par des permutations spécifiques à chacune des classes.

Remarque 185 *Modalités de vérification des résultats trouvés*

L'analyse de la fonction de détermination des classes de colorations, montre que l'ensemble produit est certainement une partition de cet ensemble. En vue de vérifier informatiquement la cohérence des résultats obtenus, nous mettons en oeuvre deux outils.

- *D'une part, nous vérifions que chacune des classes trouvées est effectivement une coloration propre. Nous disposons pour cela de deux méthodes selon les nombres respectifs des arêtes du graphe considéré et celui de ses cliques maximales. Pour la seconde approche en termes de cliques, nous vérifions que chaque classe assure l'injectivité d'une coloration sur toute clique maximale; pour la première approche, le contrôle s'effectue en termes de respect de chaque arête.*

- *D'autre part, nous évaluons le nombre des colorations via la connaissance de cette partition et vérifions, par génération de celles-ci sous le langage C, qu'il y a coïncidence des deux nombres de colorations.*

L'algorithme principal décrit ci-dessus a pour objet de produire l'ensemble de toutes les classes de colorations propres pour un graphe G, lorsque le nombre de couleurs disponibles k désigne une variable formelle, non fixée à l'avance. Nous

verrons dans le chapitre suivant comment la connaissance d'une partition des colorations propres de G permettra de retrouver le polynôme chromatique du graphe, certaines propriétés générales du graphe mais aussi un certain nombre de propriétés de permutations sur les colorations propres. En effet, comme annoncé, une part de la combinatoire est ensapsulée dans les cliques maximales; la difficulté de la coloration réside, non pas dans le nombre de sommets comme le soulignent de nombreux papiers - nous y reviendrons plus loin - mais dans la géométrie relative de ces cliques maximales, dans leurs rapports "ensemblistes" débarrassés des permutations parasites; sans elles, la situation est déjà suffisamment compliquée; mais hélas notre algorithme n'a pas la vertu de muer une situation complexe en une trivialité !

3.4 Deux "produits dérivés" de l'algorithme principal

L'algorithme principal, vu notre choix de représenter les diverses classes de colorations par des colonnes, a pour effet de générer horizontalement l'ensemble de toutes ces classes. Il est naturellement limité par le temps de calcul, et la taille des objets manipulés. Toutefois, il apparaît que deux sous-produits peuvent s'en déduire, dont l'intérêt pratique est bien connu de la communauté.

- D'une part, il est possible de ne générer qu'une partie de l'ensemble de ces classes limitée par la donnée du nombre maximal r de couleurs disponibles. Si r est strictement inférieur au nombre chromatique du graphe, l'ensemble des classes généré sera vide. On peut espérer que cette génération s'arrête suffisamment tôt lors du processus, pour transmettre l'information que le r choisi est trop petit. Hélas, on a vu, par l'exemple du graphe $grn7.0$, que c'est lors de l'introduction du dernier sommet traité que la nécessité de quatre couleurs s'exprime algorithmiquement, du moins; sans doute une autre méthode théorique devrait-elle s'appliquer qui rende visible cette situation dès

la connaissance de la "géométrie globale" de l'ensemble des cliques maximales du graphe. Nous y travaillons mais cela semble très difficile...

- D'autre part, on peut accepter de ne pas disposer de l'ensemble des classes de coloration mais souhaiter en connaître une seule afin de déterminer uniquement une solution particulière du problème. Pourtant, notre algoritme fournira toujours une classe de solutions car il ne "sait faire" que cela; par contre, il est très simple d'extraire un exemple de coloration particulière solution à partir d'une telle classe.

  Cette nouvelle problématique revient, par rapport à l'algorithme principal, à inverser le sens de parcours de recherche de solution; au lieu de chercher horizontalement, tout se passe comme si on tentait de plonger verticalement à la recherche d'une classe solution particulière. Pour autant cette inversion de tendance est extrêmement perturbant, pour l'algorithme. Si les mêmes modules sont efficients, ils sont profondément modifiés dans leurs effets et leur contrôle. Le caractère plus récent de leur mise en oeuvre nous fait encore rencontrer des difficultés pour traduire avec fiabilité le fait que tel graphe n'a pas de r-solution; mais la question devrait être totalement réglée rapidement.

3.4.1 Algorithme de détermination partielle des classes de colorations propres

Définition de l'algorithme de détermination partielle

$\mathbf{algorithme.partiel}\left(\mathcal{G}, r \to (c_j)_{1\leq j\leq N}\ ,\ charpente\ ,\ (som(i))_{1\leq i\leq n}\ ,\ cl.partielles\right)$ représente le premier produit dérivé; nous allons le définir maintenant.

Comme montré ci-dessus, l'algorithme principal fournit l'ensemble des classes de colorations propres d'un graphe donné. Or, nous pouvons en suivant exactement le même procédé, nous limiter à celles des classes de colorations qui utilisent au

plus r couleurs, où r représente en général un entier supérieur ou égal au nombre chromatique χ du graphe étudié, si toutefois celui-ci est connu; s'il n'est pas connu, cette fonction pourra contribuer à le déterminer.

La définition de cette recherche partielle est identique à celle de l'algorithme principal, hormis que le champ d'entrée complémentaire r s'impose, et que le résultat produit représente, par le principe même, une partition des colorations partielles propres, c'est-à-dire un sous-ensemble des classes issues de l'algorithme principal. En résumé nous avons le descriptif suivant, dans lequel la partie consacrée à la détermination des paramètres généraux du graphe est évidemment toujours la même, puisque incontournable.

Seule la fonction **classes.de.coloration** $\left((som(i))_{1\leq i\leq n}, charpente \rightarrow classes\right)$ se trouve modifiée, pour se réduire à la recherche des classes de colorations limitées à l'utilisation de r couleurs au plus.

Algorithme 186 *Premier produit dérivé*

algorithme.partiel $\Big(\mathcal{G}, r \rightarrow (c_j)_{1\leq j\leq N}$, *charpente* , $(som(i))_{1\leq i\leq n}$, *cl.partielles*$\Big)$

- ***Entrée:***
  - $\mathcal{G}$ *matrice triangulaire supérieure définissant le graphe étudié* G.
  - r *entier naturel non nul égal au nombre de couleurs disponibles.*
- ***Sortie:***
  - $(c_j)_{1\leq j\leq N}$ *est une suite constructive du graphe.*
  - *charpente désigne la charpente du graphe* G*; elle représente les effets relatifs des diverses cliques maximales.*
  - $(som(i))_{1\leq i\leq n}$ *est le vecteur des sommets du graphe, dans l'ordre de leur coloration.*

– *cl.partielles désigne une partition de l'ensemble des colorations propres de G, utilisant r couleurs.*

- ***Corps d'algorithme***

  Début

  1. **rech.cliques.max** $(\mathcal{G} \rightarrow tailles, cliques)$ *d'où la décomposition en cliques maximales* $\mathcal{D}(G)$.
  2. **suite.constructive** $\left(\mathcal{D}(G) \rightarrow (c_j)_{1 \leq j \leq N}\right)$
  3. **creation.charpente**$((c_j)_{1 \leq j \leq N} \rightarrow (som(i))_{1 \leq i \leq n}, charpente)$
  4. **cl.partielles** $\left((som(i))_{1 \leq i \leq n}, charpente, r \rightarrow cl.partielles\right)$

  Fin de **algorithme.partiel** ( )

Corollaire 187 *Si l'entier r est inférieur strictement au nombre chromatique, l'ensemble des classes partielles, cl.partielles, sera vide.*

Remarque 188 *Comme évoqué plus haut, l'expérience montre que selon la géométrie algébrique du graphe considéré, la détection du fait que l'ensemble des classes de colorations propres est vide peut être relativement tardif lors de l'exécution et ne pas produire l'effet de renseignement rapide espéré, surtout lorsqu'on s'approche d'une possible solution. Toutefois, en général, le gain est réel.*

Exemple d'utilisation

Nous pouvons revisiter l'exemple du graphe *grn*7.0 décrit ci-dessus dans la section 3.3.5. Comme le principe de recherche des classes est exactement le même, nous pouvons en suivant l'introduction successive de chaque sommet, déterminer fort simplement l'ensemble des classes manipulées lors de l'appel de l'algorithme partiel

pour $r = 3$, par exemple. En pratique, cela signifie que l'algorithme partiel génère les seules classes pour lesquelles les vecteurs d'interdits comportent au plus deux sommets. On remarquera qu'une classe est présente seulement jusqu'au traitement de l'avant dernier sommet; ceci consacre l'impossibilité de trouver une classe solution si l'on dispose de seulement trois couleurs.

Par contre, l'appel de l'algorithme partiel pour $r = 4$, montre que les classes produites sont au nombre de quatre finalement, fournies par les colonnes 3, 4, 5 et 6 de la table de l'ensemble total des classes. Les détails fournis dans la section 3.3.5, permettent d'en suivre la génération au niveau de l'introduction de chaque sommet nouveau.

Certes nous ne sommes pas en possibilité de fournir une étude de complexité de la fonction proposée mais pensons contribuer à montrer la cause de celle-ci. Le fait de calculer quelques moyennes sur un ensemble d'instances, même nombreuses, n'apporterait pas à nos yeux de renseignements plus pertinents !

3.4.2 Algorithme de recherche d'une solution particulière de r-coloration

Comme évoqué dans l'introduction, nous allons présenter maintenant le principe d'une seconde variante de l'algorithme principal, destinée à la recherche d'une solution particulière, dont l'étude est plus récente et nous laisse en conséquence encore quelques incertitudes quant au fait de conclure à la non-existence de solution pour un graphe et un entier r donnés.

Principe de la recherche de solution particulière

Nous décrivons dans cette section la définition du principe de la fonction:

sol.particuliere $\big((som(i))_{1\leq i\leq n}, charpente, r \rightarrow sol.part\big)$.

Nous nous appuyons évidemment sur la connaissance du fonctionnement de l'algorithme principal, et allons en décrire l'idée majeure, de manière différentielle.

L'algorithme principal, afin de produire l'ensemble des colorations propres jusqu'au niveau d'un sommet $som(i)$, commence par ajouter une ligne à toutes les colonnes représentatives de l'ensemble des colorations propres pour les sommets $som(1)$, ...,$som(i-1)$; cette ligne ajoutée est constituée de l'élément $charpente\,\{i\}$. Puis il traite chacune des colonnes en la remplaçant chaque fois qu'il existe une cellule à deux niveaux, par la sous-partition obtenue par contraction-expansion sur les éléments du second étage. De façon plus précise, il crée toutes les sous-classes de la classe initiale obtenues en traduisant qu'un élément s du second étage peut être de même couleur que l'un des sommets du premier étage, mais aussi que ce même élément s peut être de couleur distincte de tous les sommets du premier étage, pourvu dans les deux cas, que ce soit conforme à la géométrie algébrique du graphe en construction.

L'algorithme de recherche de solution particulière ne crée pas toutes les classes possibles; il en crée une seule, sachant que l'on dispose de r couleurs au plus, afin de limiter le nombre des possibles visités, en cas d'échec surtout d'ailleurs.

- Il privilégie d'abord le "risque" de la différence. En effet parmi les sous-classes qui peuvent naître dans l'algorithme principal, il construira de façon systématique en premier celle qui tente d'attribuer, lors de la présence d'une cellule à deux niveaux, une couleur pour un sommet s du second étage, distincte de toutes celles des sommets du premier étage. En effet, c'est la présence d'une diversité maximale des couleurs qui ouvre le plus de possibles pour solutionner les conflits ultérieurs.

- Par contre, il est facile à comprendre qu'en opérant ainsi, en adoptant une "solution tendue" vers la diversité maximale, nous allons risquer de provoquer

des situations sans solutions, en raison d'un excès d'exigence. Par suite, il est indispensable de mémoriser toutes les sous-classes non traitées ou plutôt de conserver la possibilité de les générer afin qu'elles constituent des solutions de repli, de retour plus précisément, en cas de blocages ultérieurs. Si le fait d'avoir imposé antérieurement trop de diversité conduit à une impasse, donc à la perte de la colonne solution en cours d'élaboration, nous pourrons la remplacer par une solution fondée sur des égalités de couleurs entre sommets, afin de relâcher un peu la tension dans les zones critiques. Il s'en déduira une pénalisation du fait d'un retour en arrière, d'une reprise d'une portion de traitement "déjà solutionnée", du moins qu'on croyait déjà solutionnée avant d'aboutir à une impasse.

- Pour des raisons d'optimisation locale de cette fonction, les remplaçantes des dernières solutions choisies seront positionnées de telle sorte qu'elles soient les premières visitées en cas d'échec ultérieur; en effet, vu que les choix précédents ont été presque satisfaisants, on peut espérer trouver une solution proche du dernier choix effectué.

- De façon plus fine, les diverses solutions en réserve seront ordonnées de telle sorte que les moins "traumatisantes" pour un déroulement aisé de la recherche, soient visitées les premières en cas d'échec ultérieur. Ainsi les plus pénalisantes, celles qui provoquent le plus de reprises, le plus de retours en arrière, ne seront étudiées qu'en cas d'échec des plus efficaces tentées initialement et des moins coûteuses proches.

- Enfin, il est impératif que les colonnes tenues en réserve représentent tous les possibles non visités. En effet, c'est le prix à payer pour qu'en cas de non sortie de solution, on puisse en déduire qu'il n'en existe effectivement aucune, pour le nombre maximal r de couleurs choisi.

Nous fournissons ci-dessous le plan de la fonction

rech.solution $\Big(\mathcal{G}, r \rightarrow (c_j)_{1\leq j\leq N}$, *charpente* , $(som(i))_{1\leq i\leq n}$, *sol.part*$\Big)$.

Algorithme 189 *Second produit dérivé*

rech.solution $\Big(\mathcal{G}, r \rightarrow (c_j)_{1\leq j\leq N}$, *charpente* , $(som(i))_{1\leq i\leq n}$, *sol.part*$\Big)$

- ***Entrée:***
  - $\mathcal{G}$ *matrice triangulaire supérieure définissant le graphe étudié* G.
  - r *entier naturel non nul égal au nombre de couleurs disponibles.*
- ***Sortie:***
  - $(c_j)_{1\leq j\leq N}$ *est une suite constructive du graphe.*
  - *charpente désigne la charpente du graphe* G*; elle représente les effets relatifs des diverses cliques maximales.*
  - $(som(i))_{1\leq i\leq n}$ *est le vecteur des sommets du graphe, dans l'ordre de leur coloration.*
  - *sol.part désigne une classe particuliere de* r*-colorations propres de* G.
- ***Corps d'algorithme***

  Début

  1. **rech.cliques.max** $(\mathcal{G} \rightarrow tailles, cliques)$

     d'où la décomposition en cliques maximales $\mathcal{D}(G)$.
  2. **suite.constructive** $\Big(\mathcal{D}(G) \rightarrow (c_j)_{1\leq j\leq N}\Big)$
  3. **creation.charpente**$((c_j)_{1\leq j\leq N} \rightarrow (som(i))_{1\leq i\leq n}, charpente)$

4. **sol.particuliere** $\left((som(i))_{1\leq i\leq n}, charpente, r \rightarrow sol.part\right)$

Fin de **rech.solution** ( )

Remarque 190 *En cas de sortie de la fonction* **rech.solution** ( ) *avec une colonne sol.part vide, nous pouvons affirmer qu'il n'existe pas de classe solution, pour le r choisi.*

EXEMPLE 1: CAS D'UN TORE

Nous considérons un graphe "torique", dont nous avons écrit un outil de génération automatique à partir de deux paramètres: sa section (ici triangulaire d'où $s3$) mais aussi le nombre des côtés du "tube", constitué ici de 4 côtés (d'où $t4$); il comporte $n = 12$ sommets. Ce type de graphe connu pour la rudesse de ses conflits, nous a servi à tester nos algorithmes sans complaisance, à les mettre en difficulté; les algorithmes existants à cette étape de notre recherche, ont traité ces cas sans émotions particulières...

Nous fournissons à la figure 3.8 une représentation simplifiée de $tore.s3t4$, avec les seuls arcs liant les sommets $\{1, 2, ..., 6\}$ sachant que pour obtenir les arêtes et cliques maximales manquantes à partir des arêtes représentées et de la liste des premières cliques maximales, il suffira de translater les numéros de sommets de multiples de 3 tout en les réduisant au delà de 12, modulo 12 précisément.

- Sommets et liste des cliques maximales

  Elles sont au nombre de 28. Elles se déduisent par translation et réduction modulo 12 au delà de 12 justement, des sept cliques maximales suivantes:

  $$\{1, 2, 3\}\,;\{1, 2, 5\}\,;\{1, 3, 4\}\,;\{1, 4, 5\}\,;\{2, 3, 6\}\,;\{2, 5, 6\}\,;\{3, 4, 6\}\,.$$

- Nos algorithmes montrent qu'il existe 622 classes de colorations propres.

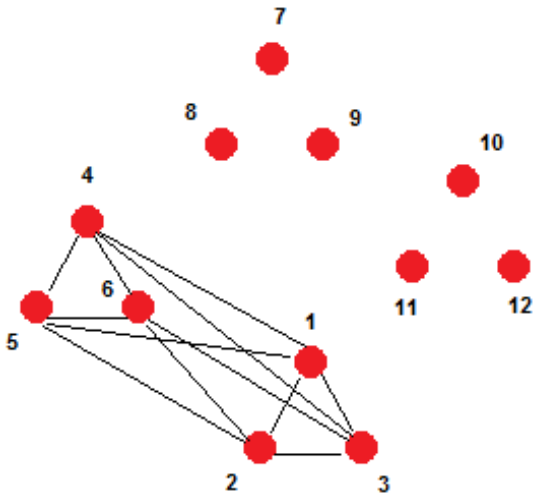

Figure 3.8: Graphe *tore.s3t4* simplifié

- La fonction **rech.solution** ( ) extrait pour $r = 4$, la r-solution donnée par les tables suivantes:

$som(i)$	$sol.part(i)$
1	[]
2	[1]
3	[2, 1]
5	[−3]
10	[3, 2, 1]
12	[−2]

$som(i)$	$sol.part(i)$
4	$[-2]$
6	$[-10]$
11	$[10, 3, 2]$
7	$[-3]$
8	$[-10]$
9	$[10, 3, 2]$

- Comparaison des temps de calcul des étapes essentielles de **rech.solution** ( )

Etapes principales	Temps utilisé en secondes
$rech.cliques.max$ ()	$0,17$
$suite.constructive$ ()	$0,65$
$sol.particuliere$ ()	$0,85$

EXEMPLE 2: CAS DE $grn50.1$

- Le graphe $grn50.1$ a été choisi aléatoirement. Il comporte $n = 50$ sommets, 626 arêtes et 997 cliques maximales dont la répartition, en termes de cardinal, est la suivante:

Cardinal des cliques maximales existantes	3	4	5	6	7
Nombre de cliques du cardinal donné	2	131	521	299	44

.

- La fonction **rech.solution** ( ) extrait comme première solution lorsque r croît (on en déduit $\chi = 10$) une 10-solution que nous ne pouvons évidemment pas transcrire ici...

- Comparaison des temps de calcul des étapes essentielles de **rech.solution** ( )

Etapes principales	Temps utilisé en secondes
rech.cliques.max ()	33, 35
suite.constructive ()	17, 5
sol.particuliere ()	299, 76

3.5 Conclusion

Dans ce chapitre, nous caractérisons une coloration propre f d'un graphe G donné, en termes de respect par injectivité de chacune des cliques maximales de sa décomposition. Ce procédé permet une détermination structurante des colorations propres.

Nous revisitons la méthode de contraction-expansion, grâce à la reconstruction totale du graphe par ajouts sucessifs de cliques maximales, prises en compte dans l'ordre d'une suite constructive de G. Nous précisons le comment de l'émergence de cette méthode, liée au départ à la seule problématique de décompte des colorations propres mais nettement étendue à la définition formelle de toute coloration propre.

Nous en déduisons un algorithme général de construction formelle des colorations propres, qui nous fait retrouver bon nombre de pratiques mises en oeuvre dans diverses heuristiques. Deux sous-produits s'en déduisent.

D'abord une version partielle, qui permet de ne construire que les sous-classes des colorations propres, limitées à l'utilisation de seulement r couleurs disponibles. L'intérêt de cette version simplifiée est réel autant par ses résultats d'existence que de non-existence de telles colorations.

Ensuite, nous détournons l'algorithme premier pour lui faire déterminer non pas l'ensemble des classes de colorations propres possibles, mais une parmi toutes. Ceci revient à parcourir l'ensemble des solutions non plus en largeur, mais en profondeur, bref dans une autre direction que l'algorithme principal et c'est là l'essentiel !

Nous fournissons quelques exemples de mise en oeuvre des diverses versions de cette méthode centrale et de ses corollaires, mais réservons pour le chapitre suivant le descriptif des résultats obtenus ou incomplets, celle des propriétés générales qui en découlent pour l'étude du graphe global et de ses colorations. De façon conjointe, nous y préciserons au fur et à mesure les différents prolongements qui à nos yeux doivent pouvoir venir compléter les résultats obtenus.

Chapitre 4

Analyse critique des résultats obtenus - Prolongements

4.1 Introduction

Nous analysons dans ce chapitre, un certain nombre d'effets des outils développés lors des chapitres 2 et 3, en exploitant les conséquences qui en découlent en termes de propriétés générales sur les graphes. De façon simultanée, nous précisons les liens avec diverses questions connexes et évoquons les prolongements qui nous semblent prometteurs; nous travaillons déjà sur quelques-uns de ceux-ci.

D'abord, nous fournissons des résultats relatifs aux temps de calculs pour quelques graphes classiquement utilisés dans la profession tout en précisant nos attentes assez limitées à cet égard.

Ensuite nous faisons une recherche des propriétés générales des colorations propres d'un graphe, déduites de l'obtention des classes de colorations issues des méthodes du chapitre 3.

- Nous y étudions en particulier les invariances des colorations propres sous certaines permutations, et parfois lorsque c'est possible, quelques résultats formels venus de la connaissance d'une partition de leur ensemble.

- Nous retrouvons des résultats de dénombrement des solutions via le polynôme chromatique du graphe et interrogeons les résultats attendus de cet outil, spécialement de l'étude de ses racines.

Enfin nous établissons une correspondance entre les sous-graphes d'un graphe donné et l'ensemble des colorations propres sur ces différents sous-graphes. Le résultat, partiel encore, doit se généraliser.

4.2 Résultats expérimentaux et questions connexes

Nous devons préciser d'emblée, que dès le début de notre travail, nous n'avons pas envisagé d'entrer dans une course à la performance pour déterminer une coloration propre d'un graphe donné. Nous avons tenté une approche globale, en quête des propriétés générales d'un graphe liées à sa coloration. Ainsi le seul fait de choisir de travailler sous Matlab et non sous le langage C, par exemple, était signifiant à cet égard; nous avons privilégié un outil qui nous apporterait un cadre existant de formalismes généraux plutôt que développer des routines très adaptées à des cas spécifiques. Pour autant, et ce point nous intéresse, il nous semble que la comparaison des coûts temporels des différentes étapes des méthodes mises en oeuvre, présente un intérêt puisqu'ils sont mesurés par rapport au même outil de calcul. A l'envers, il ne nous apparaît pas déterminant de multiplier les expérimentations car une fois mille exemples traités, nous nous trouverons très orphelins au moment d'aborder le suivant ! Nous sommes parfaitement conscients après l'étude des théorèmes du No Free Lunch, de l'impossibilité pour une méthode de traiter de façon heureuse l'ensemble des instances d'un même problème. Très vite il est apparu que dès que nous avons opté pour une méthode de traitement, les idées pour la piéger deviennent très précises !

Notre quête est très humble... Il s'agit de tenter de préciser quels éléments sont pertinents pour caractériser la difficulté de coloration d'un graphe, mais aussi de détecter les paramètres qui semblent pertinents mais ne le sont pas, en raison du fait qu'ils ne représentent pas les causes des phénomènes. Nous sommes parfaitement

désolés d'être incapables de fournir une étude générale de complexité. Mais se révéler capable de le faire, serait maîtriser la situation; et tel n'est pas le cas !

4.2.1 Quelques résultats expérimentaux

Notations

Nous adopterons pour ce paragraphe mais aussi pour le reste du chapitre, un certain nombre de notations permettant en particulier l'utilisation d'abréviations communes dans la définition des lignes et colonnes des tables suivantes. Nous présentons ci-dessous un choix d'exemples, mais disposons de bon nombre d'autres expérimentations qui mériteraient des études statistiques sérieuses que nous ne souhaitons pas présenter ici.

n désigne le nombre des sommets du graphe G considéré, qui sera désigné par un nom nom, afin d'en reconnaître certains utilisés comme exemples dans d'autres parties de notre exposé;

d désigne la densité (arrondie à la deuxième décimale), dont nous verrons qu'elle ne représente pas une caractéristique de difficulté de coloration, mais elle en demeure une fréquente accompagnatrice;

$nb.cl$ désigne le nombre de cliques maximales de la décomposition de G;

$classes$ désigne le nombre de classes de colorations propres déterminé par nos algorithmes.

k désigne le nombre de couleurs disponibles comme variable formelle; si on affecte à k une valeur particulière entière, celle-ci sera notée r. Le nombre chromatique du graphe sera noté comme d'ordinaire χ.

En termes temporels, les résultats sont fournis en secondes, sous Matlab 7.1 tournant sur une machine AMD Athlon XP 2800+, disposant de 1 Go de RAM,

sous Windows XP SP3. Les mesures fournies sont celles de cputime, données en secondes. Nous distinguerons en particulier:

- t_{dec}: le temps de décomposition en cliques maximales;
- t_{sc}: le temps de détermination d'une suite constructive du graphe considéré, ainsi que sa *charpente*;
- $t_{classes}$: le temps de détermination de l'ensemble des classes de colorations;
- t_{sol}: le temps de détermination d'une classe particulière de colorations propres lorsque r, qui désigne le nombre de couleurs disponibles, est précisément choisi, ce qui est le cas pour cette seule mesure. En effet les autres objets déterminés sont formels et par suite, indépendants du choix de r.
- L'indication n_calc signifie que l'exécution a été arrêtée avant l'obtention du résultat correspondant.

QUELQUES RÉSULTATS

a) Exemples utilisés au chapitre 3

nom	n	d	$nb.cl$	t_{dec}	t_{sc}	$classes$	$t_{classes}$
$grn7.5$	7	$0,43$	4	$0,17$	$0,27$	1	$0,33$
$grn7.0$	7	$0,5$	8	$0,20$	$0,25$	6	$0,53$
$grn7.4$	7	$0,29$	6	$0,14$	$0,28$	5	$0,62$
$tore.s3t4$	12	$0,46$	28	$0,15$	$0,48$	622	$14,39$
$grn50.1$	50	$0,49$	997	$33,35$	$17,5$	n_calc	n_calc

nom	$r = \chi$	t_{sol}
$grn7.5$	4	$0,02$
$grn7.0$	4	$0,19$
$grn7.4$	3	$0,23$
$tore.s3t4$	4	$0,92$
$grn50.1$	10	$299,76$

b) Deux grands classiques

A titre de curiosités, mais aussi pour rappeler l'existence de deux graphes qui ont joué dans l'histoire un rôle original d'empêcheurs de tourner en rond, nous proposons ci-dessous le traitement des graphes de Petersen (figure 4.1) et de Soifer (figure 4.2). Le premier a fourni des contre-exemples sans complaisance et s'est révélé un tueur de propriétés générales excessivement optimistes... Le graphe de Soifer est le premier des graphes minimaux montrant en 1997 après Heawood, Poussin que la preuve de Kempe relative au théorème des quatre couleurs, était fausse même si elle avait fourni le concept clé des chaînes dites aujourd'hui de Kempe.

Nous obtenons les résultats suivants.

nom	n	d	$nb.cl$	t_{dec}	t_{sc}	$classes$	$t_{classes}$
$petersen$	10	$0,27$	15	$0,16$	$0,42$	105	$1,97$
$soifer$	9	$0,44$	12	$0,15$	$0,38$	33	$1,09$

nom	$r = \chi$	t_{sol}
$petersen$	3	$0,50$
$soifer$	4	$0,44$

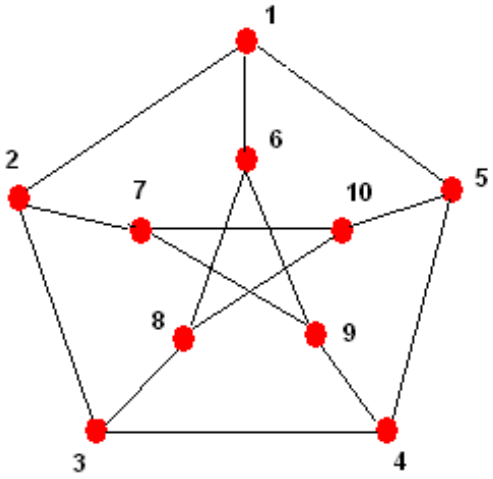

Figure 4.1: Graphe de Petersen

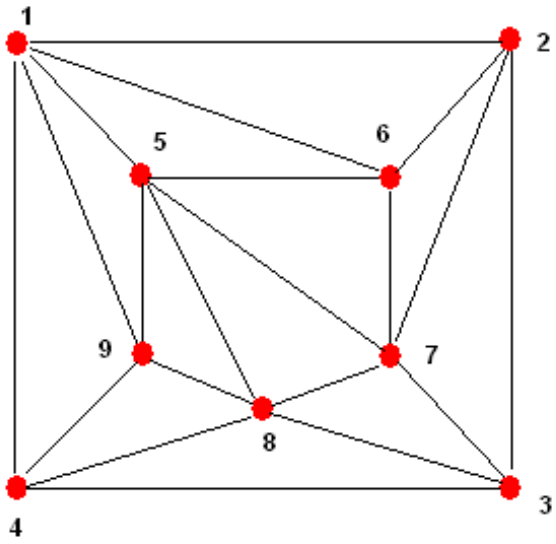

Figure 4.2: Graphe de Soifer

c) Exemples issus des données DIMACS

Nous sommes évidemment limités par les tailles des graphes considérés, et les temps de calcul qui en découlent. Nous mettons en oeuvre nos algorithmes sur quelques exemples issus des données DIMACS.

nom	n	d	$nb.cl$	t_{dec}	t_{sc}	$classes$	$t_{classes}$
$myciel3$	11	$0,30$	20	$0,17$	$0,42$	1678	$23,77$
$myciel4$	23	$0,26$	71	$0,13$	$1,48$	n_calc	n_calc
$queen5$	25	$0,49$	76	$0,42$	$1,34$	n_calc	n_calc
$queen6$	36	$0,44$	129	$1,17$	$2,04$	n_calc	n_calc

nom	$r = \chi$	t_{sol}
$myciel3$	4	$0,91$
$myciel4$	5	$3,66$
$queen5$	5	$1,58$
$queen6$	7	$303,55$

Remarque 191 *On notera à titre d'exemple que notre algorithme nous informe en* 21" *environ que le nombre de couleur* $r = 6$ *ne permet pas de trouver une solution propre pour queen*6*; par contre, il construit une classe de solutions propres pour le même graphe pourvu qu'on dispose de* $r = 7$ *couleurs. Nous prouvons ainsi que* $\chi = 7$.

d) Autres exemples

Comme nous l'avons indiqué plus haut, la complexité de l'assemblage des cliques est responsable de la difficulté de coloration. Nous présentons ci-dessous les résultats relatifs à deux graphes du type rayons à n sommets, dont nous rappelons la construction par la figure 4.3.

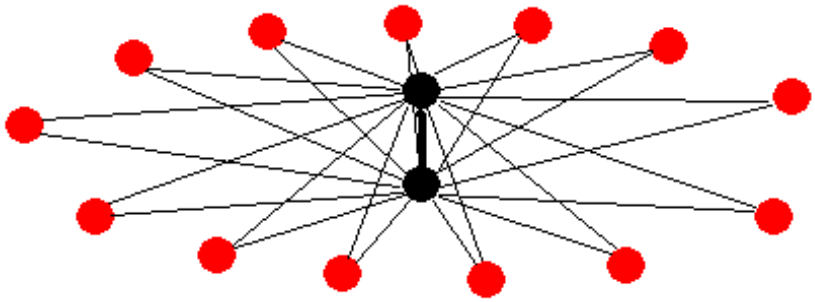

Figure 4.3: Graphe Rayons à 15 sommets

Les sommets du moyeu seront notés $\{1, 2\}$, les autres de 3 jusqu'à n. L'adjonction de sommets nouveaux a pour effet de faire augmenter les degrés des sommets $\{1, 2\}$. Pour deux valeurs de n simples, nous obtenons les résultats suivants.

nom	n	d	$nb.cl$	t_{dec}	t_{sc}	$classes$	$t_{classes}$
$grn50.2$	50	$0,08$	48	$0,17$	$6,33$	1	$1,62$
$grn100.1$	100	$0,04$	98	$0,23$	$58,80$	1	$3,97$

nom	$r = \chi$	t_{sol}
$grn50.2$	3	$0,03$
$grn100.1$	3	$0,03$

Remarque 192 *Si nous voulions obtenir une densité grande, nous pourrions adjoindre à un tel graphe de type "rayons" une clique maximale de taille arbitraire contenant* $\{1, 2\}$ *et des sommets distincts de ceux déjà utilisés. Evidemment, le temps de détermination des cliques maximales serait profondément affecté; pourtant la coloration serait toujours "facile" puisqu'il existerait toujours une seule classe de colorations propres en raison de la géométrie algébrique des cliques mises en jeu. Nous y reviendrons dans la section 4.2.2.*

4.2.2 Eléments d'analyse et prolongements

Difficulté de coloration d'un graphe

Nous avons développé des algorithmes destinés à déterminer tout ou partie de l'ensemble des classes de colorations propres d'un graphe. Au vu de ces méthodes, nous sommes en mesure d'exhiber des exemples de graphes ou de construire des classes de graphes, comportant un nombre arbitraire n de sommets, qui nous permettent de fonder les remarques qui suivent.

- La difficulté de coloration d'un graphe donné n'est pas une fonction, au sens mathématique, du nombre n de sommets, puisque certains types de graphes, y compris pour n grand, comporteront toujours une seule classe de colorations propres (par exemple un graphe de type rayons à n sommets, $rayons\_n$) tandis que d'autres en comporteront un grand nombre. Or le nombre de ces classes est le caractère accompagnateur de cette difficulté. En réalité pour notre algorithme, c'est plutôt l'inverse; c'est parce que la situation est complexe que le nombre de classes augmente...

- Il est par contre fort possible, qu'une majoration suffisamment grossière de la complexité exprimée en termes de n, restitue le caractère fonctionnel en n de l'écriture du majorant dégradé choisi.

- L'essentiel de l'information relative à la difficulté de coloration de G, se trouve dans la charpente du graphe *charpente* (voir chapitre 3), c'est-à-dire dans la manière dont sont assemblées au sens de la suite constructive retenue, les différentes cliques maximales qui figurent dans sa décomposition. Si la clique de numéro i coupe le graphe engendré par les précédentes, à l'intérieur de l'une de ces cliques, la coloration des sommets nouveaux apportés par la clique i sera "facile" . Elle sera difficile par contre lorsque l'intersection de la clique de numéro i avec le graphe engendré par les précédentes, appartient à un nombre conséquent de ces cliques, disjointes. En ce sens, la prise en compte des cliques de grandes tailles en premier lieu dans la suite constructive considérée, renforce l'effet facilitateur de la coloration, en général du moins sauf contre exemple créé spécialement pour infirmer ce résultat et piéger donc la méthode retenue...

Prolongement 193 *Approfondissement des causes de la difficulté de coloration*

Il paraît opportun, au sens d'une suite constructive $(c_j)_{1\leq j\leq N}$ *du graphe* G*, de poursuivre la recherche d'une caractérisation purement algébrique, c'est-à-dire en termes ensemblistes, des "positions relatives" des cliques maximales* c_j *relativement à celles qui les précèdent,* $(c_i)_{1\leq i<j}$*, et ce, pour* j *variable. Nous disposons de quelques pistes, encore incomplètes.*

Parallélisation de la détermination des classes de colorations propres

Prolongement 194 *Parallélisation de la recherche des classes de coloration*

La recherche des classes de colorations, que ce soit pour la détermination de leur ensemble total ou pour la détermination des seules classes relatives à l'utilisation d'un nombre donné r *de couleurs (avec en général* $r \geq \chi$*) revient à traiter un certain nombre variable de colonnes dont la hauteur* i *est liée au sommet* $som(i)$ *en*

cours de prise en compte. Chacun des niveaux relève d'un traitement parfaitement parallélisable, sans ressources partagées ou presque.

Un contact est dès lors pris pour la mise en oeuvre de cette parallélisation dans le cadre d'une collaboration avec l'université de Reims.

Analyse du fonctionnement d'heuristiques

Plusieurs auteurs font remarquer, sans toujours y accorder une grande importance, comme une bizarrerie signalée dans le corps du texte, un certain nombre de phénomènes cycliques rencontrés lors de la convergence de méthodes approchées du type tabou ou utilisant les paysages; voir par exemple les thèses [D07] et [V05]. Nous avons approfondi l'existence de ces phénomènes, pour une méthode tabou utilisée dans notre laboratoire; voir par exemple les très intéressantes études expérimentales conduites par I. Devarenne dans [D04]). Nous avons aussi échangé autour de ces questions avec nos collègues de l'université d'Angers avec qui nous avons mené un projet dans le cadre du GDR-RO07; les résultats ne sont pas évoqués ici en détail, mais ils ont contribué à attirer notre attention sur la présence d'une cause cachée pour ce type de phénomènes, comme le provoque le sentiment d'approcher une énigme !

Lors de la mise en oeuvre de telles méthodes tabous particulièrement, de temps à autre, l'heuristique semble chercher l'élément suivant de la solution en cours d'élaboration et se lance dans des cycles, de cardinalités souvent constantes avant de "plonger" vers une solution effectivement. Nous conjecturons que les cliques maximales sous-jacentes, contenant les sommets concernés, interviennent ainsi que leur agencement avec les cliques voisines, exprimé dans notre paramètre *charpente*. Nous pensons qu'il pourrait être intéressant de faire une étude systématique de ces apparitions de cycles en cours d'exécution.

Prolongement 195 *Analyse de convergence de certaines heuristiques*

Assez fréquemment dans les méthodes tabous par exemple, mais aussi dans certaines autres méthodes de coloration approchées utilisant par exemple les paysages, se produisent des phénomènes de cycles lors de la recherche des solutions, en cours d'exécution. Nous croyons intéressant d'en prévoir une étude systématique et de vérifier leur concomittance avec des géométries spécifiques d'assemblages des cliques maximales présentes dans la décomposition des graphes qui provoquent ce type de comportements des heuristiques.

Densité et difficulté de coloration

Il est possible de construire des graphes de densités arbitrairement grandes, présentant des sommets de degrés arbitrairement grands, dont nos algorithmes montreront qu'ils contiennent toujours une seule classe de colorations propres, et seront, par suite, très simples à colorer.

A titre d'exemple générique et en dehors du cas trivial mais facilement lisible d'une n-clique, il est possible de considérer un graphe comportant $2n$ sommets, dont

- les n premiers constituent une clique maximale $\{x_1, ..., x_n\}$,
- tandis que les suivants adjacents aux sommets $\{x_1, x_2\}$ constitueraient un graphe de type rayons à n sommets décomposable en n 3-cliques données par: $\{x_1, x_2, x_{n+1}\}$, $\{x_1, x_2, x_{n+2}\}$, ..., $\{x_1, x_2, x_{2n}\}$.

Un tel graphe fournit des sommets $\{x_1, x_2\}$ de degrés $2n-1$, et présente une densité d donnée par:

$$d = 2\frac{\frac{n(n-1)}{2} + 2n}{2n\,(2n+1)} = \frac{n\,(n-1) + 4n}{2n\,(2n+1)}.$$

N.B. On constatera facilement qu'il est possible, en conservant le même principe de génération, de produire un graphe de densité plus grande encore en adjoignant

à un graphe de type rayons à n sommets comme ci-dessus, une clique de taille $2n$, puis $3n$, et ainsi de suite.

Pour autant, l'ensemble des classes de colorations propres sera toujours limité à une seule, ce qui rendra sa coloration triviale pour notre algorithme, en dépit de valeurs de paramètres souvent vécus comme pénalisants.

Par contre, si la coloration est très simple de mise en oeuvre, la détermination des cliques maximales d'un tel graphe sera alourdie en termes temporels dans le cadre des algorithmes mis en oeuvre. Les théorèmes du No Free Lunch nous y avaient préparé, par principe. Toutefois, il est possible de développer des outils d'aide à la détection de grosses cliques. De nombreuses références abordent les questions de cliques maximales; voir par exemple [B83], [GM85] et [AYZ97] mais aussi [K72].

Nous lions ci-dessous la détection de grandes cliques à la recherche de motifs en triangles rectangles constitués de 1 dans la matrice triangulaire haute $\mathcal{G}$, associée au graphe G; nous fournissons seulement l'énoncé du résultat essentiel connexe.

Proposition 196 *Sous les notations précédentes, l'existence d'une clique de cardinal* 3*, ici* $\{x_i, x_{i+1}, x_{i+2}\}$ *pour un indice* i *convenable* ($i \leq n-2$)*, est caractérisée par la présence d'un triangle rectangle-isocèle constitué de* 1*, de côté* $|\{x_i, x_{i+1}, x_{i+2}\}|-1$*, dans la matrice* $\mathcal{G}$ *associée au graphe* G*.*

La preuve est évidente et le résultat se généralise à des ordres de cliques supérieurs d'une part; d'autre part il se prolonge au cas des 2-cliques représentées par un $2-1$ triangle d'éléments 1 dans , autrement dit par un 1... comme on pouvait s'y attendre ! Il peut enfin se prolonger aux cas de sommets non contigüs en faisant "abstraction" des portions de colonnes ou lignes séparatrices constituées d'éléments nuls.

Prolongement 197 *Il peut être intéressant de se doter d'outils de détection de grandes cliques qui viendront compléter la recherche de la décomposition d'un graphe*

en cliques maximales. Il conviendra pour autant de ne pas se faire trop d'illusions quant au caractère universel de leur pertinence... Il existe des travaux qui fournissent de tels outils; le fait de les infléchir par la prise en compte de l'ensemble du corpus lié à la décomposition en cliques maximales devrait pouvoir apporter un plus.

4.3 Invariance par permutations des classes de colorations propres

4.3.1 Résultat général

Les algorithmes développés au chapitre 3 permettent de mettre en évidence, pour tout graphe G, l'existence d'une partition des colorations propres en un certain nombre de classes, décrites en amont sous la forme de colonnes dont celle de numéro j sera notée $(int\{i,j\})_i$. On rappelle que pour tout sommet $som(i)$ de la suite des sommets du graphe, ordonnée par la création de la suite constructive choisie pour G, la cellule $int\{i,j\}$ représente:

- en général l'ensemble, éventuellement vide, des sommets antérieurs dans la suite des sommets, dont $som(i)$ ne doit pas porter la couleur pour assurer l'injectivité de la coloration sur les cliques maximales prises en compte à ce stade;
- en particulier, la couleur de l'un des sommets antérieurs s_a (le premier s'il en existe plusieurs de ce type): on écrira alors $int\{i,j\} = [-s_a]$.

Ainsi la transposée de la colonne $(int\{i,j\})_i$ aura en général la forme suivante:

som	$som(1)$	$som(2)$	$\cdots$	$som(i)$
${}^t((int\{i,j\})_i)$	$[] = \varnothing$	$[som(1)]$	$\cdots$	$[s_{i,1}, ..., s_{i,\alpha_i}]$ ou $[-s_{a_i}]$

$\cdots$	$som(n)$
$\cdots$	$[s_{n,1}, ..., s_{n,\alpha_n}]$ ou $[-s_{a_n}]$

Nous notons $f_j : \{x_1, ..., x_n\} \rightarrow \{y_1, ..., y_k\}$ une coloration quelconque de la classe $(int\{i,j\})_i$, utilisant k couleurs (avec $k \geq \chi$ en général).

Proposition 198 *Détermination d'un élément f_j d'une classe de coloration*

Une coloration f_j, élément de la $j^{ième}$ classe de colorations propres, est définie pour tout élément de $\{x_1, ..., x_n\}$, par la détermination de l'image des sommets dans l'ordre produit par la suite constructive retenue pour G. Pour i variant de 1 jusqu'à n, on choisit $f_j(som(i))$ comme suit.

- *Si $int\{i,j\} = [s_{i,1}, ..., s_{i,\alpha_i}]$ alors $f_j(som(i)) \in \{y_1, ..., y_k\} - f_j(\{s_{i,1}, ..., s_{i,\alpha_i}\})$;*
- *si $int\{i,j\} = [-s_{a_i}]$ alors $f_j(som(i)) = f_j(s_{a_i})$.*

Preuve. Elle est immédiate, en raison de la signification accordée à l'objet $(int\{i,j\})_i$. ■

Corollaire 199 *On peut déduire de la proposition précédente, par permutations sur les valeurs prises par f_j, les autres colorations propres de la même classe, et ainsi les obtenir toutes.*

N.B. On notera toutefois, que les permutations admissibles devront respecter la définition de la classe, et ne pourront pas s'exprimer simplement en termes de permutations de $\{y_1, ..., y_k\}$. Il semble qu'on définisse ici une sorte de permutations sous contraintes... Les pemutations licites sont celles qui respectent la géométrie algébrique du graphe

Remarque 200 *On mesure mieux, une fois saisis ces quelques éléments explicatifs du pourquoi de la validité des colorations, que tomber "au hasard" ou par un critère non caractéristique sur une solution, relève de l'évènement heureux !*

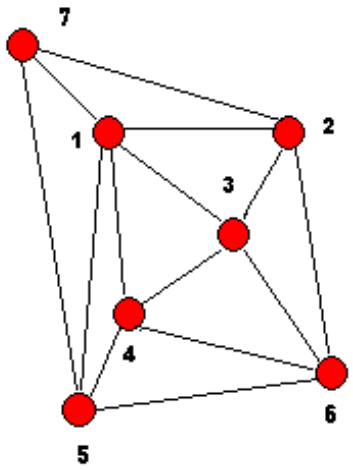

Figure 4.4: Rappel de $grn7.0$

4.3.2 Retour sur un exemple du chapitre 3

Nous réutilisons l'exemple de la coloration du graphe $grn7.0$, rappelé ici en figure 4.4, utilisé déjà dans la section 3.3.5 du chapitre 3, sous l'étude de notre algorithme principal. Nous en rappelons les résultats majeurs, obtenus lors de la détermination de l'ensemble des classes de colorations, pour un nombre formel k de couleurs disponibles, avec $k \geq \chi$.

Lors de l'avant-dernière étape, celle de la coloration du sommet $som(6) = 5$, relative au sous-graphe G de $grn7.0$ dont la décomposition en cliques maximales est donnée par:

$$\mathcal{D}(G) = \{\{1,3,4\},\{1,2,3\},\{2,3,6\},\{3,4,6\},\{1,4,5\},\{4,5,6\}\}$$

nous obtenions la table des classes de colorations propres suivantes:

$som(i)$	$int\{i,1\}$	$int\{i,2\}$	$int\{i,3\}$	$int\{i,4\}$
1	[]	[]	[]	[]
3	[1]	[1]	[1]	[1]
4	[3, 1]	[3, 1]	[3, 1]	[3, 1]
2	[4, 3, 1]	[4, 3, 1]	[−4]	[−4]
6	[2, 4, 3, 1]	[−1]	[4, 3, 1]	[−1]
5	[6, 4, 1]	[4, 1]	[6, 4, 1]	[4, 1]

Elles devenaient lors de l'ultime étape, après ajout des cliques maximales $\{1, 5, 7\}$ et $\{1, 2, 7\}$, l'ensemble des classes de coloration du graphe total $grn7.0$ fourni par:

$som(i)$	$int\{i,1\}$	$int\{i,2\}$	$int\{i,3\}$	$int\{i,4\}$	$int\{i,5\}$	$int\{i,6\}$
1	[]	[]	[]	[]	[]	[]
3	[1]	[1]	[1]	[1]	[1]	[1]
4	[3, 1]	[3, 1]	[3, 1]	[3, 1]	[3, 1]	[3, 1]
2	[4, 3, 1]	[4, 3, 1]	[4, 3, 1]	[4, 3, 1]	[−4]	[−4]
6	[2, 4, 3, 1]	[2, 4, 3, 1]	[−1]	[−1]	[4, 3, 1]	[−1]
5	[6, 2, 4, 1]	[−2]	[2, 4, 1]	[−2]	[6, 4, 1]	[4, 1]
7	[5, 2, 1]	[2, 1]	[5, 2, 1]	[2, 1]	[5, 4, 1]	[5, 4, 1]

Il serait simple de générer l'ensemble des colorations de chacune des classes produites ci-dessus. Par contre l'intérêt de cette génération semble bien faible, puisque nous bénéficions dans les tables produites, d'une version classifiée de l'ensemble des solutions dont l'explosion combinatoire fait la désespérance de la communauté ! Le rappel de ces résultats présente à nos yeux un intérêt autrement conséquent, dont nous esquissons le descriptif au paragraphe suivant.

4.3.3 Propriétés formelles des colorations propres d'un graphe et prolongements

L'avant-dernière table relative aux colorations propres du sous-graphe G engendré par les cliques de la décomposition de $grn7.0$ hormis $\{1,5,7\}$ et $\{1,2,7\}$, permet de démontrer qu'il existe une classe de colorations propres à 3 couleurs (la seule d'ailleurs, venue de la colonne $int\{i,4\}$) et que par conséquent, le nombre chromatique de G est $\chi_G = 3$. Par contre, nous pouvons facilement établir que pour $r = 3$ couleurs, les sommets 1, 2 et 5 sont de couleurs distinctes dans "toute" coloration propre de G. Par suite, lors de l'ajout des deux dernières cliques maximales qui permettent d'atteindre le graphe total, nous sommes certains qu'il ne peut exister de 3-coloration du graphe $grn7.0$; c'est à ce stade du rajout de ces deux dernières cliques que se produit le blocage qui aboutit à la non-existence d'une 3-coloration pour le graphe total.

Prolongement 201 *Obtention de propriétés formelles des colorations propres d'un graphe*

Il nous paraît fécond, à partir des classes de colorations propres d'un graphe et/ou de ses sous-graphes apparus lors du processus de génération des classes, d'étudier certaines propriétés formelles des colorations propres, de confirmer des conjectures, de rechercher des contre-exemples, des propriétés positives ou négatives relatives à des similitudes ou des différences de couleurs, de déterminer les zones de blocages qui font l'objet de nombreux travaux actuels dont on a l'intuition claire qu'elles jouent un rôle crucial en coloration.

4.4 Dénombrement des colorations propres d'un graphe

Comme vu dans la section précédente, notre algorithme général, mais aussi sa version dégradée limitée à un nombre fixé r de couleurs disponibles, nous fournit une parti-

tion de l'ensemble des colorations propres. Par là, il est facile d'obtenir une expression du polynôme chromatique du graphe étudié; on peut en déduire l'évolution du nombre de colorations propres lorsque r varie, qui permet in fine de retrouver l'expression formelle du polynôme chromatique du graphe.

Sous les notations antérieures, si nous considérons l'ensemble des classes de colorations propres d'un graphe donné, nous savons que nous pouvons traiter du polynôme chromatique formel ou de son expression limitée à un nombre donné r de couleurs disponibles.

Ces questions ont été l'objet de nombreux travaux, dont nous nous plaisons à rappeler l'un des premiers, tout à fait remarquable, venu du célèbre algébriste Birkhoff; voir par exemple [B67].

4.4.1 Ecriture formelle du polynôme chromatique

Si l'ensemble des classes de colorations propres du graphe G est représenté par un certain nombre de colonnes $(int\{i,j\})_i$ il est fort simple de fournir le dénombrement des colorations de cette classe de numéro j.

Proposition 202 *Si la classe de numéro j des colorations propres $(int\{i,j\})_i$ est définie par les tables suivantes:*

som	$som(1)$	$som(2)$	$\cdots$	$som(i)$
${}^t((int\{i,j\})_i)$	$[] = \varnothing$	$[som(1)]$	$\cdots$	$[s_{i,1}, ..., s_{i,\alpha_i}]$ *ou* $[-s_{a_i}]$

$\cdots$	$som(n)$
$\cdots$	$[s_{n,1}, ..., s_{n,\alpha_n}]$ *ou* $[-s_{a_n}]$

il existe dans cette classe représentée par $(int\{i,j\})_i$ et confondue avec elle, un nombre de colorations propres distinctes donné par le produit des valeurs disponibles pour la coloration de chacun des sommets $som(i)$; de façon plus précise, le cardinal

de la classe de numéro j est donné par:

$$|(int\{i,j\})_i| = (k-0)\,(k-1)\,...\times \begin{matrix}(k-\alpha_i)\\ ou\\ 1\end{matrix} \times ... \begin{matrix}(k-\alpha_n)\\ ou\\ 1\end{matrix},$$

selon que les sommets [som(i) et som(n)] respectivement sont interdits de [α_i et α_n] couleurs respectivement ou sont de la même couleur que [s_{a_i} et s_{a_n}] respectivement.

Preuve. Elle est immédiate; ce sont des dénombrements du type de celui des injections d'un ensemble dans un autre, faisant intervenir d'éventuelles exceptions pour lesquelles une couleur fixée ne laisse qu'un choix possible. ■

Proposition 203 *Le polynôme chromatique du graphe est défini par la somme des polynômes élémentaires issus de la proposition précédente.*

Preuve. Elle repose sur le fait qu'on dispose d'une partition des colorations propres. ■

Remarque 204 *Pour procéder à l'évaluation du polynôme chromatique $p(k)$ pour une valeur particulière entière de k, nous conviendrons que les produits apportés par les différentes classes sont positifs ou nuls, comme le nécessite leur interprétation; en aucun cas une classe ne peut apporter un nombre négatif de colorations propres... Ainsi, tout facteur $(k-\alpha_i)$ qui prendrait une valeur négative lors de son évaluation, sera forcé à 0; en effet, la signification des colonnes $(int\{i,j\})_i$ exprime clairement qu'une telle classe n'existe pas sous l'hypothèse d'un choix de k qui rendrait $(k-\alpha_i)$ négatif.*

Remarque 205 *Connaissant le degré n de $p(k)$, connaissant l'évaluation de $p(k)$ pour $n+1$ valeurs entières distinctes k, nous pourrons ainsi démontrer que le*

polynôme déterminé est convenable puisque son degré est n donc caractérisé par $n+1$ composantes dans la base canonique.

4.4.2 Polynôme chromatique relatif à r couleurs disponibles au plus

Si on étudie seulement les classes de colorations propres pour un nombre maximal r de couleurs disponibles, avec $r \geq \chi$ évidemment, nous déterminerons un sous-ensemble de l'ensemble total des classes de colorations ayant conduit à l'expression formelle du polynôme chromatique, comme on l'a vu dans le chapitre 3 lors de l'étude de l'algorithme dérivé de l'algorithme principal. Le mode de génération même de ces sous-classes conduit à la proposition suivante et la prouve.

Proposition 206 *Etant donné un graphe G et son polynôme chromatique connu par son écriture formelle $p(k)$ issue de la mise en oeuvre de l'algorithme principal de détermination de l'ensemble des classes de colorations, on peut déterminer grâce à l'algorithme dérivé l'expression formelle $p(r,k)$ du nombre des colorations propres du graphe pour tout $k \leq r$. Alors $p(r,k)$ est obtenu, en supprimant dans l'écriture de $p(k)$ tous les produits comportant des facteurs $(k-\alpha_i)$ avec $\alpha_i \geq r$.*

Remarque 207 *Nous disposons ici d'un résultat qui montre que l'expression formelle globale du polynôme chromatique du graphe est en quelque sorte un "tuilage" des expressions $p(r,k)$ successives, lorsque r augmente.*

Exemple 208 *Cas du graphe grn*7.0

- *Nous déduisons de la détermination de l'ensemble des classes de colorations propres du graphe grn*7.0*, l'expression formelle de son polynôme chromatique,*

fourni ici en respectant l'ordre des classes produites. Ainsi $p(k)$ *est donné par:*

$$\begin{aligned} p(k) &= k\,(k-1)\,(k-2) \\ &\quad \times \left[\begin{array}{c} (k-3)^2\,(k-4)^2 + (k-2)\,(k-3)\,(k-4) + (k-3)^3 + \\ (k-2)\,(k-3) + (k-3)^3 + (k-2)\,(k-3) \end{array} \right]. \end{aligned}$$

Nous voyons clairement que $\chi = 4$*, en raison du caractère factorisable de* $(k-3)$ *dans l'écriture de* $p(k)$*.*

- *Par ailleurs, nous pourrons vérifier en utilisant l'algorithme dérivé, ce qui corrobore la proposition précédente, que:*

$$\begin{aligned} p(5,k) &= k\,(k-1)\,(k-2) \\ &\quad \times \left[\begin{array}{c} (k-3)^2\,(k-4)^2 + (k-2)\,(k-3)\,(k-4) + (k-3)^3 + \\ (k-2)\,(k-3) + (k-3)^3 + (k-2)\,(k-3) \end{array} \right] \\ &= p(k), \end{aligned}$$

tandis que $p(3,k)$ *fournit une expression vide confirmant la non-existence de* 3*-coloration pour le graphe considéré alors que* $p(4,k)$ *fournit le nombre des* 4*-colorations.*

- *D'autres exemples simples feront apparaître des situations notoirement différentes quant aux expressions successives des* $p(r,k)$ *lorsque* r *croît, comparées à* $p(k)$*. Il est possible de s'intéresser par exemple au sous-graphe* G *du graphe* $grn7.0$*. On étudiera par exemple son polynôme chromatique* $p_G(k)$*, mais aussi ses sous-polynômes chromatiques* $p_G(3,k)$*,* $p_G(4,k)$ *et* $p_G(5,k)$*.*

4.4.3 Remarques et prolongements

Nous noterons que dans l'étude du polynôme chromatique du graphe $grn7.0$, deux classes fournissent un même apport $(k-3)^3$ au polynôme chromatique, mettant en

évidence et confirmant les rôles symétriques joués par les sommets 2 et 6. Il n'est pas impossible que l'arithmétique générale du polynôme chromatique, plus que ses détails, expriment la présence de symétries pertinentes.

L'analyse de cas plus complexes, laisse songeur quant à la structuration des situations par la prise en considération première du polynôme chromatique. En effet comme montré ci-dessus, mais surtout dans des exemples plus élaborés, le polynôme chromatique semble être affecté en qualité d'outil de projection, hélas non orthogonale... Ainsi des situations de nature extrêmement différentes structurellement par les propriétés du graphe sous-jacent considéré, mais aussi par la zone de graphe "engagée", fournissent pourtant des effets de projection similaires en termes d'apports au polynôme chromatique. Des papiers assez nombreux étudient avec grand soin, les "effets" des racines du polynôme, leur positionnement géométrique, des classes de graphes dont les polynômes chromatiques sont comparables.

Ces études apportent sans doute des renseignements pertinents, mais nous doutons que ce qui nous apparaît comme des effets de projections - non orthogonales - puissent être explicatifs et parviennent à exprimer véritablement les causes des phénomènes constatés. Pour notre part, nous nous garderons de fournir des expressions développées des écritures des polynômes chromatiques tant l'écriture sous forme de sommes de produits nous semble signifiante, parce qu'elles conservent trace, peut-être "souvenance" de propriétés structurelles à ne détruire que le plus tard possible et sans doute jamais !

Prolongement 209 *Analyse de la forme des polynômes chromatiques*

Il nous semble intéressant d'étudier la possible signification de l'arithmétique des polynômes chromatiques, et spécialement de comparer les propriétés des classes qui fournissent le même type d'apport au polynôme. Sans doute serait-il pertinent d'étudier aussi la manière dont le nombre de colorations propres se décompose en

sommes, mais aussi le cas particulier des classes de colorations uniques. Nous avons pu constater des propriétés de divisibilité assez étonnantes lors des travaux préparatoires à l'élaboration des algorithmes généraux. On pourra revenir dans le chapitre 3, à la section qui a permis de faire émerger les méthodes principales.

4.5 A propos de la représentation des graphes

L'approche que nous avons choisie pour aborder la coloration des graphes a imposé, comme naturellement, de privilégier l'aspect algébrique de la question. En effet, les cliques maximales et leur ordonnancement grâce à la notion de suite constructive d'un graphe, montrent que cet aspect est premier; cette façon de faire permet d'échapper pour partie du moins, à une certaine explosion combinatoire, du moins de la limiter, en évacuant les permutations inutiles et trompeuses. Toutefois, en termes applicatifs, nous savons la pertinence de la représentation des graphes ainsi que sa nécessité.

4.5.1 Suites constructives pour la représentation

Nous fournissons un exemple d'un graphe G très simple pour lequel nous avons retenu une représentation issue d'un positionnement aléatoire des sommets. Il en découle la représentation de la figure 4.5.

Si nous utilisons nos outils de décomposition en cliques maximales, adjoints à la notion de suite constructive du graphe considéré, il est alors possible de donner sens à la représentation en privilégiant les entités responsables majeures au point de vue coloration. Il en découle le graphe de la figure 4.6, bien connu déjà, dans notre travail.

Au delà du caractère d'exemple qui ne saurait apporter une preuve générale, nous pensons qu'il est intéressant de développer de façon systématique des outils mis en

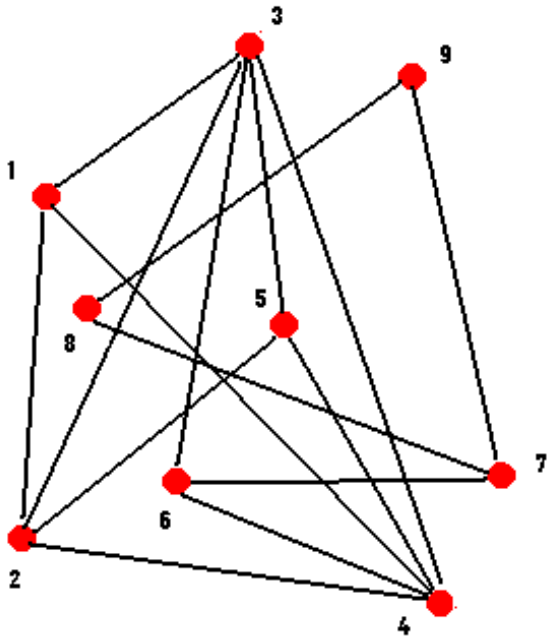

Figure 4.5: Graphe G : représentation aléatoire

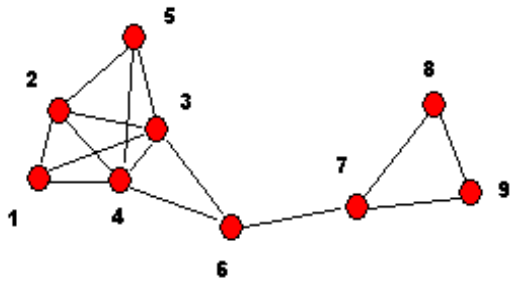

Figure 4.6: Graphe G et ses cliques maximales

oeuvre partiellement et de façon artisanale par nous-mêmes, au service seulement de diverses présentations de nos travaux.

Prolongement 210 *Représentation de graphes utilisant les suites constructives*

- *Développer le recours aux suites constructives pour une représentation des graphes plus signifiante et structurante, en termes de coloration.*
- *Mise en oeuvre systématique des outils de Matlab par exemple, avec possibilité de déformations des images représentatives de graphes et déplacements automatisés des sommets et des arêtes connexes.*
- *Développement d'outils complémentaires permettant des études locales au voisinage de tels ou tels sommets, dans le cas ordinaire des graphes complexes.*

4.5.2 A PROPOS DU THÉORÈME DES QUATRE COULEURS POUR UN GRAPHE FINI

Nous utilisons ci-dessous les points de départ de la présentation du théorème des quatre couleurs par G Gonthier fournie par [wG10]; en particulier, nous remarquons que l'on peut ramener le problème de coloration des cartes à celui des graphes connexes triangulés dont chaque face est exactement délimitée par trois routes. Soit G_T un tel graphe.

Proposition 211 *Tout graphe G_T admet pour décomposition en cliques maximales un ensemble de 3-cliques et/ou de 4-cliques.*

Preuve. Vu le caractère triangulé de G_T on sait que sa décomposition en cliques maximales ne contient pas de cliques d'ordre inférieur ou égal à 2. Par ailleurs, on sait que toute clique K_5 ne peut pas être représentée sous forme planaire, par caractérisation de Kuratovski ou Wagner - voir par exemple [wW10] -. D'où le résultat. ■

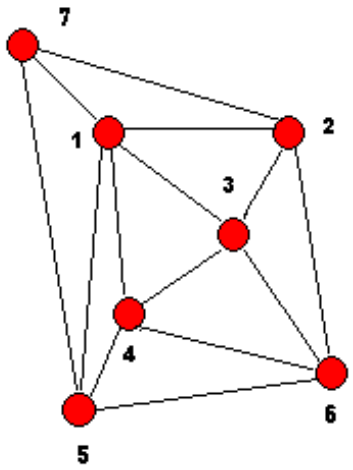

*Figure 4.7: Graphe grn*7.0

Assemblages de 3-cliques: exemple

Proposition 212 *Le graphe grn*7.0*, utilisé dans la section 4.3.2 et donné par la représentation planaire de la figure 4.7,*

est 4*-colorable, mais non* 3*-colorable.*

N.B. La preuve a été établie dans la section précitée.

Assemblages de 4-cliques: exemple

Considérons des 4-cliques figurant dans la décomposition en cliques maximales d'un graphe planaire. Leurs intersections sont des cliques d'ordre au plus 3, puisqu'elles sont distinctes. Nous nous limitons ci-dessous, aux intersections de 4-cliques formées de 3-cliques.

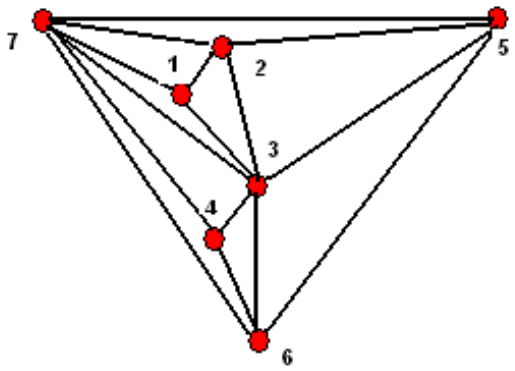

*Figure 4.8: Graphe grn*7.6

S'il existe une représentation planaire pour une 4-clique, nécessairement cette 4-clique est une pyramide de base triangulaire, de sommet un point de l'espace représenté par un point intérieur à sa base.

Proposition 213 *Le graphe grn*7.6*, donné par la représentation planaire de la figure 4.8,*

est 4*-colorable, évidemment non* 3*-colorable.*

Preuve. Le graphe *grn*7.6 admet pour décomposition en cliques maximales $\mathcal{D}(grn7.6)$, écrite ci-dessous dans l'ordre d'une suite constructive de ce graphe:

$$\mathcal{D}(grn7.6) = \{\{2,3,5,7\},\{3,5,6,7\},\{3,4,6,7\},\{1,2,3,7\}\}.$$

Vu la nature des intersections des différentes cliques avec le graphe engendré par les précédentes, - toujours constituées de 3-cliques -, on en déduit qu'il existe une

classe de colorations propres unique et que le graphe est certainement 4-colorable mais non 3-colorable.

N.B. Nous remarquons que nous disposons via l'exemple étudié ci-dessus, d'une procédure de construction récursive des assemblages des graphes dont la décompositon est un ensemble de 4-cliques lorsque leurs intersections sont des 3-cliques uniquement. Ce cas n'est pas le seul à traiter bien sûr puisque les intersections peuvent a priori, être constituées de 2-cliques, voire de 1-cliques. ■

Prolongement

Prolongement 214 *Sachant que tout graphe planaire G_T est décomposable seulement en un ensemble de 3-cliques et/ou 4-cliques, nous pensons qu'il peut être intéressant de revisiter le théorème des quatre couleurs grâce à l'étude des assemblages des cliques maximales d'une suite constructive du graphe planaire donné, en privilégiant ainsi l'approche algébrique sous-jacente, véritable cause à nos yeux des configurations géométriques qui en découlent.*

4.6 Correspondance sous-graphes - colorations propres

4.6.1 Notations

On considère un graphe simple, non orienté $G = (X, E)$, son ensemble de sommets $X = \{x_1, ..., x_n\}$, son ensemble d'arêtes E, supposé non vide en général afin d'éviter le cas de peu d'intérêt des sommets isolés, donc des 1-cliques maximales. On note $\mathcal{D}(G)$ la décomposition de G en cliques maximales, et $(c_j)_{1 \leq j \leq N}$ une suite constructive du graphe G, choisie une fois pour toutes.

- Pour tout j de $\{1, ..., N\}$, on note T_j la famille $\{c_1, ..., c_j\}$; par convention nous noterons $T_0 = \varnothing$. On note de surcroît $\mathcal{T}$ l'ensemble des familles T_j; ainsi $\mathcal{T} = \{T_j \ / \ j \in \{0, 1, ..., N\}\}$.

- Par ailleurs, on considère l'ensemble $\mathcal{F}$ des colorations de G propres ou non, c'est-à-dire l'ensemble des applications de $\{x_1, ..., x_n\}$ dans $\{y_1, ...y_k\}$, où k désigne un entier naturel donné, supposé supérieur au nombre chromatique χ de G pour éviter les situations triviales.

4.6.2 Treillis $\mathcal{T}$ et $\mathcal{P}(\mathcal{F})$

Treillis $\mathcal{T}$

Définition 215 *Ordre sur $\mathcal{T}$*

On dote $\mathcal{T}$ de l'inclusion qui en fait un ensemble ordonné.

Proposition 216 *$(\mathcal{T}, \subset)$ est un treillis. Plus précisément:*

$$T_s \wedge T_t = T_{\min(s,t)} \text{ et } T_s \vee T_t = T_{\max(s,t)}.$$

Preuve. Elle est évidente.

- Le fait que $\subset$ soit d'ordre sur $\mathcal{T}$, est bien connu.

  N.B. L'ordre est même total ici; cette propriété ne sera pas conservée dans des situations plus complexes, lorsque $\mathcal{T}$ ne sera plus constituée de chaînes ordonnées.

- Les bornes inférieure et supérieure dans $\mathcal{T}$ sont évidemment celles données ci-dessus.

■

Etude de $\mathcal{F}$

Proposition 217 *Soit f un élément de $\mathcal{F}$.*

Alors il existe un unique entier naturel maximal j de $\{0, ..., N\}$ tel que la restriction de f à toutes les cliques de $\{c_1, ..., c_j\}$ soit injective.

Preuve. Considérons l'ensemble des entiers naturels t vérifiant $1 \leq t \leq N$ pour lesquels la restriction de f à c_t est non injective.

- Si cet ensemble est vide, alors on pose $j = N$ et cet entier j fournit le résultat attendu.
- Si non, la partie de $\mathbb{N}$ considérée admet un plus petit élément t_0, en raison du bon ordre. On pose alors $j = t_0 - 1$ et j élément de $\{0, ..., N-1\}$ répond.

■

Définition 218 *j-compatibilités*

Soit f un élément de $\mathcal{F}$ quelconque et j l'entier associé défini par la proposition précédente.

- *j est appelé le degré de compatibilité de f; on notera: $j = d_{comp}(f)$.*
- *On dira alors que f est au plus j-compatible et on notera $\mathcal{F}_j$ l'ensemble des colorations de G, au plus j-compatibles.*
- *Si j est non nul, pour tout t vérifiant $1 \leq t \leq j$, on dira que f est t-compatible.*

  N.B. On remarquera que $\mathcal{F}_N$ représente l'ensemble des colorations propres de G, en raison du théorème de caractérisation des colorations propres établi au chapitre 2. En effet une coloration de G est propre si et seulement si la restriction de f à chacune des cliques maximales de sa décomposition est injective.

Proposition 219 *La famille $\{\mathcal{F}_j\}_{0 \leq j \leq N}$ de parties de $\mathcal{F}$ en constitue une partition.*

Preuve. Elle est immédiate. ■

Treillis $(\mathcal{P}(\mathcal{F}), \subset)$

Proposition 220 *L'ensemble des parties de $\mathcal{F}$, doté de l'inclusion, a une structure de treillis.*

Preuve. Le résultat est bien connu. ■

4.6.3 Correspondance de Galois entre $(\mathcal{T}, \subset)$ et $(\mathcal{P}(\mathcal{F}), \subset)$

Nous renvoyons pour ces notions à la bible des treillis de concepts [GW99], mais aussi à une contribution tout à fait novatrice et prometteuse [VS05].

Définition de la correspondance

Nous commençons en rappelant une propriété indispensable à ce stade, relative aux suites constructives d'un graphe. Soit $(c_j)_{1 \leq jN}$ une suite constructive de G.

Pour tout j de $\{1, ..., N-1\}$, on considère le graphe engendré par la famille $T_{j+1} = \{c_1, ..., c_{j+1}\}$. L'étude des suites constructives au chapitre 2, a montré qu'on se trouvera dans l'une des deux situations suivantes.

- Ou bien le graphe engendré par $T_j = \{c_1, ..., c_j\}$ est le même que celui engendré par T_{j+1}; autrement dit T_j ne suffit pas à "décrire totalement" le graphe qu'il engendre. Dans ce cas donc, il existe un plus grand entier naturel, noté r_j parmi les s tels que le graphe engendré par $T_{j+s} = \{c_1, ..., c_{j+s}\}$ soit le même que celui engendré par T_j.

- Ou bien le graphe engendré par $T_j = \{c_1, ..., c_j\}$ n'est pas le même que celui engendré par T_{j+1}. Dans ces conditions T_j suffit à "décrire totalement" le graphe qu'il engendre. Nous poserons alors $r_j = 0$.

Reste le cas particulier où $j = 0$. Le graphe engendré par T_j est alors $(X, \varnothing)$. Ici encore $T_j = T_0$ suffit à décrire totalement le graphe qu'il engendre; nous poserons donc encore $r_0 = 0$.

Définition 221 *Considérons l'application Φ de $\mathcal{T}$ dans $\mathcal{P}(\mathcal{F})$ définie comme suit.*

A tout ensemble de cliques maximales $T_j = \{c_1, ..., c_j\}$ (vide si $j = 0$), on associe le sous-graphe de G engendré par $\{c_1, ..., c_j\}$. Alors on définit $\Phi(T_j)$ comme l'ensemble des colorations de G propres pour le sous-graphe engendré par T_j.

Définition 222 *Puis on considère l'application Ψ de $\mathcal{P}(\mathcal{F})$ dans $\mathcal{T}$ définie comme suit.*

Pour toute partie F des colorations de G, on détermine le minimum j des degrés de compatibilité des éléments de F, conformément à la définition 218. Ainsi:

$$j = \min_{f \in F} \left(d_{comp}(f) \right).$$

- *Si $j \neq 0$, d'après le rappel effectué ci-dessus, nous savons qu'il existe un plus grand entier naturel $s(j)$ tel que $s(j) + r_{s(j)} \leq j$. Il correspond au graphe maximal pour lequel tout élément de F sera une coloration propre. Par suite*
  - *si $r_j = 0$ on aura $s(j) = j$.*
  - *si $r_j \neq 0$ alors $s(j)$ sera strictement inférieur à j puisque $\{c_1, ..., c_j\}$ doit être complétée pour représenter une décomposition en cliques maximales pour l'un des graphes considérés.*
- *Si $j = 0$, nous étendrons le cas précédent en posant $s(j) = s(0) = 0 = r_0$.*

Dans tous les cas nous définissons donc $\Psi(F)$ par:

$$\Psi(F) = T_{s(j)+r_{s(j)}} = \left\{ c_1, ..., c_{s(j)+r_{s(j)}} \right\}.$$

L'ensemble de cliques maximales $T_{s(j)+r_{s(j)}}$ représente la décomposition du plus grand graphe dont on est certain que F est un ensemble de colorations propres.

Théorème 223 *(de Galois)*

Le couple (Φ, Ψ) *est une correspondance de Galois entre* $(\mathcal{T}, \subset)$ *et* $(\mathcal{P}(\mathcal{F}), \subset)$.

N.B. On pourra voir cette correspondance comme celle des sous-graphes de G *- engendrés par les* T_j *- avec l'ensemble des parties des colorations du graphe total, plutôt que comme celle des ensembles de cliques générateurs de ces sous-graphes avec ces mêmes parties de l'ensemble des colorations de* G.

Preuve. Nous devons établir trois propriétés.

1. Décroissance de Φ

   Considérons deux éléments de $\mathcal{T}$, $T_j = \{c_1, ..., c_j\}$ et $T_t = \{c_1, ..., c_t\}$ tels que $T_j \subset T_t$, c'est-à-dire avec j et t éléments de $\{0, ..., N\}$ vérifiant $j \leq t$.

   - Si j est nul, le graphe engendré par la famille de cliques vide T_0 est défini par: $(X, \varnothing)$. Par suite toute coloration de G est propre pour ce graphe. Ainsi $\Phi(T_j) = \mathcal{F}$. Pour tout t, on a certainement

   $$\Phi(\{c_1, ..., c_j\}) = \mathcal{F} \supset \Phi(\{c_1, ..., c_t\}),$$

   d'où l'inégalité cherchée dans ce cas.

   - Si non, ni j ni t ne sont nuls. Vu la définition des suites constructives, on sait que le graphe G_1 engendré par $\{c_1, ..., c_j\}$ est un sous-graphe du graphe G_2 engendré par $\{c_1, ..., c_t\}$, possiblement égal. Par suite la décomposition en cliques maximales de G_2 contient certainement la décomposition de G_1. Or pour être coloration propre d'un graphe, une coloration f doit avoir une restriction à chacune des cliques de sa décomposition injective. En conséquence toute décomposition propre pour G_2 l'est pour G_1. Par conséquent dans ce cas encore:

   $$\Phi(\{c_1, ..., c_j\}) \supset \Phi(\{c_1, ..., c_t\}).$$

2. Décroissance de Ψ

   Considérons deux parties F_1 et F_2 de $\mathcal{F}$ vérifiant $F_1 \subset F_2$. Vu l'inclusion on a certainement:

$$j1 = \min_{f \in F_1} \left(d_{comp}(f) \right) \geq j2 = \min_{f \in F_2} \left(d_{comp}(f) \right).$$

   Par suite nous avons: $\Psi\left(F_2\right) = T_{s(j2)+r_{s(j2)}} \subset T_{j2} \subset T_{j1}$. Mais $s(j2)$ est un entier qui vérifie $s(j2) + r_{s(j2)} \leq j2 \leq j1$; par conséquent il est certainement inférieur au plus grand qui vérifie cette inégalité, à savoir $s(j1) + r_{s(j1)}$.

   Ainsi: $s(j2) + r_{s(j2)} \leq s(j1) + r_{s(j1)}$, ce qui s'écrit:

$$\Psi\left(F_2\right) = T_{s(j2)+r_{s(j2)}} \subset T_{s(j1)+r_{s(j1)}} = \Psi\left(F_1\right)$$

   d'où la décroissance cherchée.

3. "Effet dilatant" de $\Psi \circ \Phi$ et $\Phi \circ \Psi$

   (a) Soit $T_j = \{c_1, ..., c_j\}$ un élément quelconque de $\mathcal{T}$.

   - Si $j = 0$ alors $\{c_1, ..., c_j\} = \varnothing$; le graphe engendré par un ensemble vide de cliques maximales du graphe étudié a pour ensemble d'arêtes $E = \varnothing$. Par conséquent on a $\Phi\left(\{c_1, ..., c_j\}\right) = \mathcal{F}$ car toute coloration de G est propre pour cet ensemble d'arêtes. Ainsi $j = \min_{f \in \mathcal{F}} \left(d_{comp}(f) \right) = 0$; ainsi $\Psi\left[\Phi\left(T_0\right)\right] = T_0 \supset T_0$.
   - Si $j \neq 0$ considérons le sous-graphe de G engendré par la famille $T_j = \{c_1, ..., c_j\}$. Sa décomposition en cliques maximales, d'après l'étude menée au chapitre 2, est exactement $\left\{c_1, ..., c_{j+r_j}\right\}$. Ainsi $\Phi\left(\{c_1, ..., c_j\}\right)$ est égal à l'ensemble des colorations de G dont la restriction à chacune des cliques c_1, ..., c_{j+r_j} est injective et pour

celles-ci seulement. Par suite :

$$\min_{f\in\Phi(\{c_1,...,c_j\})}(d_{comp}(f)) = j + r_j.$$

Ainsi $\Psi[\Phi(T_j)]$ désigne l'ensemble des cliques de la décomposition du graphe engendré par T_j c'est-à-dire T_{j+r_j}. Par conséquent:

$$\Psi[\Phi(T_j)] = T_{j+r_j} = \{c_1, ..., c_{j+r_j}\} \supset \{c_1, ..., c_j\} = T_j.$$

- Dans les deux cas, $\Psi \circ \Phi$ a un effet dilatant sur T_j.

(b) Soit F une partie quelconque de $\mathcal{F}$. Considérons $j = \min_{f\in F}(d_{comp}(f))$.

- Si $j = 0$, alors $\Psi(F) = T_0$. Par conséquent le graphe engendré par $\Psi(F)$ est vide et par suite $\Phi[\Psi(F)] = \mathcal{F}$; l'ensemble des colorations F vérifie assurément: $\Phi[\Psi(F)] = \mathcal{F} \supset F$.
- Si $j \neq 0$ alors F est un ensemble de colorations dont le degré de compatibilité minimum est j; par suite:

$$\Psi(F) = T_{s(j)+r_{s(j)}} \text{ où } s(j) + r_{s(j)} \leq j.$$

$\Phi[\Psi(F)]$ est l'ensemble de toutes les colorations de G qui sont propres sur le graphe engendré par $T_{s(j)+r_{s(j)}}$. Or, vu l'inégalité $s(j)+r_{s(j)} \leq j$, tout élément de F est une coloration qui respecte injectivement les cliques éléments de $T_j = \{c_1, ..., c_j\}$, donc toutes les précédentes et spécialement celles de $T_{s(j)+r_{s(j)}}$ qui est une décomposition complète. Par suite :

$$F \subset \Phi[\Psi(F)].$$

La preuve est terminée. ■

Remarque 224 *Le résultat établi ci-dessus est très important; il s'agit d'une propriété structurelle.*

- *En fait il n'est pas nouveau. Son contenu s'est déjà exprimé, spécialement dans les algorithmes de construction des colorations dont la restriction à l'ensemble des cliques maximales de G, est injective. Il en était le principe actif. Les algorithmes du chapitre 3 existent en raison de l'existence de cette correspondance de Galois.*

- *Cette correspondance est la marque de l'extrême structuration des colorations propres pour G et ses sous-graphes. L'intuition en était certaine, mais cette correspondance l'officialise, l'institutionnalise en quelque sorte...*

Recherche d'invariants de $\Psi \circ \Phi$

Nous savons que pour tout T_j du treillis $(\mathcal{T}, \subset)$ on a : $T_j \subset \Psi \circ \Phi\,(T_j)$; de même pour tout ensemble F de colorations de G, on a : $F \subset \Phi \circ \Psi(F)$. Nous allons nous intéresser à déterminer les invariants des transformations $\Phi \circ \Psi$ et $\Psi \circ \Phi$, composées des transformations duales Ψ et Φ. En cela nous revenons sur une question ouverte laissée sans réponse au chapitre 2, lorsque nous cherchions une caractérisation du fait qu'une partie des cliques maximales de la décomposition de G soit elle-même une décomposition d'un graphe intermédiaire.

Nous adoptons les notations utilisées partout dans cette section.

Proposition 225 *Soit T_j un élément de $\mathcal{T}$.*

Alors les deux propriétés suivantes sont équivalentes :

1. *T_j est un invariant de $\Psi \circ \Phi$*

2. *T_j est la décomposition complète d'un sous-graphe de G engendré par l'un des éléments de $\mathcal{T}$.*

Preuve. Nous établissons l'équivalence en deux temps.

1. Montrons que (2) $\Rightarrow$ (1)

   Supposons que T_j est la décomposition complète d'un sous-graphe de G engendré par l'un des éléments de $\mathcal{T}$. Alors nécessairement il s'en déduit que $T_j = T_{j+r_j}$ et donc $r_j = 0$. Par conséquent, $\Phi(T_j)$ est l'ensemble des colorations de G propres pour le graphe engendré par l'ensemble des cliques éléments de T_j. C'est donc l'ensemble des colorations dont la restriction aux cliques de T_j, et seulement celles-ci, sont injectives. En notant l l'entier défini par : $l = \min_{f \in \Phi(T_j)} (d_{comp}(f))$, il vient $l = j + r_j = j$. Par suite sous les notations antérieures :

   $$s(l) + r_{s(l)} = s(j) + r_{s(j)} \text{ égal à } j$$

   vu que T_j est une décomposition complète. Ainsi $\Psi(\Phi(T_j)) = T_j$.

   N.B. Les écritures données gardent sens lorsque $j = 0$.

2. Inversement montrons (1) $\Rightarrow$ (2)

   Considérons un élément T_j de $\mathcal{T}$ invariant par $\Psi \circ \Phi$. Si $j = 0$, alors T_0 est bien une décomposition complète du graphe $(X, \varnothing)$.

   Sinon, sous les notations antérieures, montrons que $r_j = 0$.

   Si tel n'est pas le cas $r_j > 0$ et le graphe engendré par $T_{j+r_j} = \{c_1, ..., c_{j+r_j}\}$ est égal à celui engendré par T_j. Par suite $l = \min_{f \in \Phi(T_j)} (d_{comp}(f)) = j + r_j$ puisque toutes les colorations de $\Phi(T_j)$ sont certainement injectives sur $c_1, .c_{j+r_j}$. Par suite $\Psi \circ \Phi(T_j) = T_{j+r_j} \neq T_j$, ce qui est exclu pour cause d'invariance de T_j.

■

Corollaire 226 *Soit T_j de $\mathcal{T}$ invariant par $\Psi \circ \Phi$.*

Alors $\Phi(T_j)$ est l'ensemble des colorations dont la restriction aux seuls éléments de T_j est injective, autrement dit propres pour le graphe totalement décomposé engendré par T_j.

Preuve. Le graphe engendré par T_j est totalement décomposé, et T_j en représente la décomposition. ■

Remarque 227 *Les résultats précédents répondent pour partie à la question ressentie comme non triviale au chapitre 2, de la caractérisation des ensembles de cliques maximales constituant une décomposition totale d'un graphe donné...*

Structures standards associées

De façon purement canonique, à partir de la correspondance de Galois liant $(\mathcal{T}, \subset)$ et $(\mathcal{P}(\mathcal{F}), \subset)$, nous disposons des résultats suivants. On pourra se reporter encore à [VS05] mais surtout à [GW99] pour obtenir le détail des preuves.

Proposition 228 *Existence d'un treillis de Galois associé*

Sous les notations précédentes on considère l'ensemble $\mathcal{H}$ défini comme suit :

$$\mathcal{H} = \{(T_j, F) \in \mathcal{T} \times \mathcal{P}(\mathcal{F}) \ / \ T_j = \Psi(F) \text{ et } F = \Phi(T_j)\}.$$

On dote $\mathcal{H}$ de l'ordre $\leq$ défini par :

$$[(T_j, F) \leq (T_s, F')] \Leftrightarrow [F \subset F'].$$

Alors $(\mathcal{H}, \leq)$ a une structure de treillis. Il est appelé treillis de Galois associé à la correspondance de Galois (Φ, Ψ).

Proposition 229 *Relations d'équivalence induites*

- *Sur $\mathcal{T}$ on définit la relation $\sim_{\mathcal{T}}$ par :*

$$\forall (T_j, T_s) \in \mathcal{T}^2 \quad [T_j \sim_{\mathcal{T}} T_s] \Leftrightarrow [\Phi(T_j) = \Phi(T_s)].$$

- *De même sur $\mathcal{P}(\mathcal{F})$ on définit $\sim_{\mathcal{P}(\mathcal{F})}$par :*

$$\forall (F, F') \in (\mathcal{P}(\mathcal{F}))^2 \quad \left[F \sim_{\mathcal{P}(\mathcal{F})} F'\right] \Leftrightarrow [\Psi(F) = \Psi(F')].$$

Alors les relations $\sim_{\mathcal{T}}$ et $\sim_{\mathcal{P}(\mathcal{F})}$sont d'équivalence sur $\mathcal{T}$ et $\mathcal{P}(\mathcal{F})$ respectivement.

N.B. On notera la signification locale des deux équivalences évoquées.

- *Deux ensembles de cliques maximales T_j et T_s sont équivalents au sens de $\sim_{\mathcal{T}}$ si et seulement si ils engendrent le même sous-graphe de G.*
- *Deux ensembles de colorations de G sont équivalents au sens de $\sim_{\mathcal{P}(\mathcal{F})}$ si et seulement si ils respectent par injectivité les mêmes cliques maximales.*

Proposition 230 *Dans $(\mathcal{H}, \leq)$ comme dans tout treillis de Galois, on a les propriétés suivantes.*

- *Pour tout T_j de $\mathcal{T}$, $\Psi \circ \Phi(T_j)$ est l'unique plus grand élément de la classe de T_j modulo $\sim_{\mathcal{T}}$ contenant T_j.*
- *De même pour tout F de $\mathcal{P}(\mathcal{F})$, $\Phi \circ \Psi(F)$ est l'unique plus grand élément de la classe de F modulo $\sim_{\mathcal{P}(\mathcal{F})}$ contenant F.*

Par suite les éléments de $\mathcal{H}$ sont constitués de couples de représentants des classes modulo $\sim_{\mathcal{T}}$ et $\sim_{\mathcal{P}(\mathcal{F})}$respectivement.

Prolongements

Nous pensons que la situation étudiée ci-dessus mérite extension et présente différents prolongements qui exprimeraient une structuration forte de cas plus généraux. Nous signalerons comme point essentiel le fait suivant, certainement générique.

Prolongement 231 *Etant donnés deux graphes G_1 et G_2 relatifs au même ensemble de sommets $X = \{x_1, ..., x_n\}$, il serait sans doute intéressant d'étudier les graphes $G_1 \cup G_2$ et $G_1 \cap G_2$ et de scruter comment les cliques maximales de G_1 et G_2 se*

prolongent en cliques maximales de la réunion et de l'intersection afin de préciser les propriétés des colorations propres associées. Une correspondance de Galois doit venir ici généraliser le cas antérieur.

Il nous reste à apporter maintenant une conclusion générale que nous lions à celle de l'ensemble de notre travail.

CONCLUSIONS - PERSPECTIVES

Dans le chapitre 1, nous avons présenté ce qui fut l'entrée dans ce travail de recherche, à savoir un état de l'art sur les théorèmes du No Free Lunch et leurs conséquences, grâce à l'étude du papier initial de DH Wolpert et WG Macready dans sa version améliorée de 1997 suivie de l'analyse des contributions de nombreuses équipes qui, dans le monde entier, ont réagi à cette manière nouvelle d'aborder les problèmes d'optimisation; il s'agit de réaliser une approche d'ensemble liant méthodes de résolution et fonctions à optimiser, de façon à interroger les liens intimes de ces deux composants. L'ensemble constitue un corpus conséquent que nous avons présenté à plusieurs reprises sous formes d'exposés longs, à l'occasion de divers congrès.

Dès lors nous sommes sortis convaincus de la nécessité d'une approche globale des problèmes d'optimisation qui privilégierait la recherche de propriétés générales, la prise en compte de symétries spécialement, dans les problèmes étudiés ainsi que la mise en oeuvre, dès la recherche des solutions, de méthodes qui auraient un effet structurant afin d'approcher les causes des phénomènes. Nous avons alors choisi de tenter d'exercer cette manière de faire dans le champ applicatif de la coloration de graphes en raison d'un attrait personnel pour ce domaine, en raison aussi des thématiques de notre équipe d'accueil et pour l'intérêt intrinsèque de cette question, dont sa capacité à être un traducteur de problèmes d'autres natures, ce qui à nos yeux révèle sa puissance et son intérêt original.

Nous rappelons maintenant les étapes essentielles de notre construction. Nous définissons d'abord la décomposition de tout graphe en cliques maximales; elles jouent pour le graphe considéré le rôle des nombres premiers pour les entiers naturels

et, en particulier, permettent de le reconstruire dans sa totalité par adjonctions de cliques successives, pourvu qu'on les ordonne de façon convenable; d'où la notion de suite constructive d'un graphe. Apparaît alors une correspondance de Galois entre les familles emboîtées de cliques maximales et les colorations propres des sous-graphes engendrés par ces familles. Cette correspondance est la cause profonde de la structuration des colorations propres des différents sous-graphes engendrés dans leur évolution croissante vers le graphe total. En particulier, il est possible d'en déduire une partition en classes de l'ensemble des colorations propres du graphe étudié; cette propriété cruciale permet de compter les colorations propres mais surtout d'établir une équivalence entre colorations; les colorations congrues se déduisent les unes des autres par des permutations complexes qu'il est toutefois possible d'expliciter.

Nous renvoyons au chapitre 4 pour la mise en correspondance des résultats essentiels obtenus et des prolongements espérés. Nous tenons toutefois à rappeler ou esquisser quelques pistes prometteuses qui semblent s'ouvrir à l'issue de notre recherche.

- Une part conséquente de nos algorithmes, dont la fonction centrale permettant de générer les classes de colorations propres d'un graphe simple et non orienté, est certainement parallélisable. Il conviendrait d'en assurer la mise en oeuvre.

- Il est crucial de poursuivre les investigations destinées à déterminer la véritable cause de la complexité de coloration d'un graphe donné, point sur lequel nous n'avons pas su conclure. Nous avons montré qu'un certain nombre de critères ordinairement utilisés pour ce faire ne sont pas satisfaisants; ils accompagnent ordinairement les situations complexes mais ne sont pas caractéristiques, ce qui conduit à des idées fausses; les contre-exemples abondent et peuvent en quelque sorte être construits à la demande, dès qu'on a saisi un peu mieux le

pourquoi des difficultés. Pourtant, nos méthodes mettent en évidence un marqueur causal, encore insuffisant, de la complexité de coloration mais quelques intuitions connexes permettent d'espérer.

- La "géométrie algébrique" de l'ensemble des cliques maximales du graphe est responsable de cette complexité mais il est indispensable de trouver un traducteur suffisamment simple de cette géométrie, des rapports des diverses cliques maximales entre elles. L'enjeu est important; il est lié au point signalé précédemment mais revêt un intérêt propre, encore plus essentiel.

- L'analyse du fonctionnement des grandes classes d'heuristiques apparaît comme un champ à explorer de façon approfondie. En effet, dans notre équipe mais aussi en dehors, des auteurs signalent fréquemment des phénomènes cycliques lors des recherches de solutions de divers problèmes de coloration, sous des méthodes fort diverses; ces phénomènes sont signalés assez souvent, comme un peu étranges sans donner lieu à étude plus systématique, parfois évoqués un peu par hasard, même. L'absence d'outil d'étude des causes des difficultés de coloration rendrait de toute façon fort aléatoire une quelconque interprétation. Il nous semble intéressant de procéder à l'analyse comparée de l'apparition de ces cycles avec les nombres de degré de liberté des sommets concernés dans les suites constructives des graphes étudiés.

- Nos méthodes fournissent l'ensemble des classes de colorations propres d'un graphe donné. Il serait d'un grand intérêt de travailler à déterminer aussi les impropres, ou plutôt de les classifier. Nous avons commencé à étudier cette question et disposons de résultats partiels non évoqués dans ce travail. Tout ne peut pas arriver ! En effet, lorsqu'on génère l'ensemble des colorations d'un graphe, certains nombres de défauts ne sont jamais présents; ce fait est en

lien encore avec la géométrie algébrique de l'ensemble des cliques maximales présentes dans la décomposition.

- La connaissance d'une partition des colorations propres de sous-graphes d'un graphe donné sous nos méthodes, permet comme évoqué au chapitre 4 de déterminer certaines propriétés formelles des colorations sous hypothèses complémentaires relatives, par exemple, au nombre de couleurs disponibles; en particulier, ces propriétés permettent de prévoir les "goulots d'étranglement", certaines zones responsables de fort conflits qui nécessitent l'ajout d'une couleur par exemple pour débloquer une situation. Il serait intéressant sans doute de confronter les travaux des équipes qui oeuvrent à l'étude de ces zones à risques avec les outils que nous avons développés.

- L'utilisation des suites constructives d'un graphe pour sa représentation globale s'il est suffisamment modeste, pour la représentation de certaines zones locales s'il est de taille ordinaire, mais aussi la nature des intersections de ses cliques maximales semble prometteuse. Nous avons expérimenté, sur des cas simples bien sûr, l'aide considérable à la lisibilité des situations qu'apportent le concept de suite constructive. Par ailleurs, si nous persistons dans l'idée que l'essentiel des questions est algébrique, il n'en demeure pas moins que ces lectures géométriques des situations ont un intérêt certain, ne serait-ce qu'en termes historiques. En particulier, le théorème des quatre couleurs pourrait fournir l'occasion d'une analyse complète et féconde des différents modes d'assemblages de cliques maximales d'ordre 3 ou 4, repésentables sous forme planaire. Les premiers résultats obtenus semblent encourageants.

- Une tâche plus abstraite consisterait à tenter d'étudier une sorte de morphisme entre la forme de l'ensemble des graphes et sous-graphes d'un graphe donné et celle de l'écriture de son polynôme chromatique avant évidemment d'en

casser la signification, par des développements calculatoires brutaux. Il n'est pas impossible qu'on puisse obtenir des résultats de divisibillté, de décompositions d'entiers grâce à ce type d'approche.

- Nous signalons enfin l'intérêt de généraliser la correspondance de Galois mise en évidence au chapitre 4. Il serait sans doute très fécond d'étudier de façon plus générale comment deux graphes G_1 et G_2 connus par leurs décompositions en cliques maximales et leurs suites constructives associées peuvent être réunis, par exemple. Qu'advient-il alors de la décomposition en cliques maximales de leur réunion mais aussi de leur intersection, comparées aux décompositions de chacune d'elles, qu'en est-il des rapports des suites constructives des unes et des autres, comment définir les colorations propres des unes et des autres et quels sont leurs rapports ?

  Nous avons conscience de n'avoir développé qu'une part infime d'une tâche autrement conséquente.

  Mais comme l'écrit Rimbaud: "... viendront d'autres horribles travailleurs; ils commenceront par les horizons où l'autre s'est affaissé !"

BIBLIOGRAPHIE

[A94] D'Arcy Thomson (1994) *Forme et croissance* Seuil, Paris.

[A98] M. Ascher (1998) *Mathématiques d'ailleurs* Seuil, Paris.

[AYZ97] N. Alon, R. Yuster, U. Zwick (1997) *Finding and counting given length cycles.* Algorithmica, vol 17, p. 209-223.

[B00] A. Bejan (2000) *Shape and Structure from Engineering to Nature.* Cambridge University Press.

[B67] G. Birkhoff (1967) *Lattice theory* American Mathematical Society. Colloquium Publications. Vol XXV, 3^{rd} ed.

[B74] N. Biggs (1974) *Algebraic Graph Theory* Cambridge Mathematical Theory

[B83] C. Berge (1983) *Graphes* Gauthiers-Villars (3°édition) Bordas, Paris.

[B86] R.E. Bryant (1986) *Graph-based algorithms for Boolean function manipulation.*IEEE Transactions on Computers C-35, 677-691

[BH00] M. Belaidouni & J.-K. Hao (1999) *Landscapes of the Maximal Constraint Satisfaction Problem* Lecture Notes in Computer Science 1829:244-255 Springer-Verlag.

[BH99(1)] M. Belaidouni & J.-K. Hao (1999) *An Analysis of the Configuration Space of the Maximal Constraint Satisfaction Problem* PPSN 2000, 49-58.

[BH99(2)] M. Belaidouni & J.-K. Hao (1999) *A measure of combinatorial landscape difficulty for Metropolis algorithm.* Journal of Combinatorial Optimization 3(4):379-397.

[BL46] G.D. Birkhoff, D. Lewis (1946) *Chromatic polynomials.* Trans. Amer. Math. Soc.60, 355-451.

[C92] J. Culberson (1992) *Iterated greedy graph coloring and the difficulty landscape.*Technical Report TR 92-07 University of Alberta. Dept. of Computing Science (1992).

[C99] J. Culberson (1999) *On the Futility of Blind Search.* Evolutionnary Computation, 6(2):109-127.

[D04] I. Devarenne (2004) *Etudes d'heuristiques à mémoires pour l'affectation de fréquences.* Mémoire de DEA en Informatique Automatique et Productique. Université de Technologie de Belfort-Montbéliard. Belfort, septembre 2004.

[D07] I. Devarenne (2007) *Etude en recherche locale adaptative pour l'optimisation combinatoire.* Mémoire de thèse de Doctorat en Informatique. Université de Technologie de Belfort-Montbéliard. Belfort. Novembre 2007.

[DH98] R. Dorne,J.-K. Hao (1998) *Tabu search for graph coloring, T-colorings and set T-colorings.*In S. Voss,S. Martello, I.H. Osman & C. Roucairol (ed) Meta-Heuristics:Advances and Trends in Local Search Paradigms for Optimization, Kluwer, 77-92.

[DJW02] S Droste, T Jansen, I Wegener (2002). *Optimization with Randomized Search Heuristics - The (A)NFL Theorem, RealisticScenarios, and Difficult Functions.* Theoritical Computer Science, 287(1): 131-144, september 2002.

[DJW99] S Droste, T Jansen, I Wegener (1999) *Perhaps Not a Free Lunch But At Least a Free Appetizer.* In W. Banzhaf, and alter, editors, Proceedings of the First Genetic and Evolutionary Computation Conference (GECCO 99), 833-839, San Francisco CA, 13-17. Morgan Kaufman Publishers.

[DMC06] I. Devarenne, H. Mabed, A. Caminada (2006) *Optimization by extension-restriction of neighborhood in local search: application to graph coloring problem.* In ESM'2006. 20th annual European Simulation and Modelling conference. Oct.2006. Toulouse (France).

[GJ79] M.R. Garey, D.S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Vol A1.2, W.H Freeman.

[GM85] M. Gondran, M. Minoux.(1985) *Graphes et algorithmes* Eyrolles Paris.

[GR91] D. Goldberg and M. Rudnick (1991). *Genetic algorithms and the variance of fitness.* Complex Systems, 5(3):265-278.

[GW99] B. Ganter, R. Wille *Formal Concept Analysis: Logical Foundations.* Springer-Verlag, Berlin, 1999.

[H96] D.S. Hochbaum (1996). *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA.

[HH04] J.-P. Hamiez and J.-K. Hao (2004) *An analysis of solutions properties of the graph coloring problem.* in M.G.C Resende & J. Pinho de Sousa (ed) Metaheuristics:Computer Decision-Making, Vol. 86 of Applied Optimization, Kluwer, 325-346.

[HR89] T. Hagerup and C.R. Rüb (1989). *A guided tour of Chernoff bounds.* Information Processing Letters 33,305-308.

[HRW99(1)] R. Heckendorn, S. Rana and D. Witley.(1999) *Polynomial time summary statistics for a generalisation of MAXSAT.* Gecco-99, Morgan Kaufman.

[HRW99(2)] R. Heckendorn, S. Rana and D. Witley.(1999) *Tests Function Generators as Embedded Landscapes.* In Foundations of Genetic Algorithms FOGA-5 Morgan Kaufman.

[HW87] A. Hertz, D. de Werra (1987) *Using tabu search techniques for graph coloring.* Computing 39 (1987), 345-351.

[IT01] C. Igel and M. Toussaint.(2001) *On Classes of Functions for which No Free Lunch Results Hold.* Technical report, Institut für Neuroinformatik, Ruhr-Universität Bochum, ND 04 44780 Bochum - Germany.

[JH06] I. Juhos, J.I. van Hemert (2006). *Improving Graph Colouring Algorithms and Heuristics Using a Novel Representation* EvoCOP, pp123-134 DBLP, http://dblp;uni-trier.de

[JTH04] I. Juhos, A. Toth, J.I. van Hemert (2004). *Binary Merge Model Representation of the Graph Colouring Problem* EvoCOP, pp124-134 DBLP, http://dblp;uni-trier.de

[JT96] D. Johnson, M. Trick (1996) *Cliques, Coloring and Satisfiability.* American Mathematical Society, Dimacs.

[K72] R.M. Karp (1972) *Reducibility Among Combinatorial Problems.* Proceedings of a Symposium on the Complexity of Computer Computations. Plenum Press.

[L04] S. Lang (2004) *Algèbre* 3° édition Dunod, Paris.

[LV97] M. Li and P.M.B. Vitanyi. (1997) *An Introduction to Kolmogorov Complexity and its Applications*,Second Edition. Springer-Verlag, New York.

[MB71] S. Mac Lane,G. Birkhoff (1971) *Algèbre tome 1 et 2* Gauthiers-Villars, Paris.

[MW96] W.G. Macready and D.H. Wolpert.(1996) *What makes an optimization problem hard?* Complexity, vol 5, pp 40-46.

[P94] C.H. Papadimitriou (1994) *Computational Complexity.* Dover Publications, Englewood Cliffs, NJ.

[PS98] C.H Papadimitriou and K. Steiglitz (1998) *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, Englewood Cliffs, NJ.

[PZ10] A. Pêcher, X. Zhu (2010) *Claw-free circular-perfect graphs,* Journal of Graph Theory (65), pages 163-172, 2010

[R97] G. Rudolph (1997): *Convergence Properties of Evolutionary Algorithms.* Ph.D. Thesis, Verlag Dr Kovač].

[RRS95] Y. Rabani, Y. Rabinovich, and A. Sainclair (1995) *A computational view of population genetics.* Random Structures and Algorithms 12(4), 314-334.

[RS95] N.J. Radcliffe and P.D. Surry.(1995) *Fundamental limitations on search algorithms: Evolutionary computing in perspective.* In J. van Leeuwen, editor, Lecture Notes in Computer Science 1000. Springer-Verlag.

[S71] P. Samuel (1971) *Théorie algébrique des nombres.* Hermann, Paris.

[S98] R.S. Stankovic (1998) *Some remarks on terminology in spectral techniques for logic design: Walsh transform and Hadamard matrices.* IEEE trans.comput.-aided des. integr.circuits syst. 1998, vol. 17, n°11, pp 1211-1214.

[S99] U. Schöning (1999). *A probabilistic algorithm for k-SAT and constraint satisfaction problems.* In Proceeding of the 40 th Annual IEEE Symposium on Foundations of Computer Science (FOCS 99), 410-414. IEEE Press, Piscataway, NJ.

[S00] C. Schumacher.(2000). *Fundamental Limitations of Search.* PhD thesis, University of Tenessee, Department of Computer Sciences, Knoxville, TN.

[SVW01] C.Schumacher, M.D. Vose, D. Witley (2001). *The No Free Lunch and Problem Description Length.* In L. Spector, and alter, editors, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), 565-570, San Francisco, California, USA, 7-11. Morgan Kaufman Publishers.

[V05] S. Verel (2005) *Etude et exploitation des réseaux de neutralité dans les paysages adaptatifs pour l'optimisation difficile* Mémoire de thèse de Doctorat en Informatique.Université de Nice Sophia-Antipolis. Décembre 2005.

[VOT08] S. Verel, G. Ochoa, M. Tomassini (2008) *The connectivity of NK Landscapes Basins: A Network Analysis.*
In Artificial life XI, August 2008.

[VS05] V. Ventos, H. Soldano (2005) *Les treillis de Galois Alpha.*
RIA 2005.

[W00] D Whitley (2000) *Functions as Permutations: Implications for No Free Lunch, Walsh Analysis and Statistics.*
Parallel Problem Solving from Nature - PPSN VI,
pages 169-178. Springer. Berlin.

[W04] B. Weinberg (2004) *Analyse et résolution approchée de problèmes d'optimisation combinatoire: application au problème de coloration de graphe.* Thèse UST de Lille, 42-50.

[WM95] D.H. Wolpert and W.G. Macready (1995) *No Free Lunch Theorems for Search.* Technical Report SFI-TR-95-02-210,
Santa Fe Institute.

[WM97] D.H. Wolpert and W.G. Macready (1997) *No Free Lunch Theorems for Optimization.* IEEE Transactions on
Evolutionary Computation 1(1), 67-82.

[wG10] G. Gontier (2010) *Sur le théorème des quatre couleurs*
www.enseignement.polytechnique.fr/profs/informatique/
Georges.Gontier/pi2000/pro/gonthier

[wW10] Wikipedia (2010) Graphe planaire
http://fr.wikipedia.org/wiki/Graphe_planaire