UNIVERSITÉ DE MONTRÉAL

DÉCOUVERTE PAR ORDINATEUR EN THÉORIE DES GRAPHES

GILLES CAPOROSSI
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2000

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

DÉCOUVERTE PAR ORDINATEUR EN THÉORIE DES GRAPHES

présentée par : CAPOROSSI Gilles
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. SOUMIS Francois, Ph.D., président
M. SAVARD Gilles, Ph.D., membre et directeur de recherche
M. HANSEN Pierre, Ph.D., membre et codirecteur de recherche
Mme. MARCOTTE Odile, Ph.D., membre
M. SIMON Herbert A., Ph.D., membre externe

À la mémoire de Herbert A. Simon

Remerciements

Je tiens d'abord à remercier mes parents qui ont toujours essayé de m'apprendre l'ouverture d'esprit et stimulé ma curiosité. Ce sont les premiers à m'avoir fait comprendre que le plus important est de faire ce qui nous plaît. Je tiens aussi à remercier tous les autres membres de ma famille qui m'ont aussi toujours encouragé et aidé, par tous les moyens à leur disposition, à suivre la voie que je croyais mienne sans porter aucun jugement sur mes choix, même quand ils signifiaient éloignement.

On ne peut effectuer un travail de plusieurs années sans motivation. Quand cette motivation rime avec satisfaction et passion, de ce travail il reste surtout le souvenir d'une expérience riche et beaucoup de plaisir. Pour ces raisons, je remercie pleinement Pierre Hansen qui a su sans cesse stimuler mon intérêt pour la recherche en me montrant une grande confiance, en proposant toujours de nouveaux défis et en assurant le financement de ces travaux.

Je voudrais profiter de cette occasion pour remercier aussi tous mes autres coauteurs, et particulièrement Gunnar Brinkmann avec qui j'ai pu pratiquer la programmation simultanée entre deux continents lors de l'élaboration du programme d'énumération de polyhexes qui est à ce jour le plus rapide, Dragos Cvetković pour les travaux relatifs à l'énergie ainsi que l'index d'arbres avec contrainte de coloration et Ivan Gutman pour les travaux sur l'énergie, l'étude de l'indice de Randić et la recherche d'arbres H-palindromiques.

Je remercie également Herbert Simon d'avoir accepté d'être le membre externe du jury de cette thèse.

Merci aussi à Gilles Savard qui m'a fait confiance en acceptant d'être mon directeur

administratif.

Je ne voudrais surtout pas oublier les étudiants qui ont travaillé avec **AGX** à diverses reprises, Emmanuel Libeau et Florian Pujol pour leurs remarques sur le programme ayant contribué à l'améliorer sensiblement, sans oublier Fabrice Jammes qui a non seulement utilisé le programme mais a aussi contribué à l'élaboration de la preuve de la section 2.1.3.

Un travail de cette envergure ne peut certainement pas être mené à bien sans un certain soutien à l'extérieur du milieu de travail. Merci à Armelle qui a eu la patience de supporter le "vieil haïssable" pendant ces années.

Je voudrais finir en remerciant ceux, visibles et invisibles, qui m'ont aidé à trouver les idées, l'énergie et l'intérêt pour cette recherche.

Résumé

Nous nous intéressons dans cette thèse à divers moyens d'utiliser les ordinateurs pour aider les chercheurs en théorie des graphes. La plupart des problèmes en théorie des graphes s'expriment à l'aide d'invariants graphiques.

Nous avons abordé l'étude de ces invariants par la recherche de graphes pour lesquels ils ont des valeurs extrêmes. Considérant ce problème comme un problème d'optimisation combinatoire, nous avons utilisé une métaheuristique de ce domaine pour le résoudre. La gamme de problèmes pouvant être traités à l'aide d'un programme basé sur l'optimisation semblait alors assez importante pour nous encourager à concevoir un tel outil. Au fur et à mesure de son utilisation, nous avons éprouvé le besoin d'ajouter diverses fonctionnalités au programme, de sorte qu'AutoGraphiX (**AGX**) comporte aujourd'hui plus de 30 000 lignes.

Outre le module d'optimisation qui est à son origine, il fut assez rapidement doté d'une interface graphique et d'un interpréteur lui permettant de lire le problème à traiter dans un fichier de paramètres sous la forme d'un problème d'optimisation écrit à l'aide de fonctions algébriques. Dans ce fichier de paramètres, un certain nombre de variables peuvent être initialisées à des valeurs adaptées au problème à traiter. L'interface entre le programme et le chercheur se fait par divers médias. D'une part l'interface graphique permet une visualisation rapide des résultats et une certaine forme d'interaction avec le programme car il est possible de modifier les graphes obtenus afins de tester rapidement certaines hypothèses suggérées par les résultats. D'autre part, le programme génère un rapport comportant une représentation des graphes obtenus sur quelques pages, comme le montre la figure 2.1, un exposé plus

complet de ces graphes avec les valeurs de certains invariants choisis par l'utilisateur, à raison d'un graphe par page, ainsi que la liste des conjectures automatiquement trouvées et la représentation de la valeur de la fonction objectif en fonction des paramètres utilisés similaire à la figure 2.38. Le troisième module d'**AGX** a pour but de donner de manière automatisée des conjectures sur les graphes extrêmes. Les conjectures ainsi trouvées paraissent parfois simples mais sont néanmoins très utiles car elles indiquent assez rapidement la classe de graphes extrêmes rencontrée. L'analyse approfondie des graphes obtenus est généralement guidée par ces premières relations.

Le début de la thèse est dédié à la description du programme **AGX** tandis que nous exposons ensuite les études pour lesquelles il a été utilisé ainsi que les résultats qu'il a permis d'obtenir. Certaines études portent sur divers indices topologiques dans le cas général ou pour des familles particulières de graphes. D'autres encore portent sur des propriétés des graphes plutôt que sur les valeurs d'invariants. Nous exposerons dans un premier temps l'étude de l'indice de Randić utilisé en chimie pour lequel des bornes ont été trouvées et certaines démontrées par un argument original utilisant la programmation linéaire. Nous étudierons ensuite l'énergie d'un graphe, invariant également utilisé en chimie et pour lequel nous avons trouvé des résultats simples mais inconnus malgré plusieurs décénies de recherche. Dans le cas de familles spécifiques de graphes, nous avons aussi utilisé **AGX** pour étudier l'index d'arbres avec contraintes de colorations (voir section 2.3), ce qui nous a permis de trouver une conjecture qu'il est peu probable qu'un chercheur ait trouvée car elle comporte 5 invariants. Le dernier problème que nous traitons ici est celui de réfuter une conjecture sur l'existence d'arbres dont le polynôme de Hosoya est palindromique (voir section 2.4). Cette étude nous a ensuite conduit à une nouvelle conjecture pour le moins surprenante.

Lors de la seconde partie, nous exposons deux algorithmes utilisés pour l'énumération de certains types de graphes,les polyhexes. Un polyhexe est un ensemble connexe

d'hexagones tel que deux hexagones soient disjoints ou partagent exactement une arête. Si la représentation du polyhexe à l'aide d'hexagones réguliers de même taille peut se faire sur le plan sans que deux arêtes ne se superposent, on dit que le polyhexe est planaire ("planarité" n'est pas ici pris au même sens qu'en théorie des graphes). Les polyhexes que nous nous proposons d'énumérer sont simplement connectés, ce qui signifie qu'ils n'ont qu'une frontière (ou bien qu'ils ne comportent pas de trous).

La première méthode d'énumération est basée sur la méthode de recherche inversée d'Avis et Fukuda, équivalente dans ce cas à la méthode de génération ordonnée de Read et Faradzev. L'implantation tire profit des particularités du "*Boundary-edges Code*" ainsi que d'une manière appropriée de définir la relation père-fils de l'arborescence utilisée pour cette énumération. Nous avons réussi à énumérer les polyhexes planaires simplement connectés jusqu'à $h = 21$ hexagones, soit une ensemble plus de 600 fois plus important que précédemment énuméré. Dans le cas des polyhexes simplement connectés, ce facteur s'élève à 4000.

La seconde méthode d'énumération que nous avons utilisée ensuite est basée sur la possibilité de définir chaque polyhexe par son graphe dual (graphe d'adjacence des hexagones) ainsi qu'un étiquetage des sommets de ce graphe (correspondant aux hexagones du polyhexe) permettant de donner les positions relatives des hexagones dans le polyhexe. Dans ce cas, nous générons dans un premier temps les graphes duaux, puis nous définissons les étiquettes indiquant les positions relatives des hexagones de manière à définir une et une seule fois tous les polyhexes associés à chacun de ces graphes. Du fait qu'un polyhexe ne peut appartenir qu'aux mêmes classes de symétrie que le graphe dual qui lui est associé et qu'un très grand nombre de polyhexes ont le même graphe dual, un grand nombre de tests peuvent être évités. Dans le cas où nous ignorons les contraintes de planarité, il est même possible de compter les polyhexes sans pour autant les construire. Cet algorithme nous a permis de générer tous les polyhexes planaires jusqu'à $h = 24$, soit un ensemble environ 120

fois plus important qu'avec la génération ordonnée. Dans le cas ou la planarité n'est pas considérée, l'algorithme permettait d'effectuer un décompte jusqu'à $h = 26$, ce qui est environ 25 000 fois plus important qu'avec la génération ordonnée.

Table des matières

Abstract

We are concerned in this thesis with various ways to use computers to help researchers working in graph theory. Most of the problems in that field may be written using graph invariants.

We tackle the study of these invariants by the search for graphs for which they are extremal. Considering this problem as a combinatorical optimisation one, we used a metaheuristic from that field to solve it. The range of problems that may be treated by a program based upon optimisation was broad enough to encourage us in developping such a tool. While using it, we felt a need for further capabilities of the program, so that AutoGraphiX (**AGX**) has now over 30 000 lines.

Apart from the optimisation part which was its starting point, it quickly acquired a graphical interface, a translator to read the problem to be considered in a parameters file written as an optimisation problem using algebraic formulae. In this parameters file, some variables may be initialized to specific values corresponding to the problem. The interface between the program and the researcher uses various methods. On one side is the graphical interface which allows a quick visualisation of the results as well as some interaction which allows easy tests of hypotheses that may be suggested by the results. On the other side, the program generates a report with a representation of the obtained graphs on few pages, as shown on the figure 2.1, a more complete description of the graphs with a page for each giving a list of characteristics chosen by the user, a list of the conjectures and a curve representing the objective function as a function of the parameters like the curve on figure 2.38. The third module in **AGX** aims to automatically suggest conjectures on the extremal graphs. The so

found conjectures sometimes look very simple but are nevertheless very useful as they show the family of graphs to which the extremal graphs belong. A more precise study of the obtained graphs often starts with these conjectures.

The beginning of the thesis is dedicated to the description of the **AGX** program and we then show the studies for which it was used as well as the results found. Some studies are on various topological indices in the general case or for specific families of graphs. Some others studies treat some properties of the graphs rather than values of invariants. We first describe the Randić index study for which bounds were obtained and some of them were proved by an original argument using linear programming. We then study the energy of a graph, an invariant used in chemistry for which some simple but new results were found despite decades of work on that topic. In case of specific families of graphs, we used **AGX** too for a study of the index of trees given some coloring constraints (see section 2.3). In this case, we obtained a conjecture that was very unlikely to be discovered by an unaided researcher as it involves 5 invariants. The last problem we consider here is that of refuting a conjecture on the existence of trees with a palindromic Hosoya polynomial (see section 2.4). This study then led us to a new and very surprising conjecture.

During the second part, we give two algorithms used for enumeration of special kinds of graphs, the polyhexes. A polyhex is a set of connected hexagons such that any two of them either share an edge or are disjoint. If the representation of that polyhex with regular hexagons of the same size is such that there are no overlapping edges, it is said to be planar (note that planarity in that sense has nothing in common with planarity in graph theory). The polyhexes we propose to enumerate are simply connected, which means they have a single frontier (or they have no hole).

The first enumeration method is based upon the reverse search of Avis and Fukuda, which corresponds to the orderly generation of Read and Faradzev in this case.

The implementation takes advantage of particularities of the Boundary Edges Code as well as of a specific father-son relation in the enumeration tree. This allowed enumeration of planar polyhexes to $h = 21$, which corresponds to a set 600 times larger than previously enumerated. In the case of general polyhexes, this ratio raises to 4000.

The second method was based upon a definition of the polyhexes using their dual graph (adjacency graph based upon the hexagons) as well as a labelling of the vertices to give relative positions of hexagons in the polyhex. In this case, we first generate dual graphs, and then define labels allowing the generation of each polyhex corresponding to that dual graph exactly once. Since a polyhex may only belong to symmetry classes to which its dual graph belongs, and a large number of polyhexes share the same dual graph, a huge number of tests may be avoided. If we do not consider planarity constraints, it is also possible to count the polyhex without explicitely generating them. This algorithm allowed the generation of planar polyhexes to $h = 24$ which is a set about 120 times larger than generated with orderly generation (the best previous method). In the case planarity is not required, the algorithm allows to count polyhexes up to $h = 26$, which corresponds to a set 25 000 larger than the orderly generation did.

Liste des tableaux

Liste des figures

Liste des abréviations et index

Liste des annexes

Introduction

La puissance de calculs de l'ordinateur est un atout pour le chercheur en théorie des graphes. Non seulement, il permet de définir rapidement certaines caractéristiques des graphes, mais il permet aussi de générer rapidement un grand nombre de graphes et ainsi de vérifier ou réfuter certaines propriétés. L'objectif de la présente thèse étant l'utilisation de l'ordinateur comme outil de recherche, nous nous sommes dans un premier temps intéressé aux méthodes rapides de génération de certaines classes de graphes. Dans cette optique, nous avons d'abord travaillé sur le problème de la génération de polyhexes, structures géométriques représentant certains types de molécules, un problème que les chimistes étudient. C'est l'objet de la seconde partie de cette thèse.

Nous avons ensuite remarqué que la génération exhaustive de tous les éléments d'une famille combinatoire n'est pas indispensable puisqu'il suffit d'un seul élément pour réfuter une conjecture. La recherche que nous avons alors décidé d'entreprendre est celle de contre-exemples. Comme un grand nombre de conjectures en théorie des graphes sont est données sous la forme d'inégalités faisant intervenir des mesures quantitatives, nous avons choisi de les traiter comme des problèmes d'optimisation et de trouver des graphes extrêmes plutôt que de procéder par génération exhaustive. Si nous cherchons le plus petit graphe possédant certaines propriétés, la connaissance d'un exemple donnera une borne supérieure sur cette taille. La certitude que cet exemple est bien le plus petit viendra d'une recherche exhaustive parmi tous les graphes de plus petite taille. La génération de graphes étant réduite à des graphes de taille modérée, il est souvent impossible de trouver une borne supérieure par son utilisation. Il n'est pas rare que des problèmes en théorie des graphes soient résolus par

l'utilisation simultanée de la génération exhaustive et la connaissance d'un exemple obtenu par divers moyens.

Les deux approches sont en définitive complémentaires, l'approche par optimisation (heuristique) permet éventuellement de trouver un contre-exemple à une conjecture, tandis que la génération exhaustive permet de s'assurer que cette conjecture est vraie pour la classe de graphes étudiés.

La première partie de cette thèse, consistait à développer un outil efficace de recherche de graphes extrêmes par le biais de méthodes heuristiques d'optimisation combinatoire. Par leur étude, nous avons remarqué que les graphes extrêmes donnent une information précieuse sur le problème étudié et avons ensuite concentré nos efforts sur le développement d'outils facilitant l'analyse de ces graphes extrêmes. Dans le but de faciliter l'étude des graphes extrêmes, la version actuelle du programme comporte, outre la routine de recherche de graphes extrêmes, une interface graphique permettant de visualiser les graphes obtenus et de les étudier par modifications interactives, ainsi qu'un certain nombre de routines d'analyse automatique permettant de donner une caractérisation des graphes obtenus.

La découverte scientifique et l'ordinateur

Il ne fait aucun doute qu'une machine capable de reproduire à l'infini une séquence d'opérations définie par avance peut se montrer utile pour aider les humains dans diverses tâches. Cette machine, l'ordinateur, occupe du reste une place importante dans notre vie quotidienne et professionnelle. La question que nous nous posons face à ce phénomène est bien sûr de savoir quelles sont les limites des possibilités de cette machine, quelles sont les tâches qui peuvent lui être confiées et quelles sont celles qu'elle ne peut accomplir. Elle peut certainement effectuer aujourd'hui, beaucoup

plus rapidement que les humains, des tâches répétitives de calcul, mais peut-elle aussi faire preuve d'intelligence, d'imagination, d'intuition ou même avoir des sentiments ? Si oui, dans quelle mesure ? D'une part, par sa conception même, l'ordinateur d'aujourd'hui ne peut qu'effectuer des opérations simples. D'autre part, il opère avec une grande rapidité, ce qui permet un grand nombre d'opérations en un temps raisonnable, donc de simuler un comportement plus complexe qu'il nous intéresse de comparer avec la pensée humaine. L'ordinateur peut alors être utilisé comme outil de simulation de la pensée, ce qui est d'un intérêt capital en psychologie cognitive car un grand nombre d'influences sur l'expérience sont éliminées. C'est un peu selon cette perspective que le premier programme d'intelligence artificielle, *Logic Theorist*, a été conçu par Newell, Shaw et Simon en 1956. À partir de trois règles de transformations et d'une série de théorèmes, le programme déduit de nouveaux théorèmes en utilisant les règles, ce qui, ultimement, donne une séquence de transformations permettant de retrouver l'expression à démontrer à partir d'un théorème initial (dans le cas où il parvient à effectuer la preuve).

Les trois règles de transformations sont (voir [102])

1. *Substitution :* remplacer systématiquement une variable par une expression.
2. *Remplacement :* remplacer une expression par une autre expression équivalente. Par exemple, $p \Rightarrow q$ (p implique q) est équivalent à $\bar{p} \vee q$ (non p ou q).
3. *Détachement :* la règle de détachement dit que si A et $A \Rightarrow B$ sont deux théorèmes, alors B est un théorème.

Comme chaque expression peut être modifiée de diverses manières, la recherche de la démonstration d'une proposition revient à l'exploration en profondeur d'abord d'un arbre dont chaque sommet correspond à un théorème alors que les arêtes représentent des transformations valides à l'aide des règles mentionnées ci-dessus.

En se basant sur les résultats obtenus avec *Logic theorist*, les mêmes auteurs ont conçus le *General Problem Solver* peu de temps après [103]. Le principal élément

nouveau dans le *General Problem Solver* est que le problème est clairement dissocié de la méthode de résolution. Il est alors possible de résoudre différents problèmes à l'aide du même programme à la condition que ceux-ci soient formulés dans un langage approprié. Ainsi, le *General Problem Solver* n'a pas la prétention de résoudre tous les problèmes, mais de ne pas être restreint par la nature du problème à traiter.

Ces deux programmes montrèrent la possibilité qu'un programme informatique accomplisse une tâche que nous qualifions d'intelligente.

En 1965, dans un domaine tout autre, le programme *Dendral* de Feigenbaum et Lederberg [62], utilise les résultats de la spectrométrie de masse, pour donner des structures plausibles de certains composés chimiques analysés. La spectrométrie de masse donne des *pics* associés à certaines masses (de hauteur proportionelle à la fréquence d'observations de sous-composés de la masse considérée), qui indiquent que le composé éclaté a produit un sous-composé de cette masse. Il reste ensuite à déterminer la structure de ce sous-composé pour en déduire, ultimement, la structure du composé initialement étudié. Comme le nombre de structures simples de même masse est grand, la combinatoire fait que ce problème est difficile à résoudre en pratique. Pour aider le système, des règles heuristiques sont données par les spécialistes qui, grâce à leur expérience, savent quelles structures sont les plus plausibles. L'utilisation de ces règles permet d'orienter les recherches en priorité vers les avenues les plus prometteuses, ce qui donne des résultats intéressants en un temps raisonnable. Dans le cas de *Dendral*, l'objectif n'est plus de simuler la pensée humaine pour mieux la comprendre, mais de définir un outil d'aide aux chercheurs en proposant des hypothèses plausibles, but recherché aussi par les chercheurs avec l'utilisation de moyens éventuellement différents. Le programme *Dendral* est le premier système expert a avoir été assez développé pour permettre son utilisation.

Dans le domaine de la découverte scientifique, un autre pas en avant est dû à Lenat, en 1977, avec son programme *AM* [94]. Programmé en *Lisp*, ce programme explore

la théorie des ensembles à la recherche de nouveaux concepts, ce qui lui a permis par exemple de redécouvrir le concept de nombre premier. Partant de concepts simples, *AM* les utilise pour en définir de nouveaux, plus complexes. Ces nouveaux concepts sont ensuite utilisés pour des analyses (tâches) pouvant à leur tour donner lieu à la découverte d'autres nouveaux concepts. La recherche dans *AM* est guidée par deux principes. D'une part, l'agenda joue plutôt un rôle stratégique et indique quelle tâche doit être explorée en priorité ; d'autre part, le coeur, basé sur des heuristiques, indique les méthodes à utiliser de préférence pour accomplir une tâche donnée. *AM* tel qu'il était conçu faisait de la découverte scientifique non supervisée en mathématique. Il semble qu'aujourd'hui les chercheurs ne soient plus autant attirés par ce type de recherche qui conduit souvent à l'élaboration d'une forme de savoir seulement utilisable par des moyens informatiques, dissociant une "science pour les ordinateurs" de la science qui intéresse les chercheurs. En effet, l'utilisation de l'ordinateur est peu appropriée pour certaines tâches alors que ce dernier est fort précieux pour d'autres. Il est donc plus judicieux de le considérer comme un partenaire avec lequel il est possible de collaborer que comme un chercheur autonome.

Dans le but de mieux comprendre le processus de la découverte scientifique, une série de programmes nommés *Bacon* ont été écrits par Langley, Simon, Bradshaw et Zytkow [90] [91] [93] à partir de 1979. Le principe de ces programmes est la recherche de fonctions reliant deux variables x et y entre elles. Pour ce faire, quatre règles sont utilisées.

1. Si un terme est constant, une règle est trouvée.
2. Si x et y sont linéairement reliés, une loi est trouvée.
3. Si les valeurs absolues des deux termes croissent simultanément, un terme additionnel doit être considéré, à savoir le ratio $\frac{x}{y}$.
4. Si la valeur absolue d'un terme croît tandis que celle de l'autre décroît, le produit xy doit être considéré.

Le processus est répété jusqu'à ce qu'un terme constant soit rencontré. Il suffit alors d'identifier sa construction pour mettre en évidence une loi.

Afin de trouver des relations plus complexes (ou après avoir ajouté des termes aux données), des analyses paramétriques sont effectuées, donnant des relations entre paires de variables sous l'hypothèse que d'autres variables sont constantes. Par ce principe, une relation telle que la loi des gaz parfaits ($PV = nRT$ où P et V sont respectivement la pression et le volume, n est le nombre de moles de gaz, T la température et R la constante 8,32 des gaz parfaits) fut redécouverte. Les programmes *Bacon* utilisant de manière systématique des règles très simples ont permis de redécouvrir un certain nombre de lois fondamentales telles que la troisième loi de Kepler, la loi de la gravitation universelle ou la loi des gaz parfaits, par exemple. L'intérêt de ces systèmes est qu'ils contribuent à montrer que le processus de la découverte scientifique n'est pas nécessairement un processus complexe. Les relations redécouvertes à l'aide du système *Bacon* sont décrites plus en détail dans la section 1.4.3.

La dernière génération de programmes de découverte scientifique utilise les ressources disponibles sur le réseau internet et l'extraction automatique d'information dans de grandes bases de données (*Data mining* en anglais). Le meilleur exemple est le programme *Arrowsmith* de Swanson et Smalheiser [114]. Ce programme travaille à l'aide des bases de données de *MEDLINE* accessibles sur l'internet et découvre des relations en médecine. Utilisant la remarque qu'il arrive qu'un grand nombre d'articles traitent de relations entre des variables A et B tandis que d'autres associent B et C sans pourtant qu'aucune étude ne traite de relations possibles entre A et C, le programme suggère d'étudier de telles relations. Cette approche simple a déjà permis de découvrir des relations jusque-là inconnues. La manière de considérer la découverte scientifique exploitée par *Arrowsmith* est typiquement contemporaine alors que les publications scientifiques se multiplient et qu'une masse d'informations sans précédent est dorénavant disponible. *Arrowsmith* ne fait que mettre en évidence des informations déjà

accessibles aux chercheurs mais difficiles à exploiter. Le travail rapide et systématique de l'ordinateur permet alors de repérer les connaissances utiles. Alors que la disponibilité d'informations croit rapidement dans tous les domaines,*Arrowsmith* montre l'intérêt que l'on doit porter à l'exploitation de ces données, même en découverte scientifique.

Un des objectifs de l'utilisation de l'ordinateur est de permettre au chercheur l'exploration de nouveaux horizons, et d'accéder à de nouvelles découvertes. Deux approches se complètent à cette fin. La première consiste à identifier au sein du processus de recherche les tâches répétitives afin de les automatiser, ce qui permet au chercheur de se concentrer sur d'autres aspects du problème. La seconde consiste à utiliser des méthodes qui ne seraient pas envisageables sans la puissance de calcul propre aux ordinateurs dans le but de découvrir de l'information nouvelle. Si cette information émanait d'un humain, elle serait sans doute considérée comme une intuition.

Nous nous trouvons alors dans la situation où l'ordinateur se montre apte à accomplir des tâches qui seraient a priori réservées à des êtres considérés comme intelligents. Face à cette réalité, il est naturel de se poser certaines questions quant à la position que prend, devrait prendre ou peut prendre cette machine dans les laboratoires de recherche.

Valdés-Peréz [118] s'est penché sur la question de savoir pourquoi certains programmes réussissent dans le domaine scientifique. Comme réponse à cette question, il dit qu'il en est de l'ordinateur comme de n'importe quel collaborateur, à savoir que nous travaillerons avec lui s'il donne des résultats *nouveaux*, *intéressants*, *plausibles* et *intelligibles*. Cette approche met implicitement en valeur la collaboration de l'ordinateur et du chercheur. Langley [92] avance une théorie similaire en étudiant l'importance de l'intervention humaine dans les programmes de découverte scientifique. Loin de critiquer cette influence inévitable, il va jusqu'à recommander que les systèmes à venir donnent un support plus explicite à l'intervention humaine.

Utilisation des ordinateurs en théorie des graphes

L'ordinateur est très largement utilisé pour déterminer, souvent de manière répétitive, certaines caractéristiques précises de graphes donnés. Pour effectuer ces tâches, une multitude d'algorithmes et de programmes spécifiques a été conçue. Le défi de ce genre d'applications est de trouver des algorithmes assez rapides pour traiter les graphes de grande taille représentant les problèmes auxquels les praticiens sont confrontés, dans un temps de calcul satisfaisant. Un autre genre d'applications, qui nous intéresse plus spécifiquement ici, est non pas l'étude d'un graphe donné selon certains critères, mais plutôt l'étude des caractéristiques partagées par des familles de graphes, ou par tous les graphes. Bien que l'on utilise souvent le calcul de caractéristiques de graphes précis lors de telles études, l'emphase est mis sur l'information que donnent ces calculs plutôt que sur la manière de les effectuer. Les applications suivantes sont utiles en théorie des graphes et l'ordinateur peut avantageusement aider le chercheur qui s'y intéresse..

1. *Génération automatique ou interactive, analyse et description de graphes*

   Une manière courante d'utiliser l'ordinateur est pour évaluer des fonctions associant à chaque graphe une valeur indépendemment de la manière dont les sommets sont numérotés. Ce type de fonction appelée *invariant graphique* est un outil très largement utilisé en théorie des graphes. Le nombre de sommets n aussi appelé *Ordre du graphe*, le nombre d'arêtes m ou le nombre de stabilité α (nombre maximum de sommets du graphe deux à deux non adjacents) sont des exemples d'invariants graphiques.

   Il existe des programmes généraux utilisant un certain nombre d'algorithmes pour fournir une information complète sur des graphes donnés, ce qui permet de libérer le chercheur de la tâche souvent longue et fastidieuse d'évaluer à la

main ces propriétés. Le chercheur peut alors se concentrer plus efficacement sur les véritables difficultés.

Ces programmes augmentent amplement les capacités des chercheurs à travailler sur des graphes particuliers, notamment par les fonctions de visualisation immédiate.

Le système *Cabri-Graphes* (*ca*hier de *br*ouillon *i*nformatique), développé par un groupe de chercheurs de Grenoble [50], entre dans cette catégorie de programmes.

Dans une revue récente des applications du système *Graph*, [89] Cvetković et Simić mentionnent 55 articles de 16 auteurs différents [40]. La partie *Algor* de *Graph* répond exactement aux fonctionnalités que nous venons de mentionner, et d'après les auteurs de *Graph*, c'est elle qui s'est montrée de loin la plus utile.

2. *Démonstration de théorèmes assistée par ordinateur.* L'exemple le plus célèbre est le théorème des quatre couleurs dont la première preuve [1] est due à Appel et Haken [3] [4] [5]. D'importantes parties de la preuve ne peuvent être effectuées qu'à l'aide d'ordinateur. Pour une preuve plus récente du même théorème, due à Robertson, Sanders, Seymour et Thomas [110], l'ordinateur joue aussi un rôle.

3. *Démonstration automatique ou semi-automatique de théorèmes.* Un système destiné à cette application est le démonstrateur de théorèmes *Theor* développé par Cvetković et Pevac [38] [32] [33] [34] dans le cadre du système *Graph* [89] [31] [40] [45]. Des théorèmes très simples sont démontrés sans aucune intervention humaine, mais dans le cas de théorèmes plus complexes, cette dernière demeure indispensable. Selon les auteurs mêmes de *Graph*, la démonstration automatique de théorèmes en théorie des graphes reste encore très limitée.

[1] On sait que cette preuve a du être amendée, du fait d'une hypothèse implicite sur l'indépendance des configurations réductibles.

Dans le cas de démonstration semi-automatisée de théorèmes, mentionnons le système *Ingrid* de Brigham et Dutton [19] [20] [48] [21]. Ce système est exclusivement basé sur l'analyse de relations entre invariants et peut définir des relations nouvelles induites par d'autres en utilisant l'algèbre d'intervalles. À chaque invariant graphique sont associées les bornes de l'intervalle des valeurs qu'il peut prendre. Le programme traite de 36 invariants graphiques de base et comporte une base de connaissances provenant d'environ 350 théorèmes composée d'approximativement 1200 règles. Chacune de ces règles permet de redéfinir des bornes sur certains invariants en fonction de valeurs d'autres invariants. Le système *Ingrid* est interactif et donne les bornes sur les invariants qu'il traite en fonction de valeurs qui sont fournies par l'usager. En effet, si un usager s'intéresse à une classe de graphes qu'il définit par une borne sur un invariant (par exemple le degré maximum), *Ingrid* va recalculer les bornes qui peuvent être modifiées par cette information avant de propager à nouveau les nouvelles bornes obtenues, et ce, jusqu'à ce qu'aucune borne modifiée ne soit en attente d'être propagée. Comme ce système ignore tout de la nature des graphes et se restreint à une analyse d'intervalles, il ne tient compte de contraintes dues à la structure même des graphes que dans la mesure où elles sont considérées par les théorèmes, ce qui entraîne que les bornes trouvées pour des fonctions d'invariants sont parfois trop larges pour être utilisables.

4. *Génération automatique, vérification et sélection de conjectures.* Dans ce domaine, le système *Graffiti* conçu par Fajtlowicz [52] [51] [53] [54] a déjà donné des résultats intéressants et stimulé l'intérêt de chercheurs. *Graffiti* définit des conjectures du type $i_1(G) \leq i_2(G)$ ou $i_1(G) \leq i_2(G) + i_3(G)$ ou encore $i_1(G) + i_2(G) \leq i_3(G) + i_4(G)$, où les $i_j(G)$ sont des invariants graphiques, $j = 1 \ldots 4$. De la même manière, dans les versions plus récentes, des formes où apparaissent des ratios d'invariants sont considérées. Ces conjectures sont alors testées sur une base de données d'exemples de graphes. Celles qui passent ce

test sont nombreuses et parfois de moindre intérêt. Les conjectures proposées sont donc soumises à d'autres procédures qui éliminent certaines d'entre elles. Si une relation découle d'autres relations par transitivité, ou correspond à des relations déjà connues pour des classes de graphes plus larges que la classe à laquelle elle s'applique, la conjecture n'est pas retenue. Les conjectures qui restent sont proposées à la communauté mathématique. Dans le cas où une conjecture est réfutée par un contre-exemple, celui-ci est ajouté à la base de données, ce qui contribue à renforcer le système. Certaines conjectures peuvent se montrer difficiles à prouver ou réfuter. Les conjectures de *Graffiti* ont suscité et suscitent toujours un grand intérêt au sein de la communauté des chercheurs en théorie des graphes. Le texte *Written on the wall* remis à jour régulièrement [55] comporte une liste des conjectures ainsi que les références à des preuves, des contre-exemples, et d'autres remarques suscitées par ces conjectures. Ce document fait état du travail de plus de 77 mathématiciens parmi lesquels Alon, Bollobaś, Chung, Cvetković, Erdös et Seymour. Plusieurs théorèmes intéressants ont été suggérés par *Graffiti* tels que celui de Chung [29], qui démontre que la distance moyenne n'est jamais supérieure au nombre de stabilité, ou celui de Favaron, Mahéo et Saclé [96] prouvant que le résidu ne dépasse jamais le nombre de stabilité.

5. Un autre outil utile au chercheur en théorie des graphes est la recherche bibliographique automatisée basée sur des concepts de théorie des graphes plutôt que sur des mot-clés, telle que dans la partie *Biblio* de *Graph* [47]. Un tel outil permet des recherches plus efficaces que les recherches habituelles par mots clés puisque les mot clés donnés par les auteurs pour caractériser un même concept peuvent varier selon leur discipline d'origine. D'un autre côté, il est incontestablement plus difficile à maintenir à jour que les bases de données automatiquement générées par les moteurs de recherche disponibles sur internet. De ce fait, ces derniers compensent l'absence d'uniformisation des données par

l'étendue des informations dont ils disposent. Il suffit pour cela que le chercheur fasse lui même l'effort de diversifier les mot-clés qu'il utilise lors de ses requêtes. Comme nous aurons l'occasion de le remarquer lors de la section 2.2, il y a pleinement la place pour un outil structuré d'analyse et de recherche dans la littérature. Il arrive que le même concept soit décrit dans diverses publications sous différents noms génériques, mais il arrive aussi qu'un résultat soit redécouvert parce qu'une formulation équivalente décrit exactement le même résultat. Dans un tel cas, l'outil de recherche bibliographique n'est plus suffisant mais un outil capable de transformer une formulation en une autre serait utile. En ce sens, il serait sans doute intéressant de concevoir un système capable d'identifier par une requête les problèmes équivalents déjà traités. Selon une telle perspective, les concepts utilisés dans *Ingrid* combinés à ceux de la partie *Algor* de *Graph* semblent prometteurs bien que les difficultés associées à la gestion des données restent le principal obstacle à une telle tentative.

Ces exemples montrent que les ordinateurs peuvent apporter une aide précieuse aux mathématiciens en donnant des parties de preuves, des indications pour des théorèmes spécifiques, en suggérant des conjectures ou simplement en effectuant rapidement certaines opérations.

Chapitre 1

Le système AutoGraphiX

1.1 Fonctions d'AutoGraphiX

Le système **AGX** que nous allons maintenant décrire est la partie centrale des recherches présentées ici. L'objectif initial sur lequel repose la plus grande partie de ses fonctionnalités est la recherche de graphes extrêmes, c'est-à-dire des graphes pour lesquels un invariant prend la plus grande ou la plus petite valeur possible. L'idée originale à la base d'**AGX** est de recourir aux outils de l'optimisation combinatoire et, plus précisément à la métaheuristique de *Recherche à Voisinage Variable* pour obtenir heuristiquement de tels graphes ou des graphes très proches, en grand nombre, puis d'analyser les propriétés des familles de graphes ainsi obtenues.

Dès les premiers résultats, il est devenu très clair que la recherche de graphes extrêmes pouvait être utilisée pour résoudre un grand nombre de problèmes souvent considérés en théorie des graphes, ce qui donne à cette approche un aspect générique. Il en résulte que l'outil ainsi développé peut aisément être utilisé pour divers types d'applications parfois complexes. Les applications de base de la recherche de graphes extrêmes sont les suivantes :

- **Trouver un graphe satisfaisant des contraintes données.**
  Soient "$i_1(G),\ i_2(G), \ldots i_l(G)$", l invariants de G et $p_1,\ p_2, \ldots p_l$ des valeurs qui leur sont associées. À chaque contrainte C_k du type $i_k(G) \leq p_k$, nous associons la

fonction $h_k(G) = \max(i_k(G) - p_k, 0)$, à chaque contrainte C_k du type $i_k(G) \leq p_k$, nous associons la fonction $h_k(G) = \max(p_k - i_k(G), 0)$, et à chaque contrainte C_k d'égalité $i_k(G) = p_k$, nous associons la fonction $h_k(G) = |i_k(G) - p_k|$.
Considérons alors le problème

$$\min_{G \in \mathcal{G}_n} f(G) = \sum_{k=1}^{l} h_k(G), \tag{1.1}$$

où $\mathcal{G}_n$ désigne l'ensemble des graphes à n sommets, n étant un paramètre.
Tout graphe G tel que $f(G) = 0$ satisfait les contraintes C_k $\forall k = 1 \ldots l$.
Notons que les contraintes comportant des formules avec plusieurs invariants peuvent être traitées de la même manière, une formule dépendant de plusieurs invariants étant elle même un invariant.

- **Trouver des graphes de valeur optimale ou quasi-optimale pour un invariant parmi une famille de graphes définis par des contraintes.**
  Soit $i_0(G)$ l'invariant de la fonction-objectif; supposons que les contraintes sont celles définies plus tôt et considérons le problème

$$\min_{G \in \mathcal{G}_n} f(G) = i_0(G) + M \sum_{k=1}^{l} h_k(G), \tag{1.2}$$

  où M est une constante suffisamment grande pour que, pour toute paire de graphes G, $G' \in \mathcal{G}_n$ tels que $\sum_{k=1}^{l} h_k(G) = 0$ et $\sum_{k=1}^{l} h_k(G') > 0$, on ait $f(G) < f(G')$. Alors, en tout minimum de $f(G)$ les contraintes seront satisfaites si elles peuvent toutes l'être simultanément. Les valeurs maximales sont traitées de manière analogue.

- **Réfuter une conjecture.**
  Soit une conjecture $h(G) \leq g(G)$ où $h(G)$ et $g(G)$ sont des formules d'un ou

plusieurs invariants de G.

Considérons le problème

$$\min_{G \in \mathcal{G}_n} f(G) = g(G) - h(G). \tag{1.3}$$

Si un graphe G pour lequel $f(G) < 0$ est trouvé, la conjecture est réfutée.

- **Suggérer une conjecture (ou en renforcer une).**

  Ceci peut être effectué de diverses manières qui en général font appel à la paramétrisation en fonction de n ou d'autres invariants. Considérons par exemple une fonction $f(G)$ d'un ou plusieurs invariants et le problème

$$\min_{G \in \mathcal{G}_n} f(G). \tag{1.4}$$

  L'analyse qualitative ou quantitative des graphes (supposés extrémaux) obtenus par résolution de ce problème peut nous suggérer des relations entre invariants. Cette partie sera expliquée en détail dans la section 1.4 ci-dessous.

- **Donner une idée de preuve (ou montrer qu'une méthode de preuve ne peut fonctionner).**

  Soit une conjecture du type $h(G) \leq g(G)$ et un graphe supposé extrémal G_1 pour la fonction $f(G) = g(G) - h(G)$. S'il est possible de transformer n'importe quel graphe G en G_1 par une suite de transformations utilisant un voisinage V générant une séquence de graphes dont l'évaluation $f(G)$ est monotone décroissante. Le programme, par les voisinages qu'il utilise, peut suggérer que c'est le cas et préciser le voisinage V en question. Il reste alors à démontrer que la propriété observée est bien générale, ce qui est une idée de preuve. Il faut reconnaître toutefois que cette méthode n'a donné jusqu'ici que peu de résultats. En revanche, l'impossibilité de construire un graphe de la famille extrême en se restreignant à certaines transformations indique clairement qu'une autre méthode doit être utilisée. Ce type d'information permet d'éviter de perdre du temps à essayer de faire une preuve

avant de se rendre compte que l'idée sur laquelle elle repose ne peut donner les résultats escomptés.

Nous avons profité de cette possibilité lors de l'étude de la **Conjecture WoW :12** de *Written on the Wall* [56] obtenue à l'aide de *Graffiti*. Cette conjecture dit que pour tout graphe connexe, $1+Ra \geq r$. En utilisant **AGX** pour trouver les solutions au problème

$$\min f(G) = 1 + Ra - r, \tag{1.5}$$

les graphes trouvés étaient toujours des arbres (graphes connexes sans cycles). Nous avons dans un premier temps montré que cette conjecture était vraie pour les arbres [27]. L'étape suivante consistait à démontrer qu'il est possible de passer de n'importe quel graphe à un arbre par une série de transformations réduisant toujours la fonction-objectif. La transformation que nous comptions utiliser était le retrait d'une arête.

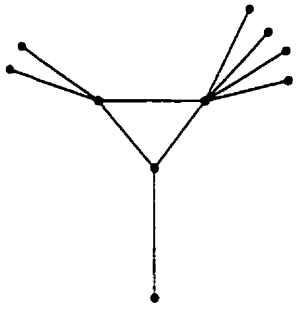


Figure 1.1 – Optimum local du problème $\min 1 + Ra - r$ pour la transformation "retrait d'une arête".

Avant d'essayer cette méthode de preuve, nous avons cherché à minimiser la fonction-objectif à partir de graphes aléatoires en restreignant les voisinages au "retrait d'une arête", dans le but de trouver des optimums locaux. Pour ce problème, un optimum local est un graphe G à partir duquel il est impossible de construire un graphe G'

tel que $f(G') < f(G)$ par un retrait d'arête. Si G n'est pas un arbre, la méthode de preuve considérée est vouée à l'échec ; c'est le cas ici puisque le programme a trouvé assez rapidement l'optimum local représenté sur la figure 1.1.

AGX exécute aussi des tâches plus simples telles que la représentation de graphes, leur modification interactive et le calcul rapide d'invariants choisis.

1.2 La recherche à voisinage variable

Comme nous l'avons mentionné plus haut, l'idée de base d'**AGX** est de chercher des graphes extrêmes, et de traiter ce problème comme un problème d'optimisation combinatoire à l'aide d'outils spécifiques, en particulier les métaheuristiques.

Depuis une vingtaine d'années, un grand nombre de métaheuristiques ont été proposées. Ce sont des cadres généraux pour construire des heuristiques, lesquelles donnent des solutions quasi-optimales et parfois optimales, sans garantie de qualité. Les métaheuristiques s'appliquent à une grande variété de problèmes. Contrairement aux heuristiques traditionnelles, celles basées sur ces méthodes ne restent pas bloquées lorsqu'elles rencontrent un optimum local. Les plus connues d'entre elles sont sans doutes les algorithmes génétiques, le recuit simulé, la recherche avec tabous et les réseaux de neurones (le lecteur intéressé par ce sujet peut se référer au livre de Reeves [109] pour une revue, et à Osman et Laporte [105] pour une bibliographie étendue). Celle que nous utiliserons ici est la récente *Recherche à Voisinage Variable* (souvent appelée par son nom anglais *Variable Neighborhood Search*, ou VNS) [100]. Cette méthode a été appliquée avec succès à un certain nombre de problèmes d'optimisation combinatoire tels que le problème du voyageur de commerce [100], le problème de la p-médiane [76] et le problème de localisation de Weber avec plusieurs sources [22].

Pour expliquer l'algorithme, nous conviendrons que le graphe G' sera dit voisin du graphe G par rapport à la transformation T (par exemple ajouter ou retirer une arête) s'il est possible de construire G' à partir de G en appliquant une fois la transformation T. L'ensemble des graphes voisins de G, $V^T(G)$, est dit *voisinage de* G pour la transformation T.

La *Recherche à Voisinage Variable* telle qu'utilisée dans **AGX** comporte trois phases imbriquées. Tous les problèmes traités par **AGX** peuvent se rapporter au problème générique $\min f(G)$ pour lequel la *Recherche à Voisinage Variable* est décrite ci-dessous.

La routine la plus centrale de l'optimisation dans **AGX** est la recherche locale qui procède comme suit :

Recherche Locale :

1. **Initialisation :**
   Définir la transformation T à utiliser, un graphe initial G, et poser
   $meilleur = 1$
2. **Tant que** $meilleur = 1$ **faire :**
   (a) $meilleur = 0$
   (b) Trouver $G' \in V^T(G)$ tel que $f(G') \leq f(G'')\ \forall G'' \in V_T(G)$
   Si $f(G') < f(G)$
   $G \leftarrow G'$
   $meilleur = 1$

   fait.
   G est un optimum local pour la fonction f, en utilisant la transformation T.

En d'autres termes, la recherche locale modifie successivement le graphe en utilisant la transformation choisie de manière à améliorer à chaque fois la valeur de la fonction-objectif. Le choix de la transformation à utiliser est crucial pour la performance de la recherche. On peut s'attendre à obtenir les meilleurs résultats avec des transformations plus générales, mais ces dernières nécessitent un temps de calcul plus long. Selon le type de problème, elles seront donc plus ou moins pertinentes et il peut être meilleur de travailler en priorité avec des transformations plus simples et rapides, au moins au début de l'optimisation, alors qu'un grand nombre de modifications de G seront effectuées. Le principe de la *Descente à Voisinage Variable* est d'effectuer successivement des *Recherches Locales* utilisant diverses transformations, comme suit :

Descente à Voisinage Variable :

1. **Initialisation** Définir une liste de transformations $T_1, T_2, \ldots, T_l$ à utiliser, un graphe initial G et poser $meilleur = 1$.
2. **Tant que** $meilleur = 1$ **faire :**
   (a) $meilleur = 0$
   (b) **Pour** $i = 1$ **à** l,
   Soit G' le graphe obtenu après avoir appliqué *Recherche locale* avec G comme graphe initial et T_i comme transformation. Si $f(G') < f(G)$,
   $G \leftarrow G'$.
   $meilleur = 1$

   fait.

   G est un optimum local pour chacune des transformations $T_1, T_2, \ldots, T_l$.

Dans **AGX**, une dizaine de transformations sont programmées. Ces transformations sont représentées sur la figure 1.2.

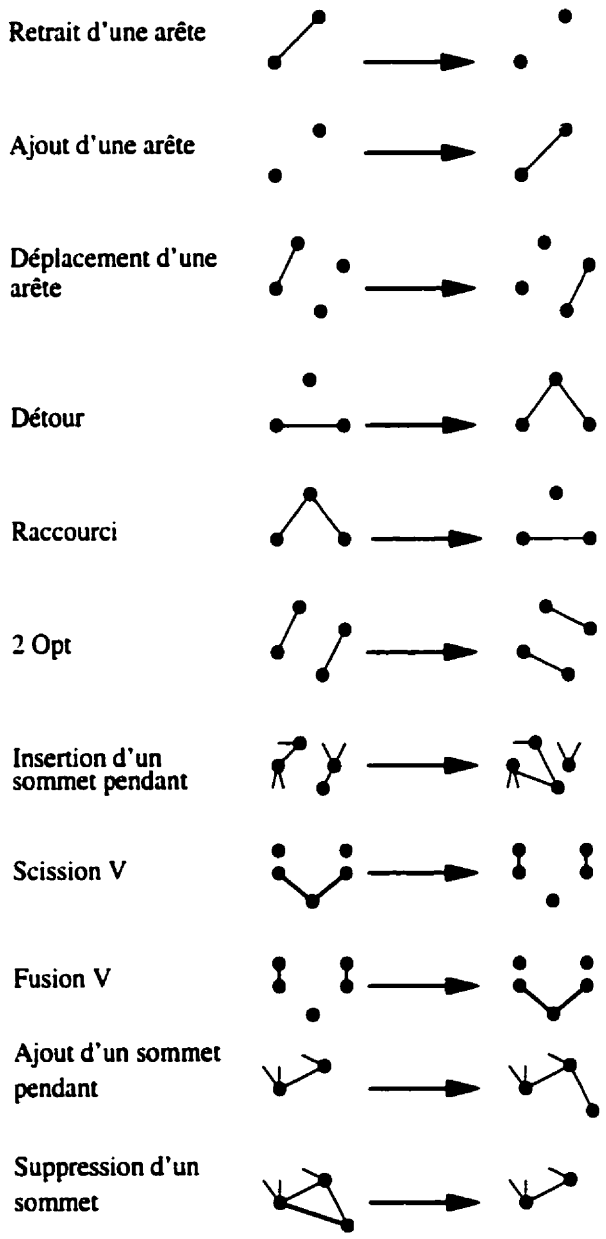


Figure 1.2 – Transformations disponibles dans AutoGraphiX

Le graphe obtenu après la *Descente à Voisinage Variable*, est un optimum local pour chacun des voisinages utilisés. Cette routine sera plus performante si un grand nombre de transformations sont utilisées ; nous l'avons donc améliorée systématiquement lors de l'utilisation du programme. Quand **AGX** trouvait un graphe G et que nous connaissions un graphe G' meilleur que G, nous cherchions une transformation permettant de passer de G à G' afin de l'intégrer au programme. Nous nous sommes limités à des transformations simples et générales, parce que nous voulions à la fois préserver la performance du programme en évitant les transformations associées à un voisinage long à explorer et parce que nous cherchions toujours des outils efficaces pour le plus grand nombre de problèmes possibles. Des améliorations dans la *Descente à Voisinage Variable* sont proposées dans la section 3.2.2 réservée aux développements envisagés.

La troisième partie de l'algorithme utilise des voisinages imbriqués qui se définissent comme suit :
Étant donnée une transformation T, nous disons que G' est dans le k^{ieme} voisinage de G ($G' \in V_k^T(G)$) si nous pouvons construire G' à partir de G en appliquant successivement k fois la transformation T à G. La *Recherche à Voisinage Variable* est alors définie comme suit :

Recherche à Voisinage Variable :
Initialisation
Définir une structure de voisinages imbriqués $V_k^T, k = 1 \ldots k_{max}$, un graphe initial G et une condition d'arrêt.

Tant que la condition d'arrêt n'est pas rencontrée

1. **Poser** $k = 1$.
2. **Tant que** $k \leq k_{max}$ **répéter** les étapes suivantes :

(a) générer aléatoirement un graphe $G' \in V_k^T(G)$.

(b) Appliquer la Descente à Voisinage Variable avec G' comme solution initiale et noter G'' l'optimum local ainsi obtenu.

(c) Si G'' est meilleur que G recentrer la recherche en posant $G \leftarrow G''$ et $k = 1$.
Sinon, $k \leftarrow k + 1$.

La condition d'arrêt peut être le temps de calcul maximum, un nombre d'itérations maximum ou de telles valeurs depuis la dernière amélioration de la fonction-objectif. L'étape *(a)* de perturbation du graphe courant G sera faite par modifications de ce graphe.

Selon le type de problème à résoudre, la phase de perturbation peut se montrer plus ou moins efficace. Si par exemple, le nombre d'arêtes est fixé, modifier le graphe par ajout ou retrait d'arêtes donnera probablement un graphe non réalisable. Il sera de ce fait inutile de considérer, dans la recherche locale, des voisinages modifiant le nombre d'arêtes. De plus, nous pouvons nous attendre à ce que, cherchant d'abord un graphe réalisable, certaines transformations de la perturbation soient annulées par les premières itérations de la recherche locale. Dans ce cas, nous reviendrons souvent sur le même optimum local. Dans le but d'améliorer la performance du programme, deux types de transformations sont utilisés pour la perturbation utilisée dans la RVV d'**AGX** . La première consiste à ajouter ou retrancher une arête au graphe tandis que la seconde, au cas où le nombre d'arêtes est fixé, consiste à déplacer une arête.

1.3 Invariants disponibles dans AutoGraphiX

Parmi les invariants disponibles dans **AGX**, mentionnons d'abord n, le nombre de sommets appelé **NBNOD** dans le fichier de paramètres et m, le nombre d'arêtes,

appelé **NBARC**. Si nous notons d_{ij} la distance entre les sommets i et j (nombre minimum d'arêtes que nous devons parcourir pour aller de i à j), nous pouvons définir $\bar{d}$ la distance moyenne entre paires de sommets, notée **AVDIST**. Le diamètre du graphe D, ou plus grande distance entre paires de sommets du graphe, est noté **DIAMETER**, le mode des distances $mode(d)$ entre paires de sommets est noté **MODEDIST** alors que le rayon du graphe r défini par :

$$r = \min_i(\max_j d_{ij}) \tag{1.6}$$

est noté **RADIUS**. Notons alors δ_i le degré du sommet i ou nombre d'arêtes incidentes au sommet i, le degré minimum δ (*resp.* maximum, Δ) de G est noté **DEGMIN** (*resp.* **DEGMAX**), tandis que le nombre de sommets pendants, ou sommets de degré 1, soit n_1, est noté **PENDING**. De manière plus générale, le nombre de sommets de degré i est noté **DEG**(i), ainsi, **DEG**(1) est équivalent à **PENDING**. La variance de la séquence des degrés $\sigma^2(\delta)$ est notée **VARDEG** tandis que la somme des inverses des degrés est notée **SUMINVDEG**. Un autre indice introduit par Randić comme mesure du branchement des alcanes [107] et fortement étudié par les chimistes sera étudié en détails plus loin. L'indice de Randić Ra, noté **RANDIC** se défini comme suit pour un graphe G composé d'un ensemble V de sommets et E d'arêtes :

$$Ra = \sum_{(ij)\in E} \frac{1}{\sqrt{\delta_i \delta_j}}. \tag{1.7}$$

Le nombre chromatique γ appelé **GAMMA** est le nombre minimum de couleurs nécessaires à la coloration de tous les sommets de G telle que deux sommets adjacents aient toujours des couleurs différentes. Le nombre de stabilité α noté **STABLEMAX** est la cardinalité du plus grand sous-graphe induit sans arêtes de G alors que la taille de la clique maximale, cardinalité du plus grand sous-graphe complet de G, est notée **CLIQMAX**. La température moyenne d'un graphe définie comme

$$AVGTMP = \frac{1}{n}\sum_{j=1}^{n} \frac{\delta_j}{n-\delta_j} \tag{1.8}$$

sera notée **AVGTMP**. L'index, plus grande valeur propre de la matrice d'adjacence $A = a_{ij}$ (où $a_{ij} = 1$ si les sommets i et j sont adjacents et 0 sinon) associée au graphe, est noté **INDEX**.

1.4 Recherche automatique de conjectures

Comme nous le montrerons dans le chapitre 2, l'examen des graphes extrémaux fournis par **AGX** permet très rapidement de proposer des conjectures. L'étape suivante consiste à automatiser cette recherche. Suite à la résolution d'un problème paramétrisé, nous disposons d'un ensemble de graphes supposés extrêmes renfermant une information assez riche pour mener à des conjectures.

La question qui se pose alors est : comment exploiter et synthétiser l'information que recèlent ces graphes. Les fonctionnalités que nous cherchons à développer relèvent de l'extraction de connaissances dans les grandes bases de données (*Knowledge Discovery in Databases* ou KDD, en anglais) qui relève du *"Data mining"* en étant par ailleurs directement reliées à la découverte scientifique par le sujet traité. En suggérant des conjectures, le programme donne au chercheur certaines intuitions quant au problème traité qu'il n'aurait éventuellement pas eues seul. Dans ce sens, nous transformons l'outil d'optimisation en un programme qualifié d'"*intelligent*".

Comme expliqué dans l'introduction, deux tendances se dégagent en découverte scientifique à l'aide de l'ordinateur. La première cherche à reproduire mécaniquement les modes de raisonnement humains, ce qui permet aussi de mieux les comprendre [91] [93]. La seconde cherche plutôt à tirer profit des spécificités de la machine dans le but de développer des méthodes plus efficaces de découverte scientifique.

Les deux approches sont utilisées actuellement dans **AGX**.

Une première méthode découlant directement de la première approche consiste à identifier les familles auxquelles les graphes trouvés par l'optimisation appartiennent dans le but de trouver, à l'aide d'information déjà connues par le système à leur sujet, une expression analytique de la fonction-objectif ne dépendant que des paramètres utilisés. C'est ce que nous appelerons la *méthode analytique*.

Les autres méthodes relèvent plutôt de la seconde approche. Lorsque nous disposons d'une base de données comportant les valeurs de certains invariants pour les graphes extrêmes obtenus, nous pouvons considérer un graphe comme un point dans l'espace des invariants utilisés. De cette manière, l'ensemble des graphes considérés peut être vu comme un nuage de points que nous chercherons à caractériser dans le but de trouver des conjectures.

La seconde méthode de recherche de conjectures consiste à essayer de trouver une direction telle que les projections orthogonales de tous les points représentant des graphes par rapport à cette direction soient identiques, ce qui signifierait que tous les points sont dans un même hyperplan orthogonal à cette direction. Dans ce cas, nous avons une relation affine respectée pour tous les graphes. C'est le principe de la *méthode numérique*.

La troisième méthode consiste à trouver des inégalités respectées par tous les points représentant les graphes dans l'espace des invariants en calculant leur enveloppe convexe : c'est la *méthode géométrique*.

1.4.1 Sélection de graphes

Les méthodes que nous utilisons pour trouver des conjectures sont toutes basées sur les graphes extrémaux obtenus. Il se peut toutefois que notre attention doive se

concentrer plus spécifiquement sur un sous-ensemble de ces graphes soit parce que l'heuristique n'a pas trouvé les graphes extrêmes (auquel cas il faudrait identifier puis retirer les graphes fautifs), soit simplement parce que certaines propriétés des graphes extrêmes ne peuvent pas être vérifiées pour toutes les valeurs des paramètres. Afin d'identifier la structure des graphes extrêmes, nous allons déterminer des méthodes pour sélectionner un sous-ensemble des graphes obtenus. La difficulté de cette tâche est que garder un graphe ne correspondant pas à la bonne famille empêche le programme d'identifier cette structure, alors que si un trop grand nombre de graphes sont éliminés, il peut ne pas en rester assez pour que cette structure soit décelable par le programme. Comme nous n'avons a priori aucune information sur les graphes recherchés, nous devrons utiliser des heuristiques pour cette sélection. À nouveau, nous courons le risque de trouver des familles ne correspondant pas à la caractérisation des graphes extrêmes recherchés. La routine de sélection donne des indications qui doivent ensuite être vérifiées par le chercheur. Afin d'accomplir cette sélection, plusieurs approches ont été considérées.

L'approche la plus simple est basée sur le principe de la moyenne flottante et procède comme suit : pour chacun des paramètres, faire une approximation linéaire de la valeur de la fonction-objectif en utilisant les graphes pour lesquels ce paramètre diffère de un (ou d'une fois le pas d'incrémentation). De cette manière, nous sélectionnerons le graphe G_i (pour lequel la valeur du paramètre est i et l'objectif vaut $f(G_i)$) si $f(G_i)$ est meilleure que $\frac{f(G_{i-1})+f(G_{i+1})}{2}$. Si i est la plus petite ou plus grande valeur possible pour le paramètre utilisé, le graphe n'est pas sélectioné afin d'éviter les effets de bords. Dans le cas où deux paramètres ont été utilisés, la sélection a lieu pour chaque paramètre, donnant deux sous-ensembles distincts de graphes. Pour chacun des cas, le graphe $G_{i,j}$, le graphe trouvé avec i et j comme paramètres, est comparé à ses "voisins" $G_{i-1,j}$ et $G_{i+1,j}$ ou bien $G_{i,j-1}$ et $G_{i,j+1}$, selon le paramètre choisi. Cette approche est actuellement inplantée dans **AGX** et permet de conserver un

assez grand nombre de graphes. Un défaut majeur est que dans le cas où la fonction-objectif a une forme concave, aucun point n'est sélectionné, comme l'illustre la figure 1.3 sur laquelle on peut voir qu'un point n'est sélectionné que s'il est en dessous de la ligne.

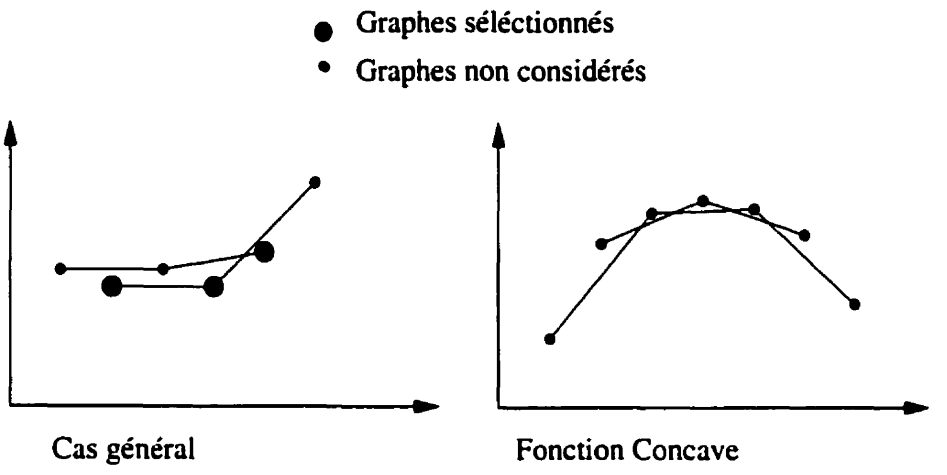


Figure 1.3 – Sélection de graphes extrêmes par la moyenne flottante

Cette première approche étant très locale, nous avons essayé de trouver d'autres méthodes utilisant plutôt une information générale. Dans un premier temps, nous cherchons, par régression non linéaire, à identifier la tendance centrale, puis nous supprimons les graphes dont la valeur est strictement moins bonne que la fonction de régression, avant de répéter le processus, le tout dans l'espoir de trouver une fonction minorante pour les valeurs obtenues. En cas de succès, la fonction obtenue permet de définir une conjecture sur la valeur de l'objectif. Lors des tests sur cette méthode, nous nous sommes rendu compte qu'à moins que la régression ne passe par tous les points assez rapidement, le nombre de graphes conservés chute très vite, de sorte que seuls deux ou trois points demeurent. Il est alors impossible de trouver quelque caractéristique que ce soit, comme l'illustre la figure 1.4. Il est parfois arrivé, malgré tout, que de bonnes conjectures soient obtenues à l'aide de cette méthode, mais ces dernières avaient déjà été trouvées par d'autres moyens. Afin que cette

méthode mène à de bon résultats, il faudrait sans doute l'arrêter avant la fin si nous voulons conserver un assez grand nombre de graphes. En se référant à la figure 1.4, nous remarquons que l'information critique de cet exemple disparaît lors de l'étape 2. La performance de cette méthode dépend du respect des hypothèses associées à la régression, et particulièrement l'homocédasticité. Malheureusement, cette hypothèse n'est que rarement satisfaite, ce qui engendre les dysfonctionnements représentés sur cette figure.

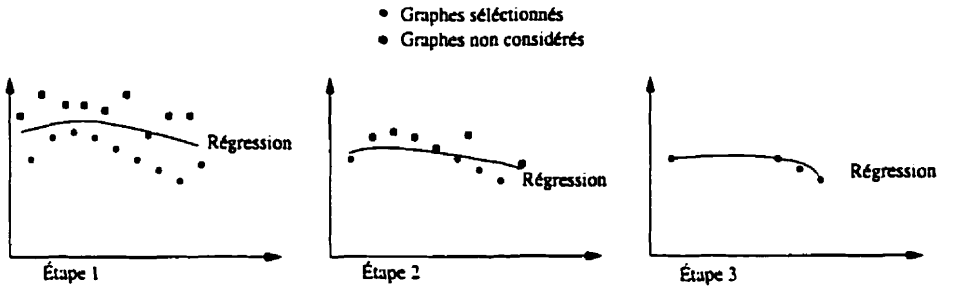


Figure 1.4 – Sélection de graphes extrêmes par régression non linéaire

1.4.2 Une méthode analytique

Cette approche peut être considérée comme une formalisation automatisée de certains aspects du raisonnement humain. La première conjecture trouvée à l'aide d'**AGX** donne une borne inférieure sur l'indice de Randić, en fonction du *nombre chromatique* γ et de l'ordre n du graphe. Rappelons que le nombre chromatique d'un graphe est le nombre minimum de couleurs requises pour qu'à chaque sommet du graphe soit associée une couleur sans que deux sommets adjacents n'aient la même couleur. Une étude assez poussée de l'indice de Randić Ra est donnée lors de la section 2.1. Cet indice a été défini dans la section 1.3.

Conjecture 1 *Pour tout arbre connexe G,*

$$Ra(G) \geq \frac{\gamma(G)-2}{2} + \frac{1}{\sqrt{n-1}}(\sqrt{\gamma(G)-1} + n - \gamma(G)) \tag{1.9}$$

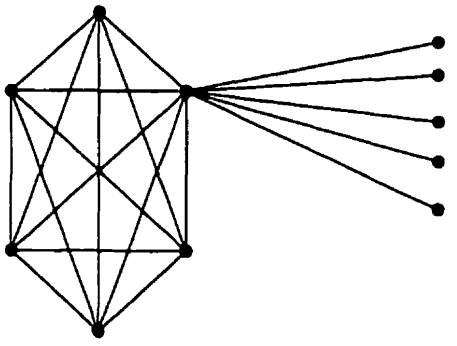


Figure 1.5 - Graphe d'indice Ra conjecturé minimum, avec $n = 11$ et $\gamma = 6$.

Elle fut trouvée en cherchant systématiquement les graphes d'indice Ra minimum, alors que l'ordre n et le nombre chromatique γ du graphe sont fixés comme paramètres. Nous avons instantanément identifié une structure commune à tous les graphes extrêmes obtenus. En effet, tous les graphes étaient composés d'une clique de cardinalité $\gamma(G)$ et de $n - \gamma(G)$ sommets pendants tous connectés au même sommet de la clique, comme l'illustre la figure 1.5.

Pour n et γ donnés, le graphe correspondant est totalement défini. De plus, $Ra(G)$ est calculable par une formule analytique avec n et γ comme paramètres. En effet, les sommets se répartissent en un sommet s de degré $n-1$, $\gamma-1$ sommets de degré $\gamma-1$, qui forment une clique avec le sommet s, et $n-\gamma$ sommets pendants reliés à s. Il s'ensuit que le graphe comporte $\frac{(\gamma-1)(\gamma-2)}{2}$ arêtes reliant des sommets de degré $\gamma-1$, $\gamma-1$ arêtes reliant un sommet de degré $n-1$ et un sommet de degré $\gamma-1$, et $n-\gamma$ arêtes reliant un sommet de degré $n-1$ à un sommet pendant. La conjecture 1 découle directement de ces informations et de la définition (1.7) de l'indice de Randić.

Nous nous sommes ensuite intéressés à la manière dont nous avons trouvé cette conjecture. Partant d'un ensemble de graphes supposés extrêmes, nous avons dans un premier temps cherché leurs caractéristiques communes. La seconde étape a consisté à identifier avec exactitude une famille de graphes à laquelle ils se rattachent. Dans la troisième étape, nous avons exprimé une relation entre la fonction-objectif et les paramètres utilisés pour trouver les graphes dérivant de la famille de graphes extrêmes identifiée. Il était alors naturel d'essayer d'automatiser cette méthode dans **AGX**.

Partant d'un ensemble de graphes supposés extrêmes, trouvés par sa partie optimisation, et d'un petit ensemble de règles, le programme identifie si ces graphes appartiennent à une classe connue. Il est surprenant de constater qu'un ensemble assez restreint de familles suffit souvent pour caractériser les graphes extrêmes.

Pour certaines de ces familles, il est possible de caractériser complètement un graphe par son nombre de sommets n. Nous appellerons ce type de familles les **familles simples** de graphes. Pour elles, les invariants peuvent en général être définis par une formule algébrique simple avec comme seul paramètre n. Les *étoiles*, *chemins*, *graphes complets* et *cycles* sont des exemples de telles familles.

Dans de nombreux cas, les familles auxquelles appartiennent les graphes extrêmes sont des *familles simples*; il est alors possible de remplacer les invariants qui apparaissent dans la fonction-objectif par une fonction de n, ce qui nous donne immédiatement une borne serrée pour la formule originale en fonction du seul paramètre n.

La méthode fut appliquée pour essayer de resserrer les 12 conjectures parmi les 30 premières de *Written on the Wall (WoW)* obtenues à l'aide de *Graffiti* [52] [56] pour lesquelles tous les invariants requis étaient déjà dans **AGX**. Ce dernier programme a été utilisé pour trouver des graphes conjecturés extrémaux, graphes pour lesquels

la différence entre le membre de gauche et celui de droite de l'inégalité est minimale. Des tests ont ensuite été faits pour identifier la classe à laquelle les graphes extrêmes appartiennent. Selon la famille extrême trouvée, il était alors possible de remplacer la fonction-objectif par une relation en n à l'aide d'une base de formules connues. 7 des 12 conjectures testées ont été automatiquement renforcées par le système.

Une routine supplémentaire a été ajoutée au système pour trouver des contre-exemples aux conjectures testées. En effet, la formule algébrique donnant la fonction-objectif a été évaluée pour $n = 5, \ldots, 10000$. Le système peut ainsi trouver des contre-exemples à certaines conjectures sans pour autant les construire, comme ce fut le cas pour la *Conjecture WoW :16.*

Les conjectures testées ainsi que les résultats obtenus sont les suivants.

- **Conjecture WoW :3**
  $RANDIC - AVDIST \geq 0$
  AGX nous donne :
  IT SEEMS THAT EXTREMAL GRAPHS ARE TREES.
  Cette information n'est pas suffisante pour trouver une formule algébrique pour $RANDIC$ ou $AVDIST$.

- **Conjecture WoW :4**
  $VARDEG + SUMINVDEG - AVDIST \geq 0$
  AGX nous donne :
  IT SEEMS THAT EXTREMAL GRAPHS ARE COMPLETE GRAPHS.
  I SUGGEST THAT :
  $VARDEG + SUMINVDEG - AVDIST \geq 0 + (NBNOD/(NBNOD - 1)) - 1$
  d'où la nouvelle conjecture
  Conjecture 2 *Pour tout graphe G :*

$$\sigma^2(\delta) + \sum_{i \in V} \frac{1}{\delta_i} - \bar{d} \geq \frac{n}{n-1} - 1 \tag{1.10}$$

qui s'écrit aussi

$$\sigma^2(\delta) + \sum_{i \in V} \frac{1}{\delta_i} - \bar{d} \geq \frac{1}{n-1} \tag{1.11}$$

cette conjecture tend asymptotiquement vers la conjecture **WoW :4** *quand n tend vers l'infini.*

- **Conjecture WoW :5**

  $MODEDIST + SUMINVDEG - AVDIST \geq 0$

  AGX nous donne :

  IT SEEMS THAT EXTREMAL GRAPHS ARE COMPLETE GRAPHS.

  I SUGGEST THAT :

  $MODEDIST + SUMINVDEG - AVDIST \geq 1+$

  $(NBNOD/(NBNOD-1)) - 1$

  d'où la nouvelle conjecture

  Conjecture 3 *Pour tout graphe G :*

$$mode(d) + \sum_{i \in V} \frac{1}{\delta_i} - \bar{d} \geq \frac{n}{n-1}. \tag{1.12}$$

- **Conjecture WoW :7**

  $RADIUS + RANDIC - MODEDIST \geq 0$

  AGX nous donne :

  IT SEEMS THAT EXTREMAL GRAPHS ARE STARS.

  I SUGGEST THAT :

  $RADIUS + RANDIC - MODEDIST \geq 1 + SQRT(NBNOD - 1) - 2$

  IF $NBNOD > 3$.

  d'où la nouvelle conjecture :

  Conjecture 4 *Pour tout graphe G d'ordre $n > 3$*

$$r + Ra - mode(d) \geq \sqrt{n-1} - 1 \tag{1.13}$$

- **Conjecture WoW :8**

  $AVDIST + RANDIC - MODEDIST \geq 0$

  AGX nous donne :

  IT SEEMS THAT EXTREMAL GRAPHS ARE STARS.

  I SUGGEST THAT :

  $AVDIST + RANDIC - MODEDIST \geq ((NBNOD - 1) * 2/NBNOD) + SQRT(NBNOD - 1) - 2$

  IF $NBNOD > 3$,

  d'où la nouvelle conjecture :

  Conjecture 5 *Pour tout graphe G d'ordre* $n > 3$

$$\bar{d} + Ra - mode(d) \geq \frac{2(n-1)}{n} + \sqrt{n-1} - 2 \tag{1.14}$$

  ou

$$\bar{d} + Ra - mode(d) \geq \sqrt{n-1} - \frac{2}{n}. \tag{1.15}$$

- **Conjecture WoW :12**

  $1 + RANDIC - RADIUS \geq 0$

  AGX nous donne :

  IT SEEMS THAT EXTREMAL GRAPHS ARE PATHS.

  I SUGGEST THAT :

  $1 + RANDIC - RADIUS \geq 1 + ((NBNOD - 3)/2 + SQRT(2))$
  $-FLOOR(NBNOD/2)$

  IF $NBNOD > 2$.

  Par une analyse de graphes extrêmes sélectionnés par la méthode de la moyenne flottante, en l'occurence des chemins, d'où la nouvelle conjecture :

Conjecture 6 *Pour tout graphe G d'ordre $n > 2$*

$$Ra - r \geq \frac{n-3}{2} + \sqrt{2} - \lfloor \frac{n}{2} \rfloor \tag{1.16}$$

ou

$$Ra - r \geq \frac{nmod2}{2} + \sqrt{2} - \frac{1}{2}. \tag{1.17}$$

- **Conjecture WoW :13**

  Pour tout graphe G connexe

  $SUMINVDEG + AVDIST - RADIUS \geq 0$

  AGX a trouvé que les graphes sélectionnés par la méthode de la moyenne flottante sont des graphes réguliers, mais cette information n'est pas suffisante pour définir avec précision un graphe. Il faudrait poursuivre l'analyse pour les graphes réguliers en fonction du degré.

- **Conjecture WoW :14**

  Pour tout graphe G connexe

  $AVDIST + RANDIC - RADIUS \geq 0.$

  AGX n'a trouvé aucun résultat.

- **Conjecture WoW :15**

  $VARDEG + RANDIC - RADIUS \geq 0$

  AGX nous donne :

  IT SEEMS THAT EXTREMAL GRAPHS ARE CYCLES.

  I SUGGEST THAT :

  $VARDEG + RANDIC - RADIUS \geq 0 + (NBNOD/2) - FLOOR(NBNOD/2),$

  d'où la nouvelle conjecture :

Conjecture 7 *Pour tout graphe G d'ordre n*

$$\sigma^2(\delta) + Ra - r \geq \frac{n}{2} - \lfloor n-2 \rfloor. \tag{1.18}$$

Cet exemple montre que l'étude de conjectures à l'aide d'un petit nombre d'exemples de petite taille conduit parfois à des résultats incertains puisque la conjecture initiale est fausse pour les chemins avec un nombre pair de sommets $n \geq 22$, comme il a par ailleurs été remarqué à l'aide d'**AGX** (voir [27]). La conjecture 7 est donc fausse elle aussi.

- **Conjecture WoW :16**

  $AVGTMP + RANDIC - RADIUS \geq 0$

  AGX nous donne :

  IT SEEMS THAT SELECTED EXTREMAL GRAPHS ARE PATHS

  I SUGGEST THAT :

  $AVGTMP + RANDIC - RADIUS \geq (2/(NBNOD-1)) + ((NBNOD-3)/2 + SQRT(2)) - FLOOR(NBNOD/2)$ *IF* $NBNOD > 2$.

  d'où la nouvelle conjecture :

$$\frac{1}{n}\sum_{j=1}^{n} \frac{\delta_j}{n-\delta_j} + Ra - r \geq \frac{2}{n-1} + \frac{n-3}{2} + \sqrt{2} - \lfloor \frac{n}{2} \rfloor \tag{1.19}$$

ou

$$\frac{1}{n}\sum_{j=1}^{n} \frac{\delta_j}{n-\delta_j} + Ra - r \geq \frac{2}{n-1} + \sqrt{2} - \frac{3}{2} + \frac{n \bmod 2}{2}. \tag{1.20}$$

En calculant la valeur du membre de droite de cette expression pour $n = 3 \ldots 10000$, le système a trouvé qu'elle était négative si n est égal à 26. Le chemin à 26 sommets est donc identifié comme contre-exemple à la conjecture **WoW :16** sans qu'il soit nécessaire de le construire. Une analyse ultérieure nous a permis de constater que tous les chemins pairs de 26 sommets ou plus sont des contre-exemples à cette conjecture.

- **Conjecture WoW :27**

  $RANDIC - SQRT(VARDEG) \geq 0$

  AGX nous donne :

  IT SEEMS THAT EXTREMAL GRAPHS ARE STARS.

  I SUGGEST THAT :

  $RANDIC - SQRT(VARDEG) \geq SQRT(NBNOD - 1) - SQRT((NBNOD - 3 + (2/NBNOD)))$

  IF NBNOD > 2.

  Nous en déduisons la conjecture

  Conjecture 8 *Pour tout graphe G d'ordre n*

$$Ra - \sqrt{\sigma^2(\delta)} \geq \sqrt{n-1} - \sqrt{n - 3 + (\frac{2}{n})}. \tag{1.21}$$

L'utilisation de la méthode analytique a aussi permis la réfutation automatique d'une des conjectures de Graffiti, soit **WoW : 16**.

Cette méthode a aussi permis de renforcer 8 conjectures sur 12. Pour la conjecture réfutée par le système, le terme renforcement n'est assurément pas adapté puisque **AGX** a plutôt proposé une correction.

Une approche possible pour améliorer cette méthode de recherche de conjectures est proposée dans la section 3.2.1.

1.4.3 Une méthode numérique

Étant donné un ensemble d'observations sur plusieurs variables, trouver efficacement des relations qu'elles respectent est un problème fondamental en découverte de connaissances dans les bases de données (*Knowledge Discovery in Databases*, ou KDD, en anglais). Du fait de l'informatisation croissante des entreprises qui disposent

donc de très grandes bases de données, cette discipline est en pleine expansion. Deux types d'applications sont considérées :

- la classification, qui permet une discrimination de la population, ce qui est très utile pour l'analyse et la segmentation de marchés par exemple,
- la recherche d'une information synthétique susceptible de décrire les données. Un modèle linéaire ou non linéaire, est par exemple utilisé pour la conception de modèles de prédiction. Dans ce cas, une fonction générique est considérée tandis que ses paramètres sont évalués grâce à la base de données.

Dans le cas qui nous concerne ici, comme pour beaucoup d'applications de découverte scientifique, l'utilisation de données et la recherche de relations qu'elles respectent est déterminante [93], [112], et s'apparente à la recherche de connaissances dans des bases de données. Plus précisément, nous cherchons des relations au sein d'un sous-ensemble des variables vérifiées par toutes les observations considérées. Les méthodes généralement utilisées relèvent de la régression ou bien, quand la taille ou la structure des problèmes ne permet pas l'utilisation de cette dernière, des réseaux de neurones, comme l'expliquent Fayyad et ses collaborateurs [61]. En effet, considérer tous les sous-ensembles de variables possibles permet de résoudre le problème mais se traduit par une explosion combinatoire. La recherche de paramètres minimisant la somme des erreurs revient à un problème d'optimisation globale si le modèle n'est pas linéaire. Nous voulons montrer ici que dans le cas déterministe, cas où il existe des relations strictement vérifiées pour toutes les observations, ce problème peut être résolu en temps polynomial pour de larges classes de relations. Plus précisément, nous proposons un algorithme polynomial pour trouver une base de toutes les relations affines entre les variables (ou des puissances, quotients, produits ou logarithmes de celles-ci). Cet algorithme utilise des résultats simples d'algèbre linéaire qui sont aussi utilisés en analyse en composantes principales [106]. Toutefois, le but recherché en analyse en composantes principales est d'expliquer les différences entre les observations alors que nous cherchons à identifier les propriétés communes à ces observations.

Nous décrirons d'abord l'algorithme utilisé, puis nous l'illustrerons avec quelques exemples classiques de découverte, déjà étudiés à l'aide du programme Bacon [93]. Une relation portant sur cinq invariants, découverte à l'aide d'une version totalement automatisée d'**AGX** sera ensuite exposée ainsi que deux autres conjectures découvertes lors de l'analyse de la première. D'autres conjectures obtenues à l'aide du système seront présentées au chapitre 2 qui porte sur les applications d'**AGX**.

Algorithme

Considérons un ensemble de m observations sur n variables quantitatives $x_1, x_2, \ldots, x_n$. Soit $X_{m\times n} = (x_{ij})$ l'ensemble de données correspondantes. Notre but est de trouver une base de l'ensemble des relations affines (les relations de la forme $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$, où les a_j et b sont des constantes) respectées par les observations. Soit $\mu = \frac{1}{m}{}^tX1_m$ le vecteur des moyennes sur les colonnes de X. Alors, toutes les relations affines entre les colonnes de X deviennent des relations linéaires (de la forme $a'_1x'_1 + a'_2x'_2 + \cdots + a'_nx'_m = 0$) entre les colonnes de $X' = X - (1_m{}^t\mu)$, la matrice des données centrées. De plus, tous les coefficients (à l'exception de b) restent inchangés. En effet, si $x_1 = \sum_{j=2}^{n} c_jx_j + d$ alors

$$\begin{aligned}\mu_1 &= \frac{1}{m}\sum_{i=1}^{m}\Big(\sum_{j=2}^{n} c_jx_{ij} + d\Big)\\ &= \sum_{j=2}^{n}\frac{1}{m}\sum_{i=1}^{m} c_jx_{ij} + \sum_{i=1}^{m}\frac{d}{m} = \sum_{j=2}^{n} c_j\mu_j + d\end{aligned}$$

et

$$\begin{aligned} x'_1 &= x_1 - \mu_1 \\ &= \sum_{j=2}^{n} c_j x_j + d - (\sum_{j=2}^{n} c_j \mu_j + d) \\ &= \sum_{j=2}^{n} c_j (x_j - \mu_j) = \sum_{j=2}^{n} c_j x'_j. \end{aligned}$$

Centrer les variables, la première étape de notre algorithme, transforme donc le problème de trouver des relations affines en celui de trouver des relations linéaires.

Considérons alors la matrice de variance-covariance V définie par $V_{n\times n} = {}^t X' X'$.

Si la relation

$$x'_j = \sum_{\ell=1,\ell\neq j}^{n} c_\ell x'_\ell \tag{1.22}$$

est vraie, alors

$$\begin{aligned} v_{jk} &= \sum_{i=1}^{m} x'_{ij} x'_{ik} = \sum_{i=1}^{m} \Big(\sum_{\ell=1,\ell\neq j}^{n} c_\ell x'_{i\ell} \Big) x'_{ik} \\ &= \sum_{\ell=1,\ell\neq j}^{n} c_\ell \sum_{i=1}^{m} x'_{i\ell} x'_{ik} = \sum_{\ell=1,\ell\neq j} c_\ell v_{\ell k}, \end{aligned}$$

ce qui veut dire que si la relation linéaire (1.22) est vraie pour les colonnes de X', elle l'est également pour celles de V. Comme V est symétrique, cette relation est encore vérifiée pour ses lignes.

La seconde étape de notre algorithme est alors de calculer V.

La troisième étape consiste à diagonaliser V (avec toutefois des lignes vides s'il y a des relations). Ceci peut être effectué par élimination Gaussienne. Dans la matrice ainsi obtenue, V', $Dim(Im(V))$ lignes contiennent des termes non nuls et correspondent à

des variables indépendantes. Les $n - Dim(Im(V))$ lignes restantes ne contiennent que des zéros et correspondent aux variables dépendantes qui peuvent s'exprimer comme combinaisons linéaires des variables indépendantes. Ces relations forment une base de l'espace nul de V. À l'aide des données originales, il est ensuite aisé de calculer les membres de droite correspondant aux relations affines.

Le cas où une variable dépendante est exprimée sous la forme d'un monôme de plusieurs variables indépendantes, au lieu d'une relation affine, peut aisément se rapporter au cas précédent.

En effet, si

$$x'_j = f \prod_{\ell=1,\ell\neq j}^{n} (x'_\ell)^{e_\ell} \tag{1.23}$$

où e_ℓ et f sont des constantes, en passant aux logarithmes des variables, cette relation devient

$$\log x'_j = \sum_{\ell=1,\ell\neq j}^{n} e_\ell \log(x'_\ell) + \log(f). \tag{1.24}$$

La méthode décrite ci-dessus peut alors être utilisée pour trouver les exposants e_ℓ (égaux à 0 si x'_ℓ n'apparaît pas dans le monôme) ainsi que la valeur f.

Les deux classes de relations décrites précédemment peuvent être aisément et substantiellement étendues en calculant des termes supplémentaires à partir des données originales. Le cas le plus simple consiste à prendre, en plus des données originales, tous les carrés et produits de paires de variables. Nous utilisons alors $n + \frac{n^2+n}{2} = \frac{n^2+3n}{2}$ variables au lieu de n. Les formules sont du type

$$x_j = \sum_{\ell=1,\ell\neq j}^{n} c_\ell x_\ell + \sum_{k=1,k\neq j}^{n} \sum_{\ell=k,\ell\neq j}^{n} c_{k\ell} x_k x_\ell + d.$$

Les ratios, puissances supérieures à deux ou les produits de trois variables peuvent

aussi être considérés. Si x_j a la forme d'un monôme, les sommes ou différences de variables peuvent être prises comme termes additionnels.

Complexité

La première étape de la méthode, *calculer les valeurs centrées*, exige le calcul des termes μ, ce qui requiert $O(mn)$ opérations. Nous devons ensuite soustraire μ_j de chaque x_{ij}, ce qui requiert à nouveau $O(mn)$ opérations.

La seconde étape, *calculer la matrice de variance-covariance* V , prend $O(mn^2)$.

La troisième étape, *la diagonalisation de* V par élimination gaussienne requiert aussi $O(n^3)$.

Afin d'éviter de trouver des relations aberrantes, il faut que $m \geq n$. On en déduit que la complexité globale de l'algorithme est $O(mn^2)$ (ou $O(n^3)$ sous l'hypothèse raisonnable que m est $O(n)$). Si les carrés et produits de termes sont considérés, la complexité s'élève à $O(n^6)$. De tels problèmes restent possibles à résoudre en un temps raisonnable lorsque n est modéré.

A titre de comparaison, utiliser la régression linéaire permet de trouver au plus une relation à la fois alors que nôtre méthode trouve une base de l'ensemble des relations présentes dans les données. De plus, la régression linéaire nécessite qu'une variable soit expliquée par les autres, ce qui implique qu'une régression doive être calculée pour chacune des $n-1$ premières variables à expliquer par les suivantes. De plus, afin de s'assurer que toutes les relations soient identifiées, il faut utiliser le critère du "meilleur sous-ensemble" pour chaque régression, ce qui implique l'essai de chacune des combinaisons de variables, un algorithme d'énumération non polynomial et une complexité exponentielle. Analyser un exemple de 20 variables et 44 observations avec le logiciel de statistique SAS requiert 10 secondes alors que 0.0013 seconde suffit

à notre méthode. Pour un exemple avec 25 variables au lieu de 20 et toujours 44 observations, le temps mis par SAS s'élève à 4 minutes et 30 secondes tandis que 0.002 seconde suffit à notre algorithme (qui est donc 130 000 fois plus rapide).

Applications

Avant d'utiliser cette méthode pour découvrir de nouvelles conjectures en théorie des graphes, nous l'avons testée sur des exemples tirés de la physique.

La troisième loi de Kepler

La troisième loi de Kepler dit que $p^2/d^3 = c$ où p représente la période de l'orbite d'une planète, d sa distance moyenne au soleil et c une constante. Comme toutes les lois exposées dans cette section, cette loi a été redécouverte par un des programmes Bacon [93], en un temps de calcul supérieur à celui de notre méthode. Notons toutefois que le but de ces programmes est différent du nôtre : comprendre le raisonnement qui conduit à la découverte plutôt que construire une méthode efficace mais dépendant de l'ordinateur pour arriver à l'identification de la relation.

Les quatre observations suivantes de p et d ont été simulées (en choisissant les unités de telle manière que $c = 1$) :

$$X = \begin{pmatrix} 9,736442 & 30,380885 \\ 1,371510 & 1,606196 \\ 1,869378 & 2,555911 \\ 2,456725 & 3,850657 \end{pmatrix}$$

ou, en passant aux logarithmes,

$$X_{log} = \begin{pmatrix} 2,275875 & 3,413813 \\ 0,315912 & 0,473868 \\ 0,625605 & 0,938408 \\ 0,898829 & 1,348243 \end{pmatrix}.$$

La matrice de variance-covariance correspondante est

$$V = \begin{pmatrix} 0,507597 & 0,507597 \\ 0,507597 & 0,507597 \end{pmatrix}.$$

Sa diagonalisation nous conduit à

$$V' = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

qui montre qu'une relation de la forme $\log(p) = c' \log(d)$ est vérifiée. Alors, en utilisant X_{log}, nous trouvons $c' = 1,5$. Une forme équivalente de la relation originale est $p = d^{1.5}$. Les temps de calcul sont trop petits pour être mesurés.

La loi des gaz parfaits

La loi des gaz parfaits s'écrit $PV = nRT$ où P est la pression (en pascals), V le volume (en mètres cubes), n le nombre de moles et T la température (en kelvins). $R = 8.32$ est la constante universelle des gaz parfaits. Le générateur a produit des simulations de données de P, V, n et T. Celles-ci ont été converties en anciennes unités afin d'être conformes à la situation qui prévalait lors de la découverte de cette loi, avant que les températures absolues ne soient utilisées. A cette époque, la pression p était mesurée en atmosphères, le volume v en litres et la température t en degrés Celsius. Les règles de conversion sont $P = 101325p$, $V = v/1000$ et $T = t + 273,15$.

En utilisant 25 observations et l'ensemble étendu de variables comprenant les carrés et produits de paires de variables nous avons obtenu, en moins d'un centième de seconde, la relation $nt = -273,15n + 12,1785pv$. En convertissant à nouveau cette relation en unités du système international, nous trouvons $nT = 0,120192PV$, qui est une forme équivalente de la loi présentée plus haut.

Notons que cette relation est encore plus facile à trouver en utilisant les unités du système international, et les logarithmes des valeurs. En ce cas un ensemble de données étendu n'est pas nécessaire. 16 observations suffisent en effet et la matrice de variance-covariance pour ces données est

$$V = \begin{pmatrix} 0,506 & 0,049 & -0,076 & 0,631 \\ 0,049 & 0,324 & 0,009 & 0,364 \\ -0,076 & 0,009 & 0,279 & -0,346 \\ 0,631 & 0,364 & -0,346 & 1,343 \end{pmatrix}.$$

Après diagonalisation, cette matrice devient

$$V' = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

qui met en évidence la relation $\log(T) = \log(P) + \log(V) - \log(n) + c$.

En utilisant les logarithmes des données, nous obtenons $c = 0,120192$. Dans ce cas, la forme équivalente $T = 0,120192PVn^{-1}$ de la loi des gaz parfaits est trouvée instantanément.

La loi de la gravitation universelle de Newton

La loi de Newton dit que $F = G\frac{mM}{D^2}$ où m et M sont les masses de deux corps, D la distance qui les sépare, F leur force d'attraction et $G = 6.67e^{-11}$ la constante

de la gravitation universelle. Dans sa forme, le problème est identique au précédent, puisque nous cherchons à nouveau un monôme impliquant trois variables.

Le générateur nous a donné 25 observations simulées de F, m, M et D. En utilisant les logarithmes des valeurs, le système a identifié la relation $F = mMD^{-2}6.67e^{-11}$ qui est équivalente à la loi donnée ci-dessus. À nouveau, cette relation fut trouvée instantanément.

En plus de ce résultat, le système a donné les relations aberrantes $m + F = m$, $M + F = M$ et $D + F = D$. Il est facile de voir qu'elles sont incorrectes, et causées par des erreurs numériques dûes à de trop grandes différences d'ordre de grandeur. Une mise à l'échelle des variables préalablement aux calculs aurait permis d'éviter cette erreur.

La loi d'Ohm

La loi d'Ohm stipule que $IL = rI + v$ où I représente l'intensité du courant, L l'inductance, r la résistance et v la tension. Le générateur a simulé 25 observations de I, L, r et v. En utilisant l'ensemble de variables étendu par ajout des carrés et produits de paires de variables, soit 14 termes (ou 10 si nous cherchons une expression sous forme d'un monôme puisqu'ajouter 2 fois la même variable est redondant), la relation $rI = -v + IL$ fut trouvée en moins d'un centième de seconde.

Notons que l'utilisation des variables originales seules n'aurait pas permis de trouver cette relation.

Exemple en théorie des graphes : découverte d'une propriété des arbres

avec contrainte de coloration d'index minimum

Nous appelons **index** d'un graphe la première (plus grande) valeur propre de sa matrice d'adjacence. La description qui suit donne un exemple d'utilisation de cette méthode de recherche de conjectures. D'autres relations ont été trouvées lors d'études plus vastes illustrant l'utilisation d'**AGX** dans son intégralité. À ce titre, elles seront exposées dans le chapitre 2 consacré aux applications du programme.

Un arbre est un graphe biparti ; on peut donc en colorier les sommets en noir et blanc sans que deux sommets adjacents n'aient la même couleur. Si les nombres de sommets noirs et blancs sont fixés (respectivement a et b avec $a \geq b$), on dit que l'arbre vérifie une *contrainte de coloration*. Dans [41], AGX est utilisé pour étudier les arbres d'index extrémal en fonction de contraintes de coloration.

Lors de cette étude, décrite dans la section 2.3, six conjectures ont été trouvées interactivement à l'aide d'**AGX**, et quatre d'entre elles ont ensuite été prouvées.

Nous nous intéresserons ici à une exploration automatisée des arbres extrémaux trouvés qui donna lieu à une conjecture de nature totalement différente. De cette nouvelle conjecture, deux autres ont ensuite été déduites, toujours à l'aide d'**AGX**. Ces deux dernières ont ensuite été prouvées.

L'algorithme décrit plus tôt a été utilisé pour trouver une base des relations affines entre les quinze invariants suivants : nombre n de sommets, nombre n_1 de sommets pendants (sommets de degré 1), nombre m d'arêtes, diamètre D, rayon r, nombre de stabilité α (nombre maximum de sommets deux à deux non adjacents), degré moyen $\bar{\delta}$, distance moyenne et somme des distances entre paires de sommets, énergie (somme des valeurs absolues des valeurs propres de la matrice d'adjacence), degré maximum,

index (plus grande valeur propre de la matrice d'adjacence), indice Hyperwiener (somme des carrés des distances entre paires de sommets), indice de Randić Ra (défini dans la section 1.3), et nombre chromatique γ.

En plus de relations connues, telles que $m = n - 1$ et $\gamma = 2$ qui sont vérifiées pour tous les arbres, la relation suivante, fort inattendue, a été découverte :

$$2\alpha - m - n_1 + 2r - D = 0 \tag{1.25}$$

et nous savons alors que les huit invariants restants sont linéairement indépendants de tous ceux considérés.

Le résultat (1.25) peut aussi s'exprimer comme suit :

Conjecture 9 *Pour tout arbre avec contrainte de couleurs et d'index minimum*

$$\alpha = \frac{1}{2}(m + n_1 + D - 2r). \tag{1.26}$$

De plus, AGX a été utilisé pour voir si cette conjecture pouvait être étendue à tous les arbres. Minimiser et maximiser le membre gauche de (1.25) nous a ensuite conduits aux résultats suivants.

Théorème AGX 1 *Pour tout arbre*

$$\alpha \leq \frac{1}{2}(m + n_1 + D - 2r). \tag{1.27}$$

Preuve:

Rappelons que pour un arbre, $D = 2r$ si D est pair et $D = 2r - 1$ si D est impair. Donc, $D - 2r = -(D \bmod (2))$.

Supposons d'abord que D soit pair. Définissons l'*étoile* d'un sommet v_i comme étant l'ensemble d'arêtes incidentes à v_i. il est clair que les étoiles des sommets d'un ensemble stable sont disjointes. Considérons un ensemble stable S de taille maximale α

avec n_1 sommets pendants et α_2 sommets intérieurs. Alors, $m \geq n_1 + 2\alpha_2 = 2\alpha - n_1$ et $\alpha \leq \frac{1}{2}(m + n_1)$.

Supposons maintenant que D soit impair. Considérons un chemin P de longueur D ; ou bien un seul sommet pendant de P appartient à S, ou bien il existe une paire de sommets consécutifs v_i et v_j de P qui n'appartiennent pas à S, et en ce cas l'arête (v_i, v_j) n'appartient à l'étoile d'aucun sommet de S. Dans les deux cas, $m \geq 2\alpha - n_1 + 1$ et $\alpha \leq \frac{1}{2}(m + n_1 - 1)$. □

Soit $\lfloor a \rfloor$ le plus grand entier inférieur ou égal à a.

Théorème AGX 2 *Pour tout arbre*

$$\alpha \geq \frac{1}{2}\left(m + n_1 + D - 2r - \left\lfloor \frac{n-2}{2} \right\rfloor\right). \tag{1.28}$$

Preuve:

Il est facile de vérifier que ce résultat est vrai pour $n = 1$ ou 2. Supposons que $n \geq 3$ et considérons l'heuristique gloutonne suivante :

Posons $S = \emptyset$, ajoutons récursivement un sommet pendant ou isolé à S et supprimons le sommet adjacent (s'il y en a un) et les arêtes de son étoile mais pas les sommets de celle-ci. De cette manière, un ensemble stable avec au moins $\lceil \frac{n}{2} \rceil$ sommets est construit, alors $\alpha \geq \lceil \frac{n}{2} \rceil$. Observons aussi que tous les sommets pendants forment un ensemble stable, d'où $\alpha \geq n_1$ et $\alpha \geq \frac{1}{2}(\lceil \frac{n}{2} \rceil + n_1)$. Alors, si n est pair, $m - \lfloor \frac{n-2}{2} \rfloor = n - 1 - \frac{n-2}{2} = \frac{n}{2} = \lceil \frac{n}{2} \rceil$, et si n est impair, $m - \lfloor \frac{n-2}{2} \rfloor = m - \frac{n-3}{2} = \frac{n+1}{2} = \lceil \frac{n}{2} \rceil$. D'où, $\alpha \geq \frac{1}{2}(m + n_1 + \lfloor \frac{n-2}{2} \rfloor)$.

Comme $D - 2r \leq 0$, le résultat s'ensuit. □

1.4.4 Une méthode géométrique

À partir d'un ensemble de graphes, il est possible de calculer les valeurs de certains invariants, comme nous l'avons fait pour rechercher des conjectures avec la méthode

numérique. Si nous calculons ensuite l'enveloppe convexe des points représentant ces graphes dans l'espace des invariants, nous obtenons par définition des inégalités respectées par l'ensemble des graphes considérés.

Nous avons pensé utiliser cette technique pour identifier des conjectures caractérisant les graphes extrêmes trouvés par **AGX**. Une telle approche permet de trouver des conjectures du type "Si G est un graphe avec entre 10 et 20 sommets tel que l'indice $I(G)$ soit minimum, alors $I_1(G) \leq I_2(G)$". Pour trouver des conjectures intéressantes, il est nécessaire qu'elles soient plus générales, et que le nombre de sommets (ou tout paramètre utilisé pour la recherche des graphes extrêmes) n'intervienne pas dans les hypothèses. Nous avons décidé alors de procéder par deux étapes successives. Dans un premier temps, nous enlevons les graphes dont les paramètres ont des valeurs extrêmes. En un second temps, nous évaluons les inégalités trouvées à l'aide de tous les graphes, et retirons celles qui ne sont alors plus valides, comme l'indique la figure 1.6. Nous espérons alors avoir trouvé des conjectures assez robustes pour être étudiées avec intérêt.

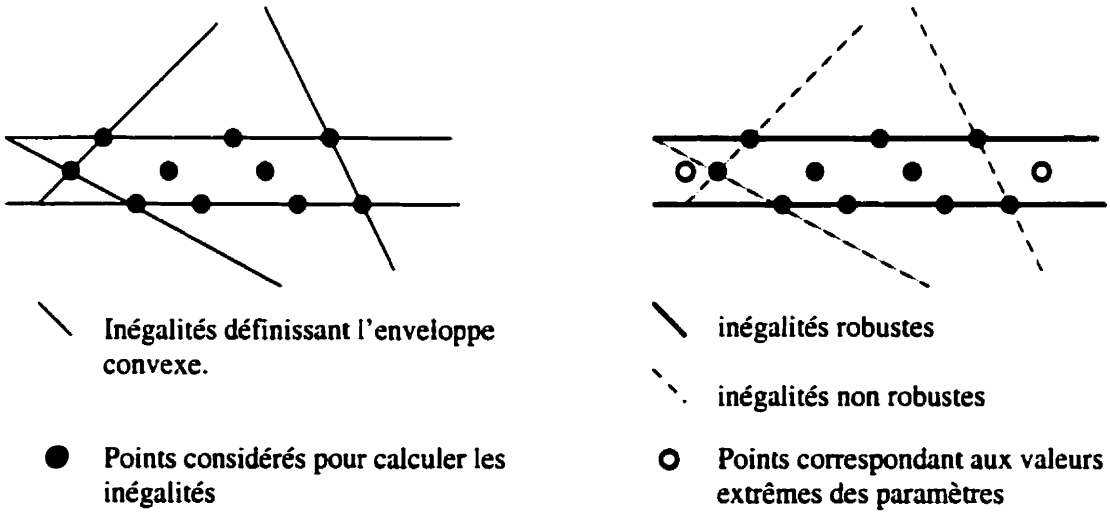


Figure 1.6 – Les deux étapes de la recherche de conjectures géométrique

Lorsque nous avons essayé cette approche, elle nous a donné des résultats peu élégants

et inappropriés à toute étude quand nous utilisions plusieurs invariants graphiques. De plus, les résultats ainsi obtenus ne permettaient pas de guider notre imagination pour l'identification de nouveaux résultats, contrairement à la méthode numérique qui conduit souvent à des caractérisations au moins partielles des graphes obtenus. Dans certains cas, toutefois, la méthode géométrique nous a indiqué des relations intéressantes quand on considérait un ou deux paramètres ainsi que la fonction-objectif. Nous devons par contre reconnaître que dans le cas où deux paramètres étaient considérés, elle n'a jamais permis d'identifier aucune conjecture qu'une autre méthode n'avait pas permis de trouver. Comme il est difficile de représenter clairement à l'écran l'enveloppe convexe en trois dimensions (deux paramètres et l'objectif), l'approche géométrique n'a pas été programmée dans **AGX** de manière générale. Une routine de calcul des équations des droites composant l'enveloppe convexe inférieure pour les problèmes de minimisation et supérieure dans le cas contraire est disponible en deux dimensions. Cette routine s'est montrée très pratique car elle utilise l'interface graphique et peut être appelée interactivement. De plus, la droite obtenue est représentée d'une couleur différente si elle est vérifiée comme une égalité par plus de deux points, ce qui donne immédiatement une conjecture. La figure 1.7 illustre cette fonction dans le cas de l'étude de la distance à la palindromicité du polynôme de Hosoya pour des arbres de diamètre impair (voir section 2.4).

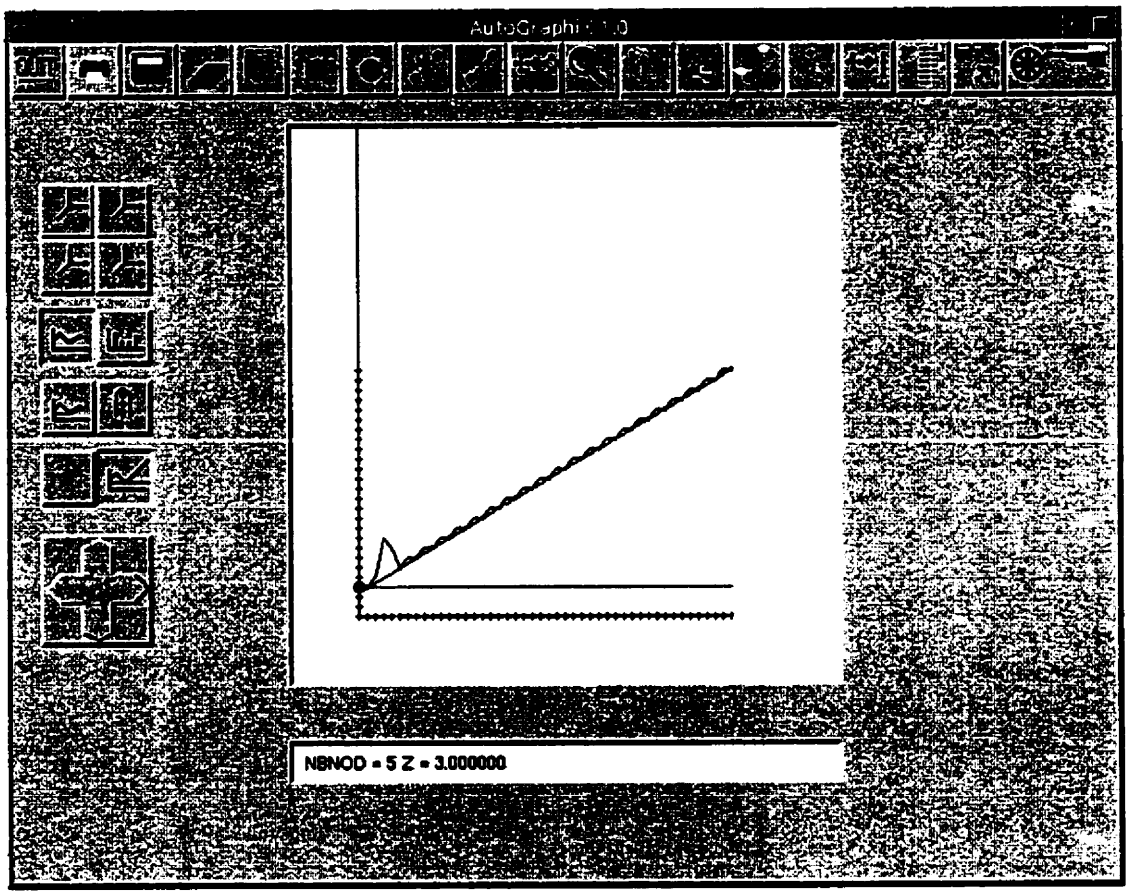


Figure 1.7 – Enveloppe convexe inférieure de la distance à la H-palindromicité d'arbres de diamètre impair donnée par AutoGraphiX

Chapitre 2

Applications

2.1 Étude de l'indice de Randić

L'indice de Randić, ou indice de connectivité, noté Ra est défini comme suit. Soit (i, j) une arête d'un graphe G, δ_i et δ_j les degrés des sommets i et j. Alors le *poids* de (i, j) est $\frac{1}{\sqrt{\delta_i \delta_j}}$ et l'*indice de connectivité* $Ra(G)$ est égal à la somme des poids de toutes les arêtes de G :

$$Ra = Ra(G) = \sum_{(i,j)\in E} \frac{1}{\sqrt{\delta_u \delta_v}}. \tag{2.1}$$

Cet indice fut conçu par Randić il y a un quart de siècle [107] pour l'étude de la compacité et de diverses autres propriétés des graphes pouvant représenter des molécules, ou **graphes chimiques**. La restriction que nous considérons pour qu'un graphe soit dit chimique est que son degré maximum est inférieur ou égal à 4, la même restriction appliquée à un arbre définit un **arbre chimique**.

C'est sans doute l'indice topologique associé aux graphes chimiques le plus étudié. Plusieurs centaines d'articles et deux livres ont été consacrés à Ra et ses généralisations [84] [85].

Les recherches mathématiques au sujet de l'indice de Randić ont principalement été

stimulées par les conjectures obtenues par le système automatisé *Graffiti*. Une revue des résultats ainsi obtenus, qui ont mené à des théorèmes ou demeurent à l'état de conjectures, est donnée dans [25]. Parmi les résultats importants notons que Bollobás et Erdős [16] ont prouvé que

$$Ra \geq \sqrt{n-1}, \tag{2.2}$$

et cette borne est atteinte pour les étoiles ; toutefois, ce résultat est d'utilité limitée pour les chimistes puisque quand $n > 5$, l'étoile n'est plus un graphe chimique.

Lors de l'étude de cet indice, nous nous sommes intéressés aux graphes chimiques d'indice de Randić extrême. Dans un premier temps, nous avons étudié les arbres d'indice Ra extrême en général, puis nous avons approfondi cette recherche en considérant les arbres chimiques dont le nombre de sommets pendants (sommets de degré 1) est fixé. Ce problème semble avoir un intérêt particulier puisqu'il a déjà été étudié par Araujo et De la Peña [6]. Cette seconde étude nous a permis d'identifier quelques erreurs dans [6] et, en nous basant sur les graphes extrêmes, de trouver des relations plus précises que celles connues précédemment. Enfin, nous avons abordé l'étude des graphes chimiques d'indice de Randić extrême en fonction du nombre cyclomatique ν (nombre de cycles élémentaires indépendants du graphe) défini dans la section 2.1.3.

2.1.1 Arbres chimiques d'indice de Randić extrême

Au début de nos recherches, nous n'avions pas d'indications quant à la structure des arbres chimiques d'indice Ra minimum et avons utilisé **AGX** afin de trouver ou d'approximer de tels arbres. Les arbres trouvés pour $5 \leq n \leq 24$ sont représentés sur la figure 2.1.

En étudiant la structure des arbres chimiques d'indice Ra minimum, nous constatons qu'ils ne sont pas uniques. En effet, considérons un arbre chimique à n sommets T_n

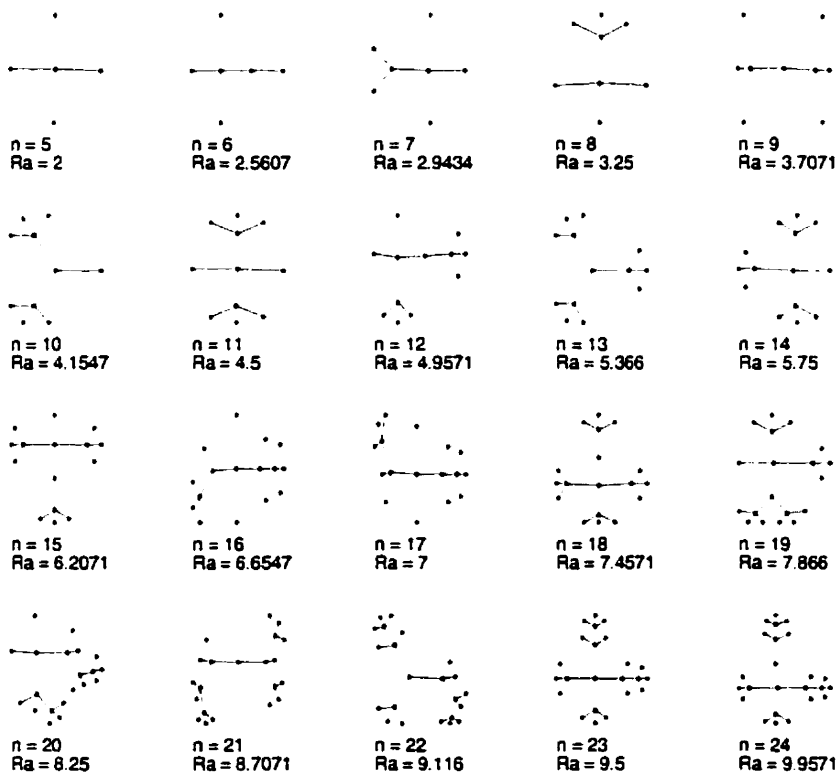


Figure 2.1 – Arbres chimiques d'indice Ra minimum d'ordre n pour $5 \leq n \leq 24$.

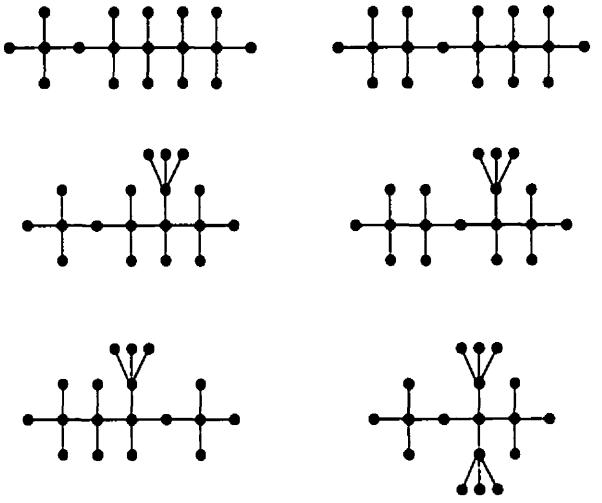


Figure 2.2 – Les six arbres chimiques non-isomorphes avec 18 sommets et un indice Ra minimum.

et notons le nombre de ses sommets de degré i par n_i , $i = 1, 2, 3, 4$. De plus, notons le nombre d'arêtes de T_n reliant un sommet de degré i à un sommet de degré j par x_{ij} . Alors, d'après l'équation (2.1),

$$Ra(T_n) = \frac{x_{12}}{\sqrt{2}} + \frac{x_{13}}{\sqrt{3}} + \frac{x_{14}}{2} + \frac{x_{22}}{2} + \frac{x_{23}}{\sqrt{6}} + \frac{x_{24}}{2\sqrt{2}} + \frac{x_{33}}{3} + \frac{x_{34}}{2\sqrt{3}} + \frac{x_{44}}{4} . \qquad (2.3)$$

De l'expression (2.3), il s'ensuit que dès que les valeurs x_{ij} sont identiques pour deux arbres, ces arbres ont le même indice Ra. En d'autres termes, l'indice de Randić dépend des valeurs de certaines structures locales fixant les degrés des sommets de chaque arête mais non de la manière dont elles sont agencées dans le graphe considéré [66] [67]. Il n'est donc pas surprenant de constater que pour n'importe quelle valeur de n (sauf $n \leq 13$ et $n = 16$) il existe plusieurs arbres chimiques non-isomorphes d'indice de Randić minimum. La figure 2.2 montre six arbres non isomorphes d'indice Ra minimum avec 18 sommets et $x_{14} = 12$, $x_{24} = 2$, $x_{44} = 3$ et $x_{12} = x_{13} = x_{22} = x_{23} = x_{33} = x_{34} = 0$.

En utilisant **AGX** de manière analogue pour les arbres d'indice de Randić maximum,

nous avons obtenu un résultat simple et attendu : l'arbre chimique à n sommets d'indice Ra présumément maximum est le chemin P_n.

Théorème AGX 3 *Si T_n et P_n sont respectivement un arbre et un chemin à n sommets, $n \geq 2$, alors*

$$Ra(T_n) \leq Ra(P_n), \tag{2.4}$$

avec égalité si et seulement si T_n est un chemin.

Preuve:

Outre les cas $n = 2$ et $n = 3$ qui sont triviaux, nous utiliserons une technique issue de la programmation linéaire [30] pour démontrer le *Théorème AGX* 3.

Soit x_{ij} le nombre d'arêtes reliant un sommet de degré i à un sommet de degré j, alors pour tout arbre nous avons

$$\sum_{i=1}^{n} \sum_{j=i}^{n} x_{ij} = n - 1, \tag{2.5}$$

pour tout arbre T_n. De même, nous avons, pour tout arbre de plus de 2 sommets ($x_{11} = 0$)

$$\sum_{i=1}^{j} x_{ij} + \sum_{\substack{i=j \\ (i,j)\neq(1,1)}}^{n} x_{ji} = jn_j \; \forall j = 1 \ldots n \tag{2.6}$$

$$\Leftrightarrow \quad n_j = \frac{1}{j}\Big(\sum_{\substack{i=1 \\ (i,j)\neq(1,1)}}^{j} x_{ij} + \sum_{\substack{i=j \\ (i,j)\neq(1,1)}}^{n} x_{ji}\Big) \; \forall j = 1 \ldots n \tag{2.7}$$

où n_j est le nombre de sommets de degré j.

Nous en déduisons

$$
\begin{aligned}
n &= \sum_{j=1}^{n} \frac{1}{j} \sum_{\substack{i=j \\ (i,j)\neq(1,1)}}^{j} x_{ij} + \sum_{j=1}^{n} \frac{1}{j} \sum_{\substack{i=j \\ (i,j)\neq(1,1)}}^{n} x_{ji} && (2.8)\\
&= \sum_{(i,j)\in\mathcal{S}_1} \frac{1}{j} x_{ij} + \sum_{(i,j)\in\mathcal{S}_1} \frac{1}{i} x_{ij} && (2.9)\\
&= \sum_{(i,j)\in\mathcal{S}_1} (\frac{1}{i} + \frac{1}{j}) x_{ij}, && (2.10)
\end{aligned}
$$

où $\mathcal{S}_1 = \{(i,j) \in \mathbb{N}^2 | 1 \leq i \leq n,\ i \leq j \leq n,\ (i,j) \neq (1,1)\}$, tandis que l'indice de Randić s'écrit

$$
Ra(T_n) = \sum_{i=1}^{n} \sum_{\substack{j=i \\ (i,j)\neq(1,1)}}^{n} \frac{x_{ij}}{\sqrt{ij}}. \tag{2.11}
$$

Le programme linéaire associé à ce problème est alors

$$
MaxZ = \sum_{(i,j)\in\mathcal{S}_1} \frac{x_{ij}}{\sqrt{ij}} \tag{2.12}
$$

sous les contraintes

$$
\begin{aligned}
\sum_{(i,j)\in\mathcal{S}_1} x_{ij} &= n-1 && (2.13)\\
\sum_{(i,j)\in\mathcal{S}_1} (\frac{1}{i} + \frac{1}{j}) x_{ij} &= n && (2.14)\\
x_{ij} \geq 0 \qquad & \forall i = 1 \ldots n\ \forall j = i \ldots n && (2.15)
\end{aligned}
$$

où les variables sont x_{ij} alors que n est un paramètre.

Soit $\mathcal{S}_2 = \{(i,j) \in \mathcal{S}_1 \,|\, (i,j) \neq (1,2)\}$, l'équation (2.13), se réécrit alors

$$
x_{12} = n - 1 - \sum_{(i,j)\in\mathcal{S}_2} x_{ij}. \tag{2.16}
$$

En remplaçant x_{12} par sa valeur dans (2.14), nous obtenons

$$\frac{3n}{2} - \frac{3}{2} - \sum_{(i,j)\in\mathcal{S}_2} (\frac{1}{i} + \frac{1}{j} - \frac{3}{2})x_{ij} = n \tag{2.17}$$

$$\Leftrightarrow \quad \sum_{(i,j)\in\mathcal{S}_2} (\frac{1}{i} + \frac{1}{j} - \frac{3}{2})x_{ij} = \frac{3-n}{2} \tag{2.18}$$

$$\Leftrightarrow \quad x_{22} = n - 3 + \sum_{(i,j)\in\mathcal{S}_3} (\frac{2}{i} + \frac{2}{j} - 3)x_{ij} \tag{2.19}$$

avec $\mathcal{S}_3 = \{(i,j) \in \mathcal{S}_2 \,|\, (i,j) \neq (2,2)\}$.

L'équation (2.16) devient alors

$$x_{12} = 2 - \sum_{(i,j)\in\mathcal{S}_3} (\frac{2}{i} + \frac{2}{j} - 2)x_{ij}. \tag{2.20}$$

En utilisant les équations (2.19) et (2.20), l'indice Ra s'écrit :

$$\begin{aligned} Ra(T_n) &= \frac{1}{\sqrt{2}}[2 - \sum_{(i,j)\in\mathcal{S}_3} (\frac{2}{i} + \frac{2}{j} - 2)x_{ij}] \\ &\quad + \frac{1}{2}[n - 3 + \sum_{(i,j)\in\mathcal{S}_3} (\frac{2}{i} + \frac{2}{j} - 3)x_{ij}] \\ &\quad + \sum_{(i,j)\in\mathcal{S}_3} \frac{x_{ij}}{\sqrt{ij}} \\ &= \frac{n-3}{2} + \sqrt{2} + \sum_{(i,j)\in\mathcal{S}_3} \bar{c}_{ij}x_{ij}. \end{aligned} \tag{2.21}$$

où

$$\bar{c}_{ij} = (\frac{1}{i} + \frac{1}{j})(1 - \sqrt{2}) + \sqrt{2} - \frac{3}{2} + \frac{1}{\sqrt{ij}}. \tag{2.22}$$

Il reste alors à démontrer que $\bar{c}_{ij} \leq 0 \,\forall (i,j) \in \mathcal{S}_3$. Nous montrerons d'abord que $\bar{c}_{ij}$ décroit avec j, ce que nous ferons en vérifiant que $\bar{c}_{ij+1} - \bar{c}_{ij} \leq 0 \,\forall (i,j) \in \mathcal{S}_3$.

$$\begin{aligned}
\bar{c}_{ij+1} - \bar{c}_{ij} &= (\frac{1}{j+1} - \frac{1}{j})(1-\sqrt{2}) + \frac{1}{\sqrt{i(j+1)}} - \frac{1}{\sqrt{ij}} && (2.23)\\
&= \frac{j-(j+1)}{j(j+1)}(1-\sqrt{2}) + \frac{1}{\sqrt{i}}\frac{\sqrt{j}-\sqrt{j+1}}{\sqrt{j(j+1)}} && (2.24)\\
&= \frac{1-\sqrt{2}}{j(j+1)} + \frac{1}{\sqrt{i}}\frac{\sqrt{j}-\sqrt{j+1}}{\sqrt{j(j+1)}} && (2.25)\\
&= \frac{1}{\sqrt{j(j+1)}}(\frac{1-\sqrt{2}}{\sqrt{j(j+1)}} + \frac{\sqrt{j}-\sqrt{j+1}}{\sqrt{i}}). && (2.26)
\end{aligned}$$

Comme $1-\sqrt{2} < 0$ et $\sqrt{j}-\sqrt{j+1} < 0$, $\bar{c}_{ij+1} - \bar{c}_{ij} < 0$.

Nous montrerons ensuite que $\bar{c}_{13}, \bar{c}_{23}$ et $\bar{c}_{ii} < 0\ \forall i = 3 \ldots n$.

$$\begin{aligned}
\bar{c}_{13} &= \frac{\sqrt{3}-\sqrt{2}-2}{3} \approx -0.06 && (2.27)\\
\bar{c}_{23} &= \frac{\sqrt{2}+\sqrt{6}-4}{6} \approx -0.02 && (2.28)\\
\bar{c}_{ii} &= \sqrt{2} + \frac{3-2\sqrt{2}}{i} - \frac{3}{2} && (2.29)\\
&= (3-2\sqrt{2})(\frac{1}{i} - \frac{1}{2}). && (2.30)
\end{aligned}$$

Étant donné que $i > 2$ dans l'équation (2.30), et que $j \geq i$ en général, $\bar{c}_{ij} < 0\,\forall (i,j) \in \mathcal{S}_3$. Toutes les variables devant être non négatives, la solution $x_{12} = 2$, $x_{22} = n-3$ qui correspond au chemin est celle qui maximise l'indice de Randić. □

Théorème AGX 4 *Soit T_n un arbre chimique quelconque à n sommets.*
(a) Si $n \equiv 2$ (mod 3) et $n \geq 5$, alors

$$Ra(T_n) \geq \frac{5n}{12} - \frac{1}{12}$$

avec égalité si et seulement si tous les sommets sont de degré 1 ou 4 (ou encore si et seulement si T_n ne comporte aucun sommet de degré 2 ou 3).

(b) Si $n \equiv 1 \ (mod\ 3)$ *et* $n \geq 13$, *alors*

$$Ra(T_n) \geq \frac{5\,n}{12} + 6\sqrt{3} - \frac{11}{12} \tag{2.31}$$

avec égalité si et seulement si tous les sommets de T_n *sont de degré 1 ou 4, à l'exception d'un seul qui est de degré 3 et adjacent à 3 sommets de degré 4.*

(c) Si $n \equiv 0 \ (mod\ 3)$ *et* $n \geq 9$, *alors*

$$Ra(T_n) \geq \frac{5\,n}{12} + 2\sqrt{2} - \frac{3}{4} \tag{2.32}$$

avec égalité si et seulement si tous les sommets sont de degré 1 ou 4 à l'exception d'un seul qui est de degré 2 adjacent à 2 sommets de degré 4.

Le *Théorème AGX* 4 ne couvre pas les cas $n \leq 4$, $n = 6$, $n = 7$ et $n = 10$ pour lesquels se produisent des "effets de bord". Les graphes d'indice Ra minimum avec $n = 4, 6, 7$ ou 10 sommets sont représentés sur la figure 2.1.

Preuve:

Pour démontrer le *Théorème AGX* 4, nous à nouveau la programmation linéaire. Si T_n est un arbre à n sommets, alors les six relations qui suivent sont vérifiées :

$$x_{12} + x_{13} + x_{14} = n_1 \tag{2.33}$$

$$x_{12} + 2\,x_{22} + x_{23} + x_{24} = 2\,n_2 \tag{2.34}$$

$$x_{13} + x_{23} + 2\,x_{33} + x_{34} = 3\,n_3 \tag{2.35}$$

$$x_{14} + x_{24} + x_{34} + 2\,x_{44} = 4\,n_4 \tag{2.36}$$

$$n_1 + 2\,n_2 + 3\,n_3 + 4\,n_4 = 2(n-1) \tag{2.37}$$

$$n_1 + n_2 + n_3 + n_4 = n\ . \tag{2.38}$$

Notre stratégie est de considérer les équations (2.33)–(2.38) comme un système de six équations linéaires dont les inconnues sont $n_1, n_2, n_3, n_4, x_{14}, x_{44}$, et de le résoudre.

La solution ainsi obtenue dépendra des autres paramètres, soit x_{12}, x_{13}, x_{22}, x_{23}, x_{24}, x_{33} et x_{34}. Les résultats d'**AGX** indiquent que pour les graphes chimiques d'indice *Ra* minimum tous, ou du moins la majorité, de ces paramètres seront nuls.

En réécrivant les équations (2.33)–(2.38) comme suit,

$$\begin{aligned}
n_1 - x_{14} &= f_1 \\
2\,n_2 &= f_2 \\
3\,n_3 &= f_3 \\
4\,n_4 - x_{14} - 2\,x_{44} &= f_4 \\
n_1 + 2\,n_2 + 3\,n_3 + 4\,n_4 &= f_5 \\
n_1 + n_2 + n_3 + n_4 &= f_6
\end{aligned}$$

où

$$\begin{aligned}
f_1 &= x_{12} + x_{13} \\
f_2 &= x_{12} + 2\,x_{22} + x_{23} + x_{24} \\
f_3 &= x_{13} + x_{23} + 2\,x_{33} + x_{34} \\
f_4 &= x_{24} + x_{34} \\
f_5 &= 2\,n - 2 \\
f_6 &= n
\end{aligned}$$

nous obtenons les solutions recherchées :

$$
\begin{aligned}
n_1 &= \frac{1}{3}\left(4\,f_6 - f_5 - f_2 - \frac{1}{3}\,f_3\right) \\
n_2 &= \frac{1}{2}\,f_2 \\
n_3 &= \frac{1}{3}\,f_3 \\
n_4 &= \frac{1}{3}\left(f_5 - f_6 - \frac{1}{2}\,f_2 - \frac{2}{3}\,f_3\right) \\
x_{14} &= \frac{1}{3}\left(4\,f_6 - f_5 - f_2 - \frac{1}{3}\,f_3\right) - f_1 \\
x_{44} &= \frac{1}{6}\left(5\,f_5 - 8\,f_6 - f_2 - \frac{7}{3}\,f_3\right) + \frac{1}{2}(f_1 - f_4)\ .
\end{aligned}
$$

Dans ce qui suit, il suffit d'avoir x_{14} et x_{44} sous forme explicite :

$$x_{14} = \frac{2n}{3} + \frac{2}{3} - \frac{4x_{12}}{3} - \frac{10x_{13}}{9} - \frac{2x_{22}}{3} - \frac{4x_{23}}{9} - \frac{x_{24}}{3} - \frac{2x_{33}}{9} - \frac{x_{34}}{9} \tag{2.39}$$

et

$$x_{44} = \frac{n}{3} - \frac{5}{3} + \frac{x_{12}}{3} + \frac{x_{13}}{9} - \frac{x_{22}}{3} - \frac{5x_{23}}{9} - \frac{2x_{24}}{3} - \frac{7x_{33}}{9} - \frac{8x_{34}}{9}. \tag{2.40}$$

La substitution des relations (2.39) et (2.40) dans l'équation (2.3) donne

$$
\begin{aligned}
Ra(T_n) &= \frac{5\,n-1}{12} + \left(\frac{1}{\sqrt{2}} - \frac{7}{12}\right) x_{12} + \left(\frac{1}{\sqrt{3}} - \frac{19}{36}\right) x_{13} + \frac{x_{22}}{12} + \left(\frac{1}{\sqrt{6}} - \frac{13}{36}\right) x_{23} \\
&\quad + \left(\frac{1}{2\sqrt{2}} - \frac{1}{3}\right) x_{24} + \frac{x_{33}}{36} + \left(\frac{1}{2\sqrt{3}} - \frac{5}{18}\right) x_{34}. \qquad (2.41)
\end{aligned}
$$

Pour vérifier la condition d'optimalité de la programmation linéaire, il faut que tous les sept coefficients présents dans le membre droit de l'équation (2.41) aient des valeurs positives, ce qui est le cas. En calculant ces coefficients, nous obtenons :

$$
\begin{aligned}
Ra(T_n) &= \frac{5\,n-1}{12} + 0.12377\,x_{12} + 0.04957\,x_{13} + 0.08333\,x_{22} + 0.04714\,x_{23} \\
&\quad + 0.02022\,x_{24} + 0.02778\,x_{33} + 0.01090\,x_{34}. \qquad (2.42)
\end{aligned}
$$

Des équations (2.41) et/ou (2.42), il s'ensuit que, pour n fixé, la valeur de $Ra(T_n)$ sera minimale si tous les paramètres $x_{12}, x_{13}, x_{22}, x_{23}, x_{24}, x_{33}, x_{34}$ sont nuls. Si ceci est impossible, nous devons chercher des arbres chimiques pour lesquels les paramètres $x_{12}, x_{13}, x_{22}, x_{23}, x_{24}, x_{33}, x_{34}$ sont proches de zéro, en sachant qu'ils doivent prendre des valeurs entières non négatives, celles qui minimisent (2.42)

- **Étape 1.**

  En posant

$$x_{12} = x_{13} = x_{22} = x_{23} = x_{24} = x_{33} = x_{34} = 0 \tag{2.43}$$

  l'indice Ra devient

$$Ra(T_n) = \frac{5\,n - 1}{12} \tag{2.44}$$

  et les solutions des équations (2.33)–(2.38) se lisent :

$$\begin{aligned} n_1 &= \frac{2}{3}(n+1) \\ n_2 &= 0 \\ n_3 &= 0 \\ n_4 &= \frac{1}{3}(n-2) \\ x_{14} &= \frac{2}{3}(n+1) \\ x_{44} &= \frac{1}{3}(n-5)\ . \end{aligned}$$

  Afin que n_1, n_4, etc. soient des entiers, le nombre de sommets n doit être égal à 2(mod 3), et pour que x_{44} soit non négatif, il faut que n soit supérieur ou égal à 5. Si ces deux conditions sont respectées, alors, il existe des arbres chimiques vérifiant l'équation (2.43), ce qui démontre la partie (a) du *Théorème AGX* 4.

- **Étape 2.**

  Si $n \not\equiv 2 \pmod 3$, alors aucun arbre chimique à n sommets ne satisfait l'équation (2.43) puisqu'il n'en existe pas sans sommet de degré 2 ou 3. Les arbres chimiques

d'indice Ra minimum doivent donc comporter au moins un sommet de degré 2 ou 3. De l'équation (2.42), nous pouvons conclure ce qui suit :

cas $n_2 = 1, n_3 = 0$:

si $x_{12} = 1$ & $x_{24} = 1$ alors Ra croit de 0.14399

si $x_{12} = 0$ & $x_{24} = 2$ alors Ra croit de 0.04044

cas $n_2 = 0, n_3 = 1$:

si $x_{13} = 2$ & $x_{34} = 1$ alors Ra croit de 0.11004

si $x_{13} = 1$ & $x_{34} = 2$ alors Ra croit de 0.07137

si $x_{13} = 0$ & $x_{34} = 3$ alors Ra croit de 0.03269 .

L'augmentation de l'indice Ra, c'est-à-dire la différence entre la valeur de Ra et$(5\,n - 1)/12$ quand $n_2 > 1$ et/ou $n_3 > 1$, ou $n_2 = n_3 = 1$ est significativement plus grande et ne nécessite pas d'être considérée.

D'après les informations précédentes, il semble que la recherche d'arbres chimiques d'indice Ra minimum doive se poursuivre par des arbres respectant une des conditions suivantes :

$$x_{34} = 3\,,\; x_{12} = x_{13} = x_{22} = x_{23} = x_{24} = x_{33} = 0 \tag{2.45}$$

ou

$$x_{24} = 2\,,\; x_{12} = x_{13} = x_{22} = x_{23} = x_{33} = x_{34} = 0\,. \tag{2.46}$$

- **Étape 3.**

En combinant les équations (2.41) et/ou (2.42) avec (2.45) l'indice de connectivité s'écrit

$$Ra(T_n) = \frac{5\,n - 1}{12} + 3\left(\frac{1}{2\sqrt{3}} - \frac{5}{18}\right) = \frac{5\,n - 1}{12} + 0.03269 \tag{2.47}$$

où les solutions des équations (2.33)–(2.38) sont :

$$
\begin{aligned}
n_1 &= \frac{1}{3}(2n+1) \\
n_2 &= 0 \\
n_3 &= 1 \\
n_4 &= \frac{1}{3}(n-4) \\
x_{14} &= \frac{1}{3}(2n+1) \\
x_{44} &= \frac{1}{3}(n-13)\ .
\end{aligned}
$$

De plus, afin que n_1, n_4, etc. soient entiers, le nombre de sommets de T_n doit respecter la condition : $n \equiv 1 \pmod 3$. Afin que x_{44} soit non négatif, il faut que $n \geq 13$.

Ceci démontre la partie (b) du *Théorème AGX* 4.

- **Étape 4.**

Les considérations dans le cas où les conditions (2.46) sont vérifiées sont analogues à celles de l'étape précédente. L'indice de connectivité est alors donné par

$$Ra(T_n) = \frac{5n-1}{12} + 2\left(\frac{1}{2\sqrt{2}} - \frac{1}{3}\right) = \frac{5n-1}{12} + 0.04044 \tag{2.48}$$

et

$$
\begin{aligned}
n_1 &= \frac{2}{3}n \\
n_2 &= 1 \\
n_3 &= 0 \\
n_4 &= \frac{1}{3}n - 1 \\
x_{14} &= \frac{2}{3}n \\
x_{44} &= \frac{1}{3}(n-6) - 1\ .
\end{aligned}
$$

Dans ce cas n_1, n_4, etc. sont entiers si $n \equiv 0 \pmod 3$, et x_{44} prend des valeurs non négatives si $n \geq 9$. La partie (c) du *Théorème AGX* 4 en découle.

Les cas considérés dans les étapes 1, 3 et 4 couvrent toutes les possibilités, ainsi la preuve du *Théorème AGX* 4 est complète. □

2.1.2 Arbres chimiques d'indice de Randić minimum en fonction du nombre de sommets pendants.

Deux bornes sur l'indice de connectivité de graphes chimiques

L'application de la routine de recherche de relations d'**AGX** aux graphes chimiques d'indice de Randić présumément minimum n'a donné au départ aucune relation. Toutefois, en considérant les graphes selectionnés par la méthode de la moyenne flottante (voir section 1.4.1), nous avons obtenu les deux relations suivantes, à l'origine des *Théorèmes* 5 et 6.

Théorème AGX 5 *Pour tout graphe chimique G :*

$$Ra(G) \geq \frac{1}{4}(n_1 + m) \tag{2.49}$$

Preuve:
Soit n_i, $i = 1, \ldots, 4$ le nombre de sommets de degré i de G et x_{ij}, $i, j = 1, \ldots, 4$; $i \leq j$ le nombre d'arêtes de G dont les extrémités sont de degrés i et j. Considérons ensuite les 6 équations :

$$\begin{aligned}
n_1 + n_2 + n_3 + n_4 &= n \\
n_1 + 2n_2 + 3n_3 + 4n_4 &= 2m \\
x_{12} + x_{13} + x_{14} &= n_1 \\
x_{12} + 2x_{22} + x_{23} + x_{24} &= 2n_2 \\
x_{13} + x_{23} + 2x_{33} + x_{34} &= 3n_3 \\
x_{14} + x_{24} + x_{34} + 2x_{44} &= 4n_4
\end{aligned}$$

obtenues en comptant les sommets et les degrés. Il est clair que ces équations sont indépendantes.

Résoudre pour $n_1, n_2, n_3, n_4, x_{14}$ et x_{44} et substituer x_{14} et x_{44} dans la fonction-objectif

$$\min\ Ra(G) = \sum_{i=1}^{4}\sum_{j=i}^{4} \frac{x_{ij}}{\sqrt{ij}} \tag{2.50}$$

nous conduit à la relation

$$\begin{aligned}\min\ Ra(G) = {} & \frac{n_1}{4}+\frac{m}{4}+\left(\frac{1}{\sqrt{2}}-\frac{1}{2}\right)x_{12}+\left(\frac{1}{\sqrt{3}}-\frac{1}{2}\right)x_{13}\\ & +\frac{x_{22}}{4}+\left(\frac{1}{\sqrt{6}}-\frac{1}{4}\right)x_{23}+\left(\frac{1}{2\sqrt{2}}-\frac{1}{4}\right)x_{24}\\ & +\frac{x_{33}}{12}+\left(\frac{1}{2\sqrt{3}}-\frac{1}{4}\right)x_{34}.\end{aligned} \tag{2.51}$$

Le résultat découle alors du fait que $x_{12}, x_{13}, x_{22}, x_{23}, x_{24}, x_{33}$ et x_{34} sont non négatifs et ont des coefficients positifs dans (2.51). □

Théorème AGX 6 *Pour tout graphe chimique*

$$Ra(G) \geq \frac{n}{3} + \frac{m}{12}. \tag{2.52}$$

Ce théorème généralise le *Théorème AGX* 4 et sa preuve est découle de la preuve du *Théorème AGX* 5. Nous l'omettrons donc ici.

2.1.3 Graphes chimiques avec nombre cyclomatique fixé d'indice de Randić extrême

Le nombre *cyclomatique* ν est défini pour des graphes connexes par

$$\nu = m - n + 1. \tag{2.53}$$

La démarche que nous avons utilisée pour cette étude est similaire à celle adoptée pour les cas précédents. **AGX** a d'abord permis de trouver des graphes chimiques d'indice Ra maximum et minimum alors que n et m étaient fixés comme paramètres.

Dans un premier temps, nous nous sommes intéressés aux graphes chimiques d'indice Ra maximum. En utilisant **AGX** avec les paramètres $12 \leq n \leq 20$ et $2 \leq \nu \leq 5$, la courbe représentant Ra en fonction de n et ν est très régulière, comme le montre la figure 2.3.

De plus, la procédure automatisée de recherche de conjectures d'**AGX** nous indique que les relations

$$n_4 = 0 \tag{2.54}$$

$$n_1 = 0 \tag{2.55}$$

$$x_{23} = 2n_2 - 2x_{22} \tag{2.56}$$

$$x_{33} = -n_2 + 1.5n_3 + x_{22} \tag{2.57}$$

(ainsi que d'autres relations triviales) sont vérifiées pour tous les graphes supposés extrêmes trouvés par le programme.

L'équation (2.56) dit que le nombre d'arêtes adjacentes à des sommets de degré 2 est deux fois le nombre de tels sommets, ce qui est trivial. L'équation (2.57) est essentiellement la même pour les sommets de degré 3, comme on peut le voir en ajoutant (2.56) à deux fois (2.57).

De ces relations nous pouvons déduire la conjecture suivante :

Conjecture 10 *Si G est un graphe chimique de nombre cyclomatique $\nu \geq 2$ et d'indice de Randić maximum, alors G n'a pas de sommets pendants ni de sommets de degré 4.*

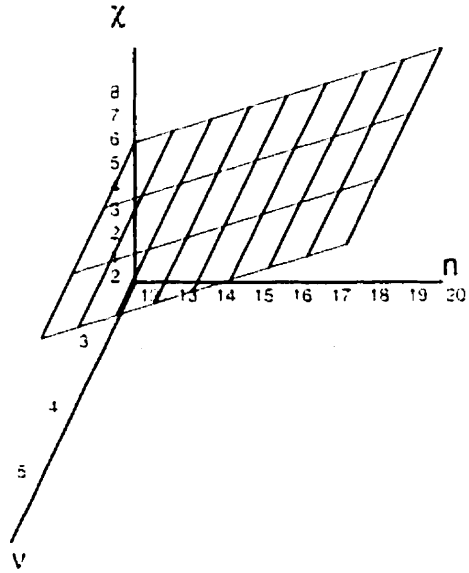


Figure 2.3 - Valeurs de Ra pour les graphes chimiques en fonction de n et ν.

Le programme a ensuite été utilisé pour diverses valeurs de ν dans le but de trouver des relations alors que n est considéré comme paramètre. Les résultats obtenus sont résumés dans le Tableau 2.1 donnant les valeurs de n_3, x_{22}, x_{23} et x_{33} en fonction de ν.

Tableau 2.1 - Relations proposées par AutoGraphiX pour les graphes d'indice Ra maximum en fonction ν, pour n assez grand.

ν	n_3	x_{22}	x_{23}	x_{33}
2	2	$n_2 - 2$	4	1
3	4	$n_2 - 1$	2	5
4	6	$n_2 - 1$	2	8
5	8	$n_2 - 1$	2	11

Ces résultats suggèrent les conjectures suivantes :

Conjecture 11 *Si G est un graphe chimique de nombre cyclomatique $\nu \geq 3$ et*

d'indice de Randić maximum, alors le sous-graphe induit par les sommets de degré 2 est un chemin.

Cette conjecture découle de la relation $x_{22} = n_2 - 1$ donnée par le programme pour $\nu \geq 3$.

Conjecture 12 *Si G est un graphe chimique de nombre cyclomatique $\nu \geq 3$ et d'indice de Randić maximum, alors G a $3\nu - 4$ arêtes reliant des sommets de degré 3.*

Ces conjectures ont été trouvées pour $\nu \leq 5$ et nous devons poser des restrictions sur leur généralisation. Bien sûr, pour un ordre n donné, si ν est assez grand, n_4 doit être positif. Afin de trouver les conditions dans lesquelles ces conjectures sont vraies et de les prouver, rappelons que

$$\nu = m - n + 1, \tag{2.58}$$

$$n = n_1 + n_2 + n_3 + n_4 \tag{2.59}$$

$$m = \frac{1}{2}(n_1 + 2n_2 + 3n_3 + 4n_4) \tag{2.60}$$

alors

$$\begin{aligned} \nu &= \frac{1}{2}(n_1 + 2n_2 + 3n_3 + 4n_4) - (n_1 + n_2 + n_3 + n_4) + 1 \\ &= -\frac{n_1}{2} + \frac{n_3}{2} + n_4 + 1. \end{aligned} \tag{2.61}$$

Les résultats ci-dessus conduisent au théorème suivant :

Théorème AGX 7 *Si G est un graphe chimique de nombre cyclomatique $\nu \geq 2$, comportant $n \geq 5\nu - 4$ sommets et d'indice de connectivité Ra maximum alors*

- *G n'a pas de sommets pendants ;*
- *G a au moins une arête reliant deux sommets de degré 2 ;*
- *G n'a pas de sommets de degré 4.*

Preuve:
Afin de démontrer ce théorème, nous traitons chacune de ses parties séparément sous forme de lemmes.

Lemme 1 *Soit $\nu > 1$ le nombre cyclomatique d'un graphe chimique G à n sommets, si $n \geq n_{min} = 5\nu - 4$ et G est d'indice de Randić maximum, alors G n'a pas de sommets pendants.*

Preuve:
Soit C l'ensemble de sommets d'une composante connexe obtenue quand un sommet v est retiré de G. Le sous-graphe induit par $T = C \cup \{v\}$ est appelé *sous-arbre* de G s'il ne comporte pas de cycle.

Lemme 1.1 Si G est un graphe chimique d'indice de Randić maximum, alors tout sous-arbre de G est un chemin.

Afin de démontrer cette affirmation, nous utilisons un argument similaire à la preuve que les arbres d'indice de Randić maximum sont des chemins (*Théorème AGX* 3 ci-dessus). Considérons un graphe comportant au moins un sous-arbre T, alors soit ce sous-arbre est un chemin soit non. Si tel n'est pas le cas, choisir dans T un sommet de branchement u tel que si u était enlevé, le graphe résiduel serait composé d'au plus une composante qui ne soit ni un chemin ni un sommet isolé. Clairement, un tel sommet existe.

Le graphe G a alors la structure décrite sur la figure 2.4.

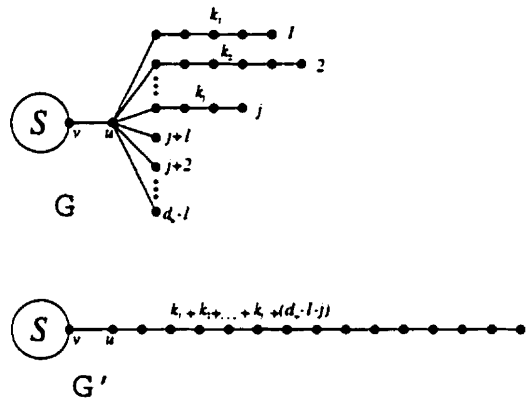


Figure 2.4 – Graphe avec un sous-arbre et ses modifications augmentant Ra

La suite de la preuve de ce lemme reprend en tous points la preuve du *Théorème* AGX^*3 donnée ci-dessus. Elle est donc omise ici.

À cette étape, nous avons démontré que si G a un indice de Randić maximum, il ne peut contenir d'autres sous-arbres que des chemins.

Soit A le sommet appartenant à un sous-arbre de G de degré maximum, alors A a un degré supérieur ou égal à 3.

Considérons d'abord le cas où $\delta_A = 3$. Soit C un voisin de A appartenant aussi à un sous-arbre, B et C les autres. D'après le *Lemme 1.1*, soit C appartient à un chemin comme le représente la figure 2.5, dont l'extrêmité est l'arête (QP) où P est un sommet pendant, soit C est un sommet pendant. Nous nous intéresserons d'abord au cas où C appartient à un chemin.

Remplacer l'arête (AB) par (BP) change Ra de

$$\begin{aligned}\Delta_{Ra} &= \frac{1}{\sqrt{(\delta_P+1)\delta_Q}} + \frac{1}{\sqrt{(\delta_P+1)\delta_B}} + \frac{1}{\sqrt{(\delta_A-1)\delta_C}} + \frac{1}{\sqrt{(\delta_A-1)\delta_D}} \\ &\quad -[\frac{1}{\sqrt{\delta_P\delta_Q}} + \frac{1}{\sqrt{\delta_A\delta_B}} + \frac{1}{\sqrt{\delta_A\delta_C}} + \frac{1}{\sqrt{\delta_A\delta_D}}] \qquad (2.62)\end{aligned}$$

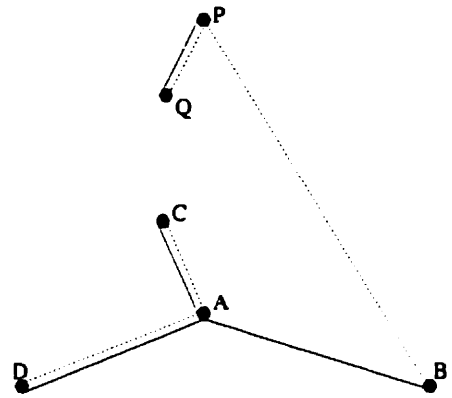


Figure 2.5 – Réduire le nombre de sommets pendants en augmentant Ra

comme $\delta_A = 3$, $\delta_P = 1$ et $\delta_Q = \delta_C = 2$, l'équation (2.62) devient

$$\begin{aligned}\Delta_{Ra} &= \frac{1}{2} + \frac{1}{\sqrt{2\delta_B}} + \frac{1}{2} + \frac{1}{\sqrt{2\delta_D}} \\ &\quad -[\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3\delta_B}} + \frac{1}{\sqrt{6}} + \frac{1}{\sqrt{3\delta_D}}] \\ &= 1 - \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{6}} + (\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{3}}) \cdot (\frac{1}{\sqrt{\delta_B}} + \frac{1}{\sqrt{\delta_D}}). \qquad (2.63)\end{aligned}$$

Puisque $1 \leq \delta_B \leq 4$ et $1 \leq \delta_D \leq 4$, il est facile de vérifier que pour chacun des 16 cas, $\Delta_R a > 0$. La transformation proposée donne un graphe toujours connexe G' de même nombre cyclomatique ν que G, avec un sommet pendant de moins et un plus grand indice de Randić.

Un raisonnement similaire nous conduit à la même conclusion dans le cas où C est un sommet pendant, et dans les cas où $\delta_A = 4$, C étant un sommet pendant ou pas.

Donc, si G a au moins un sommet pendant, son indice de Randić n'est pas maximum. □

Lemme 2 *Soit G un graphe chimique à n sommets, avec un nombre cyclomatique ν et sans sommet pendant; alors si $n \geq n_{min} = 5\nu - 4$, $x_{22} \geq 1$.*

Preuve:

Si G a $x_{22} = 0$, alors chaque sommet de degré 2 relie 2 sommets de degré supérieur à 2, et le nombre d'arêtes adjacentes à des sommets de degré 2 est inférieur ou égal à $3n_3 + 4n_4$. D'où $n_2 \leq \frac{1}{2}(3n_3 + 4n_4)$. De l'équation (2.61), comme $n_1 = 0$, nous avons

$$n_3 + 2n_4 = 2\nu - 2. \tag{2.64}$$

Considérons alors le problème d'optimisation

$$Max\ n = n_2 + n_3 + n_4 \tag{2.65}$$

sous les contraintes

$$n_2 \leq \frac{1}{2}(3n_3 + 4n_4) \tag{2.66}$$

$$n_3 + 2n_4 = 2\nu - 2 \tag{2.67}$$

òu n_1, n_3 et n_4 sont entiers.

Étant donné que n_2 n'apparaît que dans l'équation 2.66 et dans la fonction-objectif, il peut être remplacé par sa borne supérieure $\frac{1}{2}(3n_3 + 4n_4)$ alors que n_3 peut être remplacé par $2\nu - 2 - 2n_4$ en utilisant l'équation (2.67).

Le problème devient alors

Max $n = 5(\nu - 1) - 2n_4$

et la solution est $n_4 = 0$, ce qui implique $n = 5\nu - 5$.

Nous pouvons en conclure que tout graphe chimique sans sommets pendants avec un nombre cyclomatique ν et de cardinalité $n \geq 5\nu - 4$ comporte au moins une arête reliant deux sommets de degré 2. □

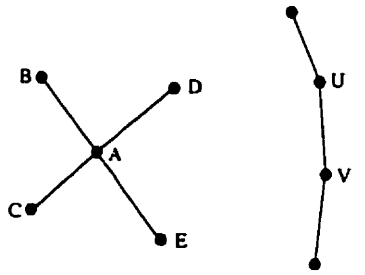


Figure 2.6 – Partie d'un graphe G_4 avec $n_4 \geq 1$

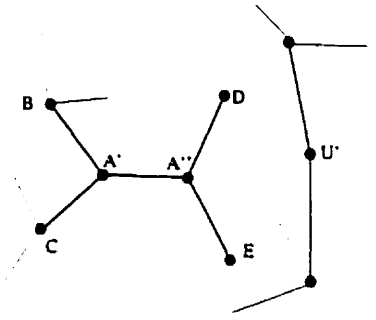


Figure 2.7 – Partie d'un graphe G_4 après modification.

Lemme 3 *Soient $\nu > 1$ le nombre cyclomatique d'un graphe chimique G à n sommets, n_{min} le nombre minimum de sommets requis pour s'assurer que $x_{22} \geq 1$. Si $n > n_{min}$ et G a un indice de Randić maximum, alors G n'a pas de sommets de degré 4.*

Preuve:

Considérons un graphe chimique G d'indice de Randić maximum avec au moins 2 sommets de degré 2 ; il est facile de vérifier que si G a au moins un sommet de degré 4 il a un tel sommet avec au moins un voisin de degré inférieur ou égal à 3. Soit A un tel sommet, B, C, D et E ses voisins, avec $\delta_E \leq 3$, et U, V deux sommets adjacents de degré 2, comme décrits sur la figure 2.6.

Si G est modifié en fusionnant U et V en un seul sommet de degré 2, et en séparant

A en deux sommets adjacents de degré 3, comme décrit sur la figure 2.7, l'indice de Randić croît de

$$\begin{aligned}\Delta_{Ra} &= \frac{1}{\sqrt{\delta_{A'}\delta_B}} + \frac{1}{\sqrt{\delta_{A'}\delta_C}} + \frac{1}{\sqrt{\delta_{A''}\delta_D}} + \frac{1}{\sqrt{\delta_{A''}\delta_E}} + \frac{1}{\sqrt{\delta_{A'}\delta_{A''}}} \\ &\quad -[\frac{1}{\sqrt{\delta_A\delta_B}} + \frac{1}{\sqrt{\delta_A\delta_C}} + \frac{1}{\sqrt{\delta_A\delta_D}} + \frac{1}{\sqrt{\delta_A\delta_E}} + \frac{1}{\sqrt{\delta_U\delta_V}}], \qquad (2.68)\end{aligned}$$

comme $\delta_A = 4$, $\delta_{A'} = \delta_{A''} = 3$, $\delta_U = \delta_V = 2$, l'équation (2.68) devient

$$\begin{aligned}\Delta_{Ra} &= \frac{1}{\sqrt{3\delta_B}} + \frac{1}{\sqrt{3\delta_C}} + \frac{1}{\sqrt{3\delta_D}} + \frac{1}{\sqrt{3\delta_E}} + \frac{1}{3} \\ &\quad -[\frac{1}{\sqrt{4\delta_B}} + \frac{1}{\sqrt{4\delta_C}} + \frac{1}{\sqrt{4\delta_D}} + \frac{1}{\sqrt{4\delta_E}} + \frac{1}{2}]. \qquad (2.69)\end{aligned}$$

Il est facile de vérifier à l'aide de l'équation (2.69) que si $1 \leq \delta_B \leq 4$, $1 \leq \delta_C \leq 4$, $1 \leq \delta_D \leq 4$ et $1 \leq \delta_E \leq 3$, $\Delta_{Ra} \geq 0$. Cette transformation, valide tant que $n_4 \geq 1$, augmente toujours Ra et réduit n_4 d'une unité ; répéter ce processus jusqu'à ce que $n_4 = 0$ génèrera un graphe toujours connexe avec le même nombre cyclomatique, et un indice de Randić plus grand. □

Des preuves des *Lemmes* 1 - 3, nous concluons que le *Théorème AGX* 7 est vrai. □

2.2 Étude de l'énergie d'un graphe

L'application présentée dans cette section est issue de la chimie. Nous rappelons d'abord quelques notions élémentaires de chimie organique afin de mieux faire comprendre les motivations de cette recherche.

Un **graphe chimique** est un graphe pouvant représenter une molécule. Les sommets de ce graphe représentent des atomes tandis que les arêtes représentent des liaisons entre atomes. En réalité, chaque atome comporte un certain nombre de **couches**

d'électrons ou **orbitales**. Selon la charge associée au noyau, un certain nombre d'électrons graviteront autour de l'atome, et il se peut que certaines *couches* ne soient pas complètes. Par exemple l'atome d'hydrogène a un noyau de charge +1 équilibrée par la présence d'un électron sur sa première couche. Cette couche peut contenir jusqu'à 2 électrons. L'atome d'Hydrogène peut mettre en commun son électron avec un électron d'un autre atome, ce qui complèterait sa dernière orbitale et formerait une liaison. Dans le cas de l'atome de carbone, la charge du noyau est +6, équilibré par 6 électrons, deux sur la première couche et quatre sur la seconde. La seconde couche pouvant comporter jusqu'à 8 électrons, l'atome de carbone peut mettre en commun quatre électrons avec d'autres atomes, ce qui complèterait sa dernière couche. Dans ce dernier cas, l'atome peut avoir quatre liaisons simples (dites liaisons σ) ou avoir des liaisons multiples (mettant en jeu plusieurs électrons et appelées liaisons π) avec certains atomes.

Une molécule composée d'atomes de carbone et d'hydrogène est appelée **Hydrocarbone** . Un hydrocarbone qui comporte une succession de liaisons simples et de liaisons doubles est appelé **hydrocarbone conjugé** . Nous appelerons **π-électrons** les électrons impliqués dans une liaison de type π.

Une des plus remarquables applications de la théorie des graphes en chimie est basée sur la correspondance étroite entre les valeurs propres de la matrice d'adjacence du graphe (souvent appelées par abus de langage valeurs propres du graphe) et les niveaux d'énergie moléculaire orbitale des π-électrons des hydrocarbones conjugués.

Pour plus de détails, le lecteur pourra consulter les ouvrages de Cvetković *et al* [36], Gutman et Polansky [74] ou Dias [44]. Si G est le graphe représentant une molécule d'hydrocarbone conjugué et $\lambda_1, \lambda_2, \ldots, \lambda_n$ sont ses valeurs propres, alors dans l'approximation de l'énergie moléculaire de liaison due à Hückel (Hückel Molecular Orbital theory, ou HMO, en anglais), l'énergie de la $i^{ème}$ orbite moléculaire est donnée

par

$$E_i = \alpha + \lambda_i \beta \tag{2.70}$$

où α et β sont des constantes. Afin de simplifier le formalisme, il est de coutume de poser $\alpha = 0$ et $\beta = 1$, et en ce cas l'énergie orbitale des π-électrons et les valeurs propres du graphe coïncident.

L'énergie totale π-électronique (E) est égale à la somme pondérée des énergies associées à chaque orbitale.

$$E = \sum_{i=1}^{n} g_i \; E_i = \sum_{i=1}^{n} g_i \; \lambda_i \tag{2.71}$$

où g_i est le nombre d'électrons dans l'orbitale. En raison de restrictions provenant du principe d'exclusion de Pauli, g_i est égal à 0, 1 ou 2.

Dans la majorité des cas qui intéressent les chimistes, $g_i = 2$ quand $\lambda_i > 0$ et $g_i = 0$ quand $\lambda_i < 0$, ce qui implique

$$E = 2 \sum_{\lambda_i > 0} \lambda_i. \tag{2.72}$$

Comme la somme des valeurs propres est nulle, l'**énergie d'un graphe** se défini comme suit :

$$E = E(G) = \sum_{i=1}^{n} |\lambda_i|. \tag{2.73}$$

Le membre de droite de l'équation (2.73) est défini pour tous les graphes (qu'ils représentent le squelette d'atomes de carbones d'un système conjugué π-électrons ou non). Si G est un graphe quelconque, on définit donc $E(G)$ à l'aide de l'équation (2.73), et on l'appelle, par extension, l'énergie du graphe G.

Un certain nombre de résultats mentionnés dans la littérature chimique au sujet de l'énergie totale π-électronique des orbitales moléculaires de Hückel d'hydrocarbones conjugués sont en fait, des résultats valides pour l'énergie d'un graphe quelconque.

Une revue des propriétés mathématiques de E, est donnée dans le chapitre 12 du livre de Gutman et Polansky [74] ainsi que l'article de synthèse de Gutman [69].

Les propriétés de l'énergie totale π-électronique des orbitales moléculaires de Hückel et en particulier ses rapports avec la structure moléculaire, attirent l'attention des chimistes théoriciens depuis plus d'un demi siècle. Parmi la multitude de résultats obtenus en ce domaine, un certain nombre sont relatifs aux graphes (moléculaires) extrémaux pour E, et/ou à des bornes inférieures ou supérieures sur E. La plupart des bornes sur E sont valides pour des classes particulières de graphes telles que : bipartis, benzenoïdes, arbres, et autres. Peu de résultats s'appliquent à tous les graphes. Les graphes d'énergie extrémale ne sont connus que pour les arbres avec n sommets et pour les arbres sur n sommets avec couplage parfait (c'est-à-dire un ensemble d'arêtes disjointes recouvrant tous les sommets). Les résultats d'une recherche par ordinateur (mais seulement jusqu'à 6 sommets) sont mentionnés dans un article de Cvetković et Gutman [37].

L'objet du travail présenté ici est de contribuer à combler ce vide.

Dans un premier temps, AGX a été appliqué au problème de trouver des graphes d'énergie minimale. Les nombres n de sommets et m d'arêtes ont été choisis comme paramètres avec pour valeurs $2 \leq n \leq 12$ et $0 \leq m \leq \frac{n(n-1)}{2}$. Lors de la recherche, tous les voisinages ont été utilisés à l'exception de ceux qui peuvent modifier les valeurs de n ou m, et le programme a fonctionné durant une fin de semaine sur un Pentium 133 MHz.

Les valeurs obtenues, conjecturées minimales pour E, en fonction de n et m sont représentées sur la figure 2.8. Il est plus facile d'interpréter ces résultats pour des valeurs fixées de n ou m. Ceux pour $n = 12$ sont présentés sur la figure 2.9.

En regardant cette fonction, nous observons certaines régularités : une courbe à la courbure décroissante (plus du bruit) sur la partie gauche, et une droite sur la

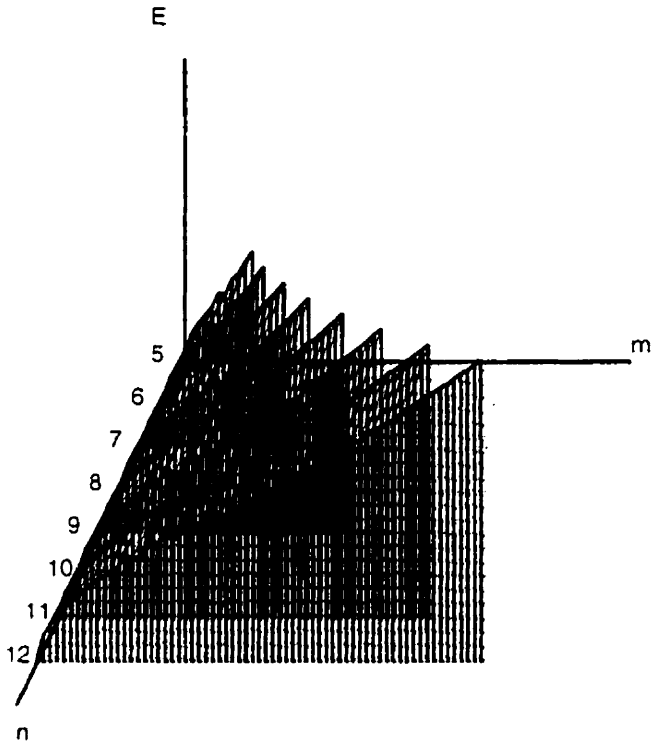


Figure 2.8 – Valeurs conjecturées minimales de E pour $n = 12$ et $m = 0$ à 66

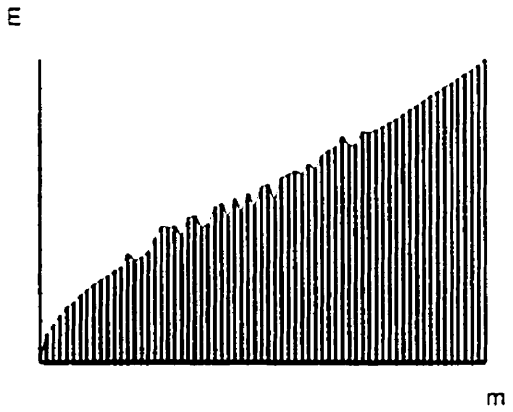


Figure 2.9 – Plus petites valeurs de E pour $n = 12$ et $m = 0$ à 66 trouvées par le programme

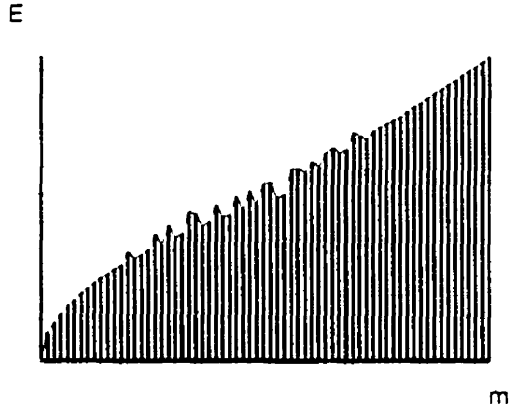


Figure 2.10 – Plus petites valeurs de E pour $n = 12$ et $m = 0$ à 66 après corrections

partie droite. Un examen des graphes extrémaux situés sur la courbe, a montré qu'il s'agissait de **graphes bipartis complets**, (c'est-à-dire dont les sommets peuvent être partitionnés en deux ensembles V_1 et V_2 tels que deux sommets dans le même ensemble ne sont jamais adjacents alors que deux sommets dans des ensembles différents le sont toujours), avec éventuellement des sommets isolés. De tels graphes, quand ils existent pour n et m donnés et sont différents des graphes obtenus par **AGX** ont une énergie inférieure à ces derniers. Ceci nous a permis d'améliorer les résultats dans deux cas sur 67. Des améliorations interactives sur la partie de droite de la courbe, dans la zone linéaire, nous ont permis d'améliorer la solution dans deux cas de plus.

Les résultats corrigés sont présentés sur la figure 2.10.

En superposant les résultats obtenus pour $n = 2$ à 12, comme l'illustre la figure 2.11, nous constatons que le côté gauche des courbes de valeurs pour n fixé coïncide dans un bon nombre de cas. Il est alors facile de voir que les points sont sur ou au dessus de

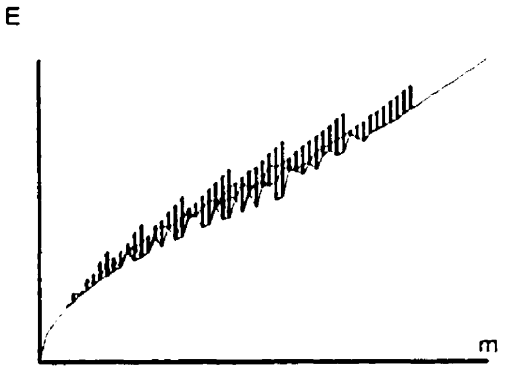


Figure 2.11 - Résultats superposés pour $n = 2$ à 12

la courbe $2\sqrt{m}$. De plus, cette valeur est atteinte quand il existe un graphe composé d'un graphe biparti complet et éventuellement des sommets isolés pour les valeurs données de n et m. Ceci se produit si et seulement si

$$\begin{cases} m & = a \times b \\ n & \geq a + b \\ a, b & entiers, \end{cases} \tag{2.74}$$

ce qui arrive 27 fois pour $n = 12$. La comparaison des résultats de plusieurs essais a montré que s'il y a plusieurs paires $(a, b), (a', b') \ldots$ qui satisfont les conditions mentionnées plus tôt, elles correspondent à des graphes de même énergie. Un exemple avec $n = 5, m = 4$ est d'une part l'étoile et d'autre part un cycle à 4 sommets auquel est ajouté un sommet isolé. Ces deux graphes ont une énergie E égale à 4.

Quand les valeurs m et n sont telles qu'il n'y a pas de paire a, b d'entiers positifs tels que $m = a \times b$ et $a + b \leq n$, les graphes extrêmes semblent être des graphes bipartis modifiés avec $a \times b < m$, et $m - a \times b$ arêtes reliant un sommet du plus petit côté à d'autres sommets du même côté. Les 10 graphes extrêmes pour $n = 12$ qui ne sont

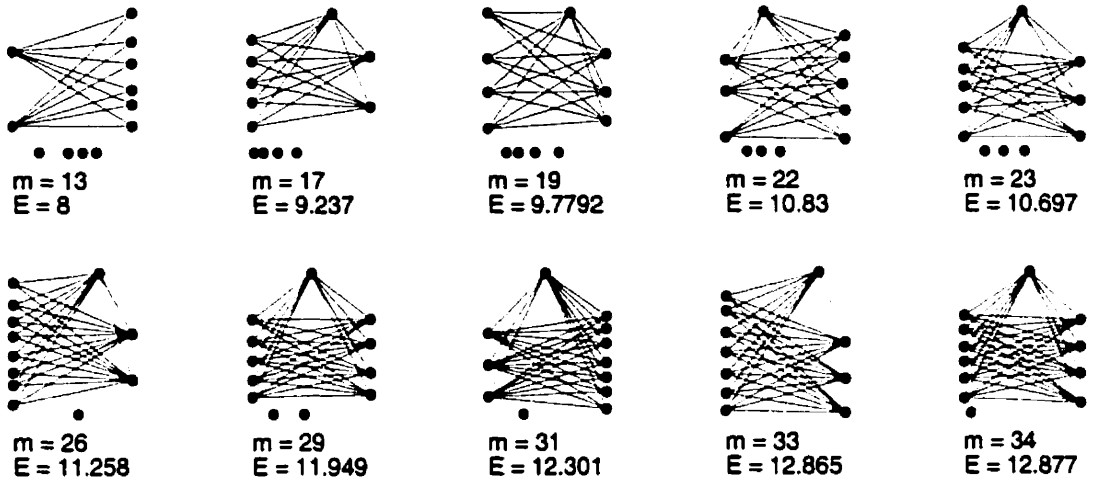


Figure 2.12 – Graphes obtenus avec AGX

pas des graphes bipartis complets, avec éventuellement des sommets isolés, et pour lesquels $m \leq \lfloor \frac{n}{2} \rfloor \times \lceil \frac{n}{2} \rceil$, le nombre maximal d'arêtes dans un graphe biparti complet sont présentés sur la figure 2.12.

Remarquons que le graphe biparti qui est modifié par addition d'arêtes n'est pas nécessairement celui qui comportait le plus d'arêtes. Par exemple, dans le cas $n = 12$ et $m = 34$ le graphe extrême avec $a = 5$, $b = 6$ et 4 arêtes ajoutées (le dernier sur la figure 2.12) a une énergie $E = 12,877$, alors que celui avec $a = 4$, $b = 8$ et 2 arêtes ajoutées a une énergie $E = 13,17$, tel que vérifié interactivement par modification du précédent. Les remarques faites jusqu'ici peuvent être résumées par les conjectures suivantes (appelées théorèmes si la preuve en a été faite).

Théorème AGX 8 *Pour tout graphe G l'énergie vérifie*

$$E \geq 2\sqrt{m}. \tag{2.75}$$

De plus, la borne est atteinte si et seulement si G est un graphe biparti complet, auquel sont éventuellement ajoutés des sommets isolés.

Preuve:

Soit G un graphe quelconque à n sommets, avec m arêtes, et notons $\lambda_1, \lambda_2, \ldots, \lambda_n$ ses valeurs propres. Trois relations bien connues pour les valeurs propres de la matrice d'adjacence d'un graphe (voir [36]) sont :

$$\sum_{i=1}^{n} \lambda_i = 0 \tag{2.76}$$

$$\sum_{i=1}^{n} (\lambda_i)^2 = 2m \tag{2.77}$$

$$\sum_{i<j} \lambda_i \lambda_j = -m. \tag{2.78}$$

Si G comporte des sommets isolés, alors à chaque tel sommet nous pouvons associer une valeur propre nulle. Ajouter des sommets isolés à G ne changera donc ni m ni E.

De l'équation (2.73) nous avons

$$E(G)^2 = \sum_{i=1}^{n} |\lambda_i|^2 + 2 \sum_{i<j} |\lambda_i||\lambda_j|. \tag{2.79}$$

En vertu de l'équation (2.77), la première somme du membre droit de l'équation (2.79) est égale à $2m$. Au vu de l'équation (2.78),

$$\sum_{i<j} |\lambda_i||\lambda_j| \geq |\sum_{i<j} \lambda_i \lambda_j| = m. \tag{2.80}$$

Par conséquent,

$$E(G)^2 \geq 4m \tag{2.81}$$

soit

$$E(G) \geq 2\sqrt{m}, \tag{2.82}$$

ce qui démontre le *Théorème AGX* 8 □

Conjecture 13 *Si G est un graphe avec n sommets, $m \leq \lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$ arêtes et d'énergie minimum, alors*
(i) s'il existe des entiers positifs a et b tels que $a \times b = m$ et $a + b \leq n$, G est un graphe biparti complet auquel sont éventuellement ajoutés des sommets isolés.
(ii) sinon, G est un graphe biparti complet $K_{a',b'}$ avec $a' \times b' \leq m$ et $a' + b' \leq n$ modifié par l'ajout de $m - a' \times b'$ arêtes reliant un sommet du plus petit côté de $K_{a',b'}$ et d'autres sommets de ce côté, auquel sont éventuellement ajoutés des sommets isolés.

La courbe (2.75), et les résultats pour $n = 12$ sont représentés sur la figure 2.13. Il apparaît que la coïncidence sur la partie gauche de la courbe, pour $m \leq 36$, c'est-à-dire le plus grand nombre d'arêtes dans un graphe biparti avec 12 sommets, est excellente puisque la borne est atteinte pour 27 valeurs sur 37. Toutefois, la courbe est bien en dessous des valeurs observées pour $m > 36$. Observons alors que si nous prolongeons la tendance linéaire qui apparaît sur la partie droite de la courbe, nous obtenons une droite passant par l'origine. Comme pour le graphe complet à n sommets $m = \frac{n(n-1)}{2}$ et $E = 2n - 2$, (voir [36], page 72) la pente de cette droite peut facilement être calculée et nous mène au théorème suivant.

Théorème AGX 9 *Pour tout graphe G, l'énergie E satisfait*

$$E \geq \frac{4m}{n}. \tag{2.83}$$

Preuve:
La plus grande valeur propre d'un graphe ne peut être inférieure au degré moyen de ce dernier (voir par exemple [36]).Sachant que le degré moyen d'un graphe est $\frac{2m}{n}$, nous avons la relation :

$$\frac{2m}{n} \leq \lambda_1. \tag{2.84}$$

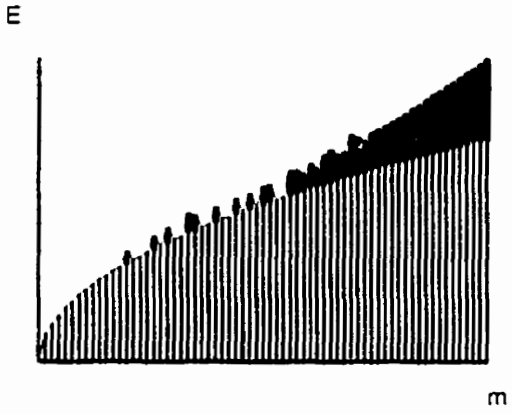


Figure 2.13 – Plus petites valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées à la *Conjecture 8*

D'autre part, si nous notons par S la somme des valeurs propres positives de G, nous avons

$$\lambda_1 \leq S = \frac{E(G)}{2} \tag{2.85}$$

en vertu de l'équation (2.76). D'où

$$E(G) \geq \frac{4m}{n}, \tag{2.86}$$

ce qui démontre le *Théorème AGX* 9. □

Pour $n = 12$ et $m \geq 37$, la borne (2.83) est atteinte par les graphes obtenus avec AGX 12 fois sur 30. Les deux courbes (2.75) et (2.83), et les résultats pour $n = 12$ sont présentés sur la figure 2.14. Les valeurs observées paraissent raisonnablement proches des courbes et coïncident dans bon nombre de cas.

Les caractérisations des graphes extrêmes pour $n = 12$ et $m \geq 37$ semblent plus difficiles que pour $m \leq 36$. Comme les arêtes sont nombreuses, plutôt que d'utiliser

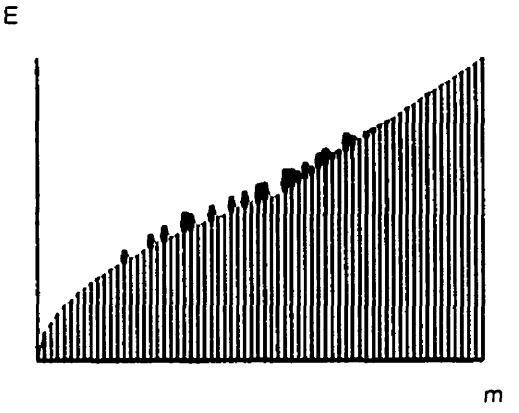
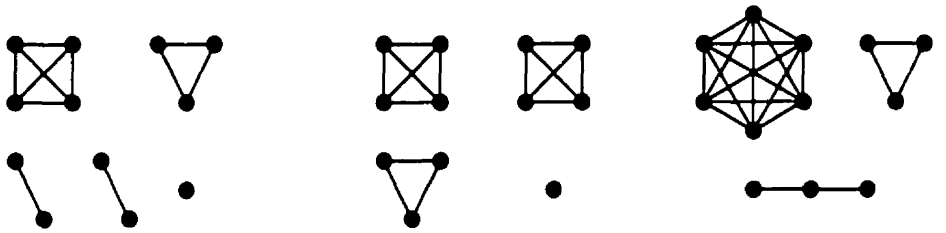


Figure 2.14 - Plus petites valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées aux deux conjectures *Conjecture 8* et *Conjecture 9*

G, il est plus approprié d'utiliser $\bar{G}$, son **complémentaire** (ou **complément**) où une arête relie deux sommets v_k et v_l de $\bar{G}$ si et seulement si ces sommets ne sont pas reliés dans G. Les compléments $\bar{G}$ de quelques graphes G trouvés par AGX pour $n = 12$ sont représentés sur la figure 2.15. Il apparaît que la plupart de ces graphes $\bar{G}$, mais pas tous, sont composés de cliques disjointes auxquelles sont ajoutés éventuellement des sommets isolés. Les cliques de tailles quasi égales semblent favorisées même si parfois elles ne sont pas disjointes. En effet, un cas dans lequel les valeurs de n et m sont telles qu'il existe une décomposition en cliques de tailles très inégales, par exemple $n = 12$, $m = 43$ et $\bar{m} = 23$, (qui se décompose en $K_7, 2K_2$ et K_1) a un graphe conjecturé extrémal G tel que $\bar{G}$ se décompose en une clique sur 5 sommets et une composante de 7 sommets composée de deux cliques sur 5 et 3 sommets partageant un sommet.

Les **graphes de Turan**, décrits par Berge [14], sont composés de cliques disjointes avec des cardinalités égales ou différant d'au plus un. Les graphes $\bar{G}$ qui sont des



n=12 m=55 n=12 m=51 n=12 m=46

Figure 2.15 – Compléments de graphes conjecturés extrémaux

graphes de Turan ne sont pas nécessairement les compléments de graphes G tels que la borne (2.83) soit atteinte. Ce qui est plus surprenant est que cette propriété ne tient même pas lorsque $\bar{G}$ est composé d'arêtes disjointes. Par exemple, si $n = 11, m = 52, \bar{m} = 3, \bar{G}$ consiste en 3 arêtes disjointes auxquelles sont ajoutés 5 sommets isolés et $E = 18.954$, alors que la borne est $\frac{4m}{n} = 18.909$.

Lors d'une seconde série d'expériences, AGX a été appliqué au problème de trouver des graphes connexes d'énergie minimum, à nouveau avec n et m comme paramètres. Les valeurs considérées étaient $2 \leq n \leq 12$ et $n - 1 \leq m \leq \frac{n(n-1)}{2}$. Le temps de calcul requis était similaire à celui de la première série d'expériences. Les valeurs conjecturées minimales pour E dans le cas des graphes connexes en fonction de n et m sont représentées sur la figure 2.16. Les valeurs minimums de E pour n donné sont toujours obtenues quand $m = n - 1$, soit, pour les étoiles.

Théorème AGX 10 *Pour tout graphe connexe G*

$$E \geq 2\sqrt{n-1} \tag{2.87}$$

et la borne est serrée si et seulement si G est une étoile.

L'équation (2.87) découle du *Théorème AGX* 8, et de la relation $E \geq 2\sqrt{m}$ vraie pour tout graphe connexe. Cette borne sera atteinte si et seulement si G est un graphe connexe biparti complet avec $n - 1$ arêtes, soit l'étoile.

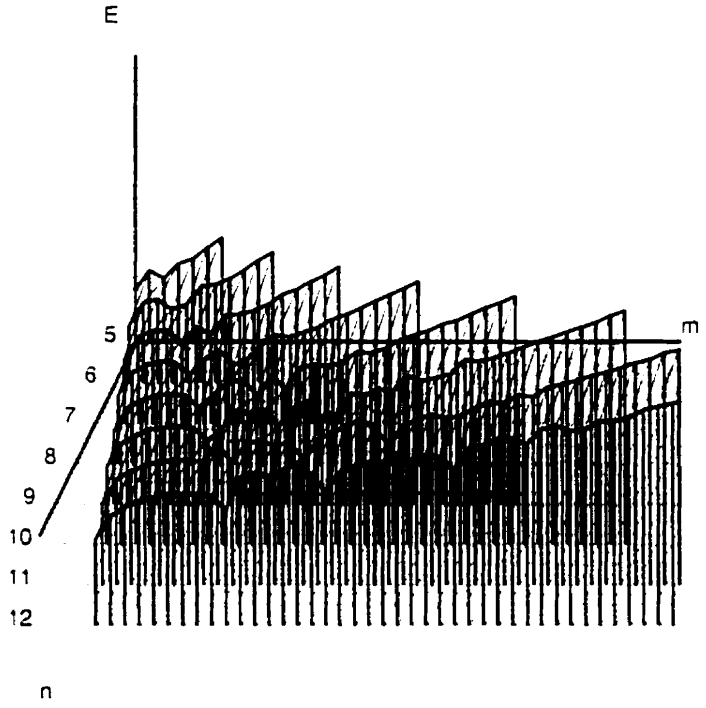


Figure 2.16 – Énergie minimum pour des graphes connexes

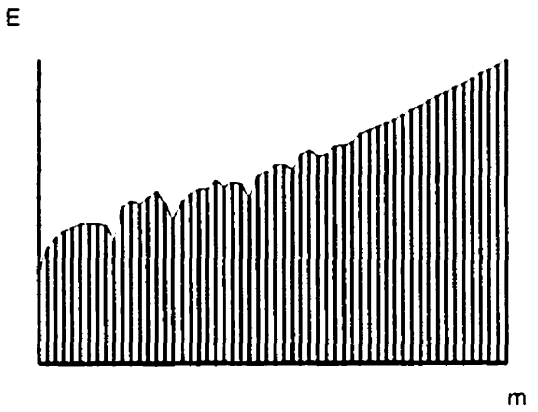


Figure 2.17 – Énergie minimum pour des graphes connexes avec $n = 12$

Les résultats pour $n = 12$ sont donnés sur la figure 2.17. La contrainte de connexité fait que les valeurs de E trouvées avec AGX sont plus grandes que celles trouvées dans le cas non contraint sur la partie gauche de la courbe excepté pour les valeurs de m telles qu'il existe un graphe biparti complet avec 12 sommets et m arêtes. Entre de telles valeurs E croit généralement dans un premier temps avant de décroître lorsque m croit. La courbe entre les deux premiers graphes sélectionnés par la moyenne flottante, qui correspondent à l'étoile et au graphe biparti complet $K_{2,10}$, est régulière. Celles entre les minima suivants le sont moins, mais ceci est peut-être dû au fait qu'AGX n'a pas trouvé les graphes d'énergie minimale dans tous les cas. La séquence de graphes obtenus pour $n = 12$ et $11 \leq m \leq 20$ est représentée sur la figure 2.18. Elle montre que les arêtes adjacentes au même sommet v_k sont ajoutées une à une jusqu'à $m = 15$, alors, l'arête reliant le sommet v_l, adjacent à tous les autres, à v_k est enlevée et deux arêtes adjacentes à v_k sont ajoutées. Ensuite, des arêtes adjacentes à v_k sont ajoutées à nouveau une à une. Les valeurs de m pour lesquelles l'arrête est enlevée et deux arêtes sont ajoutées pour $n = 5$ à 12 sont respectivement 6,7,8,10,11,13,14 et 16. Nous en déduisons la conjecture suivante :

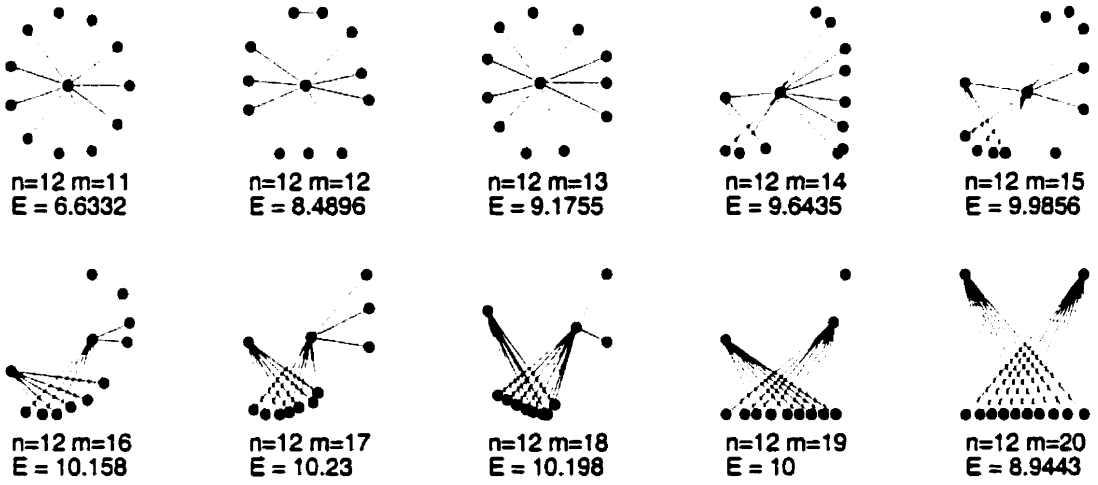


Figure 2.18 – Graphes d'énergie conjecturée minimum pour $m = 11$ à 20 et $n = 12$

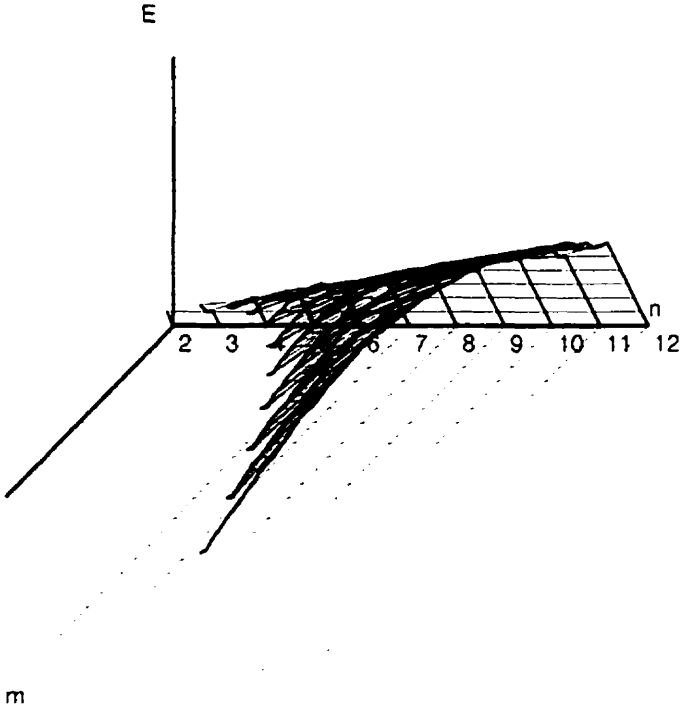


Figure 2.19 – Plus grandes valeurs de E pour $n = 5$ à 12 et $m = 0$ à $\frac{n(n-1)}{2}$

Conjecture 14 *Les graphes connexes G avec $n \geq 6$ sommets, $n-1 \leq m \leq 2(n-2)$ arêtes et une énergie minimum sont les étoiles avec $m-n+1$ arêtes supplémentaires toutes connectées au même sommet pour $m \leq n + \lceil \frac{n-7}{2} \rceil$, et sinon des graphes bipartis avec 2 sommets d'un côté, un desquels est connecté à tous les sommets de l'autre côté.*

Lors d'une troisième série d'expériences, AGX a été appliqué au problème de trouver des graphes avec une énergie maximum en utilisant à nouveau n et m comme paramètres. Les valeurs pour les paramètres sont les mêmes que pour la première série d'expériences, et le temps de calcul de nouveau similaire. Les valeurs conjecturées maximales de E en fonction de n et m sont représentées sur la figure 2.19 et les valeurs pour $n = 12$, sur la figure 2.20.

Avant de poursuivre cette étude, considérons une borne supérieure bien connue, due à McClelland [97] :

$$E \leq \sqrt{2mn}. \tag{2.88}$$

Cette borne est atteinte pour $m = 0$ ainsi que $m = \frac{n}{2}$ quand n est pair. La courbe (2.88) et les résultats pour $n = 12$ sont présentés sur la figure 2.22.

Observons que la borne (2.88) croit de manière monotone avec m tandis que la courbe des plus grandes valeurs de E passe par un maximum avant de décroître avec m (sauf une légère augmentation lorsqu'on atteint le graphe complet) quand $n \geq 7$.

Au vu des résultats obtenus à l'aide d'**AGX**, il apparaît clairement que la partie gauche de la courbe est un segment de ligne droite. De plus, des segments de lignes droites similaires, avec la même pente apparaissent pour les autres valeurs de n. Les graphes extrêmes correspondant sont composés d'arêtes disjointes auxquelles sont éventuellement ajoutés des sommets isolés. Ceci nous conduit au théorème suivant.

Théorème AGX 11 *Pour tout graphe G avec m arêtes, l'énergie vérifie*

$$E \leq 2m \tag{2.89}$$

et la borne est atteinte si et seulement si G est composé d'arêtes disjointes auxquelles sont éventuellement ajoutés des sommets isolés.

Preuve: La preuve de ce théorème, utilise les résultats de Mc Clelland décrits par l'équation (2.88) :

$$E \leq \sqrt{2mn}.$$

Supposons maintenant que G ne comporte pas de sommets isolés ; le nombre maximum de sommets d'un tel graphe est $2m$, ce qui se produit quand $G = mK_2$. Dans tout autre cas, nous avons $n < 2m$ d'où

$$E \leq \sqrt{2mn} \leq \sqrt{2m \times 2m} = \sqrt{4m^2} = 2m. \tag{2.90}$$

Sachant qu'ajouter des sommets isolés à G ne change pas son énergie, la preuve du *Théorème AGX* 11 s'en suit. □

Nous nous sommes rendu compte après avoir trouvé ce résultat que la *Conjecture 146* de *Graffiti* lui était équivalente et avait déjà été démontrée par Shearer [56]. Cette conjecture s'énonce comme suit : la somme des valeurs propres positives d'un graphe est inférieure ou égale à sa taille (ou nombre d'arêtes). En sachant que la somme des valeurs propres d'un graphe est nulle, l'équivalence des deux résultats est évidente. Toutefois, cette expérience illustre la nécessité d'une notation unifiée et d'une base de données centralisée de conjectures et théorèmes qui permettraient d'accéder aisément aux résultats cherchés même s'ils sont exprimés en termes différents.

La figure 2.21 représente les plus grandes valeurs de E pour $n = 12$ ainsi que la valeur $2m$, comme illustration du *Théorème AGX* 11.

La figure 2.23 représente les valeurs obtenues à l'aide du programme comparées avec les bornes de (2.88) et (2.89).

Les résultats pour l'énergie maximum semblent plus difficiles à analyser que ceux pour l'énergie minimum, excepté pour les bornes déjà décrites.

Les plus grande valeurs trouvées pour E, n étant donné sont indiquées sur le Tableau 2.2

Jusqu'à $n = 7$, les graphes d'énergie maximum sont les graphes complets et $E = 2n - 2$; donc l'énergie maximum croit linéairement avec n. Ensuite, ces valeurs croissent plus que linéairement. On déduit de $m \leq \frac{n(n-1)}{2}$ et (2.88) que

$$E \leq \sqrt{n^2(n-1)} < n^{\frac{3}{2}} \tag{2.91}$$

En ajustant une courbe de puissance $\frac{3}{2}$ au dessus de la différence entre les plus grandes valeurs de E et la ligne $2n - 2$ nous obtenons la conjecture suivante.

Tableau 2.2 – Plus grandes valeurs de E pour n donné.

n	E	n	E
5	8	9	17.06
6	10	10	20
7	12	11	22.91
8	14.32	12	26

Conjecture 15 *Pour tout graphe G avec n sommets, l'énergie E vérifie*

$$E \leq 2n - 2 + 0.4[\max(m-7,0)]^{\frac{3}{2}}. \tag{2.92}$$

Cette borne est atteinte pour $n \leq 7$, $n = 9$ et $n = 10$. Elle est plus compliquée que les précédentes et il est peu probable qu'elle soit atteinte pour un grand nombre de valeurs de n. Le graphe avec la plus grande énergie et $n = 10$, pour lequels $E = 26$, est le complément du graphe de Petersen qui est représenté sur la figure 2.24.

Les résultats pour $n = 12$ sont comparés à toutes les bornes citées sur la figure 2.25. Les résultats sont plus clairs pour les graphes connexes avec peu d'arêtes. Pour $m = n - 1$ seules les étoiles sont connexes. Les graphes unicycliques (avec $m = n$) de plus grande énergie sont soit des cycles, soit des cycles auxquels sont ajoutés un chemin, comme représentés sur la figure 2.27.

Les résultats pour les graphes connexes avec nombre cyclomatique 2 (soit $m = n+1$) sont représentés sur la figure 2.28.

En se basant sur les résultats présentés sur la figure 2.27 nous arrivons à la conjecture suivante. Soient P_n et C_n respectivement le chemin et le circuit à n sommets, et $C_k(h)$ le graphe obtenu en connectant un sommet du circuit C_k avec un sommet terminal du chemin P_h.

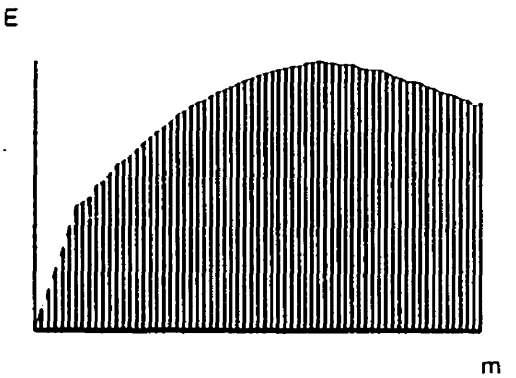


Figure 2.20 – Plus grandes valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées à la *Conjecture 15*

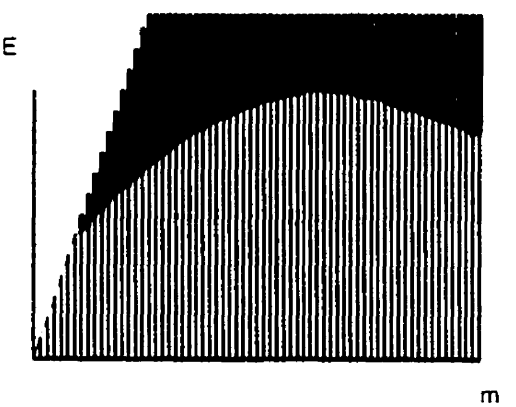


Figure 2.21 – Plus grandes valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées à la *Conjecture 11*

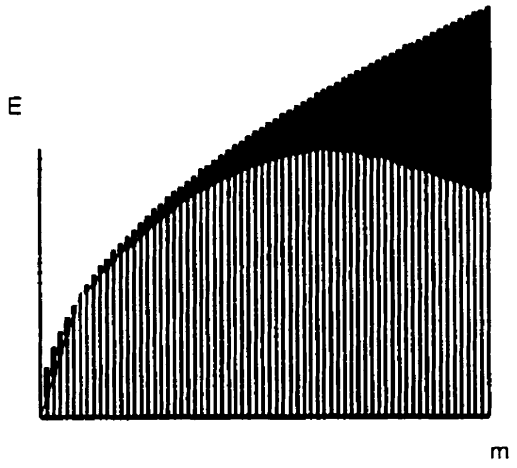


Figure 2.22 – Plus grandes valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées à la borne de McClelland

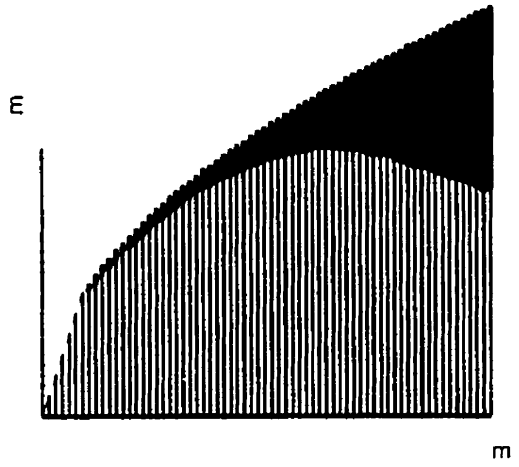


Figure 2.23 – Plus grandes valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées à la *Conjecture 11* et la borne de McClelland

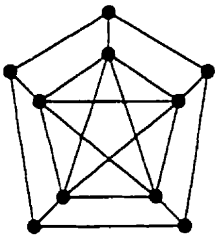


Figure 2.24 – Graphe de Petersen

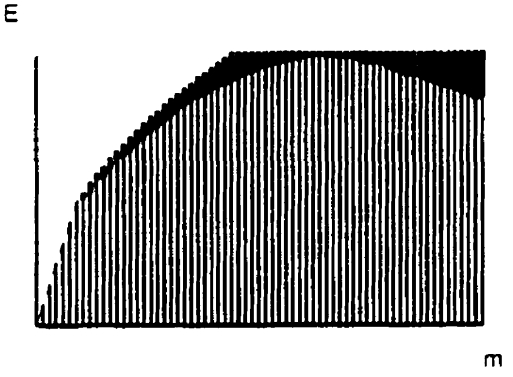


Figure 2.25 – Plus grandes valeurs de E pour $n = 12$ et $m = 0$ à 66 comparées aux bornes des *Conjecture 11, Conjecture 15* et à la borne de McClelland simultanément

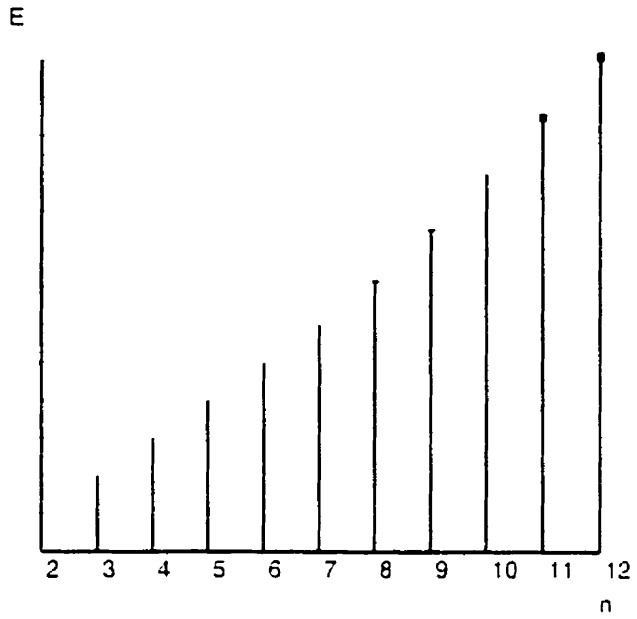


Figure 2.26 – Énergie maximum comme fonction de n

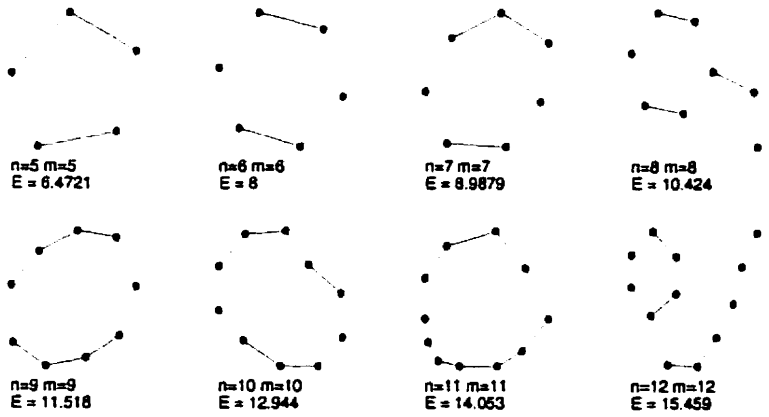


Figure 2.27 – Graphes unicycliques de plus grande énergie pour $n = 5$ à 12

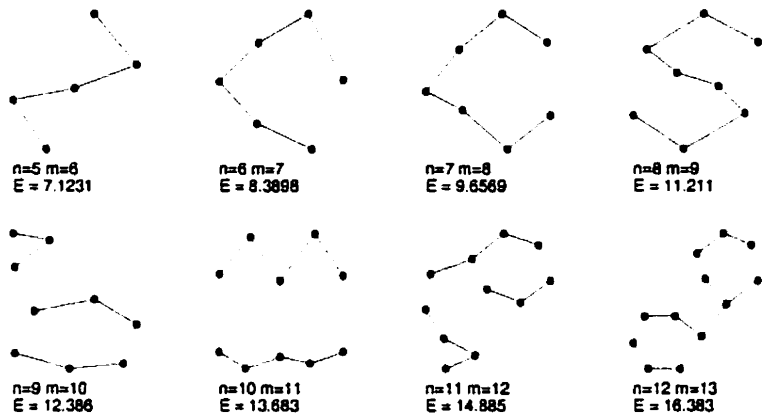


Figure 2.28 – Graphes bicycliques de plus grande énergie pour $n = 5$ à 12

Conjecture 16 *Parmi les graphes unicycliques à n sommets, le circuit C_n a l'énergie maximum si $n \leq 7$ et $n = 9, 10, 11, 13, 15$. Pour toute autre valeur de n le graphe unicyclique d'énergie maximum est $C_6(n-6)$.*

Ce résultat est intéressant du point de vue des chimistes étant donnée la fréquence des cycles hexagonaux dans les hydrocarbones.

2.3 Étude de l'index d'arbres avec contrainte de coloration

Tout arbre est un graphe biparti, ses sommets peuvent donc être coloriés avec deux couleurs, par exemple noir et blanc. Notons $\mathfrak{T}_{a,b}$ l'ensemble des arbres avec a sommets noirs et b sommets blancs. De tels arbres peuvent être vus comme des arbres de recouvrements de graphes biparti complets $K_{a,b}$. Nous pouvons supposer sans perte de

généralité que $a \geq b$. Nous nous intéressons ici au problème de trouver les arbres extrêmes $T \in \mathfrak{T}_{a,b}$ pour l'index, ou plus grande valeur propre de la matrice d'adjacence associée à T, ce qui s'écrit

$$\max_{T \in \mathfrak{T}_{a,b}} \lambda_1(T) \quad \text{et} \quad \min_{T \in \mathfrak{T}_{a,b}} \lambda_1(T)$$

pour des valeurs a et b données, et tous les arbres T pour lesquels ces extrêma sont atteints.

Une *double étoile* $S_{p,q}$ est un arbre avec un sommet de degré p, adjacent à un sommet de degré q et des sommets pendants (voir figure 2.29). Nous remarquons que $S_{a,b} \in \mathfrak{T}_{a,b}$.

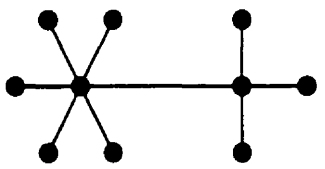


Figure 2.29 – Une double étoile

Des résultats généraux sur le comportement de la quantité $\lambda_1(G)$ suggèrent que les arbres d'index maximal doivent être proches de l'étoile tandis que ceux d'index minimal se rapprochent du chemin. La conjecture triviale que les arbres d'index maximal sont les double étoiles a déjà été faite à plusieurs reprises [35], [39]. Pour les arbres d'index minimal, il n'y a pas de candidat évident. Nous avons utilisé **AGX** pour étudier ces problèmes. Alors que le système a trouvé les double étoiles comme arbres d'index maximal dans tous les cas, les résultats pour les arbres d'index minimal étaient hétérogènes, en partie inattendus et certainement intéressants pour des raisons théoriques.

Nous allons ici présenter les résultats obtenus pour le problème de minimiser $\lambda_1(T)$ pour $T \in \mathfrak{T}_{a,b}$, avec $2 \leq n \leq 30$ et $a \geq b$.

La surface des plus petites valeurs trouvées pour $\lambda_1(T)$ est représentée sur la figure 2.30. Il est plus facile de visualiser les valeurs sur les courbes pour n constant. Trois telles courbes sont représentées sur la figure 2.31. Conjointement avec les valeurs numériques trouvées, elles conduisent au théorème suivant :

Théorème AGX 12 *Pour n fixé et $T \in \mathfrak{T}_{a,b}$, la valeur minimale de $\lambda_1(T)$ croît de manière monotone avec $a - b$.*

Nous remarquons qu'alors que les courbes de la figure 2.31 sont proches d'être convexes, si elles ne le sont pas. Nous remarquons aussi que pour a fixé et b croissant, les valeurs de $\lambda_1(T)$ ne changent pas de manière monotone. Par exemple, dans le cas où $a = 4$, les plus petites valeurs successives de $\lambda_1(T)$ pour $b = 1, 2, 3$ et 4 sont : $2, 1.9021, 1.8478$ et 1.8794.

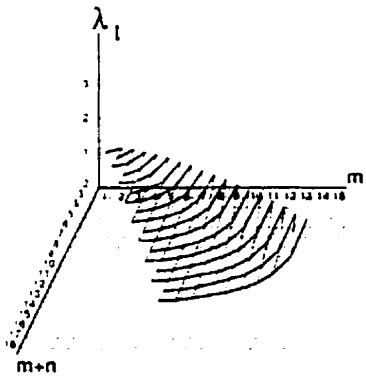


Figure 2.30 – Plus petites valeurs de $\lambda_1(T)$

Regarder les arbres extrémaux suggère une autre conjecture.

Conjecture 17 *Un sommet noir d'un arbre d'index minimal a un degré de 1 ou 2.*

Les formes des arbres peuvent changer. Ceux avec $n = 10$, $n = 20$ et avec $n = 28$ sont représentés sur la figure 2.32, 2.33 et 2.34 respectivement. Les arbres pour lesquels

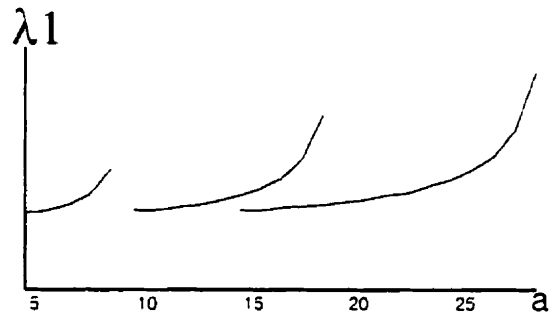


Figure 2.31 – Valeurs de $\lambda_1(T)$ pour $n = 10$, 20 et 30

a est proche des valeurs minimum ou maximum (si n est fixé) sont plus faciles à décrire, ce qui conduit à une série d'observations et de conjectures.

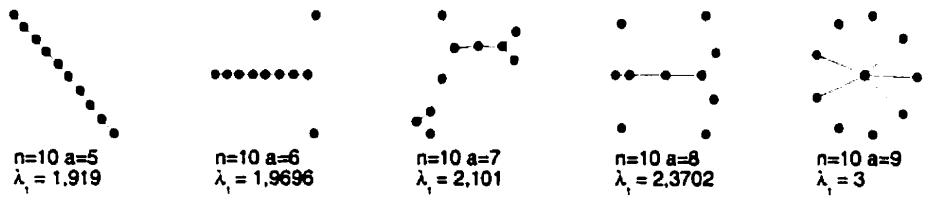


Figure 2.32 – Arbres avec plus petite valeur λ_1 et $n = 10$

Pour $a - b \leq 1$ les arbres T^* avec λ_1 minimum sont les chemins P_n. De plus

$$\lambda_1(T^*) = 2\cos\frac{\pi}{n+1} \quad \text{et} \quad \lim_{n\to\infty} \lambda_1(T^*) = 2.$$

Ceci découle du résultat de [95] qui dit que parmi les arbres à n sommets, le plus petit index (qui vaut $2\cos\frac{\pi}{n+1}$) est obtenu pour le chemin P_n. Le plus grand index (égal à $\sqrt{n-1}$) est rencontré par l'étoile $K_{n-1,1}$ qui est aussi de manière évidente le graphe d'index maximal quand $b = 1$.

Une **comète** $C_{p,r}$ est construite à partir d'une étoile $K_{p-1,1}$ ($p \geq 4$) par l'insertion de r sommets (de degré 2) à la même arête. De même, une **double comète** $D_{p,r,q}$ est

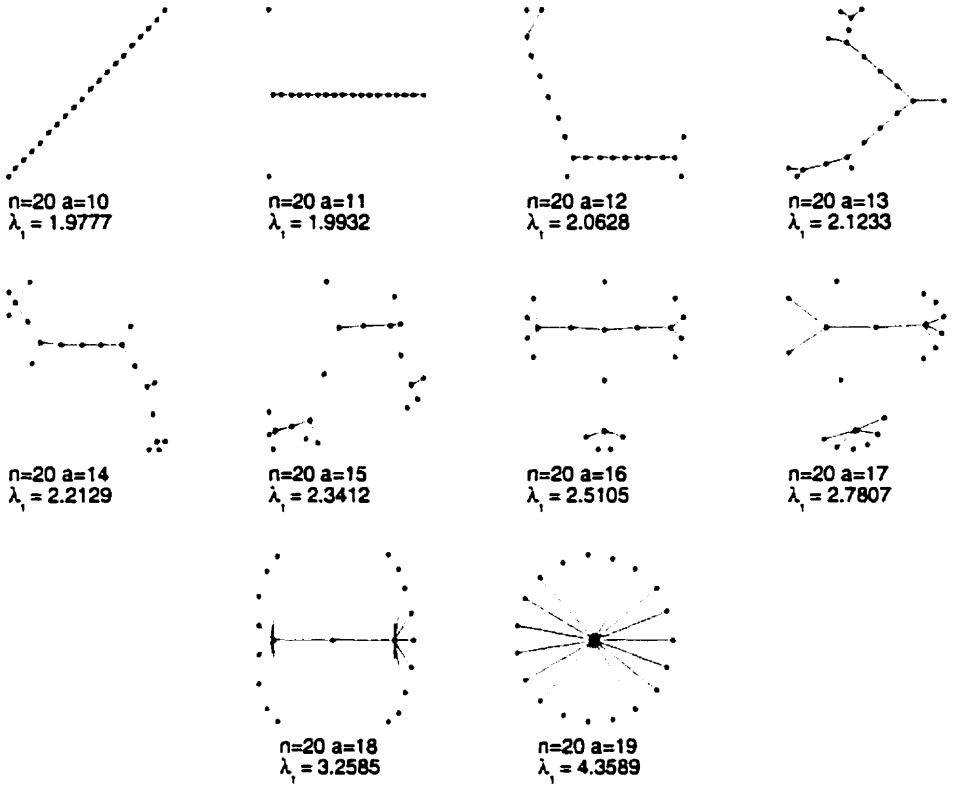


Figure 2.33 – Arbres avec plus petite valeur λ_1 et $n = 20$

obtenue d'une double étoile $S_{p,q}$ $(p, q \geq 3)$ par l'insertion de r sommets dans l'arête centrale (ou intérieure).

Théorème AGX 13 *Pour $a = b + 2$ et $n \geq 6$, les arbres T^* d'index λ_1 minimum sont les comètes $C_{4,n-4}$. De plus*

$$\lim_{n \to \infty} \lambda_1(T^*) = 2.$$

Pour $a = b + 3$ et $n \geq 7$, les arbres T^* d'index minimum λ_1 sont les doubles comètes $D_{3,n-6,3}$ et $\lambda_1(T^*) = 2$.

Rappelons qu'une **chenille** est un chemin auquel ont été ajoutées des arêtes pendantes.

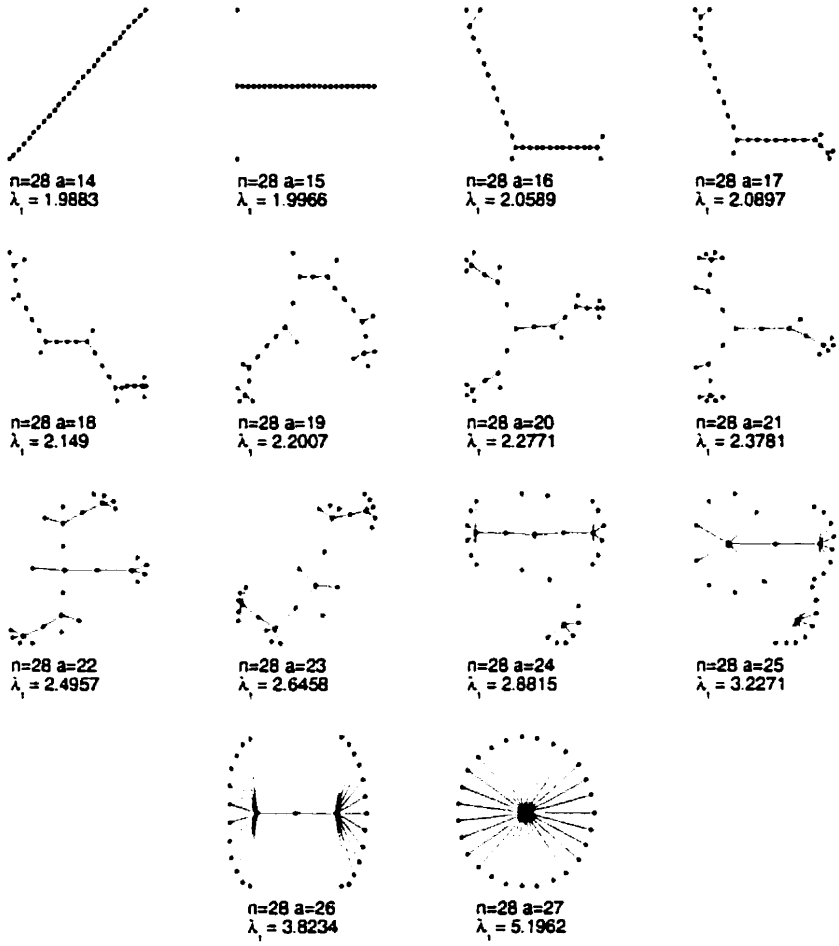


Figure 2.34 – Arbres avec plus petite valeur λ_1 et $n = 28$

Conjecture 18 *Pour $a = b+4$ les arbres T^* d'index λ_1 minimum sont des chenilles avec 5 arêtes pendantes. Pour $a = b+5$ les arbres T^* d'index minimum λ_1 sont des chenilles avec 6 arêtes pendantes.*

Les positions des arêtes pendantes changent. Pour $n = 6$ ou 7, les arbres sont des étoiles. Pour $n = 8$ les arêtes pendantes sont incidentes à un sommet de degré 4 et un sommet de degré 3. Pour $n = 9$ et 11, elles sont incidentes à deux sommets de degré 4. Dans tous les cas, les sommets de degré supérieur à 2 sont aux extrémités d'un chemin

interne (éventuellement vide). Pour $n = 10, 12, 14, 16, 18, 20, \ldots$ deux paires d'arêtes pendantes sont ajoutées chacune à un sommet extrême du chemin P_{n-5} et la dernière arête pendante est ajoutée à un sommet intérieur de ce chemin. Nous remarquons que ce sommet n'est pas nécessairement le sommet central du chemin. En effet, c'est le cas pour $n = 10$ mais pas pour $n = 12$ (en quel cas il est à distance des deux sommets de degré 3 respectivement). L'explication est que pour $n = 12, 16, 20, \ldots$ le sommet central est de couleur noire, il s'ensuit que les arêtes pendantes ne peuvent pas être ajoutées à ce sommet.

Quand $b = 1$ il n'y a qu'un seul arbre, l'étoile à n sommets. Les résultats les plus intéressants sont obtenus pour les valeurs suivantes de b.

Une double comète $D_{p,r,q}$ est **équilibrée** si $p = q$ et **quasi-équilibrée** si $|p - q| = 1$.

Théorème AGX 14 *Pour $n = 2$, les arbres T^* d'index λ_1 minimum sont soit des doubles comètes équilibrées, soit des doubles comètes quasi-équilibrées*

$$D_{\lfloor \frac{n-1}{2} \rfloor, 1, \lceil \frac{n-1}{2} \rceil}.$$

Pour $a \geq 6$ et $b = 3$ un patron semble apparaître : les arbres sont des doubles comètes avec un chemin intérieur de 5 sommets auquel des arêtes pendantes sont ajoutées sur le sommet central.

Théorème AGX 15 *Pour $b = 3$ et $a \geq 6$ les arbres T^* d'index λ_1 minimum sont des doubles comètes*

$$D_{\lceil \frac{n-1}{3} \rceil, \lceil \frac{n-1}{3} \rceil, 3}$$

au sommet central du chemin sont ajoutées $n - 2\left\lceil \frac{n-1}{3} \right\rceil - 3$ arêtes pendantes.

La plupart des arbres extrémaux trouvés sont des chenilles. Toutefois, ce n'est pas toujours le cas. Le plus petit exemple pour lequel il n'en est pas ainsi est l'arbre avec $n = 16$ et $a = 12$, qui est représenté sur la figure 2.35. Sur la même figure, nous donnons les arbres extrêmes pour $n = 26$ et $a = 19$ qui a une allure unique parmi les arbres extrémaux trouvés par **AGX**.

Les arbres d'index minimal ne sont pas toujours uniques. La figure 2.36 montre deux arbres d'index minimal pour $a = 13$ et $b = 4$. Pour chaque arbre, nous avons $\lambda_1 = 2.3702$. Nous ne connaissons pas d'autres exemples de ce type.

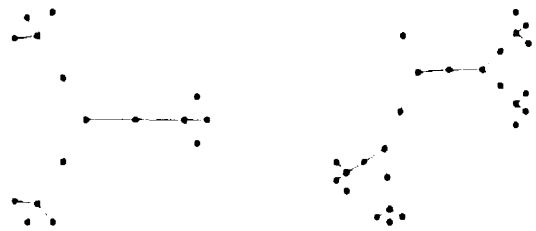


Figure 2.35 – Arbres extrêmes qui ne sont pas des chenilles

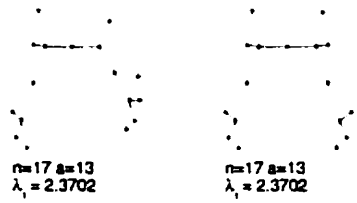


Figure 2.36 – Deux arbres d'index minimal

Les preuves des théorèmes 12, 13, 14 et 15 sont données dans [41].

2.4 Recherche d'arbres H-palindromiques

Un polynome $\sum_{k=0}^{p} a_k \lambda^k$ est dit **palindromique** si l'égalité $a_k = a_{p-k}$ est vérifiée pour tout $k = 0, 1, \ldots, p$.

Godsil et McKay [64] semblent être les premiers à avoir remarqué que les polynomes caractéristiques de certains graphes peuvent être palindromiques. Néanmoins, l'étude systématique de polynomes graphiques palindromiques a débuté avec le travail de Kennedy [83] et était initialement concentrée sur les polynomes caractéristiques et les polynomes de couplage [83] [60] [59] [70]. D'autres polynomes ont ensuite été étudiés pour leur palindromicité, tels que les polynomes d'indépendance [68] et de Hosoya (Wiener) [71] [46] [73]. Ce dernier a été introduit en 1988 par Hosoya [81], et défini comme suit. Si G est un graphe connexe, alors son polynome de Hosoya est

$$H(G) = H(G, \lambda) = \sum_{k=0}^{D} d(G, k)\, \lambda^k$$

où D représente le diamètre de G et où $d(G, k)$ est le nombre de paires de sommets de G qui sont à distance k. Par convention, $d(G, 0) = n$.

Hosoya lui même appelait $H(G)$ le *Polynome de Wiener* puisque pour $\lambda = 1$, la dérivée première de $H(G, \lambda)$ correspond à l'indice de Wiener $W(G)$. Dans le but de simplifier la notation, nous dirons que le graphe G est *H-palindromique* si son polynome de Hosoya est palindromique.

Définition 1 *Un graphe G de diamètre $D(G)$ est dit H-palindromique si $d(G, k) = d(G, D - k)$ pour tout $k = 0, 1, \ldots, \lfloor D/2 \rfloor$.*

Définition 2 *Un graphe G de diamètre $D(G)$ est dit H^*-palindromique si $d(G, k) = d(G, D - k + 1)$ pour tout $k = 1, \ldots, \lceil D/2 \rceil$.*

La conjecture qu'il n'y a pas d'arbres H-palindromiques a été faite par Gutman en 1993 [71]. Une autre recherche du même auteur, menée à l'aide de l'ordinateur [73] a étayé cette conjecture. Lors de cette étude, trois arbres H^*-palindromiques ont été trouvés (les trois seuls à moins de 13 sommets).

D'après les travaux précédents, nous étions convaincus que la conjecture était vraie. Les avantages d'une interface chercheur-ordinateur appropriée se montrent ici d'une grande importance puisque nous avons décidé de tester la conjecture à l'aide d'**AGX** à la seule condition que cela se fasse rapidement et facilement.

Afin de rechercher des arbres *H-palindromiques* avec **AGX**, nous devons trouver un problème d'optimisation dont la résolution permette de déterminer ces arbres. Dans ce but, nous utilisons la distance à la palindromicité $D^{pal}(G)$ définie par

$$D^{pal}(G) = \sum_{k=0}^{\lfloor D/2 \rfloor} |d(G,k) - d(G,D-k)|. \tag{2.93}$$

D^{pal} est supérieur ou égal à 0, et un graphe pour lequel cette valeur est nulle est *H-palindromique.*

Dans le cas des arbres avec un nombre donné de sommets, les deux premiers termes sont n et $n-1$. Les conditions sur les deux derniers termes sont donc plus restrictives que pour les autres termes (leur terme correspondant ne peut pas changer), c'est pourquoi nous avons légèrement modifié la fonction-objectif pour augmenter le poids associé à ces termes et nous avons utilisé $D^{pal'}$ défini comme suit :

$$D^{pal'}(G) = \lambda \sum_{k=0}^{1} |d(G,k) - d(G,D-k)| + \sum_{k=2}^{\lfloor D/2 \rfloor} |d(G,k) - d(G,D-k)|, \tag{2.94}$$

où λ est un nombre positif assez élevé pour que la recherche s'oriente d'abord sur des arbres dont les deux derniers termes ont les bonnes valeurs.

Après avoir ajouté à **AGX** une routine pour le calcule de $D^{pal'}$, nous avons lancé le programme un soir, et avons été surpris de trouver un contre-exemple à 21 sommets le lendemain matin.

Suite à cette expérience, nos recherches ont changé d'orientation et nous avons commencé à nous demander s'il y avait d'autres arbres *H-palindromiques*, éventuellement plus grands. Constatant, toujours avec **AGX**, qu'il y en avait en effet, nous nous sommes demandé combien et avons entrepris d'enumérer tous les arbres avec $n \leq 26$ sommets afin de les compter, à l'aide d'un programme fourni par A. Dobrynin. Les résultats de cette énumération sont résumés dans le Tableau 2.3.

Tableau 2.3 – Nombre total d'arbres à n sommets, d'arbres *H-palindromiques* et *H*-palindromiques*

n	Total	*H-palindromiques*	H*-palindromiques
4	2	0	1
5	3	0	0
6	6	0	0
7	11	0	1
8	23	0	0
9	47	0	1
10	106	0	0
11	235	0	0
12	551	0	0
13	1301	0	3
14	3159	0	0
15	7741	0	0
16	19320	0	5
17	48629	0	6
18	123867	0	0
19	317955	0	0
20	823065	0	0
21	2144505	1	39
22	5623756	2	11
23	14828074	0	2
24	39299897	2	0
25	104636890	0	410
26	279793450	0	69

Nous remarquons que le nombre d'arbres *H*-palindromiques* est relativement important alors que le nombre d'arbres *H-palindromiques* reste très modéré et ces arbres

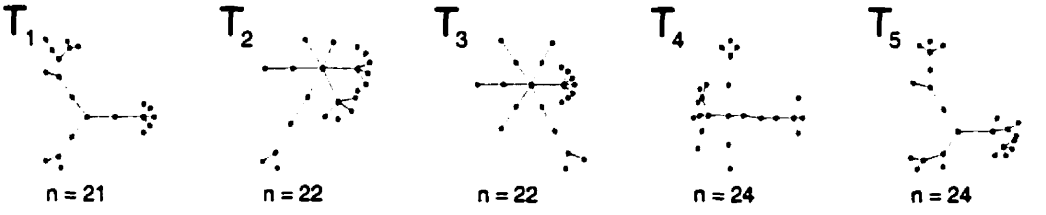


Figure 2.37 – Les cinq plus petits arbres H-palindromiques.

sont représentés sur la figure 2.37.

Les résultats de nos recherches se résument comme suit :

Théorème 1 *La conjecture qu'il n'y a pas d'arbres H-palindromiques n'est pas vraie. Le plus petit arbre H-palindromique comporte 21 sommets, il est unique et représenté sur la figure 2.37*

Le plus petit arbre *H-palindromique* a un diamètre 8. Le plus petit diamètre des arbres palindromiques trouvés est 6, et les polynômes palindromiques trouvés sont les suivants :

$$\begin{aligned}
H(T_1) &= 21 + 20\,\lambda + 34\,\lambda^2 + 25\,\lambda^3 + 31\,\lambda^4 + 25\,\lambda^5 + 34\,\lambda^6 + 20\,\lambda^7 + 21\,\lambda^8 \\
H(T_2) &= H(T_3) = 22 + 21\,\lambda + 52\,\lambda^2 + 63\,\lambda^3 + 52\,\lambda^4 + 21\,\lambda^5 + 22\,\lambda^6 \\
H(T_4) &= 24 + 23\,\lambda + 37\,\lambda^2 + 41\,\lambda^3 + 50\,\lambda^4 + 41\,\lambda^5 + 37\,\lambda^6 + 23\,\lambda^7 + 24\,\lambda^8 \\
H(T_5) &= 24 + 23\,\lambda + 39\,\lambda^2 + 41\,\lambda^3 + 46\,\lambda^4 + 41\,\lambda^5 + 39\,\lambda^6 + 23\,\lambda^7 + 24\,\lambda^8\ .
\end{aligned}$$

Tous les (cinq) arbres *H-palindromiques* que nous avons trouvés ont un diamètre pair. L'étape suivante de nos recherches fut de chercher des arbres *H-palindromiques* de diamètre impair. Une recherche intensive à l'aide d'**AGX** (sur des arbres de 32 et 36 sommets) a été infructueuse.

Nous nous sommes alors intéressés aux valeurs minimales possibles pour la fonction-objectif D^{pal}, pour des arbres avec n sommets et un diamètre impair. Notons cette valeur minimale $D^{pal}_{min}(n)$. Nous avons obtenu un résultat pour le moins inattendu :

Pour $n = 4$, 8 et 9, nous avons $D^{pal}_{min}(n) = 4$, 6 et 7, respectivement. Pour toutes les autres valeurs de n , $5 \leq n \leq 50$,

$$D^{pal}_{min}(n) = \lceil n/2 \rceil, \tag{2.96}$$

comme le montre la figure 2.38.

Les arbres correspondants sont représentés sur la figure 2.39. Nous croyons que la relation (2.96) est vérifiée pour toutes les valeurs de $n \geq 10$.

Ceci est équivalent à la conjecture suivante.

Conjecture 19 *Pour tout arbre T avec $n \geq 4$ sommets et un diamètre impair, la distance à la H-palindromicité est au moins $\lceil \frac{n}{2} \rceil$. De plus, cette borne est serrée pour tout $n \geq 10$.*

De cette conjecture découle la suivante :

Conjecture 20 *Il n'y a pas d'arbres H-palindromiques avec un diamètre impair.*

La diversité des arbres de la figure 2.39 et le fait que tout changement, même minime d'un arbre modifie considérablement son polynome de Hosoya suggèrent que les conjectures 19 et 20 seront difficiles à prouver.

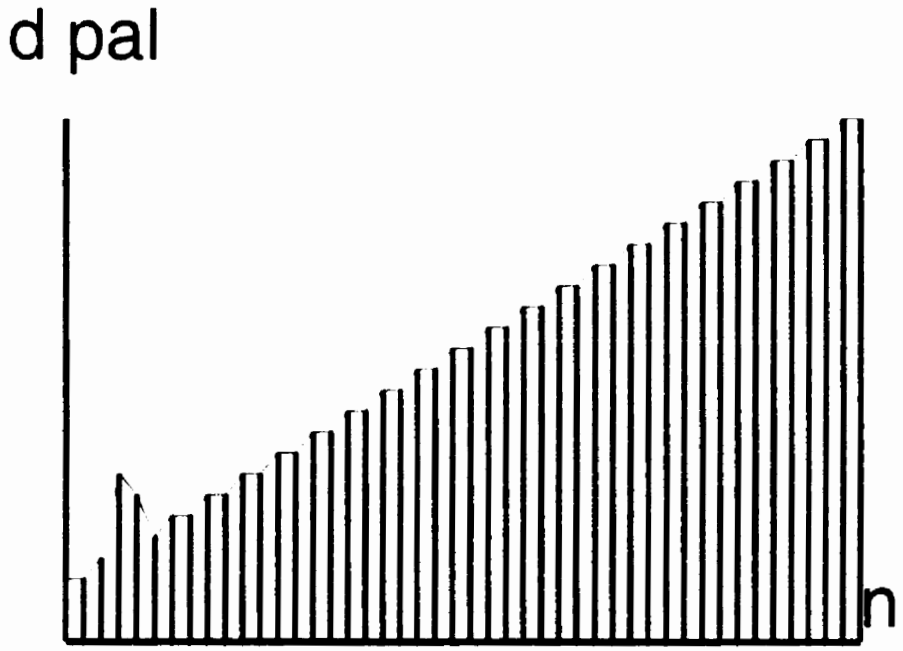


Figure 2.38 – Distance à la H-palindromicité conjecturée minimum pour les arbres de diamètre impair avec 5 à 50 sommets.

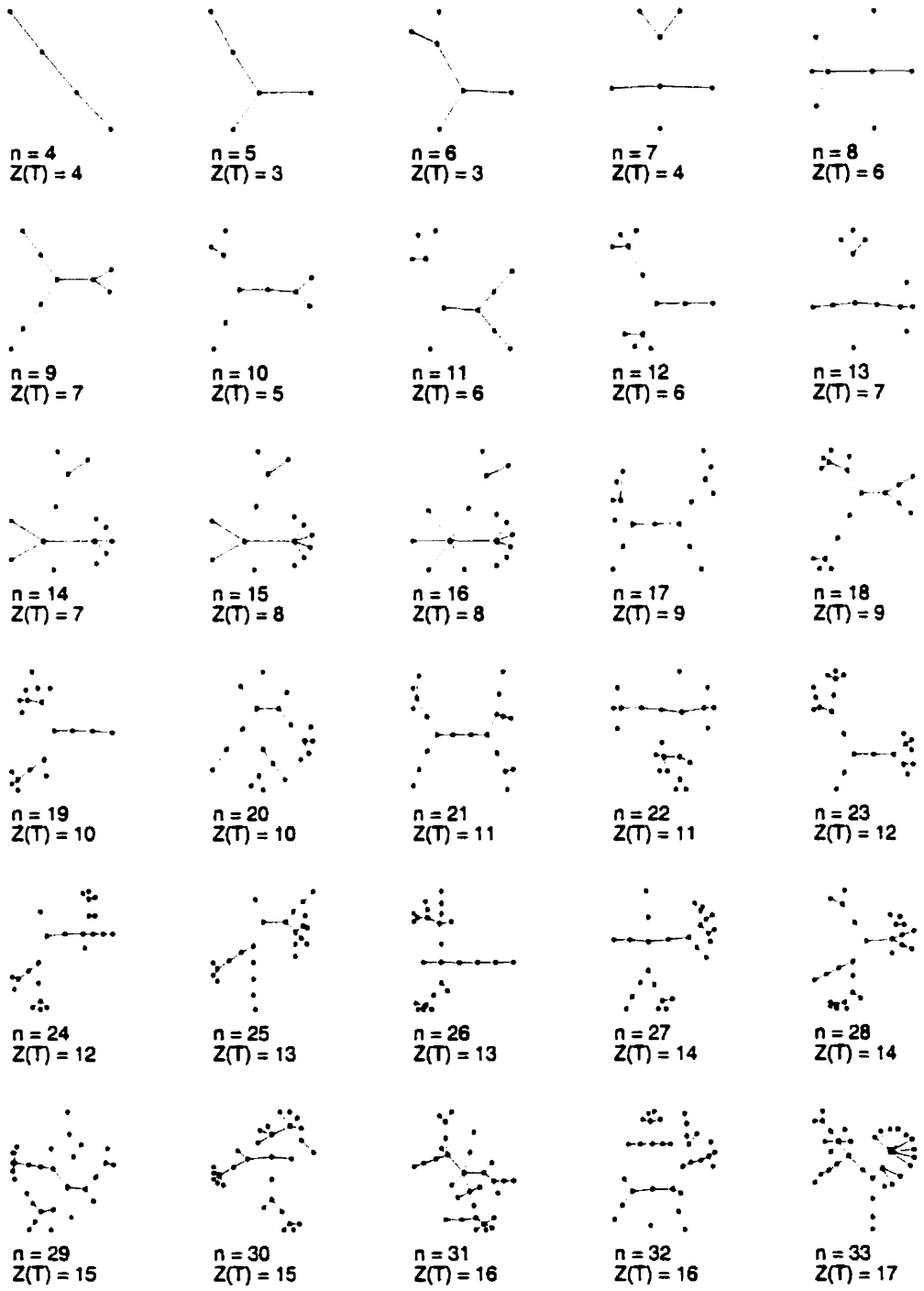


Figure 2.39 – Arbres avec n sommets, $4 \leq n \leq 33$, diamètre impair et conjecturés de distance à la H-palindromicité minimum.

Chapitre 3

Résultats et développements envisagés

3.1 Résultats

Par le biais du programme **AGX** qui était au départ un prototype sans autre ambition que d'aider à résoudre quelques problèmes très spécifiques, nous avons pu remarquer à quel point l'ordinateur peut être un collaborateur important.

Il a réussi à nous convaincre de sa pertinence en retrouvant très rapidement des résultats connus sans qu'aucune information spécifique sur le problème ne soit utilisée.

Il nous a ensuite permis de résoudre des problèmes que des mathématiciens étudiaient depuis environ 25 ans, dans le cadre de l'étude de l'énergie (voir la section 2.2 pour plus de détails).

Il a ensuite permis de trouver une conjecture d'expression élégante mais tellement inattendue que nous n'avons aucune idée de sa signification et nous sommes encore plus dépourvus lorsque nous envisageons de la démontrer. Il s'agit de la **Conjecture AGX** 19 sur la distance à la palindromicité décrite à la fin de la section 2.4.

Pour continuer la revue des résultats remarquables d'**AGX**, nous ne pouvons pas garder sous silence les conjectures obtenues de manière totalement automatisée parmi

lesquelles celle décrite dans la section 1.4.3 que nous n'aurions simplement pas cherchée car elle met en relation 5 invariants. Comme le programme travaille à partir de graphes (supposés) extrêmes, les bornes sur des invariants qu'il permet de découvrir sont en général les meilleures possibles.

3.1.1 Conjectures de Graffiti réfutées

Une fonction importante d'**AGX** est la réfutation de conjectures. Il était donc naturel de tester le programme sur un certain nombre de conjectures de Graffiti, ce qui nous a permis d'en réfuter un certain nombre. Les conjectures suivantes de Graffiti ont été réfutées par ou à l'aide d'**AGX**, elles sont numérotées conformément à la version de mai 1998 de *Written on the Wall* [56].

- **Conjecture WoW :15**

  "*Radius is not more than the variance of the degree sequence + Randic index*" ou :

  $$r \leq \sigma^2(\delta) + Ra.$$

  AGX a trouvé que le chemin à 22 sommets est un contre-exemple. Une analyse des chemins nous mène à la conclusion que les chemins pairs de 22 sommets ou plus sont tous des contre-exemples à cette conjecture.

- **Conjecture WoW :16**

  "*Radius is not more than the average temperature + Randic index*" qui peut aussi s'écrire :

  $$\frac{1}{n}\sum_{j=1}^{n}\frac{\delta_j}{n-\delta_j} + Ra - r \geq 0$$

  a été réfutée automatiquement (voir section 1.4.2).

- **Conjecture WoW :750**

"*Let e be an edge and let v be a vertex. e is called a v-horizontal edge if the distance from v to both endpoints is the same. Let h(v) denotes the number of v-horizontal edges, and let d(v) be the number of vertices at odd distance from v. Graffiti made the conjecture that for every connected graph G the independance number of G is not less than max d(v) - min h(v).*" Où : soit $h(v)$ le nombre d'arêtes (ij) du graphe G telles que $d_{iv} = d_{jv}$, et $d(v)$ le nombre de sommets à distance impaire de v. Graffiti a fait la conjecture que

$$\alpha \leq \max d(v) - \min h(v).$$

Le graphe représenté sur la figure 3.1, trouvé par **AGX**, est un contre-exemple à 9 sommets.

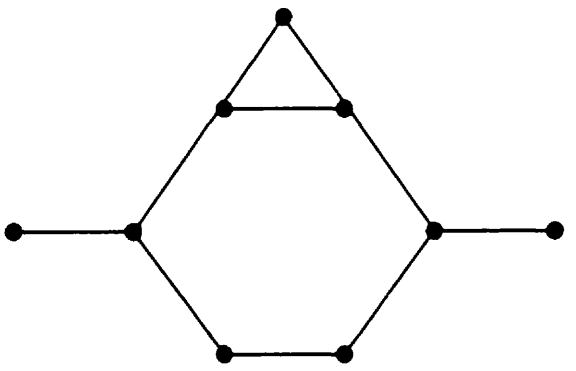


Figure 3.1 – Contre-exemple à la conjecture 750 de Graffiti

- **Conjecture WoW :834**

"*the average distance is $\leq$ 1 + maximum temperature of complement of G*" ou :

$$(1 + \delta_{min})\bar{d} \leq n.$$

AGX a trouvé en quelques minutes le graphe représenté sur la figure 3.2 qui est un contre-exemple à la conjecture 834 de Graffiti.

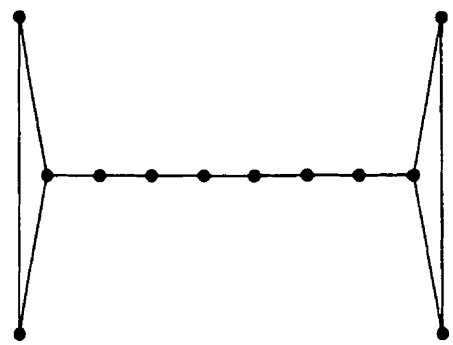


Figure 3.2 – Contre-exemple à la conjecture 834 de Graffiti

3.1.2 Conjectures obtenues

Le programme nous a permis de trouver les conjectures suivantes :

- **Conjecture AGX** 1

  Pour tout graphe connexe G :

$$Ra(G) \geq \frac{\gamma(G)-2}{2} + \frac{1}{\sqrt{n-1}}(\sqrt{\gamma(G)-1} + n - \gamma(G))$$

- **Conjecture AGX** 2

  Pour tout graphe G :

$$\sigma^2(\delta) + \sum_{i \in V} \frac{1}{\delta_i} - \bar{d} \geq \frac{n}{n-1} - 1$$

  qui s'écrit aussi

$$\sigma^2(\delta) + \sum_{i \in V} \frac{1}{\delta_i} - \bar{d} \geq \frac{1}{n-1}$$

  cette conjecture tend asymptotiquement vers la conjecture **WoW :4** quand n tend vers l'infini.

- **Conjecture AGX** 3

  Pour tout graphe G :

$$mode(d) + \sum_{i \in V} \frac{1}{\delta_i} - \bar{d} \geq \frac{n}{n-1}.$$

- **Conjecture AGX** 4

  Pour tout graphe G d'ordre $n > 3$

$$r + Ra - mode(d) \geq \sqrt{n-1} - 1$$

- **Conjecture AGX** 5

  Pour tout graphe G d'ordre $n > 3$:

$$\bar{d} + Ra - mode(d) \geq \frac{2(n-1)}{n} + \sqrt{n-1} - 2$$

  ou

$$\bar{d} + Ra - mode(d) \geq \sqrt{n-1} - \frac{2}{n}.$$

- **Conjecture AGX** 6

  Pour tout graphe G d'ordre $n > 2$:

$$Ra - r \geq \frac{n-3}{2} + \sqrt{2} - \lfloor \frac{n}{2} \rfloor$$

  ou

$$Ra - r \geq \frac{nmod2}{2} + \sqrt{2} - \frac{1}{2}.$$

- **Conjecture AGX** 7

  Pour tout graphe G d'ordre n :

$$\sigma^2(\delta) + Ra - r \geq \frac{n}{2} - \lfloor n-2 \rfloor.$$

  Cette conjecture a été réfutée (voir [27]).

- **Conjecture AGX** 8

  Pour tout graphe G d'ordre n :

$$Ra - \sqrt{\sigma^2(\delta)} \geq \sqrt{n-1} - \sqrt{n-3+(\frac{2}{n})}.$$

- **Conjecture AGX** 9

  Pour tout arbre avec contrainte de couleurs et d'index minimum :

$$\alpha = \frac{1}{2}(m + n_1 + D - 2r).$$

- **Conjecture AGX** 10

  Si G est un graphe chimique de nombre cyclomatique $\nu \geq 2$ et d'indice de Randić maximum, alors G n'a pas de sommets pendants ni de sommets de degré 4.

- **Conjecture AGX** 11

  Si G est un graphe chimique de nombre cyclomatique $\nu \geq 3$ et d'indice de Randić maximum, alors le sous-graphe induit par les sommets de degré 2 est un chemin.

- **Conjecture AGX** 12

  Si G est un graphe chimique de nombre cyclomatique $\nu \geq 3$ et d'indice de Randić maximum, alors G a $3\nu - 4$ arêtes reliant des sommets de degré 3.

- **Conjecture AGX** 13

  Si G est un graphe avec n sommets, $m \leq \lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$ arêtes et d'énergie minimum, alors

  (i) s'il existe des entiers positifs a et b tels que $a \times b = m$ et $a + b \leq n$, G est un graphe biparti complet auquel sont éventuellement ajoutés des sommets isolés,

  (ii) sinon, G est un graphe biparti complet $K_{a',b'}$ avec $a' \times b' \leq m$ et $a' + b' \leq n$ modifié par l'ajout de $m - a' \times b'$ arêtes reliant un sommet du plus petit côté de $K_{a',b'}$ à d'autres sommets de ce côté, auquel sont éventuellement ajoutés des sommets isolés.

- **Conjecture AGX** 14

  Les graphes connexes G avec $n \geq 6$ sommets, $n - 1 \leq m \leq 2(n-2)$ arêtes et

une énergie minimum sont les étoiles avec $m - n + 1$ arêtes supplémentaires toutes connectées au même sommet pour $m \leq n + \lceil \frac{n-7}{2} \rceil$, ou bien des graphes bipartis avec 2 sommets d'un côté, dont un est connecté à tous les sommets de l'autre coté.

- **Conjecture AGX** 15

  Pour tout graphe G avec n sommets, l'énergie E vérifie

$$E \leq 2n - 2 + 0.4[\max(m - 7, 0)]^{\frac{3}{2}}.$$

- **Conjecture AGX** 16

  Parmi les graphes unicycliques à n sommets, le circuit C_n a l'énergie maximum si $n \leq 7$ et $n = 9, 10, 11, 13, 15$. Pour toute autre valeur de n le graphe unicyclique d'énergie maximum est $C_6(n-6)$. Ce résultat est intéressant du point de vue des chimistes, étant donnée la fréquence des cycles hexagonaux dans les hydrocarbones.

- **Conjecture AGX** 17

  Soit $\mathfrak{T}_{a,b}$ l'ensemble des arbres dont une coloration des sommets comporte a sommets noirs et b sommets blancs avec $a \geq b$. Un sommet noir d'un arbre $T \in \mathfrak{T}_{a,b}$ d'index minimal a un degré 1 ou 2.

- **Conjecture AGX** 18

  Pour $a = b + 4$, les arbres $T^* \in \mathfrak{T}_{a,b}$ d'index minimum λ_1 sont des chenilles avec 5 sommets pendants. Pour $a = b+5$ les arbres d'index minimum λ_1 sont des chenilles avec 6 sommets pendants.

- **Conjecture AGX** 19

  Pour tout arbre T avec $n \geq 4$ sommets et un diamètre impair, la distance à la H-palindromicité est au moins $\lceil \frac{n}{2} \rceil$. De plus, cette borne est serrée pour tout $n \geq 10$.

- **Conjecture AGX** 20

  Il n'y a pas d'arbres H-palindromiques avec un diamètre impair.

Certaines des conjectures obtenues à l'aide d'**AGX** ont été démontrées. En plus des conjectures précédemment citées, les théorèmes suivants ont été trouvés grâce à **AGX** :

- **Théorème AGX 1**

  Pour tout arbre

$$\alpha \leq \frac{1}{2}(m + n_1 + D - 2r).$$

- **Théorème AGX 2**

  Pour tout arbre

$$\alpha \geq \frac{1}{2}\left(m + n_1 + D - 2r - \left\lfloor \frac{n-2}{2} \right\rfloor\right).$$

- **Théorème AGX 3**

  Si T_n et P_n sont respectivement un arbre et un chemin à n sommets, $n \geq 2$, alors

$$Ra(T_n) \leq Ra(P_n),$$

  avec égalité si et seulement si T_n est un chemin.

- **Théorème AGX 4**

  Soit T_n un arbre chimique quelconque à n sommets.

  (a) Si $n \equiv 2 \pmod 3$ et $n \geq 5$, alors

$$Ra(T_n) \geq \frac{5n}{12} - \frac{1}{12}$$

  avec égalité si et seulement si tous les sommets sont de degré 1 ou 4 (ou encore si et seulement si T_n ne comporte aucun sommet de degré 2 ou 3).

  (b) Si $n \equiv 1 \pmod 3$ et $n \geq 13$, alors

$$Ra(T_n) \geq \frac{5n}{12} + \frac{6\sqrt{3} - 11}{12}$$

  avec égalité si et seulement si tous les sommets de T_n sont de degré 1 ou 4, à l'exception d'un seul qui est de degré 3 et adjacent à 3 sommets de degré 4.

  (c) Si $n \equiv 0 \pmod 3$ et $n \geq 9$, alors

$$Ra(T_n) \geq \frac{5n}{12} + \frac{2\sqrt{2} - 3}{4}$$

  avec égalité si et seulement si tous les sommets sont de degré 1 ou 4 à l'exception d'un seul qui est de degré 2 et adjacent à 2 sommets de degré 4.

- **Théorème AGX** 5

  Pour tout graphe chimique G :

$$Ra(G) \geq \frac{1}{4}(n_1 + m)$$

- **Théorème AGX** 6

  Pour tout graphe chimique

$$Ra(G) \geq \frac{n}{3} + \frac{m}{12}.$$

- **Théorème AGX** 7

  Si G est un graphe chimique de nombre cyclomatique $\nu \geq 2$, comportant $n \geq 5\nu - 4$ sommets et d'indice de connectivité Ra maximum, alors :

  - G n'a pas de sommets pendants,
  - G a au moins une arête reliant deux sommets de degré 2,
  - G n'a pas de sommet de degré 4.

- **Théorème AGX** 8

  Pour tout graphe G l'énergie vérifie

$$E \geq 2\sqrt{m}.$$

  De plus, la borne est atteinte si et seulement si G est un graphe biparti complet, auquel sont éventuellement ajoutés des sommets isolés.

- **Théorème AGX** 9

  Pour tout graphe G, l'énergie E satisfait

$$E \geq \frac{4m}{n}.$$

- **Théorème AGX** 10

  Pour tout graphe connexe G

$$E \geq 2\sqrt{n-1}$$

  et la borne est serrée si et seulement si G est une étoile.

- **Théorème AGX** 11

  Pour tout graphe G avec m arêtes, l'énergie vérifie

$$E \leq 2m$$

  et la borne est atteinte si et seulement si G est composé d'arêtes disjointes auxquelles sont éventuellement ajoutés des sommets isolés. Après la démonstration de ce théorème, nous avons réalisé que Graffiti a donné une conjecture équivalente, prouvée par Shearer.

- **Théorème AGX** 12

  Pour un nombre fixé de sommets n et $T \in \mathfrak{T}_{a,b}$ la valeur minimale de $\lambda_1(T)$ croît de manière monotone avec $a - b$.

- **Théorème AGX** 13

  Pour $a = b+2$ et $n \geq 6$, les arbres $T^* \in \mathfrak{T}_{a,b}$ d'index λ_1 minimum sont des comètes $C_{4,n-4}$. De plus

$$\lim_{n\to\infty} \lambda_1(T^*) = 2.$$

  Pour $a = b+3$ et $n \geq 7$, les arbres $T^* \in \mathfrak{T}_{a,b}$ d'index λ_1 minimum sont des doubles comètes $D_{3,n-6,3}$ et $\lambda_1(T^*) = 2$.

- **Théorème AGX** 14

  Pour $b = 2$, les arbres $T^* \in \mathfrak{T}_{a,b}$ d'index λ_1 minimum sont soit des doubles comètes équilibrées, soit des doubles comètes quasi-équilibrées

$$D_{\lfloor \frac{n-1}{2} \rfloor, 1, \lceil \frac{n-1}{2} \rceil}.$$

- **Théorème AGX** 15

  Pour $b = 3$ et $a \geq 6$, les arbres $T^* \in \mathfrak{T}_{a,b}$ d'index minimum λ_1 sont des doubles comètes

$$D_{\lceil \frac{n-1}{3} \rceil, \lceil \frac{n-1}{3} \rceil, 3},$$

  avec $n - 2\left\lceil \frac{n-1}{3} \right\rceil - 3$ arêtes pendantes ajoutées au sommet central de son chemin.

3.2 Développements envisagés

L'utilisation d'**AGX** n'est pas restreinte aux cas mentionnés plus haut, et le programme nous donna assez régulièrement des informations ou des directions de recherche que nous serions tentés d'appeler intuition si elles n'émanaient pas d'un ordinateur.

Pour toutes ces raisons, nous pensons actuellement qu'un tel programme a réellement sa place dans un laboratoire de théorie des graphes. Le prototype ayant pleinement montré le potentiel d'un tel type de programme, nous envisageons actuellement de l'améliorer.

Nous souhaiterions dans ce cadre reconstruire le programme afin de faciliter l'intégration de fonctionalités nouvelles et l'extension des possibilités actuelles.

Les quelques exemples suivants illustrent les améliorations que nous souhaiterions apporter au programme.

3.2.1 Recherche analytique de conjectures

Dans le cas où la famille de graphes extrêmes trouvée par **AGX** n'est pas une *famille simple de graphes*, il est possible de renforcer la méthode d'analyse paramétrique des graphes obtenus en utilisant un second paramètre qui, combiné au premier, permet d'identifier une *famille simple de graphes* pour laquelle une formule analytique comportant un paramètre ajouté par le système est identifiable. Le résultat que proposerait alors le système ne serait pas une simple fonction du paramètre initial, mais une fonction de deux paramètres devant être optimisée algébriquement par rapport au paramètre ajouté.

Une illustration de cette approche est l'étude de la variance des degrés. En cherchant les graphes de la plus grande variance des degrés, **AGX** a trouvé systématiquement des *Graphes fendus complets* (*Complete Split Graphs*). De tels graphes sont composés d'une clique et d'un ensemble stable dont chaque sommet est joint à tous ceux de la clique. Malheureusement, ceci ne suffit pas à définir complètement un graphe et la valeur de la variance des *Graphes fendus complets* n'est pas constante quand n est donné. En revanche, si le degré minimum du graphe est donné, ce dernier est complètement défini, et la variance des degrés peut être calculée analytiquement. Nous avons utilisé ces propriétés pour calculer la valeur maximale de la variance des degrés dans un graphe en fonction de n et δ_{min}. En effet, pour un *Graphe fendu complet* de degré minimum δ_{min}, il y a δ_{min} sommets de degré $n-1$ et $n-\delta_{min}$ sommets de degré δ_{min}. On en déduit la valeur du degré moyen :

$$\bar{\delta} = \frac{\delta_{min}}{n}(2n - 1 - \delta_{min}).$$

La variance des degrés est alors

$$\begin{aligned} var(\delta) &= \frac{1}{n}\sum_{i=1}^{n}(\delta_i - \bar{\delta})^2 && (3.1) \\ &= \frac{1}{n}(\delta_{min}(n-1-\bar{\delta})^2 + (n-\delta_{min})(\bar{\delta}-\delta_{min})^2). && (3.2) \end{aligned}$$

La valeur maximale pour la variance des degrés dans un *Graphe fendu complet* en fonction du degré minimum est alors :

$$\max_{\delta_{min}} \frac{1}{n}(\delta_{min}(n-1-\bar{\delta})^2 + (n-\delta_{min})(\bar{\delta}-\delta_{min})^2).$$

Un programme simple calculant cette valeur pour $3 \leq n \leq 200$ en considérant toutes les valeurs possibles de δ_{min} nous a ensuite montré que la valeur de δ_{min} pour laquelle

la variance des degrés était maximale était $\lfloor \frac{n+1}{4} \rfloor$. Si ce résultat n'est pas nouveau puisqu'il fut décrit par Bell en 1992 [13], il reste néanmoins une bonne illustration des procédures à ajouter à **AGX**.

Dans certains cas, les formules analytiques pour les invariants ne sont pas disponibles, soit parce que la classe de graphes extrêmes n'a pas été considérée, soit parce que le paramètre utilisé n'est pas courant. Nous pouvons alors essayer d'utiliser la méthode numérique de recherche de conjectures décrite dans la section 1.4.3 pour donner une expression de l'invariant recherché en fonction de termes calculés avec le paramètre utilisé pour la recherche des graphes extrêmes. Par exemple, si nous utilisons le nombre de sommets n comme paramètre, il est possible de rechercher les relations entre l'invariant et n,$(n-1)$, $(n+1)$, $\sqrt{n}$, $\sqrt{n-1}$, n^2, *etc...*

Il est possible aussi de partir d'une caractérisation des graphes extrêmes pour générer une extension de la famille sans pour autant recourir à l'optimisation, ce qui permet de gagner du temps car l'optimisation pour des graphes de plus grande taille peut se montrer longue. Il est alors possible de construire un ensemble de graphes conjecturés extrêmes beaucoup plus conséquent (au risque d'une moindre qualité des graphes) dans le but de chercher des conjectures complexes (impliquant un grand nombre d'invariants), avec la méthode numérique. Il convient alors de voir si les relations simples que le programme donne pour les graphes extrêmes suffisent à définir avec exactitude chaque graphe ou si elles sont simplement un ensemble de propriétés qu'ils vérifient. La différence est que dans le cas où la caractérisation permet de définir avec exactitude un graphe en fonction des paramètres choisis, celui-ci est unique et peut ensuite être généré rapidement dans le but de trouver des contre-exemples à des conjectures, ou de renforcer ces dernières. Ce type d'approche a été décrit lors de la section 1.4.2, avec la différence que nous ne connaissons pas les formules analytiques permettant de calculer certains invariants, et que les graphes doivent ici être construits.

3.2.2 Définition automatique des voisinages à utiliser

Nous avons remarqué que le choix des transformations à utiliser, dans la recherche locale est déterminant. Ce choix est guidé par deux enjeux :
d'une part la qualité des solutions trouvées, d'autre part le temps requis pour les trouver. Il est certain que l'exploration exhaustive de tous les graphes par modifications locales impliquant k sommets donnera les meilleurs résultats parmi toutes les transformations de dimension k (impliquant k sommets). Cependant, certains des cas considérés lors d'une exploration exhaustive sont inutiles. Pour cette raison, nous avons en général plutôt cherché à caractériser des transformations pertinentes, à savoir celles qui permettent de passer du graphe courant au meilleur que nous connaissions. Au lieu de cette approche en partie basée sur notre intuition, nous pouvons utiliser la puissance de calcul de l'ordinateur pour identifier, à partir de l'optimum local rencontré, le type de transformation de dimension k permettant une amélioration de la fonction-objectif. Une approche pour identifier ces transformations consiste à faire d'abord une recherche exhaustive à partir de l'optimum local trouvé, dans le but d'identifier celles qui engendrent une amélioration de la fonction-objectif. Dans un second temps, ces transformations sont caractérisées, avant d'être mémorisées, en vue d'une utilisation systématique.

Par exemple, si nous nous intéressons aux transformations de dimension 3 (avec les sommets i, j et k), trois arêtes sont impliquées dans le sous-graphe considéré, (i,j), (i,k) et (j,k). Appelons x_{ab} la variable binaire associée à l'aête (a,b) qui vaut 1 si l'arête est présente et 0 sinon. Sans perte de généralité, nous pouvons ne considérer que les cas où $x_{ij} \geq x_{ik} \geq x_{jk}$. Il reste alors à donner les valeurs de x_{ij}, x_{ik} et x_{jk} ainsi que les valeurs x'_{ij}, x'_{ik} et x'_{jk} après transformation pour lesquelles une amélioration a été observée, ainsi que les statistiques sur ces améliorations. La dernière étape consiste à sélectionner les transformations les plus prometteuses afin d'éviter de

perdre du temps à évaluer des transformations inutiles. Selon les statistiques sur les améliorations, nous pouvons également, au début, nous contenter des transformations les plus pertinentes, pour en ajouter d'autres quand celles-ci échouent.

À l'aide d'une telle procédure, nous pouvons espérer utiliser de nouveaux voisinages plus appropriés au problème lors de la *Descente à Voisinages Variables*.

3.2.3 Définition interactive d'invariants

Dans sa version actuelle tous les invariants utilisés sont programmés dans **AGX**. Cette manière de procéder est sans doute la meilleure pour un prototype car l'usager, qui est aussi le développeur, dispose en permanence du programme source avec laquelle il est à l'aise pour ajouter de nouveaux invariants si nécessaire. Par ailleurs, il a la possibilité de programmer chaque invariant de la manière la plus efficace. Comme nous pensons qu'**AGX** est un outil qui peut se montrer utile à un bon nombre de chercheurs en théorie des graphes, nous pensons aussi à l'aspect pratique pour l'usager qui ne dispose pas de la source du programme. Ou bien il faut que tous les invariants qu'il désire utiliser soient disponibles, ou alors il doit pouvoir les ajouter aisément sans aucune connaissance de la structure interne du programme. Dans cette optique, deux approches sont à considérer : la programmation d'un plus vaste ensemble d'invariants d'une part, et la possibilité d'ajouter de nouveaux invariants d'autre part. Il est bien entendu dans nos projets d'ajouter un grand nombre d'invariants au programme. Nous pensons aussi à ajouter un module permettant à l'usager de donner ses propres invariants soit par l'intermédiaire d'un interpréteur, soit par l'ajout de modules qu'il compilerait séparément. L'interpréteur serait certainement d'une utilisation plus simple mais ralentirait l'exécution du programme en plus d'être démesurément ardu par rapport aux bénéfices envisagés. Pour ces raisons, nous envisageons plutôt de rendre possible l'intégration de modules compilés séparément par l'usager.

3.2.4 Énumération de familles de graphes

Le principe de fonctionnement d'**AGX** étant de trouver puis d'analyser des graphes extrêmes, il serait sans doute pertinent d'ajouter au programme un module d'énumération de familles de graphes qui permettrait de trouver les graphes extrêmes avec certitude, contrairement à ce que permet l'utilisation d'une heuristique. Bien sûr, l'énumération de graphes généraux est une tâche particulièrement longue, mais c'est chose relativement raisonnable si nous nous intéressons à des graphes avec contraintes (par exemple les arbres). Actuellement, **AGX** est capable de lire par l'entrée standard (*Standard input*) des graphes fournis par le générateur *geng* de McKay, et de sélectionner le meilleur des graphes proposés comme graphe optimal au lieu de le chercher heuristiquement. Cette possibilité a l'avantage de garantir l'optimalité mais au prix d'un temps de calcul ne permettant pas de traiter dans le cas général des graphes à plus d'une dizaine de sommets. Pour cette raison, il serait sans doute intéressant de nous pencher sur la recherche de propriétés des graphes extrêmes par l'étude des graphes obtenus heuristiquement, puis d'utiliser les conditions que nous avons pu démontrer pour réduire l'espace de recherche afin d'envisager l'utilisation d'énumération appliquée à la famille de graphe ainsi définie.

Chapitre 4

Énumeration de Benzenoides et Helicènes

4.1 Introduction

Comme expliqué dans le chapitre d'introduction et illustré dans les premiers chapitres, les méthodes heuristiques et les méthodes d'énumération sont des outils complémentaires pour la découverte de conjectures en théorie des graphes. Les méthodes d'énumération peuvent :
(i) réfuter des conjectures par des exemples de taille modérée,
(ii) déterminer l'ensemble des contre-exemples de taille modérée à une conjecture, ce qui peut mener à sa reformulation (comme le montre le cas des arbres H-palindromiques discuté durant la section 2.4 du présent document),
(iii) corroborer une conjecture en montrant qu'elle est vérifiée pour tous les graphes de taille modérée,
(iv) déterminer des graphes extrêmes qui n'auraient pas été découverts par une méthode heuristique, la correction des résultats ainsi faite facilitant la découverte automatisée ou assistée par l'ordinateur de nouvelles conjectures.

Dans cette partie, nous proposons deux méthodes d'énumération et de dénombrement nouvelles et les appliquons à deux problèmes classiques de la théorie des graphes chimiques : l'énumération et le dénombrement de **polyhexagones** (ou **polyhexes**) représentant des benzenoïdes et des hélicènes (les définitions sont rappelées ci-dessous).

La première méthode utilise la **recherche inversée** de Avis et Fukuda [7], qui dans le cas présent est équivalente à la recherche ordonnée de Read [108] et Faradzev [58] [57], ainsi que le **code des arêtes de frontière** ou " *boundary edges code*", appelé *BE Code*. La seconde méthode (développée avec Gunnar Brinkmann [Université de Bielefeld]) est également basée sur la recherche ordonnée. Cependant, elle fonctionne à deux niveaux : l'énumération des graphes duaux des polyhexes, puis de tous les polyhexes ayant le même graphe dual. Les deux méthodes sont appliquées aux cas des polyhexes planaires (au sens géométrique, les hexagones étant supposés réguliers) et non planaires, ou **fusènes**, cette dernière classe comprenant les polyhexes planaires et non planaires, ou hélicènes. Dans tous les cas, des ensembles de polyhexes beaucoup plus grands que ceux traités par les méthodes précédentes sont énumérés ou dénombrés.

4.2 Définitions

Un **polyhexe** est un système **planaire** et **simplement connecté** d'hexagones réguliers tel que deux hexagones quelconques partagent exactement une arête ou sont disjoints [87] [111] (voir figure 4.1(a)). Les sommets intérieurs sont de degré 3 et ceux de la frontière de degré 2 ou 3. Les polyhexes peuvent être **planaires** ou non au sens géométrique (différent de la planarité au sens de la théorie des graphes) c'est-à-dire qu'ils peuvent ou non être plongés dans un treillis hexagonal infini du plan, sans que deux arêtes ne se superposent. Si ce n'est pas le cas, c'est-à-dire en cas de superposition d'au moins une paire d'arêtes distinctes, ils sont appelés **hélicènes** (voir figure 4.1(b)) Remarquons que même si un polyhexe est dit planaire, la molécule qu'il représente peut avoir une géométrie vrillée, et ne pas être planaire dans l'espace, le lecteur peut se référer à Herndon *et al* [80] pour une discussion à ce sujet. Un **coronoïde** est un polyhexe, planaire ou non, qui comporte au moins un trou

d'au moins deux hexagones, comme illustré sur la figure 4.1(c), par opposition à un **polyhexe simplement connecté** qui n'a pas de trous, comme l'illustre la figure 4.1(a).

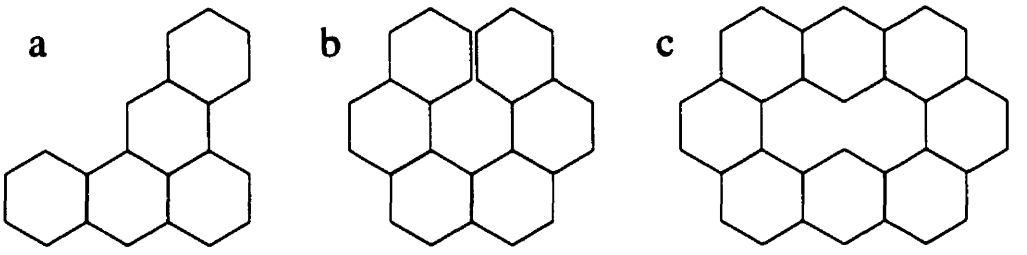


Figure 4.1 – Exemples de polyhexes planaire simplement connecté (a), hélicène (b) et coronoïde (c)

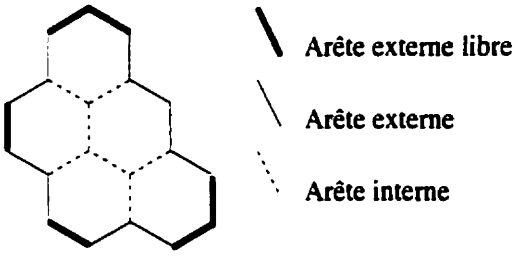


Figure 4.2 – Arêtes d'un polyhexe

Si on s'intéresse aux incidences entre arêtes et sommets, les polyhexes peuvent être considérés comme des graphes. Dans un polyhexe, les degrés des sommets sont soit deux, soit trois. Une **arête externe** d'un polyhexe simplement connecté est une arête qui n'appartient qu'à un seul hexagone, comme l'illustre la figure 4.2. Une **arête libre** est une arête externe reliant deux sommets de degré deux. La **frontière** d'un polyhexe simplement connecté est le cycle décrit par ses arêtes externes. Un **sommet externe** d'un polyhexe est un sommet faisant partie de son contour. On appelle **graphe dual d'un polyhexe** le graphe obtenu en associant un sommet à chaque hexagone du polyhexe, deux sommets étant adjacents si et seulement si les hexagones qu'ils représentent ont une arête en commun. Une **arborescence** est un arbre orienté de telle manière que chaque sommet soit à l'origine d'exactement un arc, excepté un sommet qui est appelé **racine** .

4.3 Historique de l'énumération de polyhexes planaires simplement connectés

Le **dénombrement** et l'**énumération** de molécules de diverses familles ont fait l'objet de nombreux travaux, commencés déjà au XIXième siècle, comme le montrent ceux de Flavitsky [63] par exemple.

Pour des revues comprenant un historique de ces tentatives, le lecteur peut se référer à Balaban [8], Bababan *et al.*[9], Balasubramanian [11], Bonchev et Rouvray [17] [18], Brunvoll *et al.* [42], Dias [43], Gutman et Cyvin [72], Hosoya [82]), Knop *et al.* [86], Mercier *et al.* [99] et Trinajstic *et al.* [116] [115].

Les deux activités sont distinctes : le dénombrement vise à connaître le nombre de molécules alors que l'énumération implique d'en donner aussi une courte description sous la forme d'un *code*.

L'information fournie par l'énumération est plus complète et plus utile que celle obtenue par simple dénombrement. Par exemple, lorsque nous étudions des propriétés associées à un invariant, les codes des éléments de l'ensemble considéré peuvent être utilisés pour trouver les valeurs minimum, maximum ou moyenne de cet invariant. De plus, si cet ensemble est particulièrement grand, il est possible de sélectioner un échantillon aléatoire suffisamment important lors de la génération des structures qui fournira une bonne approximation de la distribution des valeurs de cet invariant.

L'énumération de polyhexes est un problème très étudié depuis les années soixante. La croissance exponentielle du nombre de polyhexes en fonction du nombre d'hexagones (noté h) fait de ce problème un excellent problème test pour l'évaluation des performances de méthodes d'énumérations. Le premier programme informatique pour la résolution de ce problème fût proposé par Balasubramanian *et al* en 1980 [12]. Il

est basé sur le *boundary code* qui semble être le premier code informatique pour les polyhexes.

L'idée de ce code est de décrire le contour du polyhexe, utilisant la remarque qu'il est toujours possible de passer d'un sommet à chacun de ses voisins par des déplacements unitaires utilisant seulement six directions. En associant à chaque direction un chiffre, il est possible de décrire le contour du polyhexe par une série de chiffres représentant les directions utilisées successivement. Ce code étant circulaire, nous examinons ensuite les différentes représentations du polyhexe en fonction du sommet de départ et du sens utilisé pour parcourir le contour. Comme l'orientation est importante puisque les directions utilisées ont une influence sur le code lui même, nous devons considérer toutes les 12 manières de représenter le polyhexe, par rotations et symétrie axiale avant de choisir le code le représentant comme celui qui est lexicographiquement le plus grand, tel que le montre la figure 4.3. Il est aisé de se rendre compte que le *boundary code* est redondant et long.

L'énumération de polyhexes planaires simplement connectés jusqu'à $h = 10$ [88], $h = 11$ [113] et $h = 12$ [77] utilisaient ce code.

Les progrès suivants sont venus du code DAST (de l'anglais *Dualist Angle-restricted Spanning Tree* ou Arbre de Recouvrement Dual avec Restriction d'angle) [104] qui est basé sur le graphe dual associé à chaque polyhexe [10]. La première étape du calcul du code DAST consiste à identifier l'arbre de recouvrement orienté permettant d'identifier la position de chaque hexagone du polyhexe. La manière de parcourir le polyhexe à partir d'un hexagone choisi comme racine est telle que pour tout hexagone H, un angle de 120 degrés ou plus doit séparer le père P d'un fils quelconque F. Cette règle n'empêche nullement de définir chaque polyhexe car si F a un angle de moins de 120 degrés avec P, il lui est adjacent et a été considéré comme fils de P, ainsi, sur la figure 4.4, $H2$ est considéré comme fils de P et ne doit pas être considéré comme

fils de $H1$. De cette manière, chaque hexagone a un et un seul père tandis qu'il peut avoir jusqu'à trois fils à des positions bien définies par rapport au père.

Afin que le code soit compact, chaque hexagone sera caractérisé par un chiffre indiquant les positions de ses fils. Si l'hexagone a un fils non encore considéré aligné avec son père, une valeur 1 est considérée. S'il a un hexagone non encore indiqué dans le code sur la droite par rapport à la direction que donne la relation père-fils, on compte 2, s'il y en a un à gauche, une valeur 4 est considérée. En faisant la somme des valeurs ainsi définies, on trouve le chiffre représentant l'hexagone, compris entre 0 et 7, indiquant les huit combinaisons possibles associées aux trois positions ou un fils peut se trouver. Dans le code, à la suite du chiffre associé à un hexagone, nous plaçons les codes des sous-arbres connectés par l'hexagone à gauche, aligné et à droite quand ils sont présents (notons que cette présence est indiquée par le chiffre associé à H) dans cet ordre. Un exemple de construction de code, en fonction de l'arête utilisée pour "entrer" dans le polyhexe (indiquée par la flèche à l'extérieur du polyhexe) est donné sur la figure 4.5. Remarquons que chaque hexagone est considéré exactement une fois, le code DAST sera alors composé d'exactement h chiffres pour un polyhexe à h hexagones.

De la manière dont il est décrit, il est possible de construire sans ambiguïté un polyhexe à partir d'un code, mais à chaque polyhexe sont associés plusieurs codes. Dans le but de pallier ce problème, il faut définir de manière unique l'arête par laquelle nous "entrons" dans le polyhexe. Tout d'abord, il faut que l'hexagone par lequel nous "entrons" dans le polyhexe soit tel que tous les hexagones du polyhexe soient considérés. Afin de limiter le nombre de choix pour l'arête par laquelle nous "entrons" dans le polyhexe, si elle est verticale, nous la rejeterons si elle n'est pas le plus à droite possible, et dans le cas ou il y en aurait plusieurs sur la même verticale, celles qui ne sont pas le plus haut seront rejetées. À ce stade, nous définissons de manière unique l'arête par laquelle nous entrons pour chaque représentation possible du polyhexe.

Comme dans le cas du *Boundary Code*, il y a alors 12 possibilités correspondant aux 12 manières de représenter le polyhexe selon la symétrie axiale et les rotations possibles. Parmi chacun des codes ainsi définis, celui qui est lexicographiquement le plus petit sera sélectionné.

Ce code est beaucoup plus puissant que le *boundary code* et a permis l'énumération de tous les polyhexes pour $h = 13$ [101], $h = 14$ [117], $h = 15$ [104] et $h = 16$ [87]. Malgré tout, ce code comportait certaines faiblesses puisque chaque polyhexe peut avoir jusqu'à 12 représentations différentes selon l'hexagone utilisé comme racine de l'arbre de recouvrement du polyhexe. Afin d'éviter toute génération multiple, une règle spéciale fût utilisée [116]. Les dernièrs résultats dans le domaine est due à Tošić *et al* [111] qui proposent une approche originale utilisant une "cage" dans laquelle chaque polyhexe est placé. Cette méthode semble implicitement basée sur un code utilisant les positions des hexagones dans un espace cartésien, un peu comme le code *Wiswesser* pour les polyhexes [78]. Cette méthode donna lieu à l'énumération de tous les polyhexes avec $h = 17$ hexagones, mais les temps de calcul ne sont pas fournis dans l'article.

Nous présenterons dans les chapitres qui suivent les algorithmes que nous avons élaborés pour énumérer les polyhexes simplement connectés. Le premier nous a permis l'énumération des polyhexes planaires simplement connectés jusqu'à $h = 21$ et des polyhexes simplement connectés jusqu'à $h = 20$. Le second a été utilisé pour énumérer tous les polyhexes planaires simplement connectés jusqu'à $h = 24$ alors qu'il nous a permis de compter tous les polyhexes simplement connectés jusqu'à $h = 27$.

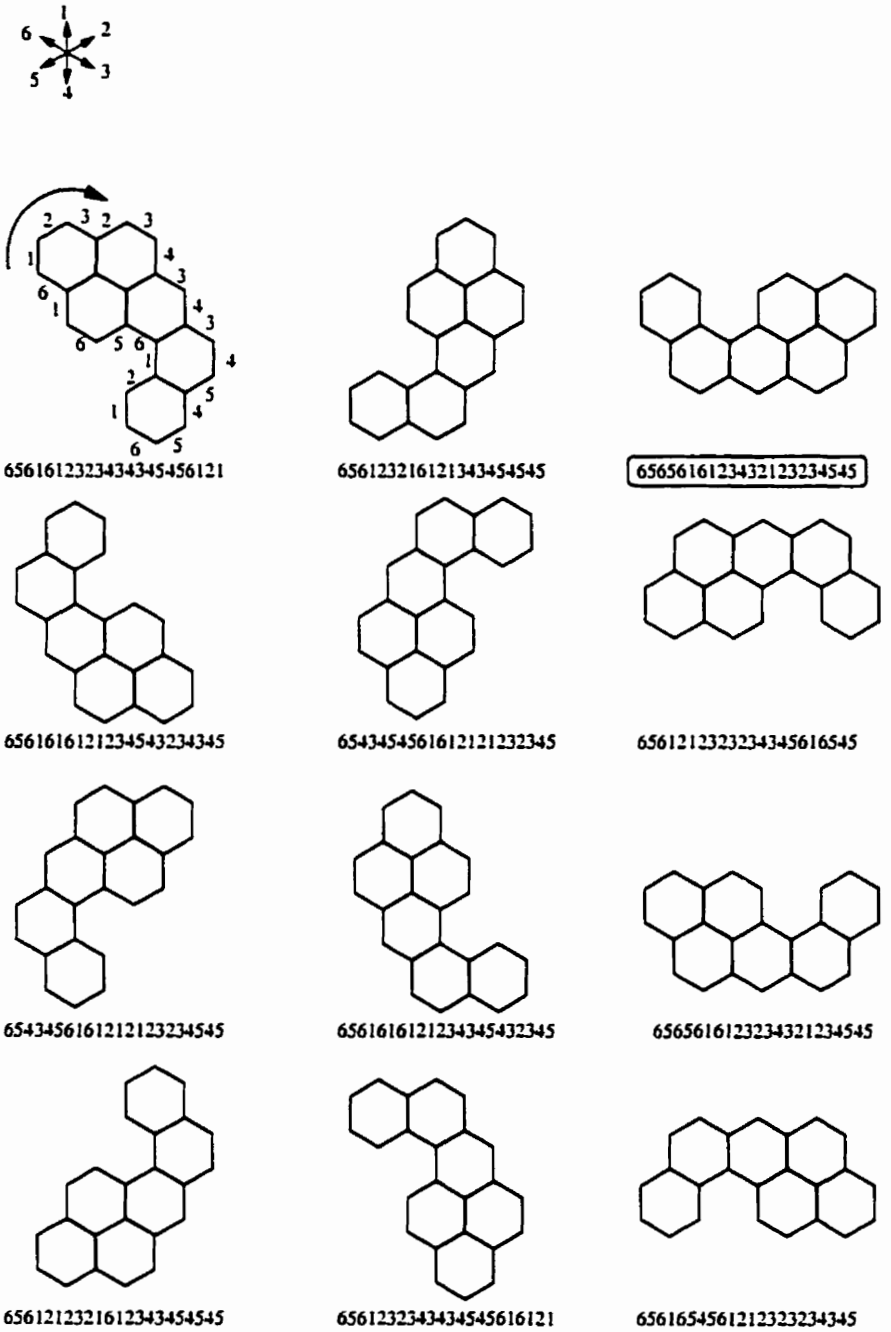


Figure 4.3 – Calcul du *boundary code* (encadré) d'un polyhexe

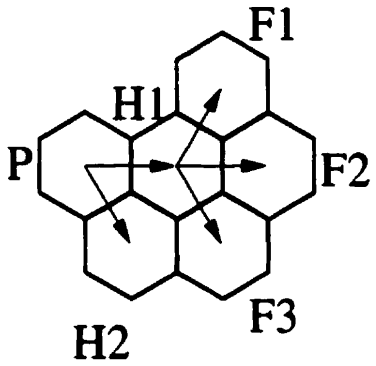


Figure 4.4 - Construction de l'arbre de recouvrement utilisé par le code DAST

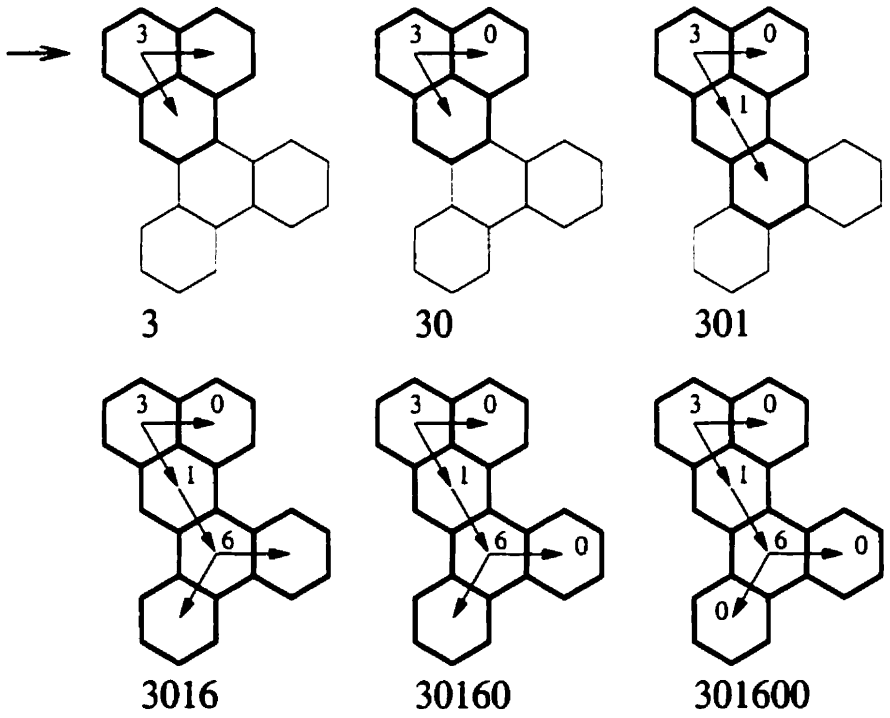


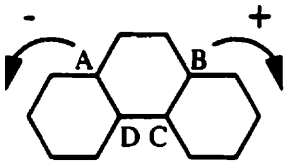
Figure 4.5 - Construction du code correspondant à un arbre de recouvrement

Chapitre 5

Une méthode d'énumération de polyhexes basée sur la génération ordonnée

5.1 Énumération de polyhexes planaires simplement connectés

Nous proposons ici un algorithme basé sur la méthode de recherche inversée proposée par Avis et Fukuda [7]. Le code utilisé est le "*Boundary-edges Code*" (ou code *BEC* défini par Hansen, Lebatteux et Zheng [75]). Dans le cas des polyhexes simplement connectés, il est équivalent au code PC-2 de Herndon et Bruce [79]. Le "*Boundary-edges Code*" est défini comme suit pour les polyhexes simplement connectés. En commençant par un sommet externe de degré 3, qui appartient donc à deux hexagones, suivre le contour du polyhexe en notant par un chiffre le nombre d'arêtes sur le contour suivi pour chaque hexagone rencontré (le même hexagone peut apparaître jusqu'à 3 fois sur le contour du polyhexe, et peut donc correspondre à 1, 2 ou 3 chiffres du code). Ensuite, appliquer, si nécessaire, des permutations circulaires ou des inversions (de la fin vers le début) afin que le code soit lexicographiquement maximum. La construction du code *BEC* d'un petit polyhexe est représentée sur la figure 5.1. Remarquons que le code est unique, mais peut être obtenu de diverses manières dans le cas où le polyhexe est symétrique.



	-	+
A	5153	3515
B	3515	5153
C	5351	1535
D	1535	5351

Figure 5.1 – Construction du code *BEC*.

Proposition 1 *Le code BEC d'un polyhexe débute toujours par un chiffre supérieur ou égal à* 3.

Preuve:
Gutman et Cyvin ([72] p23) ont montré que le nombre d'arêtes libres d'un polyhexe est $6 + b$ où b est le nombre d'arêtes externes dont chaque extrêmité a un degré 3 dans le polyhexe (ces arêtes correspondent à des "1" dans le code *BEC*). Si le premier chiffre du code est inférieur ou égal à 2, alors il n'y a aucun chiffre supérieur à 2 dans le code puisque ce dernier est lexicographiquement maximum. Dans ce cas, aucun hexagone n'a d'arêtes libres, ce qui contredit la propriété mentionnée. □

Pour simplifier l'exposé, nous utiliserons l'expression "$n^{\text{ième}}$ *hexagone*" au lieu de "*hexagone représenté par le* $n^{\text{ième}}$ *chiffre du code*".

5.2 Principe et algorithme

Comme dans la plupart des méthodes d'énumération, nous procédons par ajouts successifs d'hexagones. Ceci peut être fait en profondeur d'abord en ajoutant des

hexagones jusqu'à ce que le nombre h_{max} soit atteint, ou en largeur d'abord, en construisant tous les polyhexes à h hexagones à partir de ceux à $h-1$ hexagones, et ainsi de suite. Dans chaque cas, il peut y avoir duplication puisque le même polyhexe à h hexagones peut être construit à partir de plusieurs polyhexes à $h-1$ hexagones.

Afin d'éviter toute répétition, chaque polyhexe avec h hexagones est généré de manière légitime par un seul polyhexe à $h-1$ hexagones, en utilisant la méthode d'Avis et Fukuda [7].

La vérification de la légitimité du polyhexe (à $h-1$ hexagones) à partir duquel le polyhexe courant vient d'être généré est rapide grâce au choix d'une relation père-fils efficace.

La méthode de recherche inversée d'Avis et Fukuda et la génération ordonnée

Avis et Fukuda [7] ont introduit la méthode de *recherche inversée* pour résoudre de manière efficace un problème classique de recherche opérationelle et de géométrie calculatoire (Computational Geometry) : l'énumération de tous les sommets d'un polytope (la méthode s'étend facilement au cas de l'énumération des points extrêmes et rayons extrêmes d'un polyèdre). Les algorithmes pour ce problème explorent de différentes manières le *graphe d'adjacence* d'un polytope. Les sommets et arêtes de ce graphe correspondent respectivement aux sommets et arêtes du polytope ; deux sommets du graphe étant adjacents si et seulement si les sommets correspondant du polytope le sont. En d'autres mots, il est obtenu de l'ensemble des sommets et arêtes par les propriétés d'incidence topologique, en négligeant les propriétés métriques. Habituellement, le graphe d'adjacence est exploré par *recherche en profondeur d'abord* (voir Aho, Hopcroft et Ullman [2] pour une description de cette technique). La question est alors de savoir lorsque la procédure arrive en un sommet, s'il a déjà été

rencontré ou non. Durant plus de 25 ans, ce problème a été résolu à l'aide d'une (longue) liste des sommets rencontrés afin de la consulter pour savoir si le dernier sommet rencontré en fait partie ou non (voir Dyer [49]). Une telle procédure prend à la fois du temps et de l'espace mémoire, puisque le nombre de sommets d'un polytope peut être très grand, même si le polytope est de dimension modeste (typiquement plus d'un million pour 10 variables). Une autre méthode a été proposée par Chen, Hansen et Jaumard [28], elle utilise de listes d'adjacence entre sommets, obtenues d'abord en incluant le polytope dans un simplexe. On ajoute ensuite les facettes du polytope l'une après l'autre, tandis que les listes de sommets et d'adjacence sont mises à jour. Il est toutefois nécessaire de garder une liste relativement longue de sommets de la dernière facette ajoutée afin de déterminer leurs adjacences.

Par contre, Avis et Fukuda [7] utilisent une observation simple mais puissante pour éviter d'avoir recours à des listes. On suppose qu'une arborescence (inversée) puisse être définie sur le graphe d'adjacence (ou, en d'autres mots, qu'exactement un successeur soit associé à chaque sommet, exceptée la racine). Ceci peut se faire pour l'énumération de sommets d'un polytope par l'utilisation de la règle de Bland [15] comme critère de choix des variables entrante et sortante dans une version de l'algorithme du simplexe dont la convergence est garantie, même en cas de dégénérescence. Alors, l'exploration du graphe d'adjacence se fait par *recherche en profondeur d'abord* depuis la racine. Lorsque la procédure arrive au traitement d'un sommet, nous vérifions en "inversant la recherche" si l'itération suivante du simplexe nous donne le père de ce sommet. Si tel est le cas, le sommet est considéré comme légitime et enregistré. Sinon, nous effectuons un retour arrière. Les auteurs, puis d'autres ont rapidement remarqué que cette méthode de *recherche inversée* peut être utilisée pour un grand nombre de problèmes de géométrie calculatoire et pour d'autres domaines.

Dans le cas de l'énumération de polyhexes, un graphe est obtenu en associant un sommet à chaque polyhexe et une arête est présente entre deux sommets s'il est

possible de construire un polyhexe à partir de l'autre en lui ajoutant un hexagone. La construction d'une arborescence orientée (inversée), appelée arbre d'énumération, à partir de ce graphe est expliquée dans la section 5.2.1.

Comme nous l'a fait remarquer Gunnar Brinkmann, cette méthode d'énumération peut aussi être considérée comme une application de la *Génération Ordonnée*, déjà proposée indépendamment à la fin des années soixante-dix par Read [108] et Faradzev [58]. Nous référons le lecteur à Brinkmann [23] et Mc Kay [98] pour une présentation des variantes et des applications récentes de cette technique exploitant la symétrie.

5.2.1 Définition de l'arbre d'énumération

À chaque polyhexe P de h sommets (fils), nous associons un unique polyhexe avec $h-1$ sommets (père). En utilisant le code *BEC*, une manière directe de définir un père unique est de prendre parmi tous les polyhexes à $h-1$ sommets permettant de générer P par ajout d'un hexagone celui dont le code est lexicographiquement le plus grand. Dans ce cas, à chaque fois qu'un polyhexe est généré, nous devons examiner son code dans le but d'identifier tous ses pères potentiels, puis ces derniers doivent être construits en retirant un hexagone à P à chaque fois, dans le but d'identifier celui qui a le code lexicographiquement le plus grand. Le polyhexe sera considéré comme légitime si et seulement s'il a été construit à l'aide du père de code lexicographiquement le plus grand.

La très bonne performance de l'algorithme proposé ici est due à la manière d'identifier si oui ou non le polyhexe généré est légitime par la règle suivante :

Règle 1 *Un polyhexe sera considéré comme légitime si et seulement si le premier chiffre de son code BEC est associé au dernier hexagone ajouté.*

En d'autres mots, la *Règle 1* dit que le père d'un polyhexe P est le polyhexe obtenu en enlevant l'hexagone de P représenté par le premier chiffre du code (appelé aussi **premier hexagone** du polyhexe pour simplifier la notation).

L'utilisation de cet arbre d'énumération (représenté sur la figure 5.2) réduit significativement la complexité du programme. Au lieu d'examiner le code du polyhexe afin de trouver tous les pères potentiels comme décrit plus tôt, nous n'en considérerons qu'un. Dans le pire cas, la complexité de la phase de validation est réduite de $O(h^3)$ à $O(h^2)$, la validation étant comprise dans l'arrangement sous forme canonique du code du fils lui même. Comme il s'agit là de l'opération critique, la complexité générale de l'algorithme est réduite d'un facteur h.

Une méthode d'énumération idéale devrait permettre d'identifier *a priori* les positions où ajouter un hexagone de manière à ce que le polyhexe ainsi obtenu soit légitime, au lieu de procéder à une validation après construction de ce dernier. Il semble qu'un tel modèle d'énumération soit difficile à construire. Toutefois, l'identification *a priori* d'ajouts illégitimes est souvent possible, avant même tout calcul, en raison de la définition de l'arbre d'énumération (par exemple : ajouter un hexagone qui sera représenté comme un 3 dans le code alors que d'autres hexagones sont et demeurent représentés par des 4 ou 5 dans le même code). Quand la génération n'est pas démontrée illégitime, le réarrangement du code sous forme canonique est simple puisque le point de départ du code est déjà connu. Dans ce contexte, l'identification d'un meilleur point de départ pour le code conduit immédiatement au rejet du polyhexe avant même que son code n'ait été complètement construit.

5.2.2 Ajouts d'hexagones

Il y a trois manières valides d'ajouter des hexagones à un polyhexe.

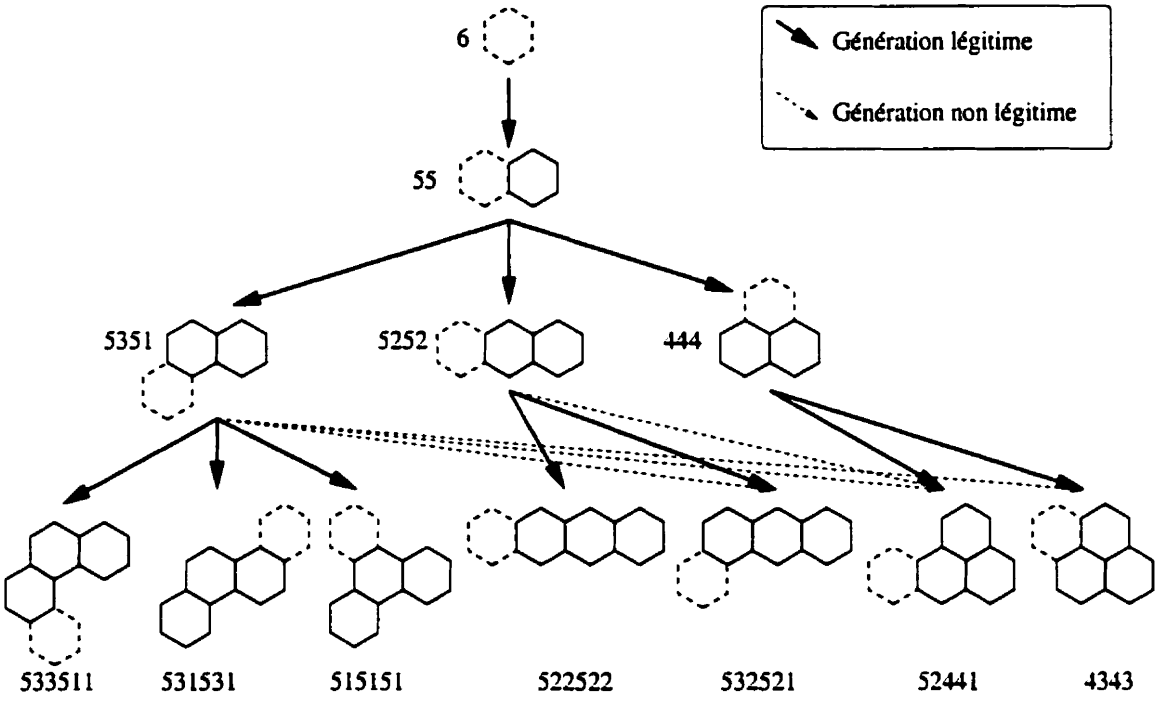


Figure 5.2 – Représentation de l'arbre d'énumération.

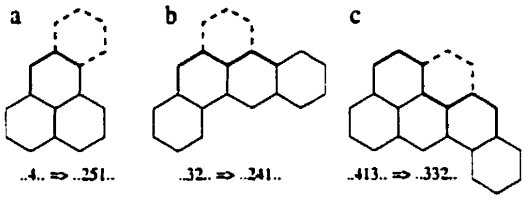


Figure 5.3 – Ajouts d'hexagones.

1. Un hexagone représenté par un chiffre $x \geq 3$ dans le code avec au moins un côté libre peut accepter un ajout. Le chiffre "x" du code est alors remplacé par la séquence "$a5b$" où $a + b + 1 = x$ et $a \geq 1$, $b \geq 1$ (voir la figure 5.3 *a*).
2. Deux hexagones adjacents représentés par la séquence "xy" (où $x > 1$, $y > 1$) dans le code peuvent accepter l'ajout d'un hexagone adjacent à chacun d'eux. La séquence "xy" du code est alors remplacée par "$(x-1)4(y-1)$" (voir la figure 5.3 *b*).
3. Trois hexagones consécutifs sur le contour représentés par la séquence "$x1y$" (où $x \geq 2$, $y \geq 2$) peuvent aussi supporter l'ajout d'un hexagone qui soit adjacent à tous les trois. La séquence $x1y$ du code est alors remplacée par "$(x-1)3(y-1)$" (voir figure 5.3 *c*).

Proposition 2 *Aucun polyhexe obtenu par ajout d'un hexagone partageant plus de trois arêtes consécutives avec le polyhexe n'est valide au sens de l'arbre d'énumération.*

Preuve:

L'hexagone ajouté aurait alors au plus 2 arêtes externes. D'après la *règle* 1, le code du polyhexe ainsi construit débuterait par un chiffre inférieur ou égal à 2. D'après la *Proposition* 1, aucun polyhexe n'a un tel code. □

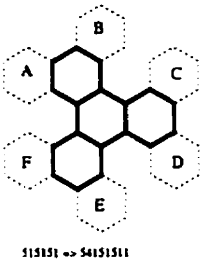


Figure 5.4 – Le polyhexe 515151 peut engendrer 6 fois le même fils.

5.2.3 Méthode pour éviter la génération multiple de polyhexes à partir du même père

Dans le cas où le père appartient à des classes non triviales de symétrie, il peut engendrer plusieurs fois le même polyhexe (voir figure 5.4 pour un exemple). Le nombre de polyhexes générés à partir du même père étant relativement petit (environ 5 en moyenne), nous en gardons simplement une liste dans laquelle nous vérifions s'il y a duplication.

Si les classes de symétrie des polyhexes avec $h-1$ hexagones sont connues lors de la génération de ceux avec h hexagones, cette étape peut être évitée. Il y a deux cas :

(i) si le père n'a pas de symétrie, (ce qui arrive le plus souvent d'après les Tableaux B.1 à B.3 plus loin) le test pour la génération multiple n'est pas effectué.

(ii) Si le père est symétrique, on doit considérer un seul ajout pour chaque orbite.

Toutefois, il ne semble pas que de tels raffinements de la méthode soient nécessaires car les gains effectués réduisent le temps de calcul de moins d'un pour cent.

5.2.4 Description de l'algorithme

Cet algorithme énumère tous les polyhexes valides au sens de l'arbre d'énumération avec h hexagones qui sont générés à partir d'un polyhexe P donné à $h-1$ hexagones.

1. **Ajout d'hexagones :** Dans le cas où nous cherchons à énumérer les polyhexes planaires simplement connectés, nous devons nous assurer que les conditions de planarité seront respectées avant d'ajouter un hexagone. Ensuite
   - Générer tous les polyhexes possibles obtenus par addition d'un "5" au code de P.
   - Si le code de P ne débute pas par un "5", générer tous les polyhexes possibles obtenus par ajout d'un "4" au code de P, sinon, ne considérer l'ajout d'un "4" qu'adjacent au premier "5" du code.
   - Si le code de P ne comporte aucun "5" et au plus deux "4", considérer l'ajout d'un "3" au code.
2. **Validation :** S'assurer que chaque polyhexe généré peut être décrit par un code débutant par le chiffre correspondant au dernier hexagone ajouté. Rejeter ceux qui ne peuvent s'exprimer ainsi.

   Les justifications de l'algorithme découlent de la *Règle* 1, de la définition du code *BEC* et plus particulièrement de ses propriétés lexicographiques.

De manière pratique, nous avons procédé par trois étapes successives pour accomplir l'énumération.

1. **Initialisation :** Nous construisons d'abord une base de données de polyhexes pour h relativement petit. Le programme peut assurément être initialisé avec le seul polyhexe à 2 hexagones, mais dans ce cas, toute l'énumération devra

être complétée sans aucune interruption et tout problème technique (tel qu'une panne d'électricité) obligera à recommencer le travail depuis le début. Comme le programme d'énumération est plutôt rapide, lire un polyhexe depuis un fichier est plus lent que de le générer à partir d'un plus petit polyhexe, ce qui nous pousse à trouver un compromis entre le temps et les risques techniques.

Pour $h = 18$ à 20, nous avons décidé de débuter l'énumération avec 67 fichiers contenant les 331 polyhexes à 7 hexagones tandis que 6018 fichiers contenant les 30086 polyhexes à 10 hexagones ont été utilisés pour $h = 21$.

2. **Enumération** : Pour chaque fichier, nous générons tous les descendants de chaque polyhexe jusqu'à la taille désirée par un appel récursif à l'algorithme décrit plus tôt.

   L'utilisation de l'arbre d'énumération permet une énumération simultanée sur différents ordinateurs sans aucune mémoire partagée (c'est ce qui a été fait pour $h = 21$). Dans le cas de traitements en parallèle des fichiers, un système de blocage des fichiers a été mis en place afin d'éviter que ceux-ci ne soient corrompus et que certaines tâches ne soient accomplies plusieurs fois.

3. **Résultats supplémentaires** : Le nombre d'atomes de carbone et d'hydrogène de chaque polyhexe est directement calculé à partir du nombre de chiffres dans son code, puisqu'il ne dépend que du nombre de sommets intérieurs et d'hexagones. Le nombre d'hexagones est relié à la profondeur de l'arbre d'énumération et le nombre de sommets internes vient directement du nombre de chiffres présents dans le code *BEC* [79], [75].

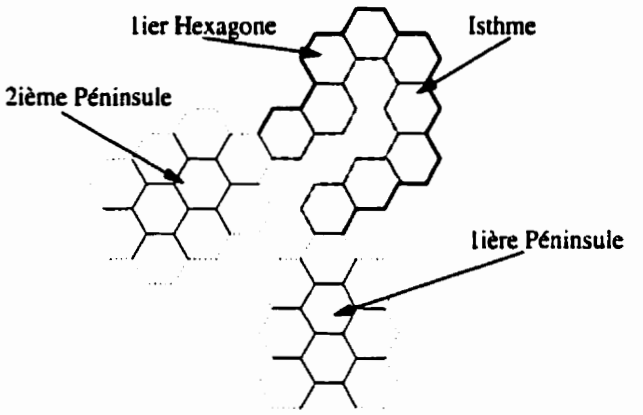
   Le nombre de baies cancérigènes, qui sont représentées par la séquence "515" dans le code *BEC* est calculé aisément. Les classes de symétrie sont aussi déterminées en utilisant des règles simples décrites par Hansen *et al* [75]. Après un traitement complet d'un fichier de données, les fichiers de résultats indiquant le nombre de chaque type de polyhexes généré sont remis à jour avant de passer au fichier suivant.

5.2.5 Validité de l'arbre d'énumération jusqu'à $h = 21$

L'arbre d'énumération est valide si et seulement si chaque polyhexe a un et un seul père. Comme le code *BEC* d'un polyhexe est unique, son premier hexagone est unique aussi, alors chaque polyhexe a au plus un père si le premier hexagone apparaît seulement une fois dans le code *BEC* du polyhexe. Il peut toutefois y avoir une difficulté dans le cas où le premier hexagone du polyhexe apparaît deux fois dans son code (il ne peut pas apparaître trois fois puisque le code d'un polyhexe ne peut pas débuter par un 1). En effet, dans ce cas, enlever le premier hexagone du polyhexe le séparerait en deux parties, et il n'aurait pas de père car une structure non connexe n'est pas un polyhexe. N'ayant pas de père, ce polyhexe est appelé *orphelin*. Le problème des orphelins n'apparaît toutefois pas dans l'énumération de polyhexes avec assez peu d'hexagones, comme nous le montrerons.

Afin d'aborder le thérème suivant et sa preuve, donnons quelques définitions supplémentaires relatives aux polyhexes.
L'**isthme** d'un orphelin est l'ensemble maximum d'hexagones consécutifs, incluant le premier hexagone, qui apparaissent tous deux fois dans le code et sont adjacents à exactement deux hexagones apparaissant deux fois dans le code. Une **péninsule** est un des deux polyhexes obtenus suite au retrait de l'isthme d'un orphelin. La **première péninsule** apparaît la première dans le code de l'orphelin alors que la **seconde péninsule** n'apparaît qu'ensuite. Comme nous utiliserons des codes partiels, il est pertinent de préciser que les codes dont nous parlerons au cours de cette preuve sont tous calculés dans le sens des aiguilles d'une montre. Le **code de l'isthme** est la chaîne de caractères représentant le contour selon l'isthme depuis la seconde péninsule vers la première (le début du code de l'orphelin est inclus dans le code de l'isthme). L'isthme a deux voisins fixés (un pour chaque péninsule). Le **code long de la péninsule** d'une péninsule donnée est la chaîne de caractères représentant les

hexagones appartenant à cette péninsule dans le code de l'orphelin, dans le même ordre. Le **code court de la péninsule** est la chaîne de caractères décrivant la péninsule si on suppose que l'isthme est enlevé, en débutant par l'hexagone adjacent à l'isthme (remarquons que ni le code court ni le code long d'une péninsule n'est ordonné lexicographiquement). La séquence **caractéristique d'une péninsule** est la séquence lexicographiquement la plus grande qui soit dans le code long de la péninsule. Ces concepts sont illustrés sur la figure 5.5.



Code de l'orphelin: 3332321132333231321211132 13233323121

Code long de la première péninsule: 132333231

Code court de la première péninsule: 33233323

Première séquence caractéristique: 333231

Figure 5.5 – Définitions relatives aux orphelins.

Théorème 2 *Aucun orphelin planaire simplement connecté n'a moins de 29 hexagones.*

La preuve utilise les quatre lemmes suivants :

Lemme 4 *Le code d'un orphelin débute par un "3".*

Preuve:

Le premier hexagone d'un orphelin doit apparaître deux fois dans son code ; ainsi il ne peut pas être un "4" ou un "5". D'après la *Proposition* 1, ce doit être un "3". □

Lemme 5 *Le code d'un orphelin ne comporte aucun "4" ou "5".*

Preuve:

La preuve découle directement du lemme 4 et de la règle lexicographique. □

Lemme 6 *Le code d'un orphelin ne peut pas comporter plus de trois "3" successifs.*

Preuve:

Comme, d'après le *lemme* 5, le code d'un orphelin ne comporte aucun chiffre supérieur à trois, et qu'il doit être ordonné de manière lexicographique, la plus grande séquence de "3" successifs doit se trouver au début du code. La présence de quatre "3" successifs impliquerait que l'orphelin comporterait un trou (ce serait alors un coronoïde, ou bien il ne serait pas planaire comme l'indique la figure 5.6). □

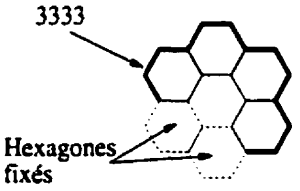


Figure 5.6 – Aucun code d'orphelin planaire ne comporte plus de trois "3" successifs.

Lemme 7 *Le code d'un orphelin ne peut comporter la séquence "333233".*

Preuve:

Supposons par contradiction que le code d'un orphelin contienne la séquence "333233". Alors, d'après les *lemmes* 4 à 6, le début de ce code doit être "333233", mais ceci

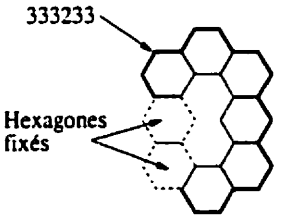


Figure 5.7 – Aucun code d'orphelin planaire ne peut comporter la séquence "333233".

implique que le polyhexe comporte un trou ou ne soit pas planaire comme le montre la figure 5.7. □

Nous démontrons maintenant le *Théorème* 2 : aucun orphelin planaire simplement connecté n'a moins de 29 hexagones.

Preuve:
La preuve se fait par énumération des petites péninsules et de leurs positions relatives en tenant compte des propriétés des orphelins établies par les lemmes 4 à 7.

La première étape consiste à énumérer toutes les petites péninsules (ne considérer que celles avec $h \leq 12$ est suffisant). Ce sont tous les polyhexes avec au plus trois "3" successifs, un "4" et aucun "5" dans leur code. En effet, un "5" ne peut pas être le premier hexagone d'une péninsule puisqu'il partage une arête avec un hexagone qui apparaît deux fois dans le code; selon la définition, il devrait alors faire partie de l'isthme lui même. Les péninsules avec jusqu'à 12 hexagones obtenues par une

énumération simple mais longue sont représentées sur la figure 5.8. Les arêtes qui peuvent être partagées avec l'isthme sont représentées en gras.

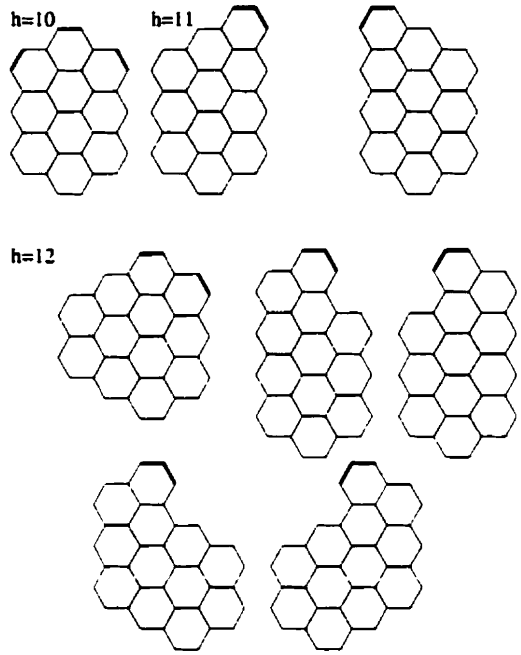


Figure 5.8 – Péninsules avec de 10 à 12 hexagones.

Les péninsules obtenues par symétrie axiale sont ici considérées comme différentes.

Le Tableau 5.1 donne la liste de tous les codes courts de péninsules ainsi que les séquences caractéristiques correspondantes. Un X dans la dernière colonne indique qu'il n'est pas nécessaire de considérer cette péninsule en vertu du lemme 7.

Soit h'_i le nombre d'hexagones de la péninsule i ($i = 1, 2$).

Notons d'abord que si la séquence caractéristique de la première péninsule débute par "33323", un hexagone doit être ajouté avant la seconde péninsule pour des raisons de planarité (voir figure 5.9 a).

1. $h'_1 = 10$. Tout orphelin comportant la péninsule à 10 hexagones doit débuter par

Tableau 5.1 – Péninsules avec de 10 à 12 hexagones et leurs séquences caractéristiques.

h	Code de péninsule	Séquence caractéristique	Lemme 7 utilisé
10	33323332	3332331	X
	33233323	333231	
	32333233	3332331	X
11	423323331	3332332	X
	423233322	333232	
	422333232	333232	
	413332332	3332332	X
12	4323233321	333232	
	4322333231	333231	
	4313332331	333233	X
	4133233313	333233	X
	4132333223	333231	
	4123332323	333232	
	332332332	332332	
	323323323	332332	

la séquence "33323". Le code de l'isthme doit comporter la séquence "1333231" ou "1333232", mais cette première configuration seule n'est pas valide à cause de la règle lexicographique (voir figure 5.9 *b*) et l'autre seule ne respecte pas la condition de planarité (voir figure 5.9 *c*).

L'étape suivante consiste à ajouter un hexagone avant la première péninsule. Il y a deux possibilités : 1333231 et 1333232.

Dans le premier cas, il y a trois manières de placer le nouvel hexagone :

(a) 13332313 : pour des raisons de planarité, un hexagone doit être ajouté avant la seconde péninsule menant au code de l'isthme 113332313 ou 213332313, mais aucun d'eux ne respecte la condition de planarité à moins qu'un autre hexagone ne soit ajouté. L'orphelin comporterait alors au moins 30 hexagones (10+10+10). Nous n'explorerons pas ce cas puisque nous trouverons un orphelin plus petit plus tard.

(b) 13332312 ou 13332311 : la règle lexicographique n'est pas respectée.

Dans le second cas, les trois manières d'ajouter le nouvel hexagone sont :

(a) 13332323 : donne un polyhexe non planaire (voir figure 5.9 *d*).

(b) 13332322 : nous pousse à ajouter un hexagone avant la seconde péninsule afin de respecter les conditions de planarité (voir figure 5.9 *e*). L'isthme devient alors "113332322" et on obtient un orphelin avec 29 hexagones (voir figure 5.9 *f*).

(c) 13332321 : nécessite l'ajout d'un hexagone avant le second isthme afin que la planarité soit vérifiée (voir figure 5.9 *g*). Alors, l'isthme comporte 9 hexagones et ne peut donner un orphelin avec moins de 29 hexagones.

2. $h'_1 = 11$.

   La séquence caractéristique de chacune des deux péninsules pour $h'_1 = 11$ est "333232" et l'isthme doit avoir au moins 7 hexagones. Il y a quatre manières de connecter la première péninsule à l'isthme (représentées sur la figure 5.9 *i*-*l*). Les deux premières donnent des polyhexes non planaires et les deux autres ne respectent pas la règle lexicographique. Dans chaque cas, un hexagone doit être ajouté à l'isthme. Le plus petit orphelin ainsi obtenu comporte au moins 10+8+11=29 hexagones. Cette configuration ne sera pas considérée puisqu'elle ne conduira à aucun orphelin plus petit que celui déjà trouvé.

3. $h'_1 = 12$. Pour les mêmes raisons que pour $h'_1 = 11$, les péninsules dont la séquence caractéristique débute par 33323 ne sont pas considérées puisqu'elles ne peuvent être utilisées pour la génération d'orphelins plus petits que celui déjà rencontré. La seule péninsule qui puisse conduire à un plus petit orphelin est "332332332" mais chaque péninsule doit alors avoir au moins 12 hexagones afin que la séquence caractéristique ne débute pas par 33323. Alors le code de l'orphelin doit débuter par 332332 ou 333. Dans le premier cas, l'orphelin

comporte au moins 30 hexagones, et dans le second, deux hexagones doivent être ajoutés pour des raisons de planarité et l'orphelin comporte alors (au moins) 29 hexagones (voir figure 5.9 m).

4. $h'_1 = 13$. Les péninsules avec 13 hexagones ont des séquences caractéristiques débutant par "332" ou "333", ce qui implique que l'isthme doit avoir au moins trois hexagones. Si la seconde péninsule comporte douze hexagones, un hexagone doit être ajouté à l'isthme avant elle ; qu'on utilise cette péninsule ou une péninsule à treize hexagones, l'orphelin ne peut comporter moins de 29 hexagones (12+4+13 ou 13+3+13). Il n'est pas pertinent de considérer des secondes péninsules plus petites, comme il a été expliqué plus tôt.

5. $h'_1 \geq 14$. Pour toute péninsule à 14 hexagones ou plus, soit la séquence caractéristique est au moins "332", et en ce cas l'isthme doit comporter au moins trois hexagones, et le plus petit orphelin comporte 12+3+14 hexagones, (la séquence caractéristique "333" requiert au moins 5 hexagones dans l'isthme, l'orphelin a alors 10+5+14=29 hexagones ou plus), soit chaque péninsule doit avoir au moins 14 hexagones, et en ce cas l'orphelin ne peut pas avoir moins de 14+1+14=29 hexagones. Aucun orphelin de moins de 29 hexagones ne peut être construit de cette manière. □

La présente méthode ne peut donc pas être utilisée pour énumérer les polyhexes planaires simplement connectés avec plus de 28 hexagones, sans l'utilisation d'une routine complémentaire de génération des orphelins.

5.2.6 Vérification de la planarité

Afin d'éviter la génération d'hélicènes, nous notons la position de chaque hexagone externe du polyhexe courant dans le plan et déterminons ensuite les arêtes sur lesquelles il est possible d'ajouter des hexagones sans que les conditions de planarité ne

cessent d'être satisfaites. La figure 5.10 montre les positions qui doivent être vérifiées selon le nombre d'arêtes libres qu'aura l'hexagone ajouté. Si l'ajout d'un hexagone implique la superposition de deux arêtes, cet ajout est interdit.

Ce programme s'est montré assez rapide pour mener à bien l'énumération et l'identification des caractéristiques mentionnées pour tous les polyhexes planaires simplement connectés jusqu'à $h = 20$ hexagones à l'aide d'un PC Pentium™I 133 MHz en 63 jours 14 heures et 25 minutes.

5.2.7 Résultats numériques

Le principal résultat nouveau obtenu fut le nombre de polyhexes planaires simplement connectés avec $h = 18, 19, 20$ et 21 hexagones. Ces résultats sont présentés dans le Tableau 5.2.

Tableau 5.2 – Nombre de polyhexes planaires simplement connectés et temps requis en fonction de h(*) sur un ensemble hétérogène d'ordinateurs difficiles à comparer avec le Pentium™133 utilisé par ailleurs.

h	Nombre de polyhexes	Temps de calcul
10	30 086	0.46 sec
11	141 229	2.32 sec
12	669 584	11.42 sec
13	3 198 256	58 sec
14	15 367 577	4 min 56 sec
15	74 207 910	25 min 33 sec
16	359 863 778	2 h 10 min 20 sec
17	1 751 594 643	11 h 10 min 02 sec
18	8 553 649 747	2 j 9 h 21 min
19	41 892 642 772	12 j 8 h 14 m
20	205 714 411 986	63 j 14 h 25 m
21	1 012 565 172 403	Environ 330 jours (*)

Tableau 5.3 – Nombres exacts et estimés de polyhexes planaires simplement connectés.

h	Nombre exact	valeur estimée	erreur	(erreur relative %)
10	30086	30087	1	(0.03)
11	141229	141183	-46	(-0.03)
12	669584	669782	198	(0.03)
13	3198256	3200916	266	(0.01)
14	15367577	15383524	15947	(0.10)
15	74207910	74277568	-69658	(0.09)
16	359863778	360078349	214571	(0.06)
17	1751594643	1751728873	134230	(0.0001)
18	8553649747	8548784328	-4865419	(-0.057)
19	41892642772	41838577888	-54064884	(-0.129)
20	205714411986	204910026644	-804385342	(-0.391)
21	1012565172403	1006213570016	-6351602387	(-0.627)

Pour $h \leq 20$ les calculs ont été effectués sur un PC Pentium™I 133 MHz. Pour $h = 21$, un réseau d'ordinateurs variés a été utilisé. Notons que dans le cas où $h = 16$, la plus grande valeur pour laquelle le temps de calcul a été donnée précédemment, ce temps était de 91 jours sur un PC 386 (20 MHz) [87], ce qui a été réduit à 2 heures 10 min et 20 secondes sur un PC Pentium™I (133MHz). Il n'y a pas de doutes quant à la vitesse du nouvel algorithme, même si le précédent [87] énumérait aussi les coronoïdes, ce que ne fait pas le nôtre. Le nombre de polyhexes pour $h = 21$, soit 1 012 565 172 403, est très grand et montre l'efficacité de la méthode d'énumération.

Une formule pour estimer le nombre de polyhexes avec h hexagones à partir du nombre de polyhexes avec $h - 1$ hexagones a été donnée par Aboav et Gutman [1]. Les prédictions sont proches des vraies valeurs pour $h \leq 17$, mais semblent sous-estimer les valeurs réelles quand h croît, et les différences relatives augmentent avec h (voir Tableau 5.3).

Tošić *et al.* [111] avaient conjecturés que le nombre de polyhexes avec $h = 18$ hexagones devrait se situer dans l'intervalle $(8549 \pm 4)10^7$, ce qui est assez proche.

Nous remarquons sur le Tableau B.3 en annexe qu'aucun des isomères de $C_{81}H_{43}$ ne comporte de symétrie. La raison en est la suivante. De la relation $H = 4h + 4 - n_i$ (Gutman et Cyvin [72] p 23), où H représente le nombre d'atomes d'hydrogène, leur nombre n_i de sommets intérieurs est égal à 1. Ces polyhexes sont donc composés de 3 hexagones mutuellement adjacents auxquels sont ajoutés au plus 2 polyhexes catacondensés (un polyhexe catacondensé est un polyhexe sans sommet intérieur). Le sommet commun à ces trois hexagones doit appartenir à tout axe de symétrie (sans quoi il ne serait pas le seul sommet intérieur). Mais avoir un axe de symétrie et un nombre impair de sommets indique qu'au moins un sommet doit être séparé en deux par cet axe. Il ne peut pas être adjacent à deux des hexagones initiaux, puisque ceci impliquerait que $n_i \geq 2$, ni ailleurs sur cet axe car le polyhexe serait un coronoïde. Étant donné que le nombre d'hexagones restant, 17, n'est pas divisible par 2 ou 3, il ne peut pas y avoir de symétrie rotationelle.

Les nombres de polyhexes planaires simplement connectés en fonction des isomères et de la symétrie étaient connus pour $h \leq 14$ [42] depuis 1992 ; ils ont été ensuite obtenus pour $h = 15, 16$ et 17 par Tošić *et al.* en 1995 [111] et furent confirmés par nôtre programme. Les valeurs pour $h = 18$ à 20 sont présentées sur les Tableaux B.1- B.3 en annexe (seules les colonnes correspondant aux classes de symétrie pour lesquelles il existe au moins un polyhexe sont indiquées).

5.3 Énumeration de polyhexes simplement connectés

L'algorithme utilisé pour l'énumération de polyhexes simplement connectés (appelés en général **fusènes**) est le même que celui utilisé pour l'énumération de polyhexes planaires simplement connectés, à l'exception du fait que le test de planarité est

omis puisque cette contrainte ne s'applique plus. La méthode basée sur le contour de la structure comporte cet avantage que si la contrainte de planarité n'est plus exigée, l'algorithme se simplifie alors que d'autres méthodes ne fonctionnent plus, en particulier celle de Tosić *et al* [111] car elles utilisent les positions cartésiennes des centres des hexagones. C'est avec surprise que nous avons alors constaté que ce problème, plus simple à nos yeux, était moins bien traité dans la littérature que le problème d'énumérer les polyhexes planaires simplement connectés, comme décrit dans la section 4.3.

Des différences entre l'énumération de polyhexes simplement connectés planaires et non nécessairement planaires interviennent toutefois à plusieur niveaux.

D'une part, la validité du code BEC même n'est pas vérifiée pour tous les polyhexes simplement connectés, comme le montrent Guo *et al.* [65], puisque deux hélicènes différents à 29 hexagones sont représentés par le même code. Ce problème ne se produit toutefois pas pour $h \leq 25$ [65].

D'autre part, la taille des plus petits orphelins est largement réduite dans le cas des hélicènes, la contrainte de planarité ne s'applicant plus. Nous montrerons donc que les plus petits orphelins comportent 20 hexagones, et qu'il y en a exactement 4, qui seront décrits.

5.3.1 Validité de l'arbre d'énumération jusqu'à $h = 20$

Théorème 3 *Le plus petit orphelin parmi les fusènes comporte $h = 20$ hexagones. De plus, il y en a exactement 4 de cette taille.*

Preuve :
La preuve se fait par énumération de cas en combinant les péninsules et les isthmes.

1. Comme, d'après le *lemme* 5, le code d'un orphelin ne peut comporter aucun "4" ou "5", il est facile de vérifier que toute péninsule comporte au moins 7 hexagones (voir figure 5.12 a). De plus, il n'y a qu'une péninsule avec 7 hexagones, une avec 8 et deux avec 9 (voir figure 5.12 a,b et c). La partie du code BEC correspondant à une péninsule à 7 hexagones comporte 5 "3" successifs (celle d'une péninsule à 8 ou 9 hexagones en comporte 4). Dès lors, l'isthme correspondant d'un orphelin doit comporter au moins 5 (ou 4) hexagones.

   En conséquence, le plus petit fusène orphelin doit comporter au moins 19 hexagones. Il n'y a qu'une manière de combiner deux péninsules à 7 hexagones avec un isthme à 5 hexagones. Le code correspondant débutant par le premier hexagone de l'isthme est alors "3333313333311111111333331". Toutefois, ce code n'est pas lexicographiquement le plus grand, on en déduit que le plus petit hélicène orphelin doit au moins comporter 20 hexagones.

2. Considérons maintenant deux péninsules à 7 hexagones et un isthme à 6 hexagones. Il y a plusieurs cas. Si le code de l'isthme comporte la séquence "333333", il y a une manière unique de le connecter aux péninsules. Le code correspondant est "3333331333331111111333331", qui est lexicographiquement maximum. Nous venons de trouver un plus petit orphelin (voir figure 5.14a). Si le code de l'isthme ne comporte pas la séquence "333333", comme il doit comporter la séquence "33333", une des péninsules doit être reliée à un hexagone de l'isthme par une arête (notée (a) sur la figure 5.13) de sorte que 3 arêtes de cet hexagone soient sur le premier côté du contour. De plus, cette péninsule doit être la seconde. En effet, si tel n'était pas le cas, le premier hexagone serait le second de l'isthme (pour que le fusène soit un orphelin) et le code débuterait par "33333133333111" mais en débutant au dernier "3" du code de la première péninsule et en renversant le code, nous obtenons un code débutant par "33333133333213" ou "33333133333113" qui est lexicographiquement plus grand. En connectant alors la première péninsule à l'arête du dernier hexagone

de l'isthme qui laisse 2 arêtes de chaque côté (noté (b) sur la figure 5.13) donne le code "3333321333331211111333331" qui est lexicographiquement maximum. Un second plus petit fusène orphelin est trouvé (voir figure 5.14 b). En reliant la première péninsule à l'isthme par une arête avec une arête du dernier hexagone du premier côté du père (noté (c) sur la figure 5.13) donne le code "3333311333331311111333331" qui n'est pas lexicographiquement maximum.

3. Considérons ensuite une péninsule de 7 hexagones et une de 8 hexagones. La dernière a une forme géométrique unique, mais peut être connectée à l'isthme sur deux différentes arêtes (arêtes représentées en gras sur la figure 5.12 b), qui ne sont pas équivalentes puisque le code est toujours calculé selon le sens des aiguilles d'une montre. À nouveau, si une péninsule a 7 hexagones, l'isthme doit en avoir au moins 5. Si l'isthme comporte 5 hexagones, que la première péninsule en a 7 et la seconde 8, il y a deux cas :

   (a) La seconde péninsule partage l'arête *(a)* avec l'isthme. Le code correspondant est "3333313333311111122333321" qui n'est pas lexicographiquement maximum.

   (b) La seconde péninsule partage l'arête *(b)* avec l'isthme. Le code correspondant est "3333313333311111112333322" qui à nouveau n'est pas lexicographiquement maximum.

   Si l'isthme comporte 5 hexagones, la première péninsule en a 8 et la seconde 7, il y a à nouveau deux cas :

   (a) La première péninsule partage l'arête *(a)* avec l'isthme. Le code est alors "3333322333321111111333331" qui est lexicographiquement maximum. Un troisième fusène orphelin est trouvé (voir figure 5.14 c).

   (b) La première péninsule partage l'arête *(b)* avec l'isthme. Le code est alors "3333312333322111111133331" qui n'est pas lexicographiquement maximum.

4. Finalement, considérons deux péninsules avec 8 hexagones et un isthme avec 4. Comme la partie du code BEC correspondant à l'isthme comporte la séquence "3333" pour que le fusène soit un orphelin, les arêtes de l'isthme auxquelles les péninsules sont reliées sont fixées et toutes deux *(a)*. Alors, en connectant chaque péninsule à une arête *(a)* donne le code "333322333321111122333321" qui est lexicographiquement maximum. Un quatrième fusène orphelin est trouvé (voir figure 5.14 d). En reliant la première (resp. la seconde ou les deux) péninsule(s) à l'isthme donne les codes "333312333322111122333321", "33332233332-1111112333322", et "333312333322111112333322" parmi lesquels aucun n'est lexicographiquement maximum. □

On déduit du *théorème* 3 que les fusènes jusqu'à $h = 20$ peuvent être énumérées par l'algorithme décrit plus tôt, si nous ajoutons les quatres orphelins qui viennent d'être identifiés.

5.3.2 Résultats numériques

Ce programme nous a permis d'énumérer pour la première fois à notre connaissance les fusènes avec $h = 16$ à 20 hexagones, ce qui représente un ensemble 4 000 fois plus important que le plus gros ensemble précédemment énuméré [24].

Les temps de calcul pour $h \geq 17$ (en utilisant une large variété de machines) sont aussi indiqués sur le Tableau 5.4. Il semble que ces temps augmentent linéairement avec le nombre de fusènes énumérés et avec le nombre d'hexagones qu'ils comportent. Les nombres de polyhexes simplement connectés pour $h = 16$ à 20 sont donnés dans les Tableaux B.7 à B.11. Il est possible avec ces Tableaux, en utilisant les nombres de benzenoides de chaque classe de symétrie et d'isomères trouvés lors de l'énumération des polyhexes planaires simplement connectés [111] [26], de déduire le nombre d'hélicènes correspondant.

Tableau 5.4 – Nombre de polyhexes simplement connectés pour $17 \leq h \leq 20$.

h	total	Temps de calcul
17	3018986040	2 jours 4h 31'
18	15927330105	11 jours 18h 22'
19	84530870455	46 jours 16h 31'
20	451069339063	242 jours 4h 40'

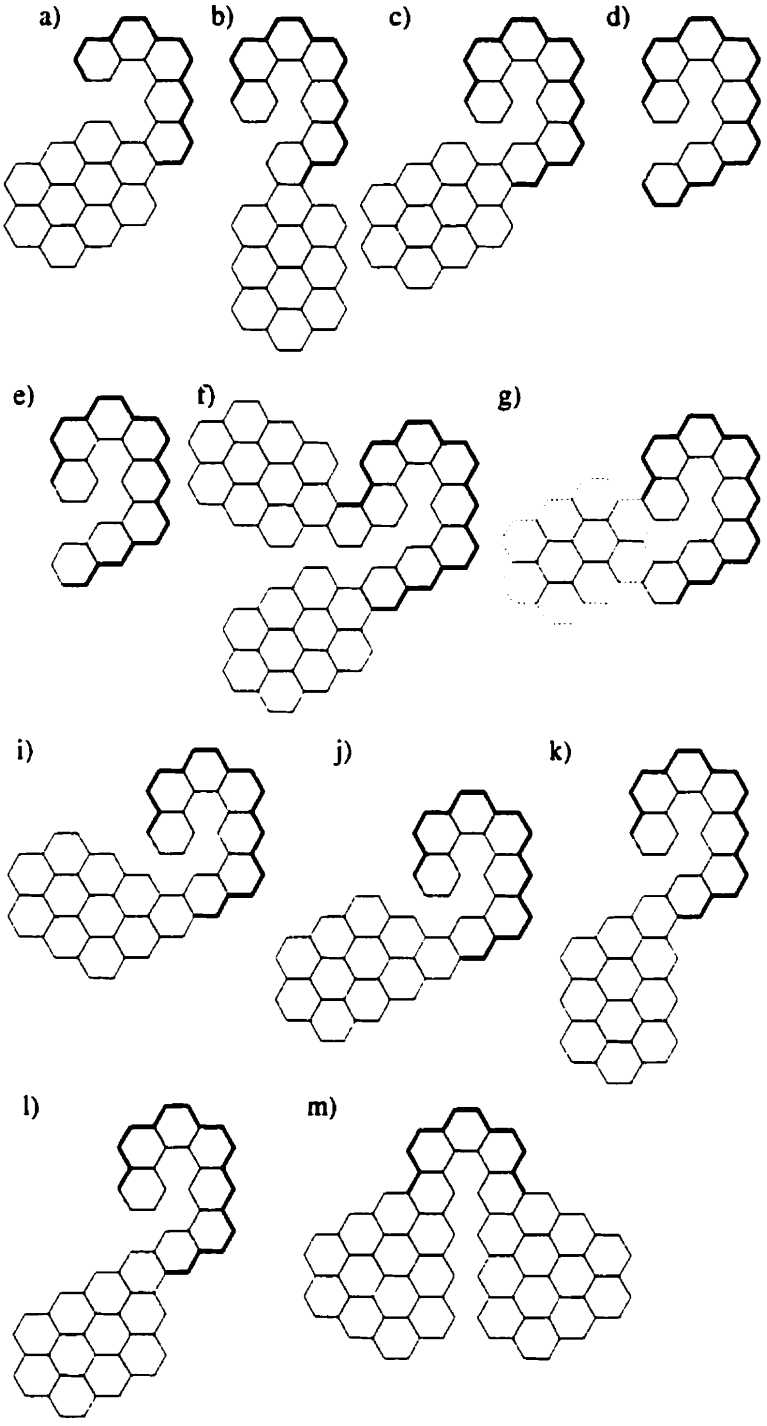


Figure 5.9 – Configurations considérées dans la preuve.

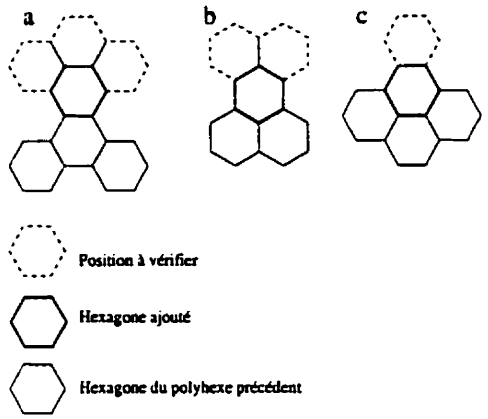


Figure 5.10 – Vérification de la planarité dans le cas où l'hexagone ajouté comporte 3 (a), 2 (b) ou 1 (c) arête libre.

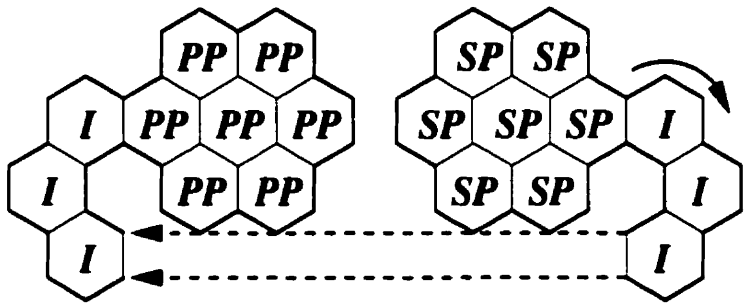


Figure 5.11 – Un orphelin (avec comme code BEC 3333331333331111111333331), son isthme I, sa première PP et seconde SP péninsules.

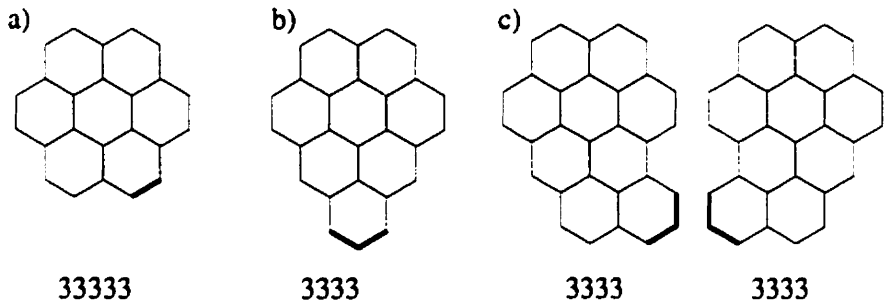


Figure 5.12 – Péninsules avec 7, 8 et 9 hexagones, et leur plus grande séquence dans le code BEC, avec les connexions possibles à l'isthme (en gras).

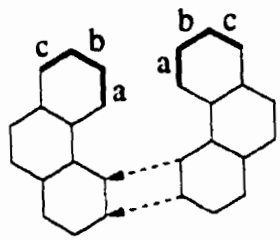


Figure 5.13 – Un isthme et les connexions possibles aux péninsules (en gras).

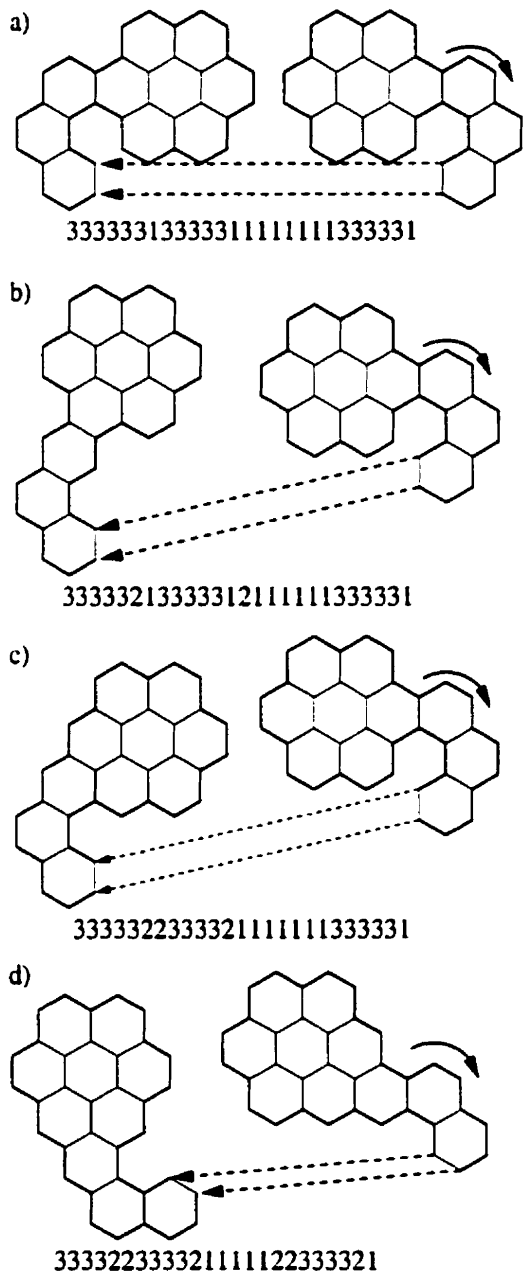


Figure 5.14 – Fusènes orphelins à 20 hexagones

Chapitre 6

Une méthode d'énumération par décomposition

6.1 Définitions et propriétés

Une **face**, définie pour les représentations planaires de graphes, est un cycle sans sommets intérieurs ni cordes. Excepté la face extérieure qui est infinie, les faces sont bornées par les arêtes sur la représentation du graphe.
Le **Graphe dual** associé à un polyhexe est défini comme suit : à chaque hexagone est associé un sommet et une arête entre deux sommets indique que les hexagones correspondants partagent une arête. Ainsi, l'aspect géométrique n'est pas considéré sur les graphes duaux (les positions de chaque sommet sur le plan n'ont pas d'importance).
Une **représentation propre** d'un graphe dual est une représentation planaire telle que toutes les faces bornées soient des triangles.

Propriété 1 *À chaque polyhexe est associé un et un seul graphe dual, mais plusieurs polyhexes peuvent être associés au même graphe dual, comme illustré sur la figure 6.1.*

Pour des raisons de simplicité d'écriture, le graphe dual d'un polyhexe sera simplement appelé graphe dual, alors que la représentation classique du polyhexe sera qualifiée de graphe original.

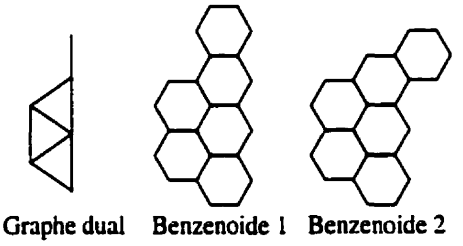


Figure 6.1 – Deux polyhexes partageant le même graphe dual

Propriété 2 *Si un polyhexe H appartient à un ensemble de classes de symétries $\mathcal{C}$, alors son graphe dual G appartient aussi à $\mathcal{C}$.*

Cette propriété est très importante puisque les tests de symétrie servent à éviter la duplication et représentent en général la partie critique des algorithmes d'énumération.

6.2 Algorithme

L'algorithme proposé est composé de deux étapes et cherche à tirer avantage des propriétés des graphes duaux mentionnées ci-dessus. La propriété 1 signifie que certains calculs peuvent être effectués seulement une fois pour un grand nombre de polyhexes alors que la propriété 2 implique que si G n'appartient qu'à la classe de symétrie triviale, on peut être assuré que les polyhexes correspondants ne seront générés qu'une fois sans qu'aucun test de symétrie ne soit effectué.

Comme les graphes duaux $G = (V, E)$ sont générés avant les polyhexes, une caractérisation de ceux-ci indépendante des polyhexes est nécessaire. Pour cette raison, la définition originale est remplacée par la suivante :
Un graphe G est un graphe dual si

i) G est planaire ;

ii) G est connexe ;

iii) toutes les faces bornées de G ont une taille de 3 ;

iv) tous les sommets intérieurs ont un degré 6 ;

v) la somme du nombre de fois t_v qu'un sommet v apparaît sur le contour de G et de son degré est au plus 6 : $t_v + \delta_v \leq 6$.

L'algorithme doit effectuer les trois tâches suivantes :

a) Générer tous les graphes duaux G comportant au plus h sommets ;

b) Trouver les classes de symétrie de chacun de ces graphes G ;

c) Générer tous les polyhexes correspondant à chaque graphe dual G, sans répétition.

La génération des graphes duaux est effectuée par génération ordonnée [58] [108]. Chaque graphe à h sommets (appelé fils) est généré à partir d'un unique graphe G' à $h-1$ sommets (appelé père). Parmi les pères potentiels, le père sera celui qui est représenté par le code lexicographiquement le plus petit (la définition du code utilisé est donnée ci-dessous). Lors de l'ajout d'un nouveau sommet w et d'une arête (v, w) à un graphe, G' (ou, en d'autres mots, lors de l'ajout d'une arête pendante (v, w) à v), nous devons aussi considérer toutes les manières possibles d'ajouter une arête incidente à w et aux sommets consécutifs $v+1, v+2\ldots$ sur la frontière de G' qui donnent un graphe dual d'après la définition donnée ci-dessus. Remarquons que dans certains cas, des arêtes supplémentaires doivent être ajoutées (voir figure 6.2) alors que dans d'autres cas elles peuvent être ajoutées ou non (voir figure 6.3).

Les règles pour l'ajout d'arêtes dépendent du degré δ_w du sommet w et sont les suivantes :

i) Si w est adjacent à un seul sommet v de G, alors $t_v + \delta_v \leq 4$. En effet, en ajoutant un sommet pendant w et une arête (vw) à G', nous augmentons t_v et δ_v de 1 chacun. La condition mentionnée ci-dessus vient de la valeur maximum de 6 pour $t_v + \delta_v$ dans G.

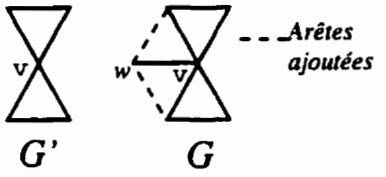


Figure 6.2 – Ajouts forcés d'arêtes après (vw), à G'.

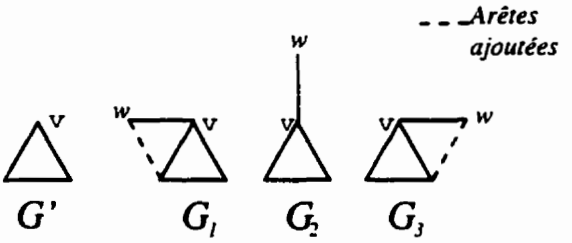


Figure 6.3 – Ajouts facultatifs d'arêtes après (vw), à G'.

ii) Si w est adjacent à deux sommets v et $v+1$ de G, alors dans G', nous avons $t_v + \delta_v \leq 5$ et $t_{v+1} + \delta_{v+1} \leq 5$. En effet, l'ajout d'un sommet w et deux arêtes (v, w) et $(v+1, w)$ à G' augmente δ_v et δ_{v+1} de 1 mais t_v et t_{v+1} restent inchangés. Le résultat découle à nouveau de la condition citée ci-dessus.

iii) Si w est adjacent à trois sommets v, $v+1$ et $v+2$ dans G, alors dans G' : $t_v + \delta_v \leq 5$, $t_{v+2} + \delta_{v+2} \leq 5$ et soit $\delta_{v+1} = 5$, soit $t_{v+1} \geq 2$. En effet, l'ajout d'un sommet w et trois arêtes (v, w), $(v+1, w)$ et $(v+2, w)$ à G' augmente δ_{v+1} de 1 et si δ_{v+1} était strictement inférieur à 5, réduit t_{v+1} de 1 ; encore une fois, le résultat découle de la condition mentionnée ci-dessus.

La recherche de symétrie des graphes duaux se fait par la méthode suivante : considérant un graphe dual G, nous définissons un court code capable d'identifier rapidement les arêtes non-équivalentes. Nous supposons qu'une représentation planaire de G, où toutes les faces intérieures sont des triangles, est connue et que pour chaque sommet, la liste de ses arêtes incidentes selon le sens des aiguilles d'une montre est donnée par des pointeurs d'une arête à la suivante. Le code est alors obtenu de la manière suivante :

a) Remplacer chaque arête non orientée par une paire d'arêtes orientées incidentes aux mêmes sommets mais de sens opposé.

b) Pour chaque arête e et orientation, construire un code comme suit :

b1) Donner l'étiquette 1 au sommet initial de e et 2 à l'autre sommet.

b2) Étiqueter les autres voisins de 1 dans l'ordre des aiguilles d'une montre à partir de 2 avec les étiquettes 3, 4, *etc...* Nous définissons l'arête de référence d'un sommet comme l'arête opposée à celle par laquelle il a été rencontré la première fois. L'arête de référence de 1 est e.

b3) Après avoir étiqueté tous les voisins des sommets 1 à $x < |V|$ de G, avec les étiquettes $1, \ldots, k$, étiqueter les voisins non encore étiquetés de $x + 1$ depuis leur arête de référence avec les étiquettes $k + 1, k + 2, \ldots$

b4) Pour chaque sommet, écrire une liste des étiquettes de ses voisins en commençant par son arête de référence. Ajouter un 0 à la fin de cette liste. Concaténer les listes dans l'ordre des sommets auxquels elles correspondent.

c) Parcourir la liste ainsi obtenue de gauche à droite et, à chaque fois qu'un sommet autre que 1 est rencontré pour la première fois, remplacer son numéro par son degré auquel est ajouté le nombre $|V|$ de sommets de G. Insérer ensuite $\delta_1 + |V|$ à la première position du code.

d) Le code choisi est celui qui est lexicographiquement le plus petit. Cette procédure de codage est illustrée sur la figure 6.5. Remarquons que les listes d'étiquettes sont calculées parallèlement avec les étiquettes.

La génération des polyhexes à partir des graphes duaux est faite par affectation d'étiquettes aux sommets de ces derniers. Chaque polyhexe peut en effet être défini à l'aide de son graphe dual ainsi que des étiquettes de chaque sommet représentant les positions relatives de ses voisins dans le plan. Dans ce but, une arborescence (arbre de recouvrement avec racine [14]) du graphe dual est définie et les

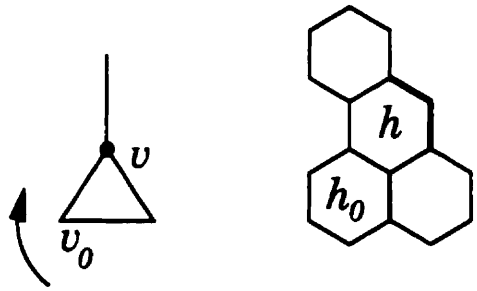


Figure 6.4 – représentation d'un polyhexe si $l(v) = 1$.

positions des hexagones sont déterminées les unes après les autres, en commençant par la racine et en respectant toutes les contraintes d'adjacences associées aux arêtes du graphe dual. Lors de ce processus, la racine ainsi qu'un de ses voisins sont placés arbitrairement. Étant donné un ensemble de sommets placés, quand nous plaçons leurs voisins, il est possible que certains sommets pour lesquels plusieurs positions respectent les contraintes d'adjacence soient rencontrés. Un tel sommet v sera dit **sommet étiquetable** et nous notons $r(v)$ le nombre de ses **positions relatives**.

La valeur de $r(v)$ est

$$r(v) = \begin{cases} 7 - \delta(v) - f(v) & si f(v) = 2 \\ 1 & sinon, \end{cases} \tag{6.1}$$

où $f(v)$, est le nombre de fois que le sommet v apparaît sur le contour de G dans une représentation propre.

Associer à chaque sommet étiquetable v une étiquette $1 \leq l(v) \leq r(v)$ suffit à complètement définir un polyhexe. Pour construire ce polyhexe, nous avons utilisé la convention que $l(v)$ correspond au nombre d'arêtes externes de l'hexagone qui lui est associé la première fois qu'il est rencontré si nous suivons le contour du polyhexe en commençant par l'hexagone associé à la racine (v_0), dans un sens donné (sens des aiguilles d'une montre ou sens trigonométrique), comme le montre la figure 6.4.

À ce moment, nous devons toujours nous assurer que le polyhexe généré est planaire. Dans ce but, une triangulation régulière du plan est mémorisée par le programme

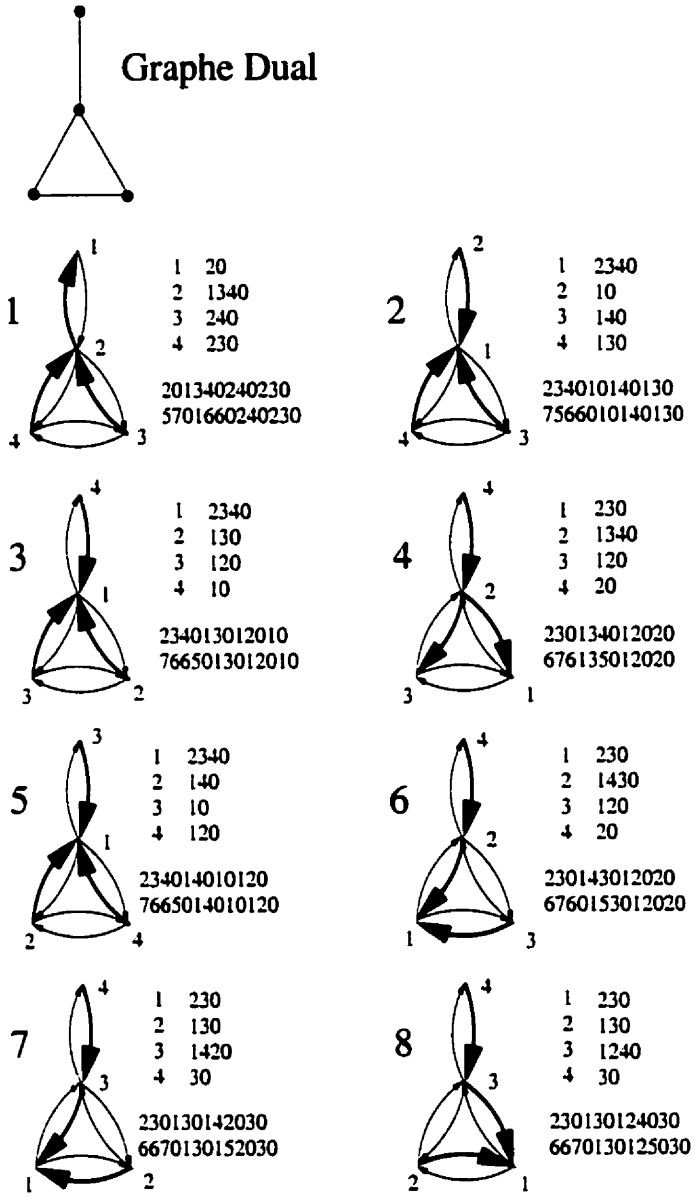
et chaque sommet v du graphe dual est associé à un nombre positif différent $n(v)$ tandis que la valeur -1 est initialement affectée à chaque sommet de la triangulation. L'étape de positionnement consiste à récursivement placer les voisins des sommets déjà placés selon les étiquettes $l(v)$. Chaque fois qu'un sommet v est placé, nous donnons la valeur $n(v)$ au sommet correspondant de la triangulation, alors, nous vérifions que tous les voisins de ce sommet de la triangulation ont la valeur -1 ou la valeur associée au voisin correspondant dans le graphe dual.

Le test de planarité (et le calcul des étiquettes $l(v)$) durant l'étape de positionement peut éviter des calculs de positionnement inutiles puisqu'il n'est pas nécessaire de continuer à définir les étiquettes d'un polyhexe qui ne peut pas être planaire.

Nous avons finalement besoin de nous assurer qu'aucun polyhexe n'est généré plusieurs fois. En effet, ceci peut se produire si plusieurs ensembles d'étiquettes distincts définissent le même polyhexe, ce qui est possible si le polyhexe appartient à des classes de symétrie non triviales. D'après la *propriété* 2, le graphe dual appartient alors aussi à des classes de symétrie non triviales, ce qui a déjà été identifié lors de la phase de génération du graphe dual. Si le graphe dual n'appartient qu'à la classe triviale de symétrie, il est alors inutile de faire quelque test que ce soit dans le but d'éviter la redondance. Étant donné que seulement une infime partie des graphes duaux ont une symétrie non triviale, comme le montre le Tableau 6.2, ce test (qui correspond à la partie critique pour la plupart des méthodes efficaces) n'est que rarement effectué. Dans le cas toutefois où le graphe dual comporte une symétrie non triviale, nous définissons une et une seule représentation **canonique** afin de nous assurer que chaque polyhexe n'est généré qu'une fois. Nous suivons la face extérieure de G (dans une représentation propre) en commençant par le sommet v_0 dans le sens des aiguilles d'une montre et tenons une liste des étiquettes des sommets étiquetables la première fois que nous les rencontrons, dans l'ordre où nous les rencontrons, ce qui définit une liste $\mathcal{L}(\mathcal{G})$ d'étiquettes. En utilisant l'information disponible sur la symétrie du

graphe dual, d'autres points de départ (définis par un sommet initial v_0^n et un sens pour suivre le contour de G) sont considérés. Pour chacun d'eux, nous calculons les listes alternatives $\mathcal{L}^{\backslash}(\mathcal{G})$ correspondantes. L'étiquetage $\mathcal{L}(\mathcal{G})$ sera dit **canonique** si la liste correspondante est lexicographiquement plus grande que chacune des listes alternatives.

Pour éviter toute redondance lors de la génération, nous ignorons tout polyhexe dont l'étiquetage n'est pas canonique car ce polyhexe sera généré par un autre ensemble d'étiquettes qui, par contre, sera canonique.



8 autres codes potentiels peuvent être obtenus en parcourrant le graphe dans le sens trigonométrique.

Le code est 5701660240230.

Figure 6.5 – Génération du code pour identifier la symétrie : graphe orienté, liste d'étiquettes, listes concaténées et code potentiel.

6.3 Résultats numériques

6.3.1 Énumération de polyhexes planaires simplement connectés

Le résultat principal obtenu à l'aide de cette méthode est le nombre de polyhexes pour $h = 22, 23$ et 24 hexagones. Pour $h = 22$ et $h = 23$, les calculs ont été effectués séparément à Montréal et Bielefeld, tandis que les résultats pour $h = 24$ ont été effectués en partie dans chaque université. La construction des graphes duaux étant assurée par génération ordonnée en ajoutant récursivement un sommet à un graphe plus petit, nous avons séparé les tâches d'une manière similaire à l'énumération décrite dans la section 5.1. La principale différence est que lors de chaque énumération indépendante, tous les graphes duaux jusqu'à une cardinalité définie au sein du code sont construits, à la suite de quoi une partie d'entre eux est considérée en s'assurant que tous les graphes duaux sont considérés une et une seule fois. De cette manière, nous avons effectué l'énumération de tous les benzenoïdes avec $h = 24$ hexagones en traitant 10 000 cas séparément (5 000 dans chaque université).

Pour l'énumération de polyhexes planaires simplement connectés, les nombres de molécules ainsi que le temps requis sont indiqués sur le Tableau 6.1 tandis que des informations complémentaires sont données sur le Tableau 6.2, pour h entre 10 et 24. Le Tableau 6.2 nous donne les valeurs des ratios des nombres de polyhexes avec h et $h - 1$ hexagones. Ces ratios semblent relativement proches de 5 et sont croissants (4.63 pour $h = 10$ et 4.91 pour $h = 20$).

Les nombres de graphes duaux sont aussi indiqués sur le Tableau 6.2, de même que le nombre de ces graphes qui appartiennent à des classes de symétrie non triviales. Le nombre de graphes duaux semble très nettement inférieur au nombre de polyhexes

planaires et décroît proportionnellement quand h croît : il y a environ 20 benzenoïdes par graphe dual pour $h = 10$ tandis que ce nombre s'élève à 1100 pour $h = 24$. Seule une faible proportion de ces graphes comportent une symétrie non triviale, à nouveau, cette proportion décroît nettement quand h augmente : 15% des graphes duaux appartiennent à une classe de symétrie non triviale pour $h = 10$ contre environ 0.0022% pour $h = 24$. Comme nous l'avons déjà mentionné, ceci est un facteur déterminant qui explique la bonne performance de l'algorithme : dans la plupart des cas, comme le graphe dual n'a que la symétrie triviale, aucun test n'est nécessaire pour assurer la génération unique de chaque polyhexe. Les énumérations dont les temps sont donnés sur le Tableau 6.1 ont été effectués sur un Pentium II™(266 MHz) pour $h \leq 21$. Pour les plus grandes valeurs, du fait que les temps de calcul sont très longs, ces calculs ont été effectués sur un ensemble hétérogène de machines, principalement la nuit, sans quoi plusieurs années auraient été nécessaires. Le Tableau 6.1 donne aussi le nombre de molécules générées par secondes. Il est intéressant de remarquer que ce nombre croît avec h, bien que les polyhexes soient plus grands. Ceci est dû au fait que le temps perdu pour la génération est inférieur au temps qui est gagné par le fait que les tests de symétrie requis sont plus rares quand h croît.

L'isomère représenté par le polyhexe pouvant par ailleurs être défini simplement à l'aide du graphe dual correspondant, le décompte des isomères est assez rapide (à nouveau car le nombre de polyhexes correspondant au même graphe dual est élevé et croît avec h).

6.3.2 Énumération de polyhexes simplement connectés

Dans le cas où nous n'avons pas les contraintes de planarité, le décompte des polyhexes peut s'effectuer incomparablement plus rapidement puisque le test de planarité est alors omis. Le test d'automorphisme afin d'éviter la redondance n'est quant à lui

Tableau 6.1 – Nombre de polyhexes planaires, temps CPU total, nombre de polyhexes générés par seconde en moyenne, en fonction de h. (*) indique des résultats obtenus sur des ensembles de machines hétérogèges, dans ce cas, le temps CPU est donné à titre indicatif.

H	N(h)	temps CPU	polyhexes par seconde
10	30086	0,0075"	400831
11	141229	0,27"	519382
12	669584	1,05"	633764
13	3198256	4,43"	721903
14	15367577	19.00"	808509
15	74207910	1' 23,61"	887485
16	359863778	6' 18,54"	950642
17	1751594643	28' 52,24"	1011172
18	8553649747	2h 14' 18,06"	1061501
19	41892642772	10h 33' 35,81"	1101979
20	205714411986	2j 3h 3' 28"	1119182
21	1012565172403	10j 3h 15' 38"	1156242
22(*)	4994807695197	313j 12h 47' 45,03"	184383
23(*)	24687124900540	4a 350j 7h 24' 40,21"	157835
24(*)	122238208783203	12a 324j 6h	300748

requis que dans le cas où le graphe dual appartient à des classes de symmétrie non triviales. Le nombre de polyhexes associés à un graphe dual est alors donné par la formule

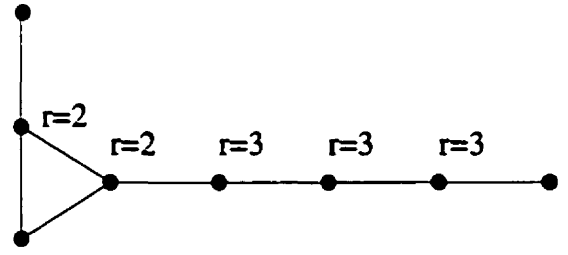
$$\prod_{v \in V} r(v), \tag{6.2}$$

comme l'illustre la figure 6.6.

Comme la très grande majorité des graphes duaux n'appartiennent qu'aux classes triviales de symétrie, cette propriété nous a permis d'effectuer le décompte des polyhexes simplement connectés jusqu'à $h = 26$, alors que les énumérations précédentes s'arrêtaient à $h = 20$ (voir section 5.3). Le nombre de polyhexes ainsi comptés s'élève à plus de 25 000 fois plus que précédemment, comme le montre le Tableau 6.3.

Tableau 6.2 – Nombre de polyhexes planaires ($N(h)$), ratio $N(h)/N(h-1)$, nombre de graphes duaux (GD), nombre de graphes duaux avec symétrie non triviale (GDS),ratio GDS/GD en pourcentages et ratio $GD/N(h)$ en fonction de h.

h	N(h)	$\frac{N(h)}{N(h-1)}$	Graphes duaux	GDS	$\frac{GDS}{GD}\%$	$\frac{N(h)}{GD}$
10	30086	4,62	1477	230	15,57	20,36
11	141229	4,69	4918	427	8,68	28,71
12	669584	4,74	16956	861	5,07	39,48
13	3198256	4,77	59494	1588	2,66	53,75
14	15367577	4,80	212364	3194	1,50	72,36
15	74207910	4,82	766753	5923	0,77	96,78
16	359863778	4,84	2796876	11963	0,42	128,66
17	1751594643	4,86	10284793	22144	0,21	170,30
18	8553649747	4,88	38096072	45009	0,11	224,53
19	41892642772	4,89	141998218	83605	0,05	295,02
20	205714411986	4,91	532301941	170059	0,03	386,46
21	1012565172403	4,92	2005638293	317341	0,01	504,86
22	4994807695197	4,93	7592441954	647393	0,008	657,87
23	24687124900540	4,94	28865031086	1210548	0,004	855,26
24	122238208783203	4,95	110174528925	2477096	0,002	1109,50



n = 2 . 2 . 3 . 3 . 3 = 108 polyhexes

Figure 6.6 – Exemple de décompte des polyhexes associés à un graphe dual sans avoir recours à la génération.

Tableau 6.3 – Nombre de polyhexes simplement connectés $N(h)$ pour $h = 21 \ldots 26$.

h	N(h)
21	2 418 927 725 532
22	13 030 938 290 472
23	70 492 771 581 350
24	382 816 374 644 336
25	2 086 362 209 298 079
26	11 408 580 755 666 756

Conclusion

Le programme **AGX** comporte plusieurs parties qui mettent en application un certain nombre de méthodes et d'algorithmes. La principale innovation est le fait de considérer la recherche de graphes extrêmes comme un problème d'optimisation combinatoire et d'utiliser la métaheuristique de *Recherche à Voisinages Variables* pour le résoudre. Nous avons illustré, par la variété de problèmes qui ont été résolus, la souplesse et la puissance de cette méthode pour la théorie des graphes. La possibilité de trouver aisément des graphes extrêmes nous a encouragés à envisager l'étude de ces graphes extrêmes comme moyen d'aborder un grand nombre de problèmes. Une autre innovation que nous proposons réside dans l'utilisation de méthodes de recherche automatique de conjectures, et dans la nature de ces méthodes. À l'aide de ces fonctionnalités, **AGX** est devenu un système assez complet pour pouvoir donner de nouvelles conjectures intéressantes de manière totalement automatique. Plus d'une trentaine de conjectures intéressantes ont été découvertes par ou à l'aide d'**AGX**. Parmi ces conjectures, près de la moitié ont été démontrées, dont certaines par une utilisation originale de propriétés liées à la programmation linéaire. Les divers outils qui composent **AGX** étant très complémentaires avec le travail du chercheur, le programme se montre aussi très adapté à une utilisation interactive et devient un collaborateur utile au chercheur en théorie des graphes.

AGX a démontré le potentiel qui réside dans les concepts qu'il utilise. Bien sûr, ce constat n'est pas une fin en soi et il reste une large place pour des améliorations dans le but que le programme soit utilisé par les chercheurs. Certaines de ces améliorations sont discutées dans la section 3.2.

Durant les derniers chapitres, nous avons appliqué de nouvelles méthodes d'énumération au problème de la génération de polyhexes. Plus important que le nombre

de polyhexes selon le nombre d'hexagones, cette étude nous a montré les gains qu'il y a à faire en utilisant les méthodes les plus apropriées. La génération ordonnée, reconnue comme une des approches les plus prometteuses quand elle est utilisable, nous a permis d'améliorer significativement les résultats connus jusque là. Toutefois, une méthode spécifique en deux phases est incomparablement plus rapide. Cette remarque quant au choix de l'algorithme reflète le compromis avec lequel nous devons toujours composer en algorithmique, et qui est très important pour les problèmes d'énumération, choisir entre la généralité et l'efficacité.

L'intégration de méthodes d'énumération au sein d'**AGX** relève de cette problématique. Nous pouvons envisager un générateur de graphes quelconques évalués par les routines d'**AGX** afin de parcourir de manière systématique l'espace des graphes jusqu'à une taille donnée. Cette possibilité est actuellement exploitée par l'intermédiaire du générateur de graphes *geng* de McKay mais se voit vite limitée par la taille des graphes. Une autre possibilité consiste à utiliser des algorithmes spécialisés en fonction des familles spécifiques de graphes à considérer. Au niveau de la performance, cette dernière approche est indéniablement meilleure mais engendre une perte de généralités. Il faudrait alors concevoir une série de routines appropriées à divers cas.

Tant par les améliorations sur le programme lui même que par les nouvelles méthodes à y ajouter, **AGX** offre un large éventail de développements possibles. En dehors de la théorie des graphes, les résultats obtenus à l'aide d'**AGX** montrent le potentiel des méthodes qu'il exploite. La généralité de ces méthodes laisse penser que nous pouvons avantageusement les adapter à d'autres problèmes combinatoires.

Bibliographie

[1] ABOAV, D., AND GUTMAN, I. (1988). *Chem. Phys. Lett. 148*, 1, 90–92. Estimation of the number of benzenoid hydrocarbons.

[2] AHO, A., HOPCROFT, J., AND ULLMAN, J. (1974). *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass.

[3] APPEL, K., AND HAKEN, W. (1977). *Illinois Journal of Math. 21*, 429–490. Every planar map is four colorable. part i. discharging.

[4] APPEL, K., AND HAKEN, W. (1977). *Illinois Journal of Math. 21*, 491–567. Every planar map is four colorable. part ii. reducibility.

[5] APPEL, K., AND HAKEN, W. (1989). *A.M.S. Contemp. Math. 98.* Every planar map is four colorable.

[6] ARAUJO, O., AND LA PEÑA, J. D. (1998). *J. Chem. Inf. Comput. Sci. 38*, 827–831. Some bounds for the connectivity index of a chemical graph.

[7] AVIS, D., AND FUKUDA, K. (1996). *Discrete Applied Mathematics 65*, 21–46. Reverse search for enumeration.

[8] BALABAN, A. (1990). Enumeration of isomers. In *Chemical Graph theory.* Abacus Press, Gordon and Breach, New York, pp. 177–234.

[9] BALABAN, A., BRUNVOLL, J., CIOLOWSKI, J., CYVIN, B., CYVIN, S., GUTMAN, I., HE, W., KNOP, J., KOVAČEVIĆ, M., ULLER, W. M., SZYMANSKI, K., TOŠIĆ, R., AND TRINAJSTIĆ, N. (1987). *Z. Naturforsch. 42a*, 863–870. Enumeration of benzenoid and coronoid hydrocarbons.

[10] BALABAN, A., AND HARARY, F. (1967). *Tetrahedron 24*, 2505–2516. Enumeration and proposed nomenclature of benzenoid cata-condensed polycyclic aromatic hydrocarbons.

[11] BALASUBRAMANIAN, K. (1990). Recent chemical applications of computational combinatorics and graph theory. In *Computational Chemical Graph Theory*, D. Rouvray, Ed. Nova Science, New York.

[12] BALASUBRAMANIAN, K., KAUFMAN, J., KOSKI, W., AND BALABAN, A. (1980). *J. Comput. Chem. 1*, 149–157. Graph theoretical characterisation and computer generation of certain carcinogenic benzenoid hydrocarbons and identification of bay regions.

[13] BELL, F. (1992). *Linear Algebra Appl. 161*, 45–54. On the Irregularity of Graphs.

[14] BERGE, C. (1970). *Graphes et Hypergraphes.* Dunod, Paris.

[15] BLAND, R. (1977). *Mathematics of Operations Research 2*, 103–107. New finite pivoting rules for the simplex method.

[16] BOLLOBAS, B., AND ERDÖS, P. (1998). *Ars Combinatorica 50*, 225–233. Graphs of Extremal Weight.

[17] BONCHEV, D., AND ROUVRAY, D. (1991). *Chemical Graph Theory, Introduction and Fundamentals.* Abacus Press, Gordon and Breach, New York.

[18] BONCHEV, D., AND ROUVRAY, D. (1992). *Chemical Graph Theory, reactivity and Kinetics.* Abacus Press, Gordon and Breach, New York.

[19] BRIGHAM, R., AND DUTTON, R. (1983). *Congressus Numeratium 39*, 337–352. Ingrid : A Software tool for Extremal Graph Research Theory.

[20] BRIGHAM, R., AND DUTTON, R. (1985). *Networks 15*, 73–107. A Compilation of Relations between Graphs Invariants.

[21] BRIGHAM, R., AND DUTTON, R. (1991). *Networks 21*, 421–455. A Compilation of Relations between Graphs Invariants. Supplement 1.

[22] BRIMBERG, J., HANSEN, P., MLADENOVIĆ, N., AND TAILLARD, E. (1997). *Les Cahiers du GERAD G-97-37*. Improvements and comparision of heuristics for solving the multisource weber problem.

[23] BRINKMANN, G. (1996). *J. Graph Theory 23*, 139–149. Fast generation of cubic graphs.

[24] BRUNVOLL, J., CYVIN, S., CYVIN, B., ZHANG, F., AND XIAOFENG, G. (1996). *Structural Chemistry 7*, 119–130. Theory of helicenic hydrocarbons : 4. further enumeration.

[25] CAPOROSSI, G., GUTMAN, I., AND HANSEN, P. (1999). *Computers and Chemistry 23*, 469–477. Variable Neighborhood Search for Extremal Graphs. 4. Chemical Trees with Extremal Connectivity Index.

[26] CAPOROSSI, G., AND HANSEN, P. (1998). *J. Chem. Inf. Comput. Sci. 38*, 610–619. Enumeration of Polyhex Hydrocarbons to $h = 21$.

[27] CAPOROSSI, G., AND HANSEN, P. (2000). *Discrete Mathematics 212*, 29–44. Variable Neighborhood Search for Extremal Graphs. 1. The Autographix System.

[28] CHEN, P.-C., HANSEN, P., AND JAUMARD, B. (1991). *Operations Research Letters 10*, 403–409. Online and offline vertex enumeration by adjacency lists.

[29] CHUNG., F. (1988). *Journal of Graph Theory 12*, 229–235. The average distance and the independence number.

[30] CHVÁTAL, V. (1983). *Linear Programming.* Freeman, New York.

[31] CVETKOVIĆ, D. (1983). *Publications de l'Institut Mathématique 47*, 29–33. Discussing graph theory with a computer ii, theorems suggested by the computer.

[32] CVETKOVIĆ, D. (1983). *Publications de l'Institut Mathématique de Beograd 48*, 37–47. Discussing graph theory with a computer iii, man-machine theore m proving.

[33] CVETKOVIĆ, D. (1983). *Proceedings of the fourth Yugoslav Seminar on Graph Theory, Nov i Sad*, 43–68. Discussing graph theory with a computer iv, knowledge organizat ion and examples of theorem proving.

[34] CVETKOVIĆ, D. (1988). *Bulletin de l'academie des Sciences et des Arts T. XCVII*, 51–70. Discussing graph theory with a computer, vi : Theorems proved wi th the aid of the computer.

[35] CVETKOVIĆ, D. (1995). *Linear and Multilinear Algebra 39*, 109–132. Star partition and the graph isomorphism problem.

[36] CVETKOVIĆ, D., DOOB, M., AND SACHS, H. (1980). *Spectra of Graphs - Theory and Applications.* Academic Press New York.

[37] CVETKOVIĆ, D., AND GUTMAN, I. (1985). *Comput. Chem. 7*, 640–644. The computer system graph : a useful tool in chemical graph theory.

[38] CVETKOVIĆ, D., AND PEVAC, I. (1988). *Artificial Intelligence 35*, 1–23. Man-machine theorem proving in graph theory.

[39] CVETKOVIĆ, D., ROWLINSON, P., AND SIMIĆ, S. (1997). *Eigenspaces of graphs.* Cambridge University Press, Cambridge.

[40] CVETKOVIĆ, D., AND SIMIĆ., S. (1994). *Bulletin de l'Academie Serbe des Sciences et des Arts T CVII*. Graph theoretical results obtained by the support of the expert system "graph".

[41] CVETKOVIĆ, D., SIMIĆ, S., CAPOROSSI, G., AND HANSEN, P. (1998). *Les cahiers du GERAD G-98-66*. Variable Neighborhood Search for Extremal Graphs. 3. On the Largest Eigenvalue of Color–Constrained Trees. à parraître dans Linear and Multilinear Algebra.

[42] CYVIN, B., BRUNVOLL, J., AND CYVIN, S. (1992). *Topics in Current Chemistry 162*, 65–180. Enumeration of benzenoid systems and other polyhexes.

[43] DIAS, J. (1987). Benzenoid hydrocarbons. In *Handbook of Polycyclic Hydrocarbons*. Elsevier, Amsterdam.

[44] DIAS, J. (1993). *Molecular Orbital Calculation Using Chemical Graph Theory*. Springer-Verlag Berlin.

[45] DIMITRIJEVIĆ, D. C. V., AND MILOSAVLJEVIĆ, M. (1996). *Variations on the Travelling Salesman Theme*. Libra Produkt, Belgrade.

[46] DOBRYNIN, A. (1994). *Graph Theory Notes of New York XXVII :8*, 50–54. Graphs with palindromic wiener polynomials.

[47] DOOB, D. C. M., AND SACHS, H. (1995). *Spectra of Graphs*. Johann Ambrosius Barth, Heidelberg.

[48] DUTTON, R. B. R., AND GOMEZ, F. (1989). *Journal of Symbolic Computation 7*, 163–177. Ingrid : A graph invariants manipulator.

[49] DYER, M. (1983). *Mathematics of Operations Research 8*, 381–402. The complexity of vertex enumeration methods.

[50] ET AL, B. O. (1993). Cabri graph 3.1 reference guide.

[51] FAJTLOWICZ, S. (1987). *Congressus Numerantium 60*, 187–197. On conjectures of graffiti - ii.

[52] FAJTLOWICZ, S. (1988). *Discrete mathematics 72*, 113–118. On conjectures of graffiti.

[53] FAJTLOWICZ, S. (1988). *Congressus Numerantium 66*, 23–32. On conjectures of graffiti - iii,.

[54] FAJTLOWICZ, S. (1990). *Congressus Numerantium 70*, 231–240. On conjectures of graffiti - iv.

[55] FAJTLOWICZ, S. (1997). Written on the wall. version 03-1997 (updated regularily).

[56] FAJTLOWICZ, S. (1998). Written On the Wall. version 05-1998 (updated regularily).

[57] FARADZEV, I. Constructive enumeration of combinatorial objects. In *Problemes Combinatoires et theorie des Graphes Colloque Internat. CNRS 260.* (1978), pp. 131–135.

[58] FARADZEV, I. (1978). Generation of nonisomorphic graphs with given degree sequence (russian). In *Algorithmic Studies in Combinatorics.* Nauka, Moscow, pp. 11–19.

[59] FARRELL, E., AND KENNEDY, J. (1992). Graphs with palindromic matching and characteristic polynomials ii non-equible palindromic graphs. preprint.

[60] FARRELL, E., KENNEDY, J., QUINTAS, L., AND WAHID, S. (1992). *Vishwa Internat. J. Graph Theory 1*, 59–76. Graphs with palindromic matching and characteristic polynomials.

[61] FAYYAD, U., PIATETSKY-SHAPIRO, G., AND SMYTH, P. (1996). *Comm. of the ACM 39*, 27–34. The kdd process for extracting useful knowledge from volumes of data.

[62] FEIGENBAUM, E., AND BUCHANAN, B. (1993). *Artificial Intelligence 59*, 233–240. Dendral and meta-dendral : Roots of knowledge systems and expert system applications.

[63] FLAVITSKY, F. (1871). *J. Russ Chem. Soc 3*, 160.

[64] GODSIL, C., AND MCKAY, B. (1978). *Bull. Austral. Math. Soc. 18*, 21–28. A new graph product and its spectrum.

[65] GUO, X., HANSEN, P., AND ZHENG, M. (1998). *Les cahiers du GERAD G-99-37*. Boundary Uniqueness of Helicenes.

[66] GUTMAN, I. (1985). *Journal of the Serbian Chemical Society 50*, 451–455. Topological properties of benzenoid systems. xli. carbon-carbon bond types and connectivity indices of benzenoid hydrocarbons.

[67] GUTMAN, I. (1986). *Zeitschrift für Physikalische Chemie (Leipzig) 267*, 1152–1158. A regularity for the boiling points of alkanes and its mathematical modeling.

[68] GUTMAN, I. (1992). *Graph Theory Notes of New York XXIII :3*, 21–24. Independant vertex palindromic graphs.

[69] GUTMAN, I. (1992). *Topics in Current Chemistry 162*, 29–63. Total π-electron energy of benzenoid hydrocarbons.

[70] GUTMAN, I. (1993). *Graph Theory Notes of New York XXIV :6*, 51–56. A contribution to the study of palindromic graphs.

[71] GUTMAN, I. (1993). *Graph Theory Notes of New York XXV :2*, 13–18. Some properties of the wiener polynomial.

[72] GUTMAN, I., AND CYVIN, S. (1989). *Introduction to the Theory of Benzenoid Hydrocarbons*. Springer-Verlag.

[73] GUTMAN, I., ESTRADA, E., AND IVANCIUC, O. (1999). Some properties of the wiener polynomial of trees. To Appear in Graph Theory Notes of New York.

[74] GUTMAN, I., AND POLANSKY, O. (1986). *Mathematical Concepts in Organic Chemistry*. Springer-Verlag Berlin.

[75] HANSEN, P., LEBATTEUX, C., AND ZHENG, M. (1996). *Theochem, Journal of Molecular Structures 363*, 237–247. The boundary-edges code for polyhexes.

[76] HANSEN, P., AND MALDENOVIĆ, N. (1998). An introduction to variable neighborhood search. In *Meta-Heuristics : Advances and Trends in Local Search Paradigms for Optimization*, V. S., M. I., O. I.H., and R. C., Eds. Kluwer, Boston, pp. 433–458.

[77] HE, W., HE, W., WANG, Q., BRUNVOLL, J., AND CYVIN, S. (1988). *Z. Naturforsch. 43a*, 693–694. Supplement to Enumeration of Benzenoid and Coronoid Hydrocarbons.

[78] HENSON, R., WINDLINX, K., AND WISWESSER, W. (1975). *Comput. Biomed. Res. 8*, 53–71. Lowest order computer-oriented "ring-index" diagrams-verifying correct orientation of fused hexagonal ring systems.

[79] HERNDON, W., AND BRUCE, A. (1987). *Stud. Phys. Theor. Chem. 51*, 491–513. Perimeter code for benzenoid aromatic hydrocarbons.

[80] HERNDON, W., NOWAK, P., DALLAS, A., CONNOR, A., AND LIN, P. (1992). *J. Am. Chem. Soc. 114*, 41–47. Empirical model calculation for thermodynamic and structural proprties of condensed polycyclic aromatic hydrocarbons.

[81] HOSOYA, H. (1988). *Discrete Appl. Math. 19*, 239–257. On some counting polynomials in chemistry.

[82] HOSOYA, H. (1990). Some recent advances in counting polynomials in chemical graph theory. In *Computational Chemical Graph Theory*, D. Rouvray, Ed. Nova Science, New York.

[83] KENNEDY, J. (1992). *Graph Theory Notes of New York XXII :8*, 27–32. Palindromic graphs.

[84] KIER, L., AND HALL, L. (1976). *Molecular Connectivity in Chemistry and Drug Research.* Academic Press, New York.

[85] KIER, L., AND HALL, L. (1986). *Molecular Connectivity in Structure–Activity Analysis.* Wiley, New York.

[86] KNOP, J., MÜLLER, W., AND SZYMANSKI, K. (1990). Computer-oriented molecular codes. In *Computational Chemical Graph Theory*, D. Rouvray, Ed. Nova Science, New York.

[87] KNOP, J., MÜLLER, W., SZYMANSKI, K., AND TRINAJSTIĆ, N. (1990). *J. Chem. Inf. Comput. Sci. 30*, 159–160. Use of small computers for large computations : Enumeration of polyhex hydrocarbons.

[88] KNOP, J., SZYMANSKI, K., JERICEVIĆ, Z., AND TRINAJSTIĆ, N. (1983). *J. Comput. Chem. 4*, 23–32. Computer enumeration and generation of benzenoid hydrocarbons and identification of bay regions.

[89] KRAUS, D. C. L., AND SIMIĆ, S. (1981). *Univ. Beograd Publ. Elektrotehn. Fak*, 100–104. Discussing graph theory with a computer i. implementation of graph theoretic algorithms.

[90] LANGLEY, P. (1979). *Proceedings of the International Joint Conference on Artificial Intelligence*. Rediscovering physics with Bacon 3.

[91] LANGLEY, P. (1981). *Cognitive Science 5*, 31–54. Data-driven discovery of physical laws.

[92] LANGLEY, P. (1998). *Proceeding of the First International Conference on Discovery Science*. The computer-aided discovery of scientific knowledge.

[93] LANGLEY, P., SIMON, H., BRADSHAW, G., AND ZYTKOW, J. (1987). *Scientific Discovery. Computational Exploration of the Creative Process*. MIT Press, Cambridge, Massachusetts.

[94] LENAT, D. (1977). *Proceedings of the fifth International Joint Conference on Artificial Intelligence*, 833–842. Automated theory formation in mathematics.

[95] LOVÁSZ, L., AND PELIKAN, J. (1973). *Periodica Math. Hung. 3*, 175–182. On the eigenvalues of trees.

[96] MAHÉO, O. F. M., AND SACLÉ, J.-F. (1991). *Journal of Graph Theory 15*, 39–64. On the residue of a graph.

[97] McClelland, B. (1971). *J. Chem. Phys 54*, 640–643. Properties of the latent roots of a matrix : Estimation of π-electron energies.

[98] McKay, B. (1998). *Journal of Algorithms 26*, 306–324. Isomorph-free exhaustive generation.

[99] Mercier, C., Sobel, Y., and Dubois, J.-E. (1990). Darc/pelco method : A topological tool for QSAR search and its reliable predictive capability. In *Chemical Graph theory*. Abacus Press, Gordon and Breach, New York, pp. 199–258.

[100] Mladenović, N., and Hansen, P. (1997). *Computers and Operations Research 24*, 1097–1100. Variable neighborhood search.

[101] Müller, W., Szymanski, K., and Knop, J. (1989). *Croat. Chem. Acta 62*, 481–483. On counting polyhex hydrocarbons.

[102] Newell, A., Shaw, J., and Simon, H. (1957). *Proceedings of the 1957 Western Joint Computer Conference*, 218–230. Empirical explorations of the logic theory machine.

[103] Newell, A., Shaw, J., and Simon, H. (1959). *Proceedings of the International Conference on Information Processing*, 256–264. Report on a general problem-solving program.

[104] Nikolić, S., Trinajstić, N., Knop, J., Müller, W., and Szymanski, K. (1990). *J. Math. Chem. 4*, 357–375. On the concept of the weighted spanning tree of dualist.

[105] OSMAN, I., AND LAPORTE, G. (1996). *Annals of Operations Research 63*, 513–628. Metaheuristics : a bibliography.

[106] PONTIER, J., DUFOUR, A., AND NORMAND, M. (1990). *Le modèle Euclidien en analyse des données.* Éditions de l'Université Libre de Bruxelles, Bruxelles.

[107] RANDIĆ, M. (1975). *journal of the American Chemical Society 97*, 6609–6615. On characterization of molecular branching.

[108] READ, R. (1978). *Annals Discrete Maths 2*, 107–120. Every one a winner.

[109] REEVES, C. (1990). *Modern Heuristic Techniques for Combinatorical problems.* Blackwell, Oxford.

[110] ROBERTSON, N., SANDERS, D., SEYMOUR, P., AND THOMAS, R. (1997). *Journal of Combinatorial theory, Ser. B 70*, 2–44. The four-colour theorem.

[111] SIĆ, R. T., MAŠULOVIĆ, D., STOJMENOVIĆ, I., BRUNVOLL, J., CYVIN, B., AND CYVIN, S. (1995). *J. Chem. Inf. Comput. Sci. 35*, 181–187. Enumeration of Polyhex Hydrocarbons to $h = 17$.

[112] SIMON, H., VALDÉZ-PERÉZ, R., AND SLEEMAN, D. (1997). *Artificial Intelligence 91*, 177–181. Scientific Discovery and Simplicity of Method.

[113] STOJMENOVIĆ, I., SIĆ, R. T., AND DOROSLOVACKI, R. Generating and Counting Hexagonal Systems. In *Graph Theory, Proceedings of the sixth Yougoslav Seminar on Graph Theory* (1985).

[114] SWANSON, D., AND SMALHEISER, N. (1997). *Artificial Intelligence 91*, 183–203. An interactive system for finding complementary litteratures : A stimulus to scientific discovery.

[115] TRINAJSTIĆ, N. (1992). *Chemical Graph Theory.* CRC Press, Boca Raton.

[116] TRINAJSTIĆ, N., NIKOLIĆ, S., KNOP, J., MÜLLER, W., AND K., K. S. (1991). *Computational Chemical Graph Theory.* Ellis Horwood.

[117] ULLER, W. M., SZYMANSKI, K., KNOP, J., NIKOLIĆ, S., AND TRINAJSTIĆ, N. (1990). *J. Comput. Chem. 11*, 223–235. On the enumeration and generation of polyhex hydrocarbons.

[118] VALDÉS-PERÉZ, R. (1999). *Artificial Intelligence 107*, 335–346. Principles of Human Computer Collaboration for Knowledge Discovery in Science.

Annexe A

Utilisation d'AutoGraphiX

Pour que l'utilisation du programme soit conviviale et facile, nous avons établi que les fonctionnalités suivantes doivent être présentes dans le système.

- Saisie des données relatives à la tâche demandée, le problème à traiter et des informations complémentaires permettant d'améliorer le traitement et d'indiquer la manière dont l'usager désire lire les résultats.
- Communication des résultats obtenus : génération de rapports par exemple.
- Interface graphique permettant une étude interactive du problème.
- Représentation adéquate des graphes trouvés.

A.1 Définition d'un problème, le fichier de paramètres

Pour traiter un problème, du nom **monPb**, nous lançons le programme par la commande "agx monPb". Il lit alors le fichier appelé **monPb.par** dans lequel sont donnés le problème ainsi que les valeurs de certain paramètres. De ce fichier, seules les instructions entre les mots clés **BeginFile** et **EndFile**. Par ailleurs, toute portion de texte comprise entre "/*" et "*/", considérée comme commentaire, est ignorée.

Le fichier de paramètres doit au moins comporter la définition du problème à traiter qui est exprimée sous la forme d'expressions d'invariants graphiques et d'opérateurs tels que :

1. $+ - */$ les quatre opérateurs classiques,
2. $ABS(x)$ la valeur absolue,
3. $SQRT(x)$ la racine carrée,
4. $x \wedge y$ élévation de x à la puissance y,
5. $LOG(x)$ le logarithme,
6. $EXP(x)$ l'exponentielle,
7. $FLOOR(x)$ le plus grand entier inférieur ou égal à x,
8. $CEIL(x)$ le plus petit entier supérieur ou égal à x,
9. $ROUND(x)$ l'entier le plus proche de x (dans le cas ou x est un multiple de $\frac{1}{2}$, c'est $x - \frac{1}{2}$ qui est choisi.
10. $SUP(x; y)$ le plus grand des nombres x et y (notez ici que le séparateur doit impérativement être " ;"),
11. $INF(x; y)$ le plus petit des nombres x et y.

La description du problème d'optimisation à résoudre doit impérativement débuter par le mot clé **Problem** et finir par **End**. La première ligne décrit la fonction-objectif précédée du mot clé **Min** : ou **Max** : selon le type de problème. Si le problème est contraint, le mot clé **SubjectTo** doit être inscrit après la fonction-objectif. Les contraintes sont ensuite données sous forme d'égalités ou d'inégalités dont le second membre doit être une constante.

Les contraintes sont traitées par relaxation lagrangienne, leurs violations sont multipliées par un poids avant d'être ajoutées à la fonction-objectif. Par défaut, le poids associé à chaque contrainte est de 1000, mais l'utilisateur a la possibilité de le changer en indiquant la valeur souhaitée entre crochets juste après le membre droit de la contrainte. Un graphe sera identifié comme réalisable si la valeur de sa fonction-objectif est comprise entre -**Range** et **Range**. **Range** est fixé à 100 par défaut mais

peut être modifié pour la valeur réelle **n** par la commande **Range n** dans le fichier de paramètres. Il revient à l'utilisateur de s'assurer que la valeur **Range** est cohérente avec les poids associés à chaque contrainte. Étant donné que les poids associés aux contraintes sont fixés a priori, il est conseillé de ne pas utiliser dans les contraintes des valeurs continues mais plutôt des valeurs entières sans quoi leur respect sera difficilement garanti.

Il est possible, lors de la définition du problème d'utiliser des contraintes disjonctives. Si le mot clé **OR** est donné entre deux contraintes, seule la plus petite des pénalités sera considérée. Par conséquent, aucune pénalité ne sera considérée dès l'instant qu'une des contraintes est respectée.

Certains paramètres peuvent être ajustés dans le fichier afin d'améliorer le comportement de l'optimiseur selon les spécificités du problème, parmi lesquels **Infty** donne une valeur au dessus de laquelle tout nombre est traité comme l'infini. Par défaut, cette valeur est fixée à 10^6. Pour des raisons de traitement numérique, il est possible aussi de fixer **Eps**, une petite valeur telle que nous considérons que si $-Eps \leq x - y \leq Eps$, alors $x = y$.

Pour certains problèmes avec contraintes et dont la fonction objectif est assez longue à calculer, il peut être utile pour accélérer la résolution, de débuter la phase d'optimisation avec un graphe particulier. Il est dans ce but possible de préciser à **AGX** le type de graphe initial à générer par la commande : **InitalGraph type** où **type** peut être **Tree** pour générer un arbre, **Path** pour un chemin, **Star** pour une étoile, **Circle** pour un cercle, **Complete** pour un graphe complet, ou tout simplement **Graph** pour un graphe aléatoire auquel cas il est peut être souhaitable de changer la densité du graphe initial à 50% par défaut pour **x**% par la commande **Density x**. Le paramètre **NbNod n** indique enfin le graphe initial comportera **n** sommets.

Si ces fonctions suffisent à définir un graphe initial réalisable, il est possible ensuite de passer directement à l'optimisation proprement dite en donnant le mot clé **NoInit**.

Dans le cas contraire, **AGX** résoudra d'abord un programme initial défini exactement comme le problème lui même, à l'exception du premier mot clé qui sera **Init** au lieu de **Problem**, notons que si aucun problème initial n'est explicitement donné, le programme utilisera le problème principal lors de la phase d'initialisation. Du point de vue de l'algorithme, une seule descente locale sera effectuée lors de la phase initialisation, bien que celle-ci puisse utiliser différentes transformations.

La manière la plus sûre d'obtenir les meilleurs résultats lors d'une descente locale avec **AGX** est d'utiliser toutes les transformations successivement. Le problème est alors que la descente sera longue. D'un autre côté, n'utiliser que les transformations rapides donnera des résultats plus rapidement mais ces derniers seront moins bons, sinon réellement mauvais. Il est alors important de définir une stratégie de choix des transformations à utiliser. Si la phase d'initialisation a permis de trouver un graphe réalisable, il sera possible d'identifier des transformations ne permettant pas conserver la réalisabilité (par exemple l'ajout d'arêtes tandis que le nombre de celles-ci est fixé). De telles transformations sont donc vouées à l'échec lors de l'optimisation. Il est donc possible de les enlever de la liste des transformations à utiliser sans risque de perdre au niveau de la qualité des graphes obtenus. Si par exemple nous nous intéressons à des arbres, dès qu'un d'entre eux est obtenu, il n'est pas nécessaire d'essayer d'améliorer la fonction-objectif par des transformations modifiant systématiquement le nombre d'arêtes sans changer le nombre de sommets. La stratégie est dans ce cas plutôt de trouver le plus rapidement possible un arbre et d'éviter ensuite toute transformation ne conservant pas cette caractéristique. Selon les problèmes étudiés, l'utilisateur peut utiliser ses connaissances pour sélectionner lui même les transformations à utiliser en les indiquant dans l'ordre par une liste débutant par le mot **BeginDesc** et finissant par **End**. Pour la phase d'initialisation, **BeginDesc** sera remplacé par **BeginInit**. Dans le cas où la liste de transformations n'est pas donnée explicitement, le programme utilisera toutes celles qui sont disponibles lors

de la phase d'initialisation. Si le graphe trouvé à la fin de cette phase est réalisable, chaque type transformation sera ensuite testé pour l'itération suivante dans le but d'évaluée sa performance. Certaines transformations seront alors éliminées si elles ne permettent pas de conserver les propriétés désirées, tandis que les autres seront triées par ordre décroissant de performance. Ce tri a pour but que la majeure partie du travail se fasse en priorité avec les transformations les plus appropriées. Si le mot clé **VnsCentered** est présent dans le fichier de paramètres, la recherche se fera en recommençant par le début de la liste des transformations à chaque fois que la meilleure solution est améliorée, sans quoi le programme continuera à suivre la liste de manière circulaire. Cette possibilité vise à tirer un meilleur profit des transformations les plus appropriées. Lors de la descente locale à l'aide d'une transformation donnée, il y a plusieurs options possibles. Nous pouvons par exemple décider d'effectuer le premier mouvement intéressant rencontré alors que le voisinage du graphe courant est examiné, ou nous pouvons décider de les considérer tous avant de sélectionner le meilleur. Entre ces deux extrêmes, nous avons la lattitude de sélectionner la meilleure des **n** premières améliorations rencontrées. Ce type de stratégies se définit en ajoutant **Improve n** après le nom du voisinage dans la liste des voisinages donnée par l'utilisateur. Si un voisinage est très long à explorer, nous pouvons chercher à l'utiliser seulement pour sortir d'un optimum local commun aux voisinages plus rapides. Dans ce cas, nous passerons au voisinage suivant dans la liste dès qu'un nombre limité **n** d'améliorations a été obtenu avec ce dernier. Il suffit pour cela de donner le mot clé **NbIt n** après le nom du voisinage concerné (avant ou après l'option **Improve n**). Certains voisinages enfin peuvent être utilisés avec des paramètres spéciaux. **Specs x** peut alors être utilisé d'une manière similaire à **Improve** ou **NbIt**. Pour la descente appelée **Local** consistant à ajouter ou retirer une arête à la fois par exemple, nous pouvons ajouter **Specs -1** pour se restreindre à des retraits d'arêtes, ou **Specs 1** pour en ajouter seulement.

La partie perturbations de la recherche à voisinages variables comporte quelques paramètres. Nous pouvons ainsi donner la taille minimale x de la perturbation en

ajoutant la commande **VnsMin x**, ou la taille maximale de la perturbation y, qui sert de critère d'arrêt de la recherche avec la commande **VnsMax y**. Il est encore possible de faire plusieurs essais (**x** par exemple) pour chaque taille de perturbation avant de changer cette taille en ajoutant la commande **VnsTry x**. Si nous désirons incrémenter la perturbation d'une valeur différente de 1 à chaque fois, nous pouvons entrer la commande **VnsStep x**. Il est certains types de problèmes pour lesquels la perturbation classique n'est pas appropriée. Si le nombre d'arêtes par exemple est fixé, le voisinage consistant à ajouter ou retirer une arête est peu pertinent. Nous pouvons alors utiliser une autre perturbation consistant à déplacer une arête. Changer la perturbation par défaut pour celle-ci s'effectue en ajoutant la commande **VnsType Move** au fichier de paramètres. Il est enfin parfois intéressant de répéter plusieurs fois l'optimisation à partir de graphes initiaux différents, pour ce faire, nous ajoutons la ligne **MaxStart x**.

Dans le cas où nous voulons faire une recherche paramétrique, nous pouvons lancer **AGX** avec une ou deux boucles du type **for PARAM = EXP1 to EXP2 step x** où **PARAM** est un invariant, en général **NBNOD** pour la première boucle et **NBARC** pour la seconde. Le paramètre peut aussi bien être un paramètre existant ou un paramètre symbolique intervenant par ailleurs dans les contraintes. Dans le cas ou le paramètre est un invariant, une contrainte est associée à chaque boucle avec un poids de 100000 pour la première boucle, et 10000 pour la seconde. Si ces valeurs ne sont pas appropriées au problème, il est possible de changer le nom du paramètre pour le changer en paramètre symbolique qui peut ensuite intervenir dans une contrainte d'égalité fixant la valeur du paramètre choisi, bien que le poids soit alors celui de cette dernière contrainte. De cette manière, dans l'exemple donné dans l'annexe A.5,
for PENDING = 2 to FLOOR((2*NBNOD-4)/3+2) step 1 est changé pour
for P = 2 to FLOOR((2*NBNOD-4)/3+2) step 1
et **P - PENDING = 0 [600]**,
ce qui réduit le poids associé au paramètre **PENDING** de 10000 à 600.

Il est à noter que si **NBNOD** est fixé par la paramétrisation, il prévaut sur l'option **NbNod x** exposée plus tôt. **EXP1** et **EXP2** sont des expressions écrites sous le même format que la fonction objectif ou le membre gauche de n'importe quelle contrainte. Un graphe aléatoire est alors généré pour évaluer les expressions. Il convient donc de prendre garde à l'utilisation de ces expressions qui peuvent être très hasardeuses. Il est conseillé qu'elles soient exclusivement basées sur des valeurs numériques et fonctions du nombre de sommets **NBNOD**, seules valeurs alors certaines. Si **step x** est omis, la valeur de l'incrément est assumée à 1.

Lorsqu'**AGX** est utilisé en mode paramétrique, selon qu'un ou deux paramètres sont utilisés, le fichier "monPb-p1.cmb" ou "monPb-p1-p2.cmb" est créé pour chaque valeur des paramètres utilisés et comportent une description des graphes pour les paramètres **p1** et, le cas échéant **p2** dans le format "combinatorica", comportant une ligne pour chaque sommet, dont le premier nombre est le numéro du sommet suivi d'un couple de valeurs réelles comprises entre 0 et 1 donnant les coordonnées de ce sommet, enfin sont donnés sous forme d'une simple liste les sommets adjacents. Si le mot clé **AutoSavePs** est présent, chaque graphe trouvé est aussi sauvegardé comme une image en PostScript avec comme extension ".ps" au lieu de ".cmb". Si en plus, le mot clé **SaveCaractPs** est présent dans le fichier de paramètres, le graphe en format PostScript sera accompagné de la valeur de quelques invariants.

Un exemple de fichier de paramètres est donné en annexe.

A.2 L'interface graphique

Puisque le seul dessin du graphe n'est en général pas suffisant pour tirer les conclusions cherchées, **AGX** dans son **mode interactif** offre un certain nombre de fonctionnalités supplémentaires. Lorsque la commande **Interactive n** ou **n** est un entier

est présente dans le fichier de paramètres, la valeur donnée à **n** indique les événements qui justifient de lancer le mode interactif. Quand **n** vaut 5, il est lancé pour chaque optimum local rencontré. Pour une valeur de **n**$\geq$ 4, il sera lancé au cours de l'optimisation, chaque fois que la solution est améliorée, afin d'observer l'évolution des graphes et déceler certaines caractéristiques du problème pouvant nous guider vers de meilleures solutions ou pouvant aider à les trouver plus rapidement. Quand **n** $\geq$ 3, il sera aussi appelé pour le graphe obtenu après la phase d'initialisation. Quand **n** $\geq$ 2, il est lancé pour chaque graphe supposé extrême trouvé, ainsi que pour le graphe initial généré aléatoirement. Pour **n** $\geq$ 1, il sera appelé à la fin de l'optimisation. Si cette dernière est paramètrique, c'est depuis le **mode d'étude paramétrique** qu'il sera invoqué.

A.2.1 Le mode interactif

L'interface en mode interactif est présentée en figure A.1 et se compose de 5 parties. La partie centrale donne une représentation de graphe modifiable par déplacement des sommets à l'aide de la souris. La partie de gauche donne les valeurs d'un certain nombre d'invariants graphiques choisis parmi une liste de manière interactive, ou depuis le fichier de paramètres. La partie de droite donne le cas échéant une liste des familles auxquelles le graphe appartient ainsi que la taille du stable maximum, de la clique maximum et leur nombre. La fenêtre en dessous du graphe peut ou bien indiquer la valeur de la fonction-objectif ainsi qu'une valeur mémorisée interactivement, dans le but d'évaluer rapidement l'effet de certaines modifications que l'utilisateur désire observer, ou bien donner des informations spécifiques à un sommet quand la souris s'en approche.

La partie supérieure, la barre des boutons est composée comme suit :

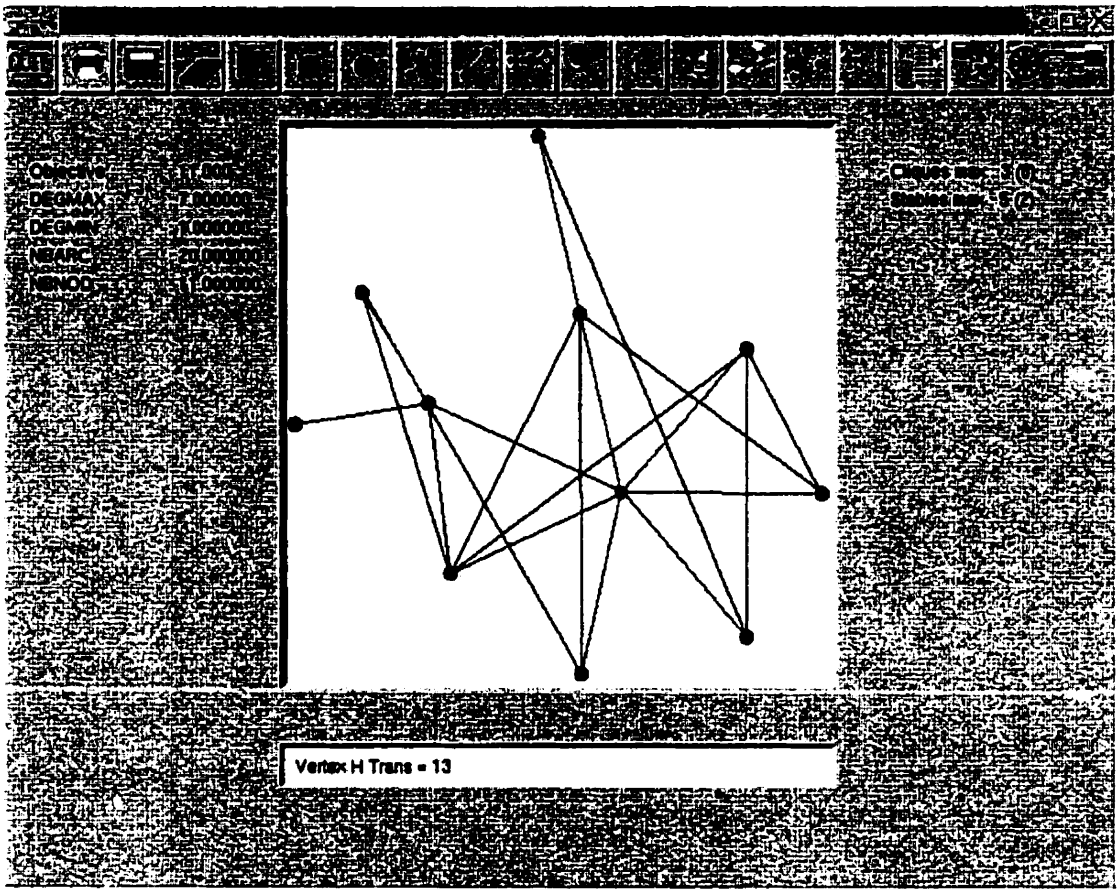


Figure A.1 – Fenêtre d'AutoGraphiX en mode interactif

Quitter le programme.

Imprimer le graphe ou la courbe qui est dans la zone centrale.

Sauvegarder le graphe sous forme PostScript ou combinatorica.

Lancer l'aide contextuelle.

Afficher le problème.

Supprimer un sommet.

Ajouter un sommet.

Supprimer une arête.

Ajouter une arête.

Afficher le complément du graphe au lieu du graphe original.

Agrandir une région.

Mettre en évidence les arêtes critiques.

Fixer le degré de certains sommets.

Interdire le retrait ou l'ajout d'une arête lors de l'optimisation.

Changer la règle de coloration des sommets.

Positionner les sommets automatiquement.

Sélectionner les invariants à afficher.

Mettre la valeur de l'objectif en mémoire.

Poursuivre l'optimisation si elle n'est pas complétée.

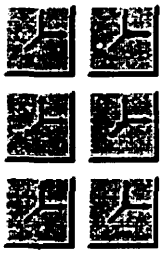
A.2.2 Le mode paramétrique

Par rapport au mode interactif, la structure de la fenêtre principale change un peu. Tout d'abord, si la barre des boutons reste affichée, seuls les boutons , ,

et restent actifs, mais la description du graphe ainsi que les valeurs de certains invariants devenues inutiles sont remplacées par des nouveaux boutons, comme le montrent les figures A.3 et A.4.

Ces boutons sont répartis en deux groupes. Les boutons à gauche correspondent à des fonctions permettant de changer la manière dont la courbe est représentée. Les fonctions des boutons de gauche sont les suivantes :

Modification de l'angle que forme le second paramètre avec le premier sur la représentation en trois dimensions.



Changement de l'échelle selon chaque axe.

Permet de déplacer l'ensemble de la courbe.

Il est à noter que certains dessins ne sont pas exactement identiques pour la représentation de la courbe en deux dimensions (un paramètre) et la courbe en trois dimensions (deux paramètres).

Les fonctions associées aux boutons de droite sont les suivantes :

Affichage des lignes joignant deux points correspondant à des paramètres successifs.

Représentation de chaque point par une ligne verticale indiquant la valeur de l'objectif.

Échange des axes correspondant au premier et second paramètre sur la représentation.

Réinitialisation de la représentation à la configuration par défaut.

Lorsqu'un de ces boutons est appuyé, cliquer sur un point représentant un graphe lance le module d'analyse paramétrique à un paramètre sur le premier (ou le second selon le bouton appuyé) paramètre, l'autre étant fixé à la valeur pour le graphe courant.

Demande à l'utilisateur une fonction qui sera affichée dans la zone principale, comme comparaison avec la courbe étudiée.

Ouvre la fenêtre de sélection d'invariants représentée sur la figure A.2 et recherche les relations affines qui sont respectées entre ces invariants à l'aide le l'algorithme décrit dans la section 1.4.3.

Lance le mode de défilement des graphes obtenus, il suffit alors d'appuyer sur le bouton pour travailler sur les graphes les uns après les autres sans passer par la courbe générale.

Dans le cas oú l'étude paramétrique porte sur un seul paramètre, certains des boutons mentionnés plus tôt ne sont pas pertinents. En revanche le bouton qui apelle le calcul de l'enveloppe convexe inférieure (si le problème est un problème de minimisation) ou supérieure (dans le cas contraire) n'est disponible que si l'étude porte sur un seul paramètre. En plus d'afficher les équations des droites représentant cette enveloppe convexe, cette fonction en dessine une représentation sur la fenêtre principale.

Dans le cas ou une équation est vérifiée comme égalité par plus de deux graphes, elle est considérée comme information importante et la droite comme son équation sont représentées d'une couleur différente.

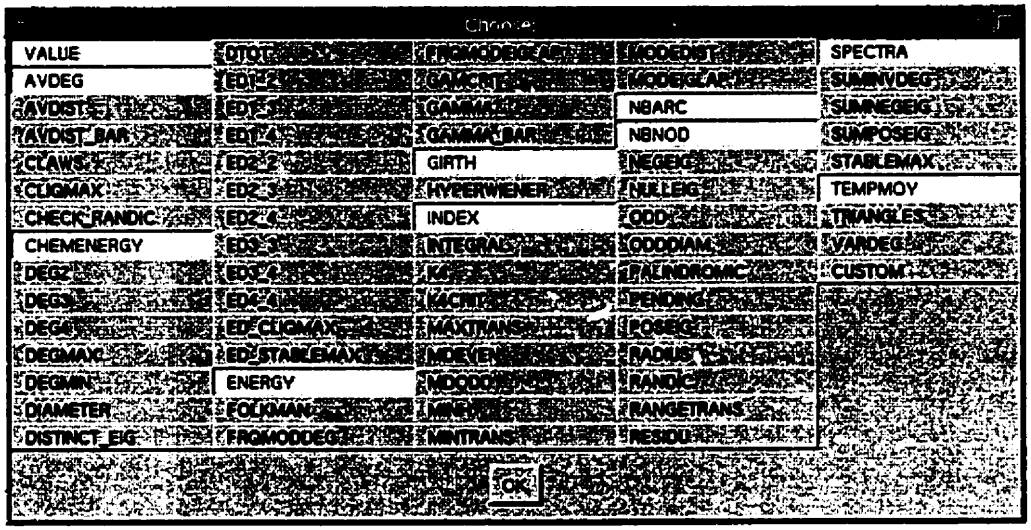


Figure A.2 – Fenêtre d'AutoGraphiX pour la sélection d'invariants

Le mode paramétrique permet d'étudier les valeurs extrêmes trouvées pour l'objectif en fonction des paramètres du problème. En déplaçant la souris dans la fenêtre centrale, nous voyons dans la fenêtre du bas les valeurs des paramètres et de la fonction-objectif correspondant à la solution pointée, laquelle est mise en valeur par un point bleu sur la fenêtre centrale. Il est alors possible de passer en mode interactif pour visualiser le graphe correspondant en cliquant avec la souris.

A.3 La génération de rapport

Après la résolution d'un problème de manière paramétrique, **AGX** peut créer un fichier PostScript comportant des représentations des graphes obtenus en taille réduite. L'échelle utilisée alors dépend de la commande **CompactPs X Y** dans le fichier de paramètres, où **X** et **Y** sont respectivement le nombre de graphes sur chaque

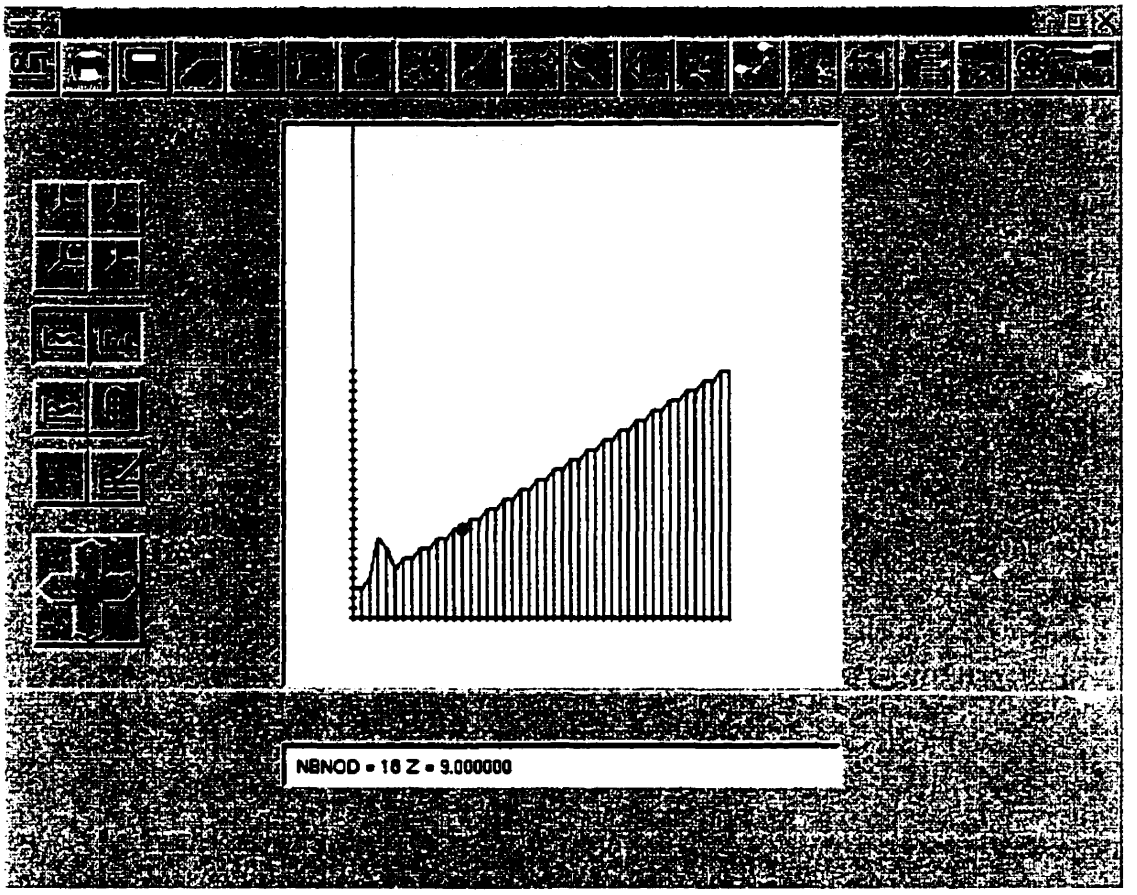


Figure A.3 – Fenêtre d'AutoGraphiX en mode d'étude paramétrique avec 1 paramètre

ligne et le nombre de lignes dans chaque page. Le fichier généré alors porte le nom "monPb-sum.ps". Il est aussi possible de générer un rapport plus détaillé composé de plusieurs parties. Sur quelques pages sont représentés les graphes dans le format qui vient d'être décrit pour visualisation globale. Une seconde partie comporte, si le nombre de graphes conjecturés extrêmes n'est pas trop grand (20 par défaut bien que cette valeur puisse être changée à partir du fichier de paramètres par la commande **MaxGraphsInReport n** où **n** est le nombre souhaité), une page pour chaque graphe comprenant une image de ce dernier ainsi qu'une description comportant une liste d'informations demandées par l'usager dans le fichier de paramètres. Cette liste est donnée entre les mot-clés **Report** et **End**. Il est ainsi possible de demander des va-

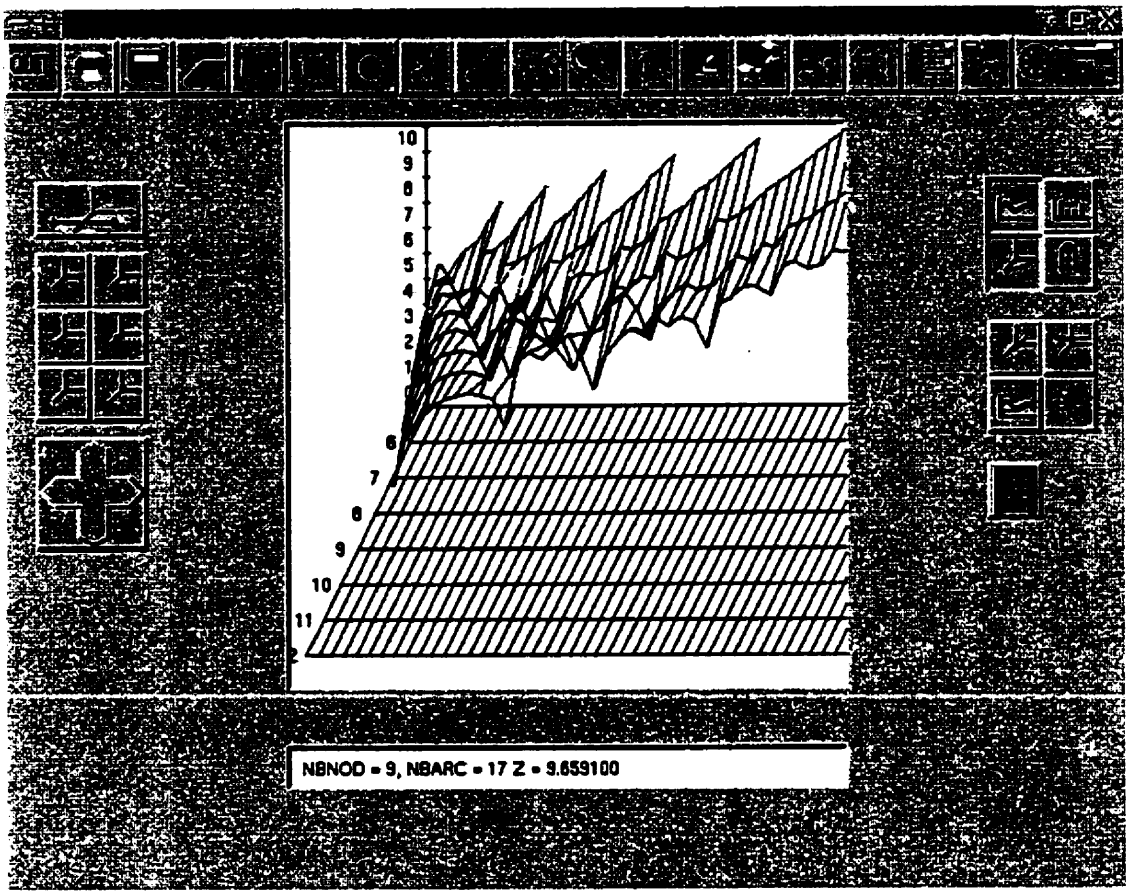


Figure A.4 – Fenêtre d'AutoGraphiX en mode d'étude paramétrique avec 2 paramètres

leurs d'invariants ainsi que des informations plus générales telles que les coefficients de polynôme de Hosoya par exemple. Une autre partie du rapport comporte une courbe représentant la fonction-objectif en fonction des paramètres, et la dernière partie est consacrée aux conjectures trouvées automatiquement par le programme.

A.4 La représentation des graphes

Un aspect déterminant quant à la performance du programme est la manière dont il assiste le mathématicien dans sa tâche. Il n'est pas question de demander au chercheur de dessiner le graphe obtenu en partant d'une information brute, mais plutôt de faire dessiner le graphe par l'ordinateur de la manière la plus proche possible de ce que l'utilisateur désire.

La manière de représenter des graphes est à elle seule un problème aussi important que difficile. La première règle, impérative, est l'interdiction de voir un sommet sur une arête car cette configuration engendre une ambiguïté. Une manière d'éviter ce problème est de positionner les sommets du graphe sur un cercle, toutes les arêtes étant alors strictement à l'intérieur de ce dernier. D'une manière générale, cette règle est assez facile à respecter, mais ne suffit pas à définir une manière de positionner les sommets que nous qualifierons d'esthétique, ou de satisfaisante. Si nous devons aussi tenir compte d'aspects esthétiques, ou d'autres critères, alors le problème devient très complexe, et justifie que la représentation des graphes soit actuellement une discipline à part entière à laquelle est consacré un symposium annuel depuis 1992, dont les actes sont publiés par *Springer Verlag* dans la série *Lecture Notes in Computer Science.*

Sans aucune prétention dans le domaine de la représentation des graphes, nous avons simplement essayé de considérer quelques cas particuliers afin de rendre l'interface graphique plus facile à utiliser. Tout d'abord, nous avons remarqué que selon le graphe considéré, la représentation adéquate diffère totalement. Dans l'application que nous en faisons ici, nous demandons au programme qu'il représente le graphe de manière à mettre en évidence sa structure. L'objectif que nous recherchons pour représenter le graphe varie donc selon sa structure. Par exemple, la minimisation du nombre de croisements d'arêtes est particulièrement peu appropriée aux graphes

bipartis pour lesquels nous cherchons plutôt à voir rapidement une partition des sommets en deux stables (sous-graphe ne comportant aucune arête). Dans ce dernier cas même, il y a une exception puisque nous ne nous attendons pas à voir un arbre représenté ainsi. Si le graphe n'est pas connexe, nous nous attendons à ce que chaque composante connexe soit représentée séparément. Partant de ces remarques, nous avons décidé d'orienter la représentation sur la famille à laquelle appartient le graphe, et d'appeler un algorithme différent pour chacune d'elles. Si le graphe n'appartient à aucune famille répertoriée, c'est la représentation par défaut qui est appliquée, à savoir placer les sommets sur un cercle. Parmi les familles traitées, nous avons porté un intérêt particulier aux arbres parce que nous les avons étudiés plus en détails. L'algorithme que nous avons utilisé pour les arbres est le suivant :

1. Initialisation : Soit C un sommet de degré maximum. Posons $\mathcal{P} = \emptyset$ et $\mathcal{L} = \{C\}$. Placer C à la position $(0, 0)$.
2. **Tant que** $\mathcal{L} \neq \emptyset$ $\forall v \in \mathcal{L}$ Soit $\mathcal{A}(v)$ l'ensemble des sommets adjacents à v.
   - **Si** $\delta_v > 1$

     Si $\mathcal{P} = \emptyset$,

     placer les sommets de $\mathcal{A}(v)$ régulièrement sur un cercle de rayon 1 autour de C. $\mathcal{P} \leftarrow \mathcal{P} \cup \{v\}$

     $\mathcal{L} \leftarrow (\mathcal{L} \cap \{v\}) \cup (\mathcal{A}(v) \cap \mathcal{P})$

     Si non

     Soit $w = \mathcal{A}(v) \cap \mathcal{P}$, placer les sommets $u \in \mathcal{A}(v) \cap \mathcal{P}$ régulièrement sur le demi-cercle ayant pour centre v, rayon $\frac{1}{d_{uC}-1}$ de manière que l'angle $\widehat{wv, vu} \geq \frac{\pi(1+\frac{1}{\delta_v})}{2}$.
   - **Si non** $\mathcal{P} \leftarrow \mathcal{P} \cup \{v\}$

     $\mathcal{L} \leftarrow \mathcal{L} \cap \{v\}$

3. Mettre à l'échelle le graphe obtenu selon chacun des deux axes afin qu'il occupe le mieux possible la place sur la fenêtre.

Cet algorithme très rapide donne en général une représentation des arbres tout à fait satisfaisante comme le montre la figure A.5.

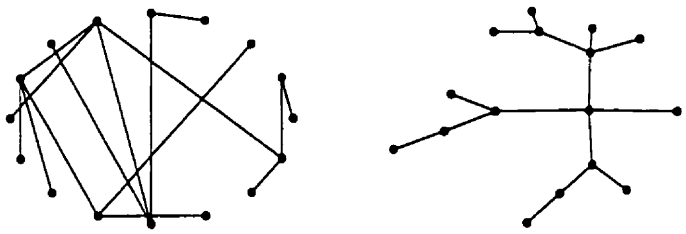


Figure A.5 – Un arbre à 15 sommets avant et après positionnement par le programme

Toujours dans le but de faciliter la lecture des graphes proposés, **AGX** offre la possibilité de donner des couleurs différentes aux sommets selon plusieurs critères tels que la coloration propre (affectation d'un nombre minimum de couleurs aux sommets pour que deux d'entre eux qui sont adjacents n'aient jamais la même couleur), la coloration propre du graphe complémentaire, le degré des sommets (les sommets de degré 1 sont bleus, ceux de degré 2 sont rouge, etc...) ou les composantes connexes (une couleur est affectée à chaque composante). Dans le cas oú le graphe comporte plus de 25 sommets, toutefois, les colorations propres ne sont pas utilisées à cause de temps parfois long qui peut s'avérer nécessaire aux calculs. Le choix du mode de coloration peut se faire depuis le fichier de paramètres mais peut aussi être modifié à l'aide de l'interface. Une autre fonction ajoutée récemment alors que nous étudiions la transmission au sein de graphes (la transmission d'un sommet est la distance du sommet qui lui est le plus éloigné) est l'affichage d'information relative aux sommets.

Lorsque nous plaçons le curseur sur un sommet, une fenêtre indique le nom du sommet pointé ainsi que sa transmission alors que les arêtes de l'arbre des plus courts chemins associé au dit sommet sont présentées en rouge (au lieu de bleu), et l'étiquette ou le point représentant chaque sommet se change pour une étiquette indiquant la distance au sommet pointé. Si l'utilisateur clique alors sur le bouton central de la souris, les sommets du graphe sont placées selon l'algorithme pour le positionement des arbres décrit plus tôt, à partir du sommet pointé et en utilisant l'arbre des plus courts chemins mis en évidence. Cette méthode permet de mieux visualiser l'arbre des plus courts chemins mais n'assure nullement qu'aucun sommet ne se trouve sur une arête. À ce titre, elle ne peut être utilisée qu'à titre d'information complémentaire.

A.5 Exemple de fichier de paramètres

Exemple de fichier de paramètres utilisé pour trouver des arbres chimiques d'indice de Randić minimum avec nombre de sommets pendants fixés.

```
BeginFile

/* Graphes d'ordre variant de 5 a 19                                  */
for NBNOD = 5 to 19 step 1

/* nombre de sommets pendants, toutes les valeurs possibles
pour des arbres a NBNOD sommets, l'invariant PENDING
n'est pas utilisé ici  mais plutôt mis dans les contraintes
sous la forme P - PENDING = 0 [600] afin de pouvoir ajuster
le poids associé à cette contrainte, sans quoi le programme
```

```
qui accorde un poids beaucoup plus élevé aux paramètres
ne parviendra pas bien à trouver des graphes avec le bon nombre
d'arêtes                                                        */

for P=2 to FLOOR((2*NBNOD-4)/3+2) step 1

/* Description du problème: maximiser l'indice de Randic pour des
arbres (NBNOD-NBARC=1) chimiques (degré maximum de 4, soit pas
de sommets de degré 5, ni de degré 6, etc...
Noter que la pénalité croît avec la valeur du degré afin de donner
une direction de descente au programme (un sommet de degré 5 sera
préféré à un sommet de degré 6, etc.
On remarque aussi que la contrainte sur le nombre d'arêtes a un
poids implicite de 1000 qui prévaut sur les contraintes de degré*/

Problem
Max: RANDIC
SubjectTo
NBNOD - NBARC = 1
P - PENDING = 0 [600]
DEG(5) = 0 [25]
DEG(6) = 0 [50]
DEG(7) = 0 [75]
DEG(8) = 0 [100]
DEG(9) = 0 [125]
DEG(10) = 0 [150]
DEG(11) = 0 [175]
DEG(12) = 0 [200]
```

```
DEG(13) = 0 [225]
DEG(14) = 0 [250]
DEG(15) = 0 [275]
DEG(16) = 0 [300]
DEG(17) = 0 [325]
DEG(18) = 0 [350]
DEG(19) = 0 [375]
DEG(20) = 0 [400]
End

Connex          /* On ne veut que des graphes connexes            */
InitialGraph Tree /* Lors de la génération aléatoire du premier
                     graphe, on génèrera un arbre                  */
MaxStart 1      /* Nombre de fois ou l'on lance VNS               */
VnsMin 2        /* Taille minimale d'une perturbation             */
VnsMax 20       /* Taille maximale d'une perturbation             */
VnsStep 1       /* À chaque fois, on augmentera la taille
                     de la perturbation de  1                      */
VnsTry 2        /* On fait deux essais pour chaque taille
                     de perturbation.                              */
VnsType Move    /* Les perturbations consisteront seulement en des
                     déplacements d'arêtes car ajouter ou retirer des
                     arêtes n'est pas permis                       */
VnsCentered     /* On  reviendra au premier voisinage de la séquence
  de descente chaque fois que la meilleure solution
                     est améliorée                                 */
ProblemName "Min Ra, arbres chim, n1 et n fixes"
AutoPos 1       /* Pour le repositionnnement des graphes à l'écran*/
```

```
SaveCaractPs   /* donne une petite description du graphe dans le
                  .ps  décrit ci dessous                        */
AutoSavePs     /* donne le graphes extrémaux a x et y fixes,
                  en .ps                                        */
Interactive 1  /* degre d'interactivite de l'utilisateur lors de
                  l'exécution
                  1: affichage uniquement à la fin de
  l'optimisation                                  */
DisplayColors DEG /* La couleur associée à chaque sommet sur la
                  fenêtre correspond à son degré                */
DrawDots       /* Sur la fenêtre, chaque sommet est représenté par
                  un point au lieu de la lettre qui lui est associée
                  (ce choix peut être modifié interactivement en
                  cliquant sur le bouton droit de la souris dans la
  fenêtre où est représenté le graphe)            */
PrintCmd "lpr" /* Commande à effectuer pour imprimer un fichier
                  PostScript dans le système                    */

CompactPs 5 6  /* La représentation de graphes sur le sommaire et le
                  rapport se fera en en mettant 5 par ligne, et 6
                  lignes par page                               */

Caracterise 1  /* niveau de l'information requise a propos de
          graphes obtenus                                  */
Verbose 3      /* niveau de l'information ecrite dans le fichier
                  .txt, 3 signifie que l'on a toutes les solutions
  et leurs ameliorations, s'il y en a             */
PrintPar       /* necessaire a la creation du fichier .txt      */
```

```
PrintCaract    /* Donne les caractéristiques des graphes obtenus
                  dans le rapport (fichier-rep.ps) et les fichiers
                  PostScripts ou sont représentés chaque graphe   */

/* racines du generateur aleatoire */
Seed1 3588
Seed2 746576
Seed3 788687
Seed4 3309469

Eps 0.00000001 /* Différence maximum entre 2 valeurs pour que le
                    programme les considère identiques            */
Infty 1000000  /* Valeur associée à l'infini pour des raisons
                  numériques                                      */

Range 20       /* Valeur de la fonction-objectif au-dessus de
                  laquelle on considérera qu'une contrainte n'est
                  pas respectée                                   */
EndFile
```

A.6 Quelques astuces de modélisation pour AutoGraphiX

Comme la méthode de recherche de graphes est heuristique, aucune garantie n'est donnée quant à la qualité des solutions trouvées. Il est alors très important de procéder à un examen du problème à traiter dans le but de trouver la manière de le représenter pour tirer le meilleur profit de l'optimisation dans **AGX**.

Un problème particulièrement critique apparaît avec certains invariants qui prennent un nombre restreint de valeurs. Dans ce cas, il est possible qu'un grand nombre de voisins de G aient la même valeur pour la fonction-objectif que lui, et qu'aucun ne soit meilleur. Pour éviter tout risque de cyclage, **AGX** n'effectue alors aucun mouvement. D'un autre coté, il n'explore pas ces *plateaux* (ensemble de graphes voisins avec même valeur pour la fonction-objectif) et reste comme dans un optimum local. Une manière de faire progresser le programme est d'ajouter à la fonction-objectif un critère secondaire qui donne à l'algorithme une information au sujet de la direction à suivre dans le but de poursuivre la descente. Il faut alors s'assurer que le critère secondaire soit doté d'un poids assez faible pour ne pas altérer le critère principal. Pour illustrer cette approche, prenons l'exemple suivant, suggéré par Paul Seymour lors d'une visite à Montréal pour un atelier sur la coloration des graphes : trouver un graphe cubique (dont tous les sommets ont un degré 3), de diamètre 3 (le diamètre d'un graphe est la plus grande distance qui soit entre sommets de ce graphe) avec le plus de sommets possibles. Notre approche fut d'abord de fixer le nombre de sommets, puis de chercher à maximiser le diamètre. Après avoir trouvé un graphe vérifiant les contraintes, nous cherchons un graphe avec deux sommets de plus, jusqu'à ce qu'**AGX** ne trouve plus de graphes respectant les contraintes. Le problème, c'est que le diamètre prend un nombre restreint de valeurs et **AGX** n'a pu trouver de graphes à plus de 16 sommets vérifiant les conditions posées. La raison de cet échec est que le diamètre d'un graphe peut prendre un nombre réduit de valeurs différentes. Comme conséquence, la plupart de ses voisins peuvent avoir le même diamètre (ou avoir un diamètre supérieur). Dans ce cas, **AGX** ne dispose d'aucune direction vers une amélioration de la fonction objectif. Pour éviter les risques de cyclage, il n'accepte de déplacements que s'ils sont strictement bénéfiques et reste sur ce plateau comme dans un optimum local. Un graphe G peut avoir un diamètre $D(G)$ parce qu'un grand nombre de paires de sommets sont à une distance $D(G)$, dans ce cas, il faudra déplacer un important nombre d'arêtes pour construire un

graphe G' de diamètre supérieur. Nous disons alors que G est sur un plateau. Une telle situation se traduit par l'absence de direction de descente et les perturbations de la Recherche à Voisinages Variables ne suffit pas à en sortir. Comme **AGX** n'effectue un déplacement que s'il est strictement bénéfique, les plateaux se comportent comme des cas particuliers d'optimums locaux où la fonction-objectif ne change pas sur une grande distance. Il est alors très difficile d'en sortir, même avec la *Recherche à Voisinages Variables* dont les perturbations nous mènent en général à des graphes sur le même plateau. Il devient alors indispensable de donner au programme une direction de descente secondaire vers laquelle on puisse espérer trouver un graphe dont l'objectif soit meilleur. Bien sur, cette direction secondaire doit être assez faible pour ne pas se substituer à l'objectif original à rechercher mais être utilisée uniquement dans le cas où un plateau est rencontré.

Dans le cas qui nous intéresse ici, avec l'intuition que la distance moyenne $\bar{d}$ (moyenne des distances entre paires de sommets) évolue en général dans le même sens que le diamètre, nous avons ajouté à la fonction-objectif un millième de la distance moyenne et **AGX** a trouvé en quelques minutes un graphe cubique, de diamètre 3 comportant 20 sommets, qui est la taille du plus grand graphe cubique de diamètre 3.

Annexe B

Tables de résultats d'énumeration

B.1 Polyhexes planaires simplement connectés

B.2 Polyhexes simplement connectés

Tableau B.1 – Isomères pour $h = 18$ en fonction des classes de symétrie.

Isomère	Nombre	C2h	D2h	C3h	D3h	C2ν
$C_{52}H_{18}$	3	1	0	0	1	1
$C_{53}H_{19}$	53	0	0	0	0	5
$C_{54}H_{20}$	471	14	2	0	0	23
$C_{55}H_{21}$	2437	0	0	3	0	24
$C_{56}H_{22}$	10587	59	5	0	0	91
$C_{57}H_{23}$	39143	0	0	0	0	71
$C_{58}H_{24}$	133713	206	6	8	0	308
$C_{59}H_{25}$	424429	0	0	0	0	165
$C_{60}H_{26}$	1262442	601	6	0	0	871
$C_{61}H_{27}$	3515696	0	0	15	2	354
$C_{62}H_{28}$	9295801	1602	5	0	0	2252
$C_{63}H_{29}$	23000043	0	0	0	0	715
$C_{64}H_{30}$	53396021	3832	12	39	1	5191
$C_{65}H_{31}$	115992011	0	0	0	0	1139
$C_{66}H_{32}$	234832415	8050	16	0	0	10535
$C_{67}H_{33}$	435901284	0	0	99	2	1695
$C_{68}H_{34}$	738076219	14601	17	0	0	18721
$C_{69}H_{35}$	1123527420	0	0	0	0	1971
$C_{70}H_{36}$	1503478059	22472	47	134	0	28086
$C_{71}H_{37}$	1676318405	0	0	0	0	1474
$C_{72}H_{38}$	1460056378	24071	13	0	0	29539
$C_{73}H_{39}$	878110881	0	0	125	0	0
$C_{74}H_{40}$	296275836	14756	27	0	0	17445
Total	8553649747	90265	156	423	6	120676

Tableau B.2 – Isomères pour $h = 19$ en fonction des classes de symétrie

Isomère	Nombre	C2h	D2h	C3h	D3h	C6h	D6h	C2ν
$C_{54}H_{18}$	1	0	0	0	0	0	1	0
$C_{55}H_{19}$	18	0	0	0	0	0	0	3
$C_{56}H_{20}$	256	3	1	0	0	0	0	17
$C_{57}H_{21}$	1647	0	0	1	0	0	0	14
$C_{58}H_{22}$	7885	17	2	0	0	0	0	92
$C_{59}H_{23}$	32042	0	0	0	0	0	0	60
$C_{60}H_{24}$	116648	66	6	4	1	0	0	316
$C_{61}H_{25}$	388180	0	0	0	0	0	0	175
$C_{62}H_{26}$	1223950	189	7	0	0	0	0	814
$C_{63}H_{27}$	3622656	0	0	6	2	0	0	495
$C_{64}H_{28}$	10119046	589	10	0	0	0	0	2323
$C_{65}H_{29}$	26745152	0	0	0	0	0	0	1402
$C_{66}H_{30}$	67072867	1677	18	26	3	1	0	6037
$C_{67}H_{31}$	157998026	0	0	0	0	0	0	3113
$C_{68}H_{32}$	350234105	3829	17	0	0	0	0	12851
$C_{69}H_{33}$	727273192	0	0	41	0	0	0	6200
$C_{70}H_{34}$	1405607063	7948	24	0	0	0	0	24710
$C_{71}H_{35}$	2494602124	0	0	0	0	0	0	11851
$C_{72}H_{36}$	4038939706	15649	23	93	8	1	1	46152
$C_{73}H_{37}$	5870405899	0	0	0	0	0	0	18123
$C_{74}H_{38}$	7465388988	25543	47	0	0	0	0	72151
$C_{75}H_{39}$	7889345133	0	0	169	0	0	0	18007
$C_{76}H_{40}$	6500036300	26931	0	0	0	0	0	74547
$C_{77}H_{41}$	3704740320	0	0	0	0	0	0	8970
$C_{78}H_{42}$	1178741568	15472	34	203	5	0	0	40141
Total	41892642772	97913	189	543	19	2	2	348564

Tableau B.3 – Isomères pour $h = 20$ en fonction des classes de symétrie.

Isomère	Nombre	C2h	D2h	C2ν
$C_{57}H_{19}$	4	0	0	1
$C_{58}H_{20}$	129	8	1	15
$C_{59}H_{21}$	1009	0	0	17
$C_{60}H_{22}$	5726	42	5	66
$C_{61}H_{23}$	25050	0	0	68
$C_{62}H_{24}$	97607	178	5	273
$C_{63}H_{25}$	345834	0	0	181
$C_{64}H_{26}$	1140529	576	5	845
$C_{65}H_{27}$	3540792	0	0	439
$C_{66}H_{28}$	10464798	1650	10	2360
$C_{67}H_{29}$	29228956	0	0	1013
$C_{68}H_{30}$	77591586	4498	8	6134
$C_{69}H_{31}$	195628098	0	0	2016
$C_{70}H_{32}$	468024447	10935	22	14544
$C_{71}H_{33}$	1055012528	0	0	3709
$C_{72}H_{34}$	2241109396	23872	31	30661
$C_{73}H_{35}$	4460933006	0	0	5663
$C_{74}H_{36}$	8257827535	46560	42	57964
$C_{75}H_{37}$	14051002527	0	0	7914
$C_{76}H_{38}$	21801082567	78191	40	95530
$C_{77}H_{39}$	30306079909	0	0	8333
$C_{78}H_{40}$	36713205973	109470	90	130675
$C_{79}H_{41}$	36879305655	0	0	5442
$C_{80}H_{42}$	28838451119	105790	14	123735
$C_{81}H_{43}$	15621799283	0	0	0
$C_{82}H_{44}$	4702507923	58460	37	65905
Total	205714411986	440230	310	563503

Tableau B.4 – Isomères pour $h = 21$ en fonction des classes de symétrie

	Number	C2h	D2h	C3h	D3h	C6h	D6h	$C2\nu$
$C_{59}H_{19}$	1	0	0	0	0	0	0	1
$C_{60}H_{20}$	47	1	1	0	0	0	0	7
$C_{61}H_{21}$	587	0	0	2	2	0	0	12
$C_{62}H_{22}$	3838	13	3	0	0	0	0	70
$C_{63}H_{23}$	18876	0	0	0	0	0	0	36
$C_{64}H_{24}$	79472	45	5	4	1	0	0	235
$C_{65}H_{25}$	296025	0	0	0	0	0	0	167
$C_{66}H_{26}$	1029521	188	8	0	0	0	0	867
$C_{67}H_{27}$	3352432	0	0	21	1	0	0	517
$C_{68}H_{28}$	10337238	587	12	0	0	0	0	2453
$C_{69}H_{29}$	30314357	0	0	0	0	0	0	1262
$C_{70}H_{30}$	84877350	1588	16	44	1	0	0	6150
$C_{71}H_{31}$	225927887	0	0	0	0	0	0	3642
$C_{72}H_{32}$	573384355	4547	16	0	0	0	0	16214
$C_{73}H_{33}$	1384916403	0	0	103	2	0	0	8863
$C_{74}H_{34}$	3175427434	11377	42	0	0	0	0	37733
$C_{75}H_{35}$	6874424366	0	0	0	0	0	0	18149
$C_{76}H_{36}$	14033798842	24111	31	232	1	0	0	75274
$C_{77}H_{37}$	26842275823	0	0	0	0	0	0	34783
$C_{78}H_{38}$	47720986689	47515	43	0	0	0	0	138929
$C_{79}H_{39}$	78025561519	0	0	480	0	0	0	60330
$C_{80}H_{40}$	116224121999	85645	45	0	0	0	0	239646
$C_{81}H_{41}$	154775144598	0	0	0	0	0	0	82662
$C_{82}H_{42}$	179004609494	124260	73	577	2	0	0	335306
$C_{83}H_{43}$	171367411124	0	0	0	0	0	0	74046
$C_{84}H_{44}$	127557881410	117943	30	0	0	0	0	311055
$C_{85}H_{45}$	65842485473	0	0	460	0	0	0	33940
$C_{86}H_{46}$	18806505243	61547	40	0	0	0	0	152826
Total	1012565172403	479367	365	1923	10	0	0	1635175

Tableau B.5 – Isomères pour $h = 22$ en fonction des classes de symétrie

	Number	C2h	D2h	C3h	D3h	C6h	D6h	$C2\nu$
$C_{62}H_{20}$	16	2	1	0	0	0	0	4
$C_{63}H_{21}$	290	0	0	1	0	0	0	8
$C_{64}H_{22}$	2467	32	3	0	0	0	0	56
$C_{65}H_{23}$	13652	0	0	0	0	0	0	54
$C_{66}H_{24}$	62027	138	3	2	3	0	0	211
$C_{67}H_{25}$	247989	0	0	0	0	0	0	178
$C_{68}H_{26}$	903415	510	6	0	0	0	0	770
$C_{69}H_{27}$	3066067	0	0	8	0	0	0	493
$C_{70}H_{28}$	9893810	1627	16	0	0	0	0	2349
$C_{71}H_{29}$	30276022	0	0	0	0	0	0	1193
$C_{72}H_{30}$	88445658	4641	19	21	4	0	0	6479
$C_{73}H_{31}$	247093330	0	0	0	0	0	0	2788
$C_{74}H_{32}$	660724668	12646	24	0	0	0	0	16960
$C_{75}H_{33}$	1685369866	0	0	58	1	0	0	5908
$C_{76}H_{34}$	4107270241	31360	30	0	0	0	0	40657
$C_{77}H_{35}$	9536543476	0	0	0	0	0	0	11169
$C_{78}H_{36}$	21028831864	70888	30	149	6	0	0	89378
$C_{79}H_{37}$	43844651607	0	0	0	0	0	0	19138
$C_{80}H_{38}$	86215717266	144276	50	0	0	0	0	176325
$C_{81}H_{39}$	158793911206	0	0	234	0	0	0	27984
$C_{82}H_{40}$	271745847969	261789	90	0	0	0	0	311332
$C_{83}H_{41}$	427742492665	0	0	0	0	0	0	36524
$C_{84}H_{42}$	612649644824	409747	62	512	18	0	0	478536
$C_{85}H_{43}$	782779458259	0	0	0	0	0	0	35025
$C_{86}H_{44}$	866100354968	524600	163	0	0	0	0	599661
$C_{87}H_{45}$	792069689506	0	0	767	0	0	0	20387
$C_{88}H_{46}$	562684058255	462848	37	0	0	0	0	517568
$C_{89}H_{47}$	277402920664	0	0	0	0	0	0	0
$C_{90}H_{48}$	75380203150	232842	81	755	9	0	0	251310
Total	4994807695197	2157946	615	2507	41	0	0	2652445

Tableau B.6 – Isomères pour $h = 23$ en fonction des classes de symétrie

	Number	C2h	D2h	C3h	D3h	C6h	D6h	$C2\nu$
$C_{64}H_{20}$	4	0	1	0	0	0	0	2
$C_{65}H_{21}$	126	0	0	0	0	0	0	9
$C_{66}H_{22}$	1448	6	3	0	0	0	0	33
$C_{67}H_{23}$	9349	0	0	0	0	0	0	38
$C_{68}H_{24}$	47167	45	3	0	0	0	0	235
$C_{69}H_{25}$	200545	0	0	0	0	0	0	124
$C_{70}H_{26}$	771061	145	6	0	0	0	0	704
$C_{71}H_{27}$	2745113	0	0	0	0	0	0	435
$C_{72}H_{28}$	9185903	510	13	0	0	0	0	2283
$C_{73}H_{29}$	29232372	0	0	0	0	0	0	1437
$C_{74}H_{30}$	89047381	1706	19	0	0	0	0	6954
$C_{75}H_{31}$	259056456	0	0	0	0	0	0	3628
$C_{76}H_{32}$	723113653	4697	24	0	0	0	0	17864
$C_{77}H_{33}$	1936428286	0	0	0	0	0	0	9528
$C_{78}H_{34}$	4970254505	12446	35	0	0	0	0	43798
$C_{79}H_{35}$	12200281933	0	0	0	0	0	0	24237
$C_{80}H_{36}$	28643742341	32297	45	0	0	0	0	106293
$C_{81}H_{37}$	64109618941	0	0	0	0	0	0	53593
$C_{82}H_{38}$	136335211131	73731	79	0	0	0	0	228216
$C_{83}H_{39}$	274374476732	0	0	0	0	0	0	104754
$C_{84}H_{40}$	520693871834	147440	70	0	0	0	0	431864
$C_{85}H_{41}$	925231748273	0	0	0	0	0	0	188701
$C_{86}H_{42}$	1527029503977	274532	108	0	0	0	0	760883
$C_{87}H_{43}$	2317626216529	0	0	0	0	0	0	300925
$C_{88}H_{44}$	3196330255841	454304	70	0	0	0	0	1212591
$C_{89}H_{45}$	3924113646972	0	0	0	0	0	0	373411
$C_{90}H_{46}$	4161661164458	594505	169	0	0	0	0	1536669
$C_{91}H_{47}$	3643586919511	0	0	0	0	0	0	305586
$C_{92}H_{48}$	2476073860335	514849	0	0	0	0	0	1298746
$C_{93}H_{49}$	1168340063265	0	0	0	0	0	0	129710
$C_{94}H_{50}$	302754225098	245895	103	0	0	0	0	586556
Total	24687124900540	2357108	748	0	0	0	0	7729807

Tableau B.7 – Isomères pour $h = 16$ en fonction des classes de symétrie

Isomer	Number	C2h	D2h	C3h	D3h	C6h	D6h	C2ν
$C_{47}H_{17}$	1	0	0	0	0	0	0	1
$C_{48}H_{18}$	30	3	2	0	1	0	0	4
$C_{49}H_{19}$	223	0	0	0	0	0	0	9
$C_{50}H_{20}$	1121	20	2	0	0	0	0	31
$C_{51}H_{21}$	4501	0	0	1	1	0	0	28
$C_{52}H_{22}$	16310	80	0	0	0	0	0	121
$C_{53}H_{23}$	52795	0	0	0	0	0	0	64
$C_{54}H_{24}$	163881	215	5	4	1	0	0	348
$C_{55}H_{25}$	473555	0	0	0	0	0	0	147
$C_{56}H_{26}$	1278169	586	7	0	0	0	0	945
$C_{57}H_{27}$	3207845	0	0	7	0	0	0	280
$C_{58}H_{28}$	7579108	1404	9	0	0	0	0	2341
$C_{59}H_{29}$	16381817	0	0	0	0	0	0	497
$C_{60}H_{30}$	32280641	2864	14	17	5	0	0	4934
$C_{61}H_{31}$	57289337	0	0	0	0	0	0	676
$C_{62}H_{32}$	90549291	5057	31	0	0	0	0	9031
$C_{63}H_{33}$	119314280	0	0	39	0	0	0	675
$C_{64}H_{34}$	123231368	6335	5	0	0	0	0	11537
$C_{65}H_{35}$	87508095	0	0	0	0	0	0	0
$C_{66}H_{36}$	36695369	4635	15	66	5	0	0	9265
Total	576027737	21199	90	134	13	0	0	40934

Tableau B.8 – Isomères pour $h = 17$ en fonction des classes de symétrie

Isomer	Number	C2h	D2h	C3h	D3h	C6h	D6h	C2ν
$C_{50}H_{18}$	9	0	1	0	0	0	0	2
$C_{51}H_{19}$	117	0	0	0	0	0	0	6
$C_{52}H_{20}$	763	6	3	0	0	0	0	36
$C_{53}H_{21}$	3414	0	0	0	0	0	0	17
$C_{54}H_{22}$	13301	20	2	0	0	0	0	104
$C_{55}H_{23}$	46688	0	0	0	0	0	0	73
$C_{56}H_{24}$	153158	68	3	0	0	0	0	311
$C_{57}H_{25}$	467966	0	0	0	0	0	0	235
$C_{58}H_{26}$	1363062	230	11	0	0	0	0	1009
$C_{59}H_{27}$	3718409	0	0	0	0	0	0	652
$C_{60}H_{28}$	9515597	606	7	0	0	0	0	2554
$C_{61}H_{29}$	22797671	0	0	0	0	0	0	1495
$C_{62}H_{30}$	51188565	1349	10	0	0	0	0	5437
$C_{63}H_{31}$	105316193	0	0	0	0	0	0	3369
$C_{64}H_{32}$	197855777	3052	18	0	0	0	0	12209
$C_{65}H_{33}$	334891034	0	0	0	0	0	0	6509
$C_{66}H_{34}$	500794983	5941	21	0	0	0	0	23327
$C_{67}H_{35}$	621832355	0	0	0	0	0	0	7897
$C_{68}H_{36}$	603819104	7189	20	0	0	0	0	28776
$C_{69}H_{37}$	405289754	0	0	0	0	0	0	4438
$C_{70}H_{38}$	159918120	4635	15	0	0	0	0	18555
Total	3018986040	23096	111	0	0	0	0	117011

Tableau B.9 – Isomères pour $h = 18$ en fonction des classes de symétrie

	Number	C2h	D2h	C3h	D3h	C6h	D6h	C2ν
$C_{52}H_{18}$	3	1	0	0	1	0	0	1
$C_{53}H_{19}$	53	0	0	0	0	0	0	5
$C_{54}H_{20}$	471	14	2	0	0	0	0	23
$C_{55}H_{21}$	2437	0	0	3	0	0	0	24
$C_{56}H_{22}$	10588	59	5	0	0	0	0	91
$C_{57}H_{23}$	39334	0	0	0	0	0	0	74
$C_{58}H_{24}$	136344	206	6	8	0	0	0	315
$C_{59}H_{25}$	443380	0	0	0	0	0	0	178
$C_{60}H_{26}$	1362022	601	6	0	0	0	0	946
$C_{61}H_{27}$	3943217	0	0	15	2	0	0	410
$C_{62}H_{28}$	10891716	1641	6	0	0	0	0	2683
$C_{63}H_{29}$	28258301	0	0	0	0	0	0	876
$C_{64}H_{30}$	69045923	4025	12	39	1	0	0	6597
$C_{65}H_{31}$	158325287	0	0	0	0	0	0	1604
$C_{66}H_{32}$	339201122	8842	19	0	0	0	0	14970
$C_{67}H_{33}$	666871615	0	0	107	3	0	0	2696
$C_{68}H_{34}$	1198097947	16793	17	0	0	0	0	29202
$C_{69}H_{35}$	1936345760	0	0	0	0	0	0	3290
$C_{70}H_{36}$	2746277704	26936	65	146	0	0	0	48447
$C_{71}H_{37}$	3224895584	0	0	0	0	0	0	2809
$C_{72}H_{38}$	2957778430	30050	15	0	0	0	0	55253
$C_{73}H_{39}$	1883445328	0	0	137	0	0	0	0
$C_{74}H_{40}$	701957539	19768	51	0	0	0	0	39536
Total	15927330105	108936	204	455	7	0	0	210030

Tableau B.10 – Isomères pour $h = 19$ en fonction des classes de symétrie

	Number	C2h	D2h	C3h	D3h	C6h	D6h	C2ν
$C_{54}H_{18}$	1	0	0	0	0	0	1	0
$C_{55}H_{19}$	18	0	0	0	0	0	0	3
$C_{56}H_{20}$	256	3	1	0	0	0	0	17
$C_{57}H_{21}$	1647	0	0	1	0	0	0	14
$C_{58}H_{22}$	7885	17	2	0	0	0	0	92
$C_{59}H_{23}$	32086	0	0	0	0	0	0	61
$C_{60}H_{24}$	117978	66	6	4	1	0	0	328
$C_{61}H_{25}$	399935	0	0	0	0	0	0	193
$C_{62}H_{26}$	1296573	189	7	0	0	0	0	857
$C_{63}H_{27}$	3975642	0	0	6	2	0	0	585
$C_{64}H_{28}$	11566647	597	11	0	0	0	0	2625
$C_{65}H_{29}$	31990818	0	0	0	0	0	0	1824
$C_{66}H_{30}$	84280474	1758	19	26	3	1	0	7423
$C_{67}H_{31}$	209206245	0	0	0	0	0	0	4422
$C_{68}H_{32}$	490133377	4134	19	0	0	0	0	17053
$C_{69}H_{33}$	1078246412	0	0	44	0	0	0	9705
$C_{70}H_{34}$	2211468431	8933	25	0	0	0	0	35729
$C_{71}H_{35}$	4167613253	0	0	0	0	0	0	20174
$C_{72}H_{36}$	7175697743	18633	30	110	10	1	1	74104
$C_{73}H_{37}$	11087497850	0	0	0	0	0	0	33994
$C_{74}H_{38}$	14950114520	31582	60	0	0	0	0	124350
$C_{75}H_{39}$	16657928358	0	0	201	0	0	0	36177
$C_{76}H_{40}$	14487876140	33795	0	0	0	0	0	135180
$C_{77}H_{41}$	8780346115	0	0	0	0	0	0	18570
$C_{78}H_{42}$	3101072051	19768	51	269	5	0	0	79118
Total	84530870455	119475	231	661	21	2	2	602598

Tableau B.11 – Isomères pour $h = 20$ en fonction des classes de symétrie

	Number	C2h	D2h	C3h	D3h	C6h	D6h	C2ν
$C_{57}H_{19}$	4	0	0	0	0	0	0	1
$C_{58}H_{20}$	129	8	1	0	0	0	0	15
$C_{59}H_{21}$	1009	0	0	0	0	0	0	17
$C_{60}H_{22}$	5726	42	5	0	0	0	0	66
$C_{61}H_{23}$	25057	0	0	0	0	0	0	69
$C_{62}H_{24}$	98126	178	5	0	0	0	0	274
$C_{63}H_{25}$	352663	0	0	0	0	0	0	195
$C_{64}H_{26}$	1189968	576	5	0	0	0	0	889
$C_{65}H_{27}$	3810058	0	0	0	0	0	0	477
$C_{66}H_{28}$	11696535	1666	11	0	0	0	0	2634
$C_{67}H_{29}$	34099127	0	0	0	0	0	0	1203
$C_{68}H_{30}$	94866661	4653	8	0	0	0	0	7452
$C_{69}H_{31}$	251646989	0	0	0	0	0	0	2647
$C_{70}H_{32}$	635388995	11750	26	0	0	0	0	19364
$C_{71}H_{33}$	1515224373	0	0	0	0	0	0	5239
$C_{72}H_{34}$	3413973431	26713	33	0	0	0	0	44397
$C_{73}H_{35}$	7220165711	0	0	0	0	0	0	9232
$C_{74}H_{36}$	14216730749	54644	60	0	0	0	0	93539
$C_{75}H_{37}$	25743534475	0	0	0	0	0	0	14436
$C_{76}H_{38}$	42549878007	96802	51	0	0	0	0	169680
$C_{77}H_{39}$	62936891732	0	0	0	0	0	0	15911
$C_{78}H_{40}$	80871221611	141596	120	0	0	0	0	256311
$C_{79}H_{41}$	85761644654	0	0	0	0	0	0	11913
$C_{80}H_{42}$	70974548080	143059	15	0	0	0	0	265101
$C_{81}H_{43}$	41052409755	0	0	0	0	0	0	0
$C_{82}H_{44}$	13779935438	85659	51	0	0	0	0	171318
Total	451069339063	567346	391	0	0	0	0	1092380