Aix*Marseille université
Socialement engagée

NNT/NL : 2020AIXM0001/001ED000

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 13 décembre 2022 par

Adrien Varet

Programmation par Contraintes et Chimie Théorique : Utilisation du Formalisme CSP pour Résoudre des Problématiques Liées aux Benzénoïdes

Discipline
Informatique

École doctorale
ED 184 : Mathématiques et Informatique
ED 250 : Sciences Chimiques de Marseille

Laboratoire/Partenaires de recherche
Laboratoire d'Informatique et Systèmes (LIS)
Institut des Sciences Moléculaires de Marseille (iSm2)

Composition du jury

Christophe LECOUTRE CRIL, Université d'Artois	Rapporteur
Simon DE GIVRY MIAT, INRAE Toulouse	Rapporteur
Christine SOLNON CITI, INSA Lyon	Présidente du jury
Aude SIMON LCPQ, Université de Toulouse	Examinatrice
Philippe VISMARA LIRMM, Institut Agro Montpellier	Examinateur
Stéphane HUMBEL iSm2, Aix-Marseille Université	Examinateur
Cyril TERRIOUX LIS, Aix-Marseille Université	Directeur de thèse
Yannick CARISSAN iSm2, Aix-Marseille Université	Co-directeur de thèse

LIS LABORATOIRE D'INFORMATIQUE & SYSTÈMES UMR 7020

iSm2 Institut des Sciences Moléculaires de Marseille UMR 7313

Affidavit

Je soussigné, Adrien VARET, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de Cyril TERRIOUX et Yannick CARISSAN, dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Marseille le 15/08/2022

Liste de publications et participation aux conférences

Liste des publications réalisées dans le cadre du projet de thèse :

1. Yannick CARISSAN, Denis HAGEBAUM-REIGNER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   Using Constraint Programming to Generate Benzenoid Structures in Theoretical Chemistry.
   In *Proceedings of the 26th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 690-706, 2020.
2. Yannick CARISSAN, Chisom-Adaobi DIM, Denis HAGEBAUM-REIGNIER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   Computing the Local Aromaticity of Benzenoids Thanks to Constraint Programming.
   In *Proceedings of the 26th International Conference on Principles and Practice of Constraint Programming*, pages 673-689, 2020.
3. Yannick CARISSAN, Denis HAGEBAUM-REIGNER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   Exhaustive Generation of Benzenoid Structures Sharing Common Patterns.
   In *Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming*, pages 19 :1-19.18, 2021.
4. Yannick CARISSAN, Denis HAGEBAUM-REIGNER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   Utiliser la PPC pour générer des structures de benzénoïdes en chimie théorique.
   In *Actes des 16èmes Journées Francophones de Programmation par Contraintes (JFPC)*, 2021.
5. Yannick CARISSAN, Denis HAGEBAUM-REIGNER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   Prise en compte de motifs et génération de structures de benzénoïdes.
   In *Actes des 16èmes Journées Francophones de Programmation par Contraintes (JFPC)*, 2021.
6. Yannick CARISSAN, Denis HAGEBAUM-REIGNER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   How constraint programming can help chemists to generate Benzenoid structures and assess the local Aromaticity of Benzenoids.
   In *Constraints Journal*, 27(3), pages 192-248, 2022.
7. Yannick CARISSAN, Denis HAGEBAUM-REIGNER, Nicolas PRCOVIC, Cyril TERRIOUX, Adrien VARET.
   BenzAI : A Program to Design Benzenoids with Defined Properties Using Constraint Programming.
   In *Journal of Chemical Information and Modelling*, 62(11), pages 2811-2820, 2022.

Participation aux conférences et écoles d'été au cours de la période de thèse :

1. Institut d'Automne en Intelligence Artificielle (IA2), 2019
2. International Symposium on Artificial Intelligence and Mathematics (ISAIM), 2020
3. Principles and Practice of Constraints Programming (CP), 2020
4. Joint ACP, ANITI, CNRS GDR IA, GDR RO International Autumn school on Combinatorial Optimization, Constraint Programming and Machine Learning, 2020
5. Journées Francophones de Programmation par Contraintes (JFPC), 2021
6. Principles and Practice of Constraints Programming (CP), 2021
7. Journées Francophones de Programmation par Contraintes (JFPC), 2022

Résumé

La *programmation par contraintes (PPC)* est une branche de l'intelligence artificielle apparue dans les années 70. Elle est située à l'interface d'autres disciplines telles que la théorie des graphes, la recherche opérationnelle, ou encore les bases de données relationnelles et s'attache à résoudre des problèmes fortement combinatoires (a minima NP-complets). Le principal avantage de la PPC est que pour résoudre un problème, il suffit de le décrire dans un formalisme donné et de fournir l'instance ainsi produite à un solveur existant. Ainsi, contrairement à d'autres paradigmes comme la programmation impérative, il n'est pas nécessaire de concevoir un nouvel algorithme de résolution pour chaque nouveau problème que l'on souhaite résoudre. La programmation par contraintes repose sur différents formalismes. Dans le cadre de cette thèse, nous nous focalisons sur celui des *problèmes de satisfaction de contraintes (CSP)* qui consiste à représenter le problème par un ensemble de variables, chacune possédant un ensemble de valeurs qu'elle peut prendre (appelé domaine). Ces variables vont être liées entre elles par des contraintes. L'objectif ici est d'affecter toutes les variables tout en prenant soin de respecter toutes les contraintes.

Les *hydrocarbures aromatiques polycycliques (HAP)* sont des molécules uniquement constituées de carbones et d'hydrogènes dont les atomes de carbones forment des cycles fusionnés de différentes tailles (majoritairement des cycles benzéniques, c'est-à-dire composés de six atomes de carbone en forme d'hexagones). Ces molécules possèdent des propriétés intéressantes en termes de stabilité énergétique, de structure moléculaire ou de spectre optique et à ce titre, elles sont étudiées dans de nombreux domaines tels que la science des matériaux, la nanoélectronique moléculaire, l'astrochimie, ... Les *benzénoïdes* constituent une sous-famille des HAP et sont constitués uniquement de cycles benzéniques fusionnés.

Dans ce travail, nous nous sommes intéressés à deux problématiques liées aux benzénoïdes : la génération exhaustive de toutes les structures de benzénoïdes satisfaisant un ensemble donné de propriétés (structurelles ou chimiques) d'une part, et la détermination de leur aromaticité d'autre part. L'aromaticité étant une notion induite par la délocalisation des électrons des HAP. Ces deux questions nécessitent de résoudre des problèmes fortement combinatoires pour lesquels la programmation par contraintes peut constituer une approche efficace et flexible.

Dans cette thèse, dans un premier temps, nous traitons de la génération exhaustive de structures de benzénoïdes en proposant d'abord un modèle CSP général, Puis nous présentons plusieurs extensions de ce modèle dont certaines permettent de prendre en compte la notion de motifs qui est actuellement l'objet de nombreuses études en chimie théorique. Dans un second temps, nous proposons une méthode de calcul de l'énergie de résonance basée sur un modèle CSP qui nous permet d'évaluer l'aromaticité d'un benzénoïde. Enfin, nous utilisons la PPC pour résoudre plusieurs problématiques secondaires (comme le calcul des structures de Clar) et décrivons le logiciel BenzAI qui implémente l'ensemble des travaux de cette thèse. Nous présentons également les premières applications de ces contributions dans le domaine de la chimie théorique.

Mots clés : programmation par contraintes, CSP, énumération, théorie des graphes, chimie théorique, hydrocarbures aromatiques polycycliques, benzénoïde, aromaticité

Abstract

Constraint programming (CP) is a branch of artificial intelligence that appeared in the 70s. It is located at the interface of other disciplines such as graph theory, operations research, or even relational databases and is committed to solving problems that are strongly combinatorics (at least NP-complete). The main advantage of CP is that to solve a problem, it suffices to describe it in a given formalism and to give the instance thus produced to an existing solver. Thus, unlike other paradigms like imperative programming, there is no need to design a new solving algorithm for each new problem you want to solve. Constraint programming is based on various formalisms. As part of this thesis, we focus on one of them called constraint satisfaction problems (CSP) which consists in representing the problem by a set of *variables*, each having a set of values it can take (called *domain*). These variables will be linked together by *constraints*. The objective here is to assign all the variables while satisfying all the constraints.

Polycyclic aromatic hydrocarbons (PAH) are molecules made up solely of carbons and hydrogens whose carbon atoms form merged cycles of different sizes. These molecules have interesting properties in terms of energy stability, molecular structure or optical spectrum and as such, they are studied in many fields such as materials science, molecular nanoelectronics, astrochemistry, ... *Benzenoids* constitute a sub-family of PAHs and consist only of merged benzenic cycles (therefore of hexagons).

In this work, we are focused on two issues related to benzenoids: being able of exhaustively generate all the structures of benzenoids corresponding to a given set of properties (structural or chemical) or even of estimating the *aromaticity* of a benzenoid. Aromaticity is a notion induced by the delocalization of the electrons of a PAH. Behind these two issues, we find different highly combinatorial problems for which constraint programming can be an effective and flexible approach.

In this thesis, firstly, we deal with the exhaustive generation of benzenoid structures by first proposing a general CSP model. Then we present several extensions of this model, some of which allow us to take into account the notion of patterns that is currently the subject of many studies in theoretical chemistry. Secondly, we propose a method for calculating the resonance energy based on a CSP model which allows us to evaluate the aromaticity of a benzenoid. Finally, we use CP to solve several secondary problems (such as the calculation of Clar structures) and describe the BenzAI software that implements all the work of this thesis. We also present the first applications of these contributions from the point of view of theoretical chemistry.

Keywords: constraint programming, CSP, enumeration, graph theory, theoretical chemistry, polycyclic aromatic hydrocarbons, benzenoid, aromaticity

Remerciements

Je souhaite profiter de la rédaction de cette thèse afin de remercier toutes les personnes y ayant contribué, de près comme de loin.

Les travaux présentés dans ce manuscrit sont avant tout issus d'un travail d'équipe important. Ainsi, je souhaiterais dans un premier temps remercier mes encadrants Cyril TERRIOUX et Yannick CARISSAN ainsi que Nicolas PRCOVIC et Denis HAGEBAUM REGNIER sans lesquels je n'aurais jamais pu mener ce projet à bien.

Je tiens également à remercier mes rapporteurs Christophe LECOUTRE et Simon DE GIVRY pour leur lecture minutieuse de mon manuscrit. Au même titre, je souhaiterais remercier Christine SOLNON, Aude SIMON, Philippe VISMARA ainsi que Stéphane HUMBEL d'avoir accepté de participer au jury.

Je remercie bien entendu les membres de mon équipe pour l'atmosphère conviviale qui règne au sein du laboratoire ainsi que tout le personnel du LIS qui nous facilite la vie au quotidien.

Pour finir, je souhaite remercier mes parents ainsi que tous mes amis de m'avoir aidé à tenir bon et soutenu dans les moments difficiles.

Table des matières

Table des figures

Liste des tableaux

Liste des modèles et des algorithmes

Liste des notations

- $\mathscr{B}_1$: ensemble de 48 benzénoïdes bien connus des chimistes 143
- $\mathscr{B}_2$: ensemble de 50 benzénoïdes catacondensés ayant entre 5 et 10 hexagones 144
- $\mathscr{B}_3$: ensemble des benzénoïdes rectangulaires de 5 lignes et de 1 à 20 colonnes 144
- $B = (V, E)$: un benzénoïde représenté sous la forme d'un graphe 55
- $B_h = (V_h, E_h)$: le graphe d'hexagones du benzénoïde $B = (V, E)$ 82
- B_h^k graphe d'hexagones étendu du benzénoïde B auquel on ajoute k couronnes d'hexagones négatifs 119
- $B_h^{c(k(n))}$: le graphe d'hexagones du coronénoïde de taille $k(n)$ 83
- $B - B[\mathscr{C}]$: sous-graphe du benzénoïde B obtenu en retirant à B le sous-graphe induit par le cycle $\mathscr{C}$ et son intérieur 141
- $\mathscr{C}$: cycle de $\mathscr{C}^*$ 140
- $E(b, h)$: énergie de résonance locale de l'hexagone h du benzénoïde B 140
- e_i : variable booléenne (utilisée par $\mathscr{M}_{i_1}$) traduisant la présence du fragment F_i dans le benzénoïde généré 122
- $E_P = \{P^1, \ldots, P^l\}$: ensembles de motifs spécifiés par l'utilisateur 126
- f_i : variable entière (utilisée par $\mathscr{M}_{i_2}$) spécifiant à quel hexagone du motif l'hexagone i du coronénoïde englobant est relié 123
- f_i^j : variable entière (utilisée par $\mathscr{M}_{m_2}^1$) spécifiant à quel hexagone du motif P^j l'hexagone i du coronénoïde englobant est relié 128
- h_{UB} : borne supérieure sur le nombre d'hexagones en fonction des critères choisis par l'utilisateur 110
- $\mathscr{K}(B)$: l'ensemble des structures de Kekulé du benzénoïde B 56
- $|\mathscr{K}(B)|$: le nombre de structures de Kekulé du benzénoïde B 56
- k_m : ordre du motif M 118
- $k(n)$: la taille du plus petit coronénoïde pouvant englober tous les benzénoïdes de n hexagones 83
- $M = (P_+, P_-, P_\circ)$: motif 118
- m_c : le nombre d'arêtes de $B_h^{c(k(n))}$ 84

Première partie

Introduction

Dans cette thèse, nous nous intéressons à des problèmes de chimie, et plus particulièrement de chimie théorique. Ce domaine étant extrêmement vaste, nous nous sommes concentrés sur des problématiques liées aux *hydrocarbures aromatiques polycycliques* (ou *HAP*). Les HAP sont une sous-famille des *hydrocarbures* (des molécules contenant exclusivement du carbone et de l'hydrogène) dont les atomes de carbone forment des cycles fusionnés de différentes tailles. Ces cycles sont généralement des cycles benzéniques, c'est-à-dire des cycles composés de six atomes de carbones en forme d'hexagones. Les HAP contenant uniquement des cycles de taille 6 sont appelés *benzénoïdes*. Ces derniers constituent une grande partie des HAP.

Les HAP sont étudiés et utilisés dans de nombreux domaines. Dans un environnement naturel, ils sont créés par la combustion incomplète du carbone contenu dans les combustibles (LUCH 2005). Ils constituent également un sujet de recherche populaire dans les sciences des matériaux. Par exemple, ils sont utilisés pour stocker ou transporter de l'énergie en nanoélectronique moléculaire (WU, PISULA et MÜLLEN 2007 ; AUMAITRE et MORIN 2019) ou encore en synthèse organique (RIEGER et MÜLLEN 2010 ; NARITA, WANG, FENG et al. 2015). Ils sont aussi beaucoup étudiés en chimie interstellaire, car on les soupçonne d'être présents dans des environnements interstellaires ou circumstellaires. Ils joueraient notamment le rôle de catalyseur pour des réactions chimiques ayant lieu dans l'espace (DRAINE 2011).

L'étude des HAP au fil du temps

L'histoire des hydrocarbures aromatiques polycycliques débute en 1825 avec la découverte du benzène (voir la figure 0.1 (a)) par Faraday (FARADAY 1825). Cette découverte est à l'origine de tous les travaux sur les HAP qui ont été effectués par la suite. Cette molécule présente des capacités peu communes, se distinguant notamment par sa grande stabilité énergétique.

Suite aux travaux de Faraday, certaines molécules présentant des propriétés similaires à celles du benzène ont commencé à être qualifiées d'aromatiques. L'origine de cet adjectif provient du fait que ces dernières ont tendance à émettre une odeur relativement plaisante. Depuis le début, les chimistes ont éprouvé des difficultés à caractériser l'aromaticité et différentes définitions lui ont été données au fil du temps comme nous allons le voir par la suite.

Les structures de Kekulé

Soixante ans plus tard, Kekulé propose en 1865 la représentation 1,3,5-cyclohexatriène pour le benzène. Cela introduira la notion de structure de Kekulé (KEKULÉ 1865) (voir la figure 0.1 (b-c)). En 1897, Thomson découvre les électrons (NAVARRO 2012 ; THOMSON 1897 ; THOMSON 1907). On en déduit ensuite que les doubles liaisons des structures de Kekulé des HAP sont formées par les électrons délocalisés de chaque atome (baptisés *électrons-π*). Ces travaux ont permis de mettre en évidence que ces électrons-π sont constamment en mouvement et forment des liaisons supplémentaires entre eux. Ces dernières correspondent aux liaisons doubles que l'on retrouve dans les structures de Kekulé des composés aromatiques.

La règle de Hückel

En 1937, Hückel (HÜCKEL 1937) présente une nouvelle définition de l'aromaticité encore utilisée de nos jours. Cette dernière stipule qu'un composé aromatique doit être cyclique, planaire et posséder $4n+2$ électrons-π délocalisés. Cette délocalisation induit des propriétés moléculaires intéressantes telles qu'une stabilité thermodynamique, une uniformisation de la taille des liaisons des cycles, des propriétés magnétiques, une réactivité particulière, ... Ces propriétés sont introuvables dans des systèmes non aromatiques. De manière plus formelle, l'aromaticité correspond alors au gain d'énergie obtenu en rendant cyclique une molécule non cyclique (la figure 0.2 illustre ce procédé pour le benzène). En effet, le fait de créer un cycle

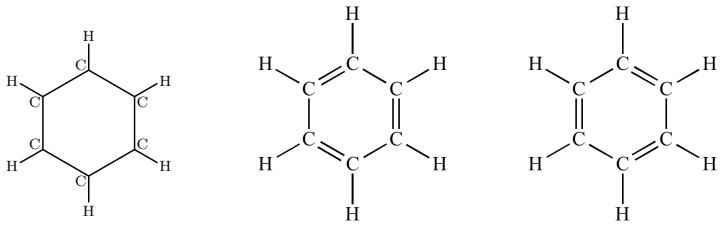


FIGURE 0.1. – Le benzène (a) et ses deux structures de Kekulé (b-c)

permet de stabiliser le système, cette stabilisation se traduisant par un gain d'énergie. Il est impossible de calculer précisément l'aromaticité d'un benzénoïde autre que le benzène étant donné que ce dernier est le seul à avoir une unique version cyclique. La figure 0.3 décrit ce problème pour le cas du naphtalène. Par conséquent, on se contente d'estimations pour les molécules autres que le benzène. De manière générale, l'aromaticité peut être estimée localement en attribuant une valeur à chaque cycle de la molécule considérée, ou globalement en lui attribuant une valeur unique. On parle alors d'aromaticité locale ou globale. De plus, les chimistes considèrent différents types de méthodes permettant d'estimer l'aromaticité d'un benzénoïde. Elles peuvent reposer notamment sur des critères *topologiques*, *énergétiques* ou *magnétiques*.

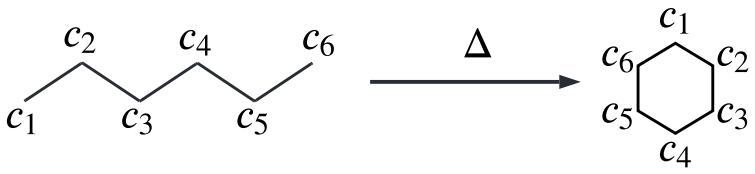


FIGURE 0.2. – Aromaticité du benzène

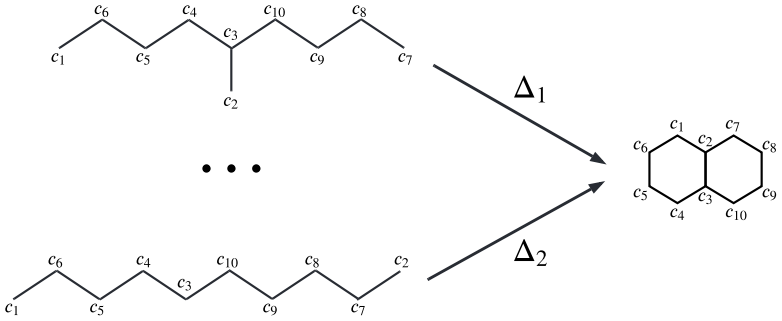


FIGURE 0.3. – Aromaticité du naphtalène

L'énergie de résonance

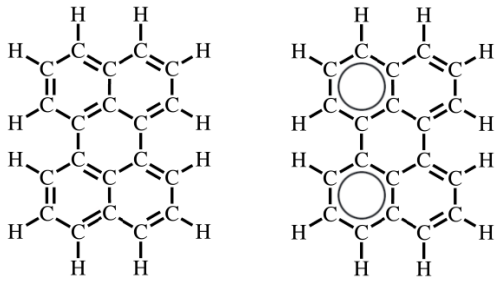
En 1952, Dewar (DEWAR 1952) introduit la notion d'*énergie de résonance*, qui consiste à caractériser l'aromaticité d'une molécule par l'énergie induite par le déplacement de ses électrons. Ensuite, en 1977, Randić (RANDIĆ 1976a) propose une méthode pour calculer cette énergie basée sur des éléments de théorie des graphes. Cette méthode consiste à énumérer les *circuits conjugués* de chacune de ses structures de Kekulé. Un circuit conjugué est un cycle dont les arêtes alternent entre liaison simple et double. Elle est néanmoins spécifique aux benzénoïdes. En 1999, Lin et Fan (LIN et FAN 1999) présentent un algorithme capable de

calculer l'énergie de résonance d'un benzénoïde donné. Ce dernier consiste à énumérer l'intégralité des structures de Kekulé du benzénoïde pour pouvoir ensuite identifier les différents circuits conjugués. Le principal inconvénient de cette méthode est qu'elle requiert l'énumération de l'ensemble des structures de Kekulé de la molécule considérée. Or, ce nombre est potentiellement exponentiel et l'algorithme devient rapidement inefficace quand on considère des benzénoïdes de grandes tailles. Pour pallier ce problème, Lin a présenté en 2000 (LIN 2000) un autre algorithme n'ayant pas à effectuer cette énumération. En contrepartie, il se limite à l'identification des circuits constitués d'au plus 18 liaisons et se contente donc de fournir une approximation de l'énergie de résonance au lieu de la calculer exactement.

Pour finir, en 1935, Pauling et al. introduisent la notion de *ring bond order (RBO)* (PAULING, BROCKWAY et BEACH 1935) qui permet également d'estimer l'aromaticité d'un benzénoïde. Cette notion se veut complémentaire à l'énergie de résonance. Le ring bond order d'un hexagone est calculé en fonction du nombre de fois où chacune de ses arêtes correspond à une liaison double dans une structure de Kekulé.

Les couvertures de Clar

En 1972, Clar (CLAR 1972) introduit la notion de *couverture de Clar* (également appelée *structure de Clar*). Une couverture de Clar peut être vue comme une factorisation de deux structures de Kekulé : des ronds (appelés *ronds de Clar*) sont placés sur les hexagones qui admettent trois doubles liaisons. La figure 0.4 décrit une des structures de Kekulé du périlène (a) et la couverture de Clar qui lui est associée (b). Clar présente également une méthode permettant d'estimer l'aromaticité locale d'un benzénoïde en attribuant à chaque hexagone une valeur dépendant du nombre de ronds qu'il admet dans l'intégralité des couvertures de Clar de la molécule traitée.



(a) (b)

FIGURE 0.4. – Une structure de Kekulé du périlène (a) et sa couverture de Clar associée (b)

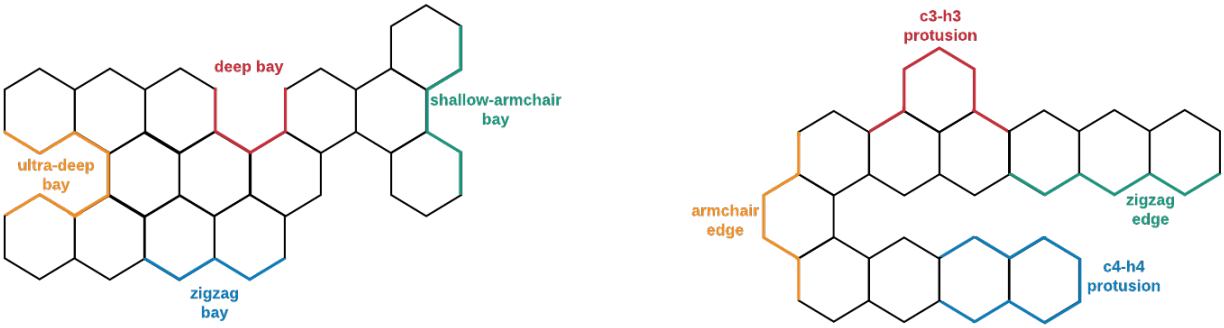
La méthode *Nucleus Independent Chemical Shift (NICS)*

En 1996, Dransfeld et al. présentent une nouvelle méthode capable d'estimer l'aromaticité d'un PAH baptisée *Nucleus Independent Chemical Shift (NICS)* (DRANSFELD, SCHLEYER et VAN EIKEMA HOMMES 1996). Cette dernière repose sur des calculs de chimie quantique et consiste à appliquer un champ magnétique perpendiculaire à la molécule traitée, puis à observer le comportement de ses électrons-π. Elle est actuellement la méthode de référence des chimistes. Son principal inconvénient est son coût très élevé dû aux lourds calculs de chimie quantique requis. Pour plus d'informations sur la méthode NICS, nous conseillons fortement au lecteur de se référer à (CHEN, WANNERE, CORMINBOEUF et al. 2005).

Étude de la topologie de la bordure des HAP

Depuis ces dernières années, les chimistes ont tendance à caractériser les HAP en fonction de la topologie de leur bordure (LIU et FENG 2020; DUMSLAFF, GU, PATERNÒ et al. 2020; YANO, MITOMA, ITO et al. 2019; EHRENHAUSER 2015). Pour cela, ils considèrent deux familles de bordures : les *baies* qui sont constituées

des bordures dénommées *deep bay* (ou *cove*), *shallow-armchair bay* (ou *armchair*), *zigzag bay* et *ultra-deep bay* (ou *fjord*) et les *protusions*, qui elles, sont constituées des bordures nommées *c3-h3 protusion*, *c4-h4 protusion*, *zigzag edge* (ou *zigzag*) et *armchair edge*. Chacun de ces types de bordures est défini dans la figure 0.5. Les benzénoïdes admettant ces types de bordures peuvent présenter des propriétés optiques et magnétiques différentes, ce qui justifie l'intérêt que leur portent les chimistes. Par la suite, nous utiliserons également la notion de *motifs* pour définir ces bordures.



(a) (b)

FIGURE 0.5. – Les *baies* (a) et les *protusions* (b)

Les objectifs de la thèse

Dans le cadre de cette thèse, nous nous focaliserons donc plus particulièrement sur les benzénoïdes. Plus précisément, nous nous intéressons principalement à deux problématiques : être capable de générer, de manière exhaustive, des structures de benzénoïdes possédant des propriétés spécifiques et estimer l'aromaticité d'un benzénoïde donné. Dans cette partie, nous expliquons l'importance de ces deux problématiques pour les chimistes et expliquerons pourquoi nous avons choisi de les étudier.

Générer des structures de benzénoïdes satisfaisant des propriétés spécifiques

Cette problématique consiste à être capable de générer des ensembles exhaustifs de structures de benzénoïdes répondant à un ensemble de propriétés spécifiques. Ces dernières peuvent être structurelles (posséder des trous, admettre une forme spécifique, ...) ou encore chimiques (nombre de carbones/hydrogènes/hexagones, ...). Cette problématique s'accompagne de plusieurs contraintes. Premièrement, il doit exister une garantie d'exhaustivité sur les ensembles générés. Pour justifier cela, nous pouvons, par exemple, citer les travaux de Bouwmann et al. (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019), qui ont présenté en 2019 une étude sur les spectres infrarouges de certains benzénoïdes. Ces derniers étaient issus de la *NASA Ames PAH database* (https://www.astrochemistry.org/pahdb/). Or, nous nous sommes rendus compte que l'ensemble de benzénoïdes utilisé n'était pas exhaustif. Être capable de générer des ensembles exhaustifs de benzénoïdes satisfaisant diverses propriétés pourrait permettre d'obtenir des résultats plus complets. Ensuite, la seconde contrainte est d'avoir un outil qui soit relativement efficace en termes de temps, ce qui n'est pas forcément évident à la vue de la forte combinatoire du problème. Pour finir, une troisième contrainte est d'être capable de facilement pouvoir ajouter de nouvelles propriétés et de les combiner entre elles afin de rester à l'écoute des besoins des chimistes.

Notons qu'il existe déjà des outils de génération performants en termes de temps d'exécution (on peut notamment citer l'algorithme de Brinkmann et al. (BRINKMANN, CAPOROSSI et HANSEN 2002), l'algorithme de

Caporossi et Hansen (CAPOROSSI et HANSEN 1998) ou encore le logiciel CaGe (https://caagt.ugent.be/CaGe/). Toutefois, bien que ces méthodes soient efficaces, elles présentent plusieurs inconvénients majeurs, le principal étant qu'elles admettent peu de critères de génération, ce qui se traduit par un manque de flexibilité. De plus, intégrer le moindre nouveau critère requiert de délicates modifications au niveau de l'algorithme de génération, ce qui demande un lourd effort en termes de conception et de programmation. Pour pallier ce problème, nous nous fixons l'objectif de proposer un outil plus flexible, admettant un grand nombre de critères de génération et permettant à l'utilisateur de sélectionner/combiner ces derniers tout en conservant des performances raisonnables.

Estimer l'aromaticité d'un benzénoïde donné

Nous avons vu plus haut que l'aromaticité n'était pas quelque chose de mesurable, mais qu'il était seulement possible de l'estimer. Nous avons ensuite évoqué différentes méthodes capables de faire cela. Tout d'abord, la méthode *Nucleus Independent Chemical Shift (NICS)* (DRANSFELD, SCHLEYER et VAN EIKEMA HOMMES 1996), qui repose sur des calculs de chimie quantique, constitue la méthode de référence des chimistes, bien qu'elle soit relativement coûteuse. Nous avons ensuite mentionné différentes méthodes exploitant des notions de théorie des graphes. Cependant, ces dernières présentent un inconvénient commun : chacune d'entre elles demande d'énumérer des ensembles admettant une forte combinatoire comme l'ensemble des structures de Kekulé ou l'ensemble des structures de Clar. De plus, il n'existe, à notre connaissance, aucune implémentation disponible de ces méthodes (mis à part pour NICS). L'objectif ici est de fournir un outil se basant sur l'une de ces méthodes capables d'estimer l'aromaticité d'un benzénoïde donné de manière efficace, et de le positionner par rapport à l'existant.

La programmation par contraintes, un candidat idéal

Les problématiques de chimie théorique que nous venons d'évoquer peuvent être vues comme des problèmes d'informatique classiques. Il convient donc maintenant de déterminer le meilleur paradigme afin de les résoudre le plus efficacement possible. Intéressons-nous, par exemple, au problème de génération de structures de benzénoïdes. Afin de résoudre correctement cette problématique, différentes conditions doivent être satisfaites :

- l'efficacité : pour être utilisable par les chimistes, l'approche que nous considérerons doit être capable de générer les structures de benzénoïdes désirées dans un temps aussi raisonnable que possible.
- l'exhaustivité : nous souhaitons naturellement avoir une garantie de complétude sur les ensembles de structures qui seront générés.
- la flexibilité : nous voulons être capable de générer des structures de benzénoïdes pouvant admettre un grand nombre de propriétés différentes. Nous souhaitons également pouvoir les combiner entre elles. Cela constitue une collection de problèmes différents, mais, qui sont relativement proches les uns des autres.

Tâchons maintenant de déterminer quelle approche serait la plus susceptible de satisfaire les besoins énoncés ci-dessus. Si l'on considère le critère de flexibilité. Ce dernier exclut l'idée de mettre au point une méthode ad hoc pour chaque nouvelle propriété que l'on souhaite modéliser (ce qu'ont fait Brinkmann et al. (BRINKMANN, CAPOROSSI et HANSEN 2002) en proposant un algorithme capable de générer les structures de benzénoïdes admettant un nombre donné d'hexagones). En effet, cela rendrait l'ajout et la combinaison de propriétés beaucoup trop compliqués et demanderait de gros efforts en termes de programmation. Ce critère exclut donc, a priori, l'utilisation de paradigmes tels que la programmation impérative ou fonctionnelle. Des approches comme la *recherche opérationnelle (RO)* (HILLIER et LIEBERMAN 2001) et la *programmation par contraintes (PPC)* (ROSSI, BEEK et WALSH 2006a) semblent plus adaptées.

La *programmation par contraintes* est un paradigme de l'intelligence artificielle apparu dans les années 1970. Résoudre un problème à l'aide de la programmation par contraintes consiste à représenter ce dernier selon un formalisme donné, puis à résoudre l'instance ainsi produite à l'aide d'un solveur. Les différents formalismes qu'elle propose consistent à représenter un problème sous la forme d'une *instance* constituée d'un ensemble de *variables*. Les valeurs possibles de ces variables sont définies par leurs *domaines* et ces dernières vont être liées entre elles par des *contraintes*. Une instance modélisée de la sorte peut ensuite être résolue par un *solveur* générique. Par générique, on entend ici que le solveur n'est pas dédié à un type d'instances particulières, mais, est en mesure de traiter des instances de nature et d'origine quelconques. Un solveur est généralement le fruit d'une savante combinaison de plusieurs techniques (heuristiques de choix de valeur/variables, algorithmes de filtrage, ...). Ces solveurs sont améliorés en permanence et mis à l'épreuve lors de compétitions telles que la compétition *XCSP* annuelle (http://www.xcsp.org/competitions/). Parmi les plus performants, nous pouvons citer ACE [1] ou Choco Solver [2] (FAGES, LORCA et PRUD'HOMME s. d.)). Il en résulte que ces solveurs ont atteint une maturité et une efficacité qui leur permettent de traiter des problèmes dits difficiles en raison de leur forte combinatoire. Tout cela confère à la PPC une grande flexibilité qui permet de s'attaquer à des problèmes de natures variées (qu'ils soient académiques ou industriels) parmi lesquels nous pouvons citer le commerce, le transport, la bio-informatique (SIMONCINI, ALLOUCHE, DE GIVRY et al. 2015 ; WU 2004), la chimie (MANN et THIEL 2013), la fabrication automobile (ANTUORI, HÉBRARD, HUGUET et al. 2020), la robotique (COLLET, GOTLIEB, LAZAAR et al. 2020), ...

À ce stade, la programmation linéaire et la programmation par contraintes sont, en tous points, comparables. Ce qui peut faire la différence, en tout cas dans le cadre de cette thèse, c'est que la PPC est en mesure de proposer des variables et des contraintes spécifiques permettant de simplifier la modélisation des problèmes. Nous pouvons citer notamment les *variables de graphes* qui admettent des ensembles de graphes comme domaines et sur lesquelles nous pouvons appliquer des contraintes dédiées imposant, par exemple, la connexité du graphe qu'elles représentent. Les problématiques que nous traitons pouvant être vues comme des problèmes liés aux graphes (nous pouvons facilement voir qu'une molécule peut être représentée sous la forme d'un graphe), ce type de variable nous permettrait de les modéliser relativement simplement. De manière plus générale, nous pouvons aussi mentionner ici les *contraintes globales* (contraintes de différence, contraintes de sommes, ...) qui, en plus de simplifier la modélisation, contribuent à l'efficacité pratique des solveurs grâce à leurs algorithmes de filtrage parfaitement adaptés.

La programmation par contraintes semble donc être la candidate idéale. En effet, même si elle pourrait être, dans certains cas, moins performante qu'une méthode ad hoc, elle reste néanmoins un très bon compromis entre le temps de résolution et la flexibilité tout en réduisant fortement les temps de conception et de développement.

Plan de la thèse

La suite de cette thèse est organisée de la manière suivante :

- La partie II détaillera toutes les notions liées à la programmation par contraintes (chapitre 1) et à la chimie théorique (chapitre 2) nécessaires pour comprendre les travaux qui seront présentés par la suite.

- La partie III détaillera l'intégralité des travaux effectués pendant cette thèse :

  - Les chapitres 3 et 4 seront consacrés à décrire les modèles que nous avons mis en place afin de générer des structures de benzénoïdes et à les évaluer expérimentalement. Le chapitre 3 présentera le problème de génération de structures de benzénoïdes dans toute sa généralité. Il détaillera également toutes les propriétés que nous avons modélisées. Le chapitre 4 sera, quant à

1. https://github.com/xcsp3team/ace
2. https://choco-solver.org/

lui, exclusivement consacré à la description de modèles permettant de générer des benzénoïdes contenant (ou non) une (ou plusieurs) forme(s) donnée(s).

- Le chapitre 5 décrira les travaux effectués liés à l'estimation de l'aromaticité d'un benzénoïde donné en calculant son énergie de résonance. Il présentera d'abord une implémentation de la méthode de Lin & Fan (LIN et FAN 1999) utilisant la programmation par contraintes. Puis, il décrira une amélioration de l'algorithme de Lin (LIN 2000) reposant, elle aussi, sur la PPC. Pour finir, il proposera une comparaison expérimentale entre les deux méthodes énoncées ci-dessus et la méthode NICS.

- Le chapitre 6 sera consacré aux outils développés au cours de cette thèse. Il présentera d'abord notre logiciel *BenzAI* dans la section 6.1, qui implémente les modèles et méthodes proposées dans les chapitres 3 à 5 ainsi que des fonctionnalités additionnelles comme le calcul des structures de Clar. Par la suite, il décrira notre base de données BenzDB dans la section 6.2. Cette dernière contient un certain nombre d'informations (valeurs NICS, spectres infrarouges, ...) sur l'ensemble des benzénoïdes possédant au plus 9 hexagones.

- Le chapitre 7 détaillera, pour sa part, les premières contributions à la chimie pouvant découler de nos travaux et est organisé de la manière suivante :

  — La section 7.2 présentera les résultats de deux méthodes permettant d'estimer l'aromaticité d'un benzénoïde plus rapidement que les méthodes présentées dans le chapitre 5 respectivement basées sur l'énumération des *couvertures de Clar* (voir sous-section 2.4.5) et sur les calculs des *Ring Bond Order (RBO)* (voir sous-section 2.4.6). En contreparties, les résultats obtenus par ces méthodes admettent un manque de finesse comparé à celles présentées dans le chapitre 5 et à NICS.

  — La section 7.3 présente deux manières d'utiliser notre logiciel BenzAI (présenté dans la section 6.1) afin d'accompagner les chimistes dans la synthèse de molécules. La sous-section 7.3.1 présentera une manière d'exhiber des benzénoïde pouvant être synthétisées à partir d'autres benzénoïdes partiellement aromatiques. La sous-section 7.3.2 présentera quant à elle une manière d'utiliser BenzAI afin de mettre en évidence des benzénoïdes admettant un grand nombre d'électrons radicalaires synthétisables par trimérisation.

  — La section 7.4 détaillera pourquoi notre base de données BenzDB (présentée dans la section 6.2) est capable de fournir aux chimistes une meilleure interprétation des spectres infrarouges des benzénoïdes possédant au plus neuf hexagones.

- Pour finir, la partie IV conclura cette thèse et décrira différentes perspectives aux travaux présentés.

Deuxième partie

État de l’art

1. Programmation par contraintes

1.1. Introduction

La *Programmation par contraintes (PPC)* (ROSSI, BEEK et WALSH 2006b) est une branche de l'intelligence artificielle apparue dans les années 70. Elle se situe à l'interface d'autres disciplines telles que la théorie des graphes, la recherche opérationnelle, ou encore les bases de données relationnelles et s'attache à résoudre des problèmes fortement combinatoires (a minima NP-complets). Contrairement à des paradigmes tels que la programmation impérative ou fonctionnelle, qui consistent à écrire un algorithme afin de résoudre un problème, la programmation par contraintes est une forme de programmation déclarative qui consiste simplement à décrire ce problème. Eugène C. Freuder (FREUDER et ELFE 1996), un des pionniers de cette discipline, l'a décrite de la manière suivante :

≪ *Constraint programming represents one of the closest approaches computer science has made to the Holy Grail of programming : the user states the problem, the computer solves it.* ≫

La plus grande force de la programmation par contraintes est donc sa forte flexibilité. En effet, l'utilisateur va pouvoir décrire divers problèmes selon un formalisme donné, et ensuite, les fournir en entrée à un des solveurs existants afin de les résoudre. Ainsi, l'effort de conception et de développement s'en trouve réduit comparé à un cadre de programmation plus classique où il aurait fallu, pour chacun de ces problèmes, écrire un nouvel algorithme et l'implémenter. Ce premier point ainsi que l'efficacité des solveurs récents permettent de l'utiliser afin de résoudre des problèmes fortement combinatoires issus de domaines très variés comme le commerce, le transport, la bio-informatique, l'agriculture, … Dans ce manuscrit, nous l'exploiterons d'ailleurs pour résoudre des problèmes de chimie théorique.

Ce paradigme repose sur différents formalismes permettant de représenter les problèmes. Le plus simple est celui reposant sur le problème SAT (satisfaisabilité d'une formule booléenne (MARQUES SILVA, LYNCE et MALIK 2009)). Ce formalisme consiste à représenter un problème sous la forme de clauses (c'est-à-dire une conjonction de disjonction de variables booléennes ou de leurs négations). Dans le cadre de cette thèse, nous nous intéressons au formalisme des *problèmes de satisfaction de contraintes* (*Constraint Satisfaction Problem (CSP)*), qui consiste à représenter le problème par un ensemble de *variables*, chacune possédant un ensemble de valeurs qu'elle peut prendre (appelé *domaine*). Ces variables vont être liées entre elles par des *contraintes*. L'objectif ici est d'affecter toutes les variables sans qu'aucune contrainte ne soit violée. Par ailleurs, il faut savoir qu'il existe des variantes de ce formalisme permettant d'exprimer des problèmes d'optimisation, comme les formalismes *COP* (*Constraint Optimisation Problem* (DECHTER 2006)) ou *WCSP* (*Weighted Constraint Satisfaction Problem* (COOPER et SCHIEX 2004 ; FREUDER et WALLACE 1992)).

Bien que résoudre un CSP soit un problème complexe, la communauté a proposé des solveurs performants capables de résoudre un grand nombre d'instances. Ces derniers ont notamment bénéficié de fortes améliorations au cours des dernières décennies.

Dans ce chapitre, nous allons rappeler les principales notions et techniques qui seront utilisées dans la suite de ce manuscrit. Pour plus d'informations au sujet des CSP, nous nous permettons de rediriger le lecteur vers (APT 2003a ; ROSSI, BEEK et WALSH 2006b ; DECHTER 2003 ; LECOUTRE 2009). Il convient également de garder à l'esprit que l'objectif de cette thèse n'est pas d'améliorer la résolution des CSP, mais plutôt d'utiliser cette approche comme une *boite noire* permettant de modéliser et résoudre des problèmes de chimie théorique.

Ce chapitre est organisé de la manière suivante : la section 1.2 présentera de manière générale le problème de satisfaction de contraintes (CSP). Dans la section 1.3, nous parlerons des principales définitions liées à ce problème. La section 1.4 abordera, quant à elle, les principales méthodes utilisées de nos jours afin de résoudre une instance CSP (elle mentionnera notamment les algorithmes de parcours et de filtrage, ainsi que les heuristiques de choix de variables/valeurs et les solveurs). Pour finir, la section 1.5 détaillera les différentes problématiques liées à la modélisation et introduira de nouveaux types de variables et de contraintes plus sophistiquées permettant de modéliser des problèmes plus efficacement.

1.2. Le Problème de Satisfaction de Contraintes (CSP)

Comme mentionné dans la section 1.1, une instance du *problème de satisfaction de contraintes (CSP)* se définit par un ensemble de *variables* qui auront pour but de représenter les objets du problème que l'on souhaite résoudre et/ou certaines de leurs propriétés. Chacune de ces variables admet un *domaine* qui contient toutes les valeurs que cette dernière peut prendre. Pour finir, nous considérons également un ensemble de *contraintes* ayant pour but d'exprimer les propriétés et/ou les relations entre les différents objets du problème. Plus formellement, une instance CSP peut être définie de la manière suivante :

Définition 1.2.1 (MONTANARI 1974) *Une* instance du problème de satisfaction de contraintes (CSP) *est définie par un triplet* $P = (X, D, C)$ *où :*

- $X = \{x_1, x_2, \ldots, x_n\}$ *est un ensemble de n variables,*
- $D = \{D_{x_1}, D_{x_2}, \ldots, D_{x_n}\}$ *est un ensemble de domaines finis. La variable* x_i *prend ses valeurs dans le domaine* D_{x_i}*,*
- $C = \{c_1, c_2, \ldots, c_m\}$ *est un ensemble de m contraintes. Une contrainte est caractérisée par un couple* $(S(c_i), R(c_i))$ *où :*
  1. $S(c_i) = \{x_{i_1}, x_{i_2}, \ldots, x_{i_q}\} \subseteq X$ *est la portée de* c_i *(c'est-à-dire l'ensemble de variables sur lesquelles porte la contrainte) avec* $|S(c_i)|$ *l'arité de la contrainte* c_i*,*
  2. $R_{c_i} \subseteq D_{x_{i_1}} \times D_{x_{i_2}} \times \ldots D_{x_{i_q}}$ *est sa relation de compatibilité. Le rôle de cette relation est de définir les combinaisons de valeurs autorisées par la contrainte.*

L'arité maximale des contraintes (notée r) est égale à $max_{c_i \in C}|S(c_i)|$*. La taille maximale des domaines (notée d) est égale à* $max_{x_i \in X}|D_{x_i}|$*. On note* $n = |X|$ *le nombre de variables de l'instance.*

Une contrainte portant sur une unique variable est appelée *contrainte unaire*. Si elle porte sur deux variables, on parle de *contrainte binaire* et dans tous les autres cas, on parle de *contrainte n-aire* (ou *non binaire*). Cette notion de binarité se transmet également à l'instance CSP :

Définition 1.2.2 *Une instance CSP est dite* binaire *si chacune de ses contraintes* $c_i \in C$ *est binaire ou unaire. Si au moins une de ses contraintes est n-aire, alors l'instance est dite n-aire.*

D'un point de vue mathématique, une relation de compatibilité d'une contrainte est définie comme l'ensemble de combinaisons de valeurs que la contrainte autorise. D'un point de vue pratique, la manière la plus simple de représenter une relation est de lister tous les tuples de valeurs qu'elle autorise (on parle alors de *supports*). Dans le cas où une contrainte admet moins de tuples interdits que de tuples autorisés, on peut souhaiter représenter plutôt ces tuples interdits (on parlera alors de *conflits*) afin de réduire l'espace consommé en mémoire. Dans les deux cas, une relation exprimée de cette manière est appelée *relation en extension*. Afin de représenter facilement ce type de contraintes, on se propose de définir les contraintes $Table^+$ et $Table^-$ (appelées contraintes de table positive et négative respectivement dans (LECOUTRE 2009)) :

Définition 1.2.3 *Soient Y un sous-ensemble de variables de X et* R^+ *(resp.* R^-*) un ensemble de tuples autorisés (resp. interdits) défini sûr* $\prod_{y \in Y} D_y$*. La contrainte* $Table^+(X, R^+)$ *(resp.* $Table^-(X, R^-)$*) est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = R^+$ *(resp.* $R(c) = R^-$*).*

L'exemple 1.2.1 nous montre une manière de représenter le problème bien connu du sudoku sous la forme d'une instance CSP utilisant des contraintes en extension.

Exemple 1.2.1 *Le sudoku est un jeu bien connu consistant à remplir une grille de taille* 9×9 *avec une série de chiffres allant de 1 à 9 de telle sorte que chaque ligne, colonne ou région (les 9 sous-grilles de taille* 3×3 *constituant la grille totale) n'ait pas de valeurs identiques. Les chiffres initialement présents dans la grille sont appelés* indices *On souhaite dans cet exemple représenter ce problème sous la forme d'une instance CSP. Dans notre cas, les variables représenteront l'association d'un chiffre à une case. Ainsi, nous allons considérer une variable* x_i *pour chaque case de la grille (*$1 \leq i \leq 81$*) ayant chacune pour domaine l'ensemble des entiers compris entre 1 et 9 (inclus). Si la variable* x_i *est égale à* j*, cela signifiera que la* i*-ème case de la grille aura pour valeur* j*. Les contraintes seront utilisées pour spécifier qu'une valeur ne pourra pas apparaître deux fois dans la même ligne, région ou colonne. Plus formellement, le problème du sudoku peut être représenté sous la forme de l'instance CSP* $P = (X, D, C)$ *suivante :*

- $X = \{x_1, x_2, \dots x_{81}\}$ *(une variable pour chaque case de la grille, la valeur de la variable* x_i *représente celle de la* i*-ème case). Ces variables sont numérotées comme illustré dans la figure 1.1), on considère les sous-ensembles de* X *suivants (voir figure 1.2) :*
  - $\{L_1, L_2, \dots, L_9\}$, L_i *représentant l'ensemble des variables de la* i*-ème ligne,*
  - $\{C_1, C_2, \dots, C_9\}$, C_i *représentant l'ensemble des variables de la* i*-ème colonne,*
  - $\{R_1, R_2, \dots, R_9\}$, R_i *représentant l'ensemble des variables de la* i*-ème région.*
- $D = \{D_{x_1}, D_{x_2}, \dots, D_{x_{81}}\}$ *avec* $D_{x_i} = \{1, 2, \dots, 9\}, 1 \leq i \leq 81$
- $C = C_{lignes} \cup C_{colonnes} \cup C_{régions} \cup C_{indices}$ *avec :*
  - $C_{lignes} = \bigcup_{1 \leq i \leq 9} C_{L_i}$*, avec :*
    $C_{L_i} = \{Table^-(\{x_j, x_k\}, \{(1,1),(2,2),\dots,(9,9)\}) | (x_j, x_k) \in L_i \times L_i \text{ t.q. } j \neq k\}$,
  - $C_{colonnes} = \bigcup_{1 \leq i \leq 9} C_{C_i}$*, avec :*
    $C_{C_i} = \{Table^-(\{x_j, x_k\}, \{(1,1),(2,2),\dots,(9,9)\}), (x_j, x_k) \in C_i \times C_i \text{ t.q. } j \neq k\}$,
  - $C_{régions} = \bigcup_{1 \leq i \leq 9} C_{R_i}$*, avec :*
    $C_{R_i} = \{Table^-(\{x_j, x_k\}, \{(1,1),(2,2),\dots,(9,9)\}), (x_j, x_k) \in R_i \times R_i \text{ t.q. } j \neq k\}$.
  - $C_{indices}$ *est un ensemble de contraintes unaires ayant pour but de fixer les valeurs des cases initialement remplies. Par exemple, si l'on considère la grille de la figure 1.3(a), on aurait :*

$$C_{indices} = \left\{ \begin{array}{l} Table^+(\{x_1\}, \{2\}), \\ Table^+(\{x_2\}, \{4\}), \\ Table^+(\{x_3\}, \{1\}), \\ \dots, \\ Table^+(\{x_{76}\}, \{3\}) \end{array} \right\}$$

D'une certaine manière, les contraintes de C_{lignes}*,* $C_{colonnes}$ *et* $C_{régions}$ *permettent d'exprimer le problème sudoku dans toute sa généralité, tandis que les contraintes de* $C_{indices}$ *permettent de préciser quelle grille spécifique de sudoku, nous souhaitons résoudre.*

Hormis les contraintes unaires de $C_{indices}$, nous avons choisi de représenter les tuples interdits des relations de chaque contrainte (c'est-à-dire les conflits). Dans notre cas, il est plus judicieux de représenter

les conflits (9 tuples) que les supports (72 tuples). Le choix de la représentation dépend grandement de la nature du problème initial.

Notons qu'un problème peut admettre plusieurs modélisations différentes. Le principal inconvénient des contraintes exprimées en extension est que le nombre de tuples énumérés peut être très élevé. Pour pallier ce problème, une possibilité est de représenter ces tuples sous la forme de prédicats mathématiques formés à partir d'opérateurs usuels $(+, -, \leq, <, =, \neq, >, \geq, \ldots)$ et portant sur les variables de l'instance. Par exemple, spécifier que deux variables x_1 et x_2 doivent avoir différentes valeurs peut être fait à l'aide du prédicat $x_1 \neq x_2$. Plus formellement :

Définition 1.2.4 (Contrainte en intention) *Soient Y un sous-ensemble de variables de X et ρ un prédicat portant sur les variables de Y. La contrainte $Intension(Y, \rho)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y | \rho(t)\ est\ vrai\}$.

Par abus de notation, on désignera généralement la contrainte $Intension(Y, \rho)$ sous la forme ρ, l'ensemble Y étant alors implicitement défini par les variables apparaissant dans ρ.

Pour illustrer davantage l'intérêt de ces contraintes, nous allons considérer une nouvelle modélisation du problème sudoku à l'aide de contraintes exprimées en intention. À noter que seul l'ensemble de contraintes sera modifié par rapport à l'exemple 1.2.1) :

Exemple 1.2.2 *Modélisation $P = (X, D, C)$ du problème du sudoku à l'aide de contraintes exprimées en intention :*

- $X = \{x_1, x_2, \ldots x_{81}\}$ *(on considère encore les sous-ensembles de X définis dans l'exemple 1.2.1),*
- $D = \{D_{x_1}, D_{x_2}, \ldots, D_{x_{81}}\}$ *avec* $D_{x_i} = \{1, 2, \ldots, 9\}, 1 \leq i \leq 9$,
- $C = C_{lignes} \cup C_{colonnes} \cup C_{régions} \cup C_{indices}$ *avec :*
  - $C_{lignes} = \bigcup_{1 \leq i \leq 9} C_{L_i}$, *avec* $C_{L_i} = \{x_j \neq x_k | (x_j, x_k) \in L_i \times L_i\ t.q.\ j \neq k\}$,
  - $C_{colonnes} = \bigcup_{1 \leq i \leq 9} C_{C_i}$, *avec* $C_{C_i} = \{x_j \neq x_k | (x_j, x_k) \in C_i \times C_i\ t.q.\ j \neq k\}$,
  - $C_{régions} = \bigcup_{1 \leq i \leq 9} C_{R_i}$, *avec* $C_{R_i} = \{x_j \neq x_k | (x_j, x_k) \in R_i \times R_i\ t.q.\ j \neq k\}$,
  - $C_{indices} = \{x_1 = 2, x_2 = 4, x_3 = 1, \ldots, x_{76} = 3\}$

Toute contrainte exprimée en intention peut également être exprimée en extension. En revanche, un tel changement n'est pas sans conséquences, chacune de ces représentations présentant ses avantages et ses inconvénients. En effet, représenter une contrainte en extension peut nécessiter un espace mémoire très important à cause du nombre de tuples énumérés. Une contrainte en intention demandera un espace mémoire moindre, mais son test de satisfaction sera souvent plus coûteux. De plus, cela a également un impact sur le coût de la propagation.

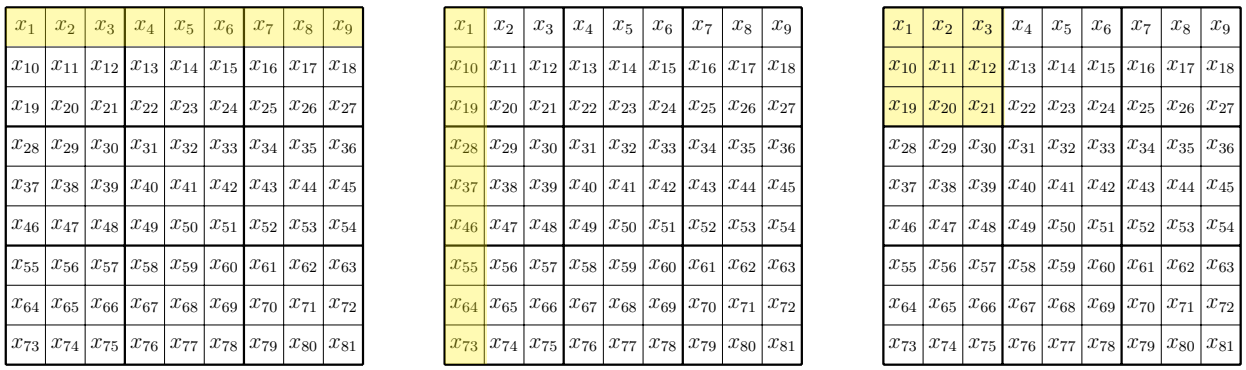
1.3. Définitions préliminaires

Dans cette partie, nous détaillerons les différentes notions mises en place afin de résoudre un CSP. Tout d'abord, introduisons la notion d'*affectation* (aussi appelée *instanciation* ou *assignation*) :

Définition 1.3.1 (Affectation) *Soit $P = (X, D, C)$ une instance CSP. On appelle* affectation *d'une variable x_i de X, l'attribution d'une valeur $v_i \in D_{x_i}$ à la variable x_i. Elle est notée $x_i = v_i$. Elle peut être étendue à tout*

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}
x_{28}	x_{29}	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}
x_{37}	x_{38}	x_{39}	x_{40}	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{46}	x_{47}	x_{48}	x_{49}	x_{50}	x_{51}	x_{52}	x_{53}	x_{54}
x_{55}	x_{56}	x_{57}	x_{58}	x_{59}	x_{60}	x_{61}	x_{62}	x_{63}
x_{64}	x_{65}	x_{66}	x_{67}	x_{68}	x_{69}	x_{70}	x_{71}	x_{72}
x_{73}	x_{74}	x_{75}	x_{76}	x_{77}	x_{78}	x_{79}	x_{80}	x_{81}

FIGURE 1.1. – Grille de sudoku, chaque variable entière représente une case



(a) (b) (c)

FIGURE 1.2. – Les ensembles de variables $L_1(a)$, $C_1(b)$ et $R_1(c)$

sous-ensemble $Y = \{x_{i_1}, x_{i_2}, \ldots, x_{i_q}\}$ *de* X. *Une affectation de* Y *est l'ensemble* $\{x_{i_1} = v_{i_1}, x_{i_2} = v_{i_2}, \ldots, x_{i_q} = v_{i_q}\}$ *qui associe à chaque variable* x_{i_p} *de* Y *(avec* $1 \leq p \leq q$*) une valeur* $v_{i_p} \in D_{x_{i_p}}$. *Si* $\mathscr{A}$ *est une affectation, on désignera par* $X_{\mathscr{A}}$ *l'ensemble des variables sur lesquelles elle porte.*

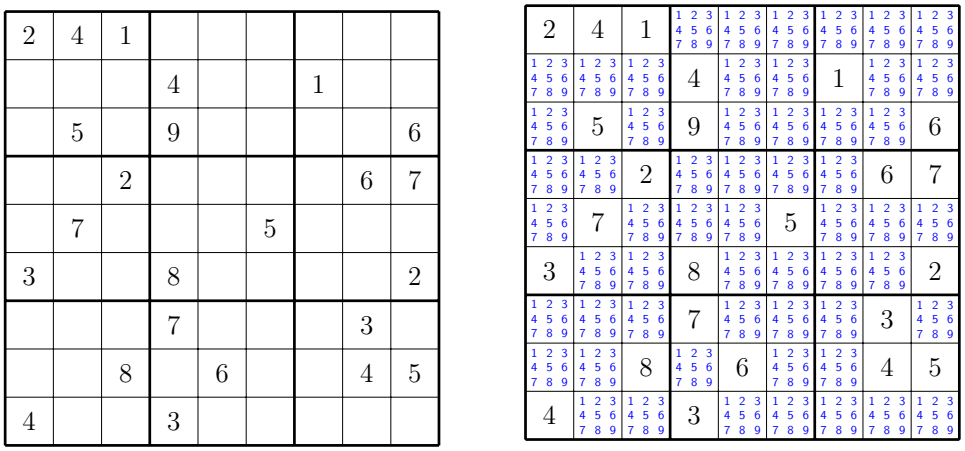
On peut représenter une affectation $\mathscr{A}$ sous la forme d'une association variable/valeur : $\mathscr{A} = \{x_{i_1} = v_{i_1}, x_{i_2} = v_{i_2}, \ldots, x_{i_q} = v_{i_q}\}$. On peut aussi la représenter sous la forme d'un tuple de valeurs $(v_{i_1}, v_{i_2}, \ldots, v_{i_q})$ (dans ce cas, l'ordre des variables est implicite). On dit qu'une affectation est *complète* si elle porte sur l'ensemble des variables de X. Dans le cas contraire, elle est dite *partielle*. On appelle *projection* la restriction de l'affectation $\mathscr{A}$ à un sous-ensemble de $X_{\mathscr{A}}$:

Définition 1.3.2 (Projection) *Soient* $\mathscr{A}$ *une affection et* Y *un sous-ensemble de* X. *La projection de* $\mathscr{A}$ *sur* Y, *notée* $\mathscr{A}[Y]$, *est la restriction de* $\mathscr{A}$ *aux variables de* Y.

Une contrainte peut être *satisfaite*, *violée* (ou ni l'un ni l'autre) par une affectation :

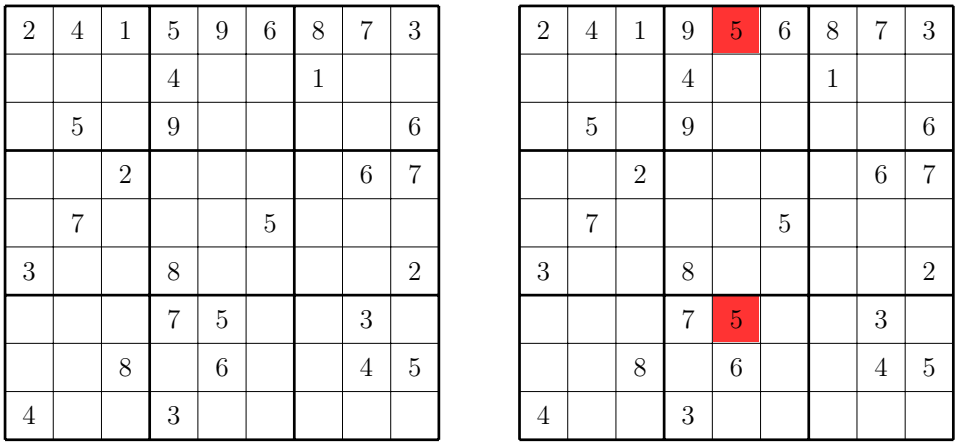
Définition 1.3.3 *Soient* $P = (X, D, C)$ *une instance CSP et* $\mathscr{A}$ *une affectation donnée. On dit que* $\mathscr{A}$ satisfait *la contrainte* $c_i = (S(c_i), R(c_i))$ *de* C *si* $S(c_i) \subseteq X_{\mathscr{A}}$ *et* $\mathscr{A}[S(c_i)] \in R(c_i)$. *Si* $S(c_i) \subseteq X_{\mathscr{A}}$ *et* $\mathscr{A}[S(c_i)] \notin R(c_i)$, *on dit que* $\mathscr{A}$ viole *la contrainte* c_i.

Par exemple, si l'on considère l'instance CSP associée à la grille de sudoku décrite dans la figure 1.3 (a), l'affectation partielle $\mathscr{A} = \{x_1 = 2, x_2 = 4, x_3 = 1, x_4 = 5, x_5 = 9, x_6 = 6, x_7 = 8, x_8 = 7, x_9 = 3, x_{59} = 5\}$ ne



(a) (b)

FIGURE 1.3. – La grille de sudoku initiale (a), les domaines initiaux des variables (b)



(a) (b)

FIGURE 1.4. – Une première affectation des variables $\{x_1, x_2, \dots x_9\}$ ne violant aucune contrainte (a), et une seconde violant une contrainte (b)

viole aucune contrainte (figure 1.4 (a)). En revanche, l'affectation partielle $\mathscr{A}' = \{x_1 = 2, x_2 = 4, x_3 = 1, x_4 = 9, x_5 = 5, x_6 = 6, x_7 = 8, x_8 = 7, x_9 = 3, x_{59} = 5\}$ viole la contrainte spécifiant que les valeurs des variables x_5 et x_{59} doivent être différentes (figure 1.4 (b)). Notons que les affectations $\mathscr{A}$ et $\mathscr{A}'$ sont des projections des affectations représentées sur ces deux figures.

Introduisons maintenant la notion d'*affectation cohérente* :

Définition 1.3.4 (Affectation cohérente) *Soit $P = (X, D, C)$ une instance CSP. On dit qu'une affectation $\mathscr{A}$ d'un sous-ensemble de variables de X est* cohérente *ssi :*

$$\forall c_i \in C \text{ telle que } S(c_i) \subseteq X_{\mathscr{A}}, \mathscr{A} \text{ satisfait } c_i.$$

Pour résumer, une affectation est cohérente si elle ne viole aucune contrainte. De plus, notons que la vérification de la cohérence d'une affectation se fait en temps polynomial. Nous introduisons ensuite la définition d'une *solution* d'une instance CSP.

Définition 1.3.5 (Solution) *On appelle* solution *d'une instance CSP $P = (X, D, C)$ une affectation complète cohérente. L'affectation porte donc sur toutes les variables de P et ne viole aucune de ses contraintes. L'ensemble des solutions de P est noté Sol_P.*

2	4	1	5	3	6	8	7	9
7	9	6	4	8	2	1	5	3
8	5	3	9	1	7	4	2	6
5	8	2	1	4	3	9	6	7
1	7	9	6	2	5	3	8	4
3	6	4	8	7	9	5	1	2
6	1	5	7	9	4	2	3	8
9	3	8	2	6	1	7	4	5
4	2	7	3	5	8	6	9	1

FIGURE 1.5. – Une solution de notre grille de sudoku

Par exemple, considérons encore une fois l'instance CSP associée à la grille de sudoku décrite dans la figure 1.3(a). L'affectation complète $\mathscr{A} = \{x_1 = 2, x_2 = 4, x_3 = 1, x_4 = 5, \ldots, x_{80} = 9, x_{81} = 1\}$ telle que décrite dans la figure 1.5 est une solution de cette instance. Nous introduisons ensuite la notion de *cohérence* d'une instance CSP :

Définition 1.3.6 *Une instance CSP est cohérente ssi* $Sol_P \neq \emptyset$.

Le *problème de décision CSP* se définit de la manière suivante :

entrée : une instance CSP $P = (X, D, C)$.
question : est-ce que P admet une solution ?

Ce problème se focalise donc sur la décision et est NP-Complet (GAREY et JOHNSON 1979). Il en résulte qu'il est fortement combinatoire avec une taille de l'espace de recherche en $O(d^n)$ et qu'à l'heure actuelle, il n'existe aucun algorithme permettant de le résoudre en temps polynomial. Il existe, dans la littérature, un certain nombre de variantes de ce problème. On peut notamment citer :

- Le *problème de satisfaisabilité booléenne (SAT)* (MARQUES SILVA, LYNCE et MALIK 2009) :

  Le problème SAT est un problème de décision prenant en entrée une formule booléenne en forme normale conjonctive (c'est-à-dire une conjonction de disjonctions de variables booléennes ou de leurs négations, également appelée ensemble de clauses) et consiste à déterminer s'il existe une affectation d'un sous-ensemble des variables propositionnelles rendant la formule vraie (contrairement aux autres problèmes évoqués ici, l'affectation ne doit pas forcément concerner l'ensemble des variables). C'est un problème pionnier de la théorie de la complexité. En effet, ce fut le premier problème dont on a prouvé la NP-Complétude (COOK 1971) et qui a entre autre servi à démontrer celle de beaucoup d'autres problèmes. Ce problème admet un grand nombre de domaines d'applications, chaque instance CSP pouvant être réduite à une instance SAT et inversement.

- Le *problème de satisfaction de contraintes pondérées (WCSP)* (COOPER et SCHIEX 2004 ; FREUDER et WALLACE 1992) :

  Ce problème consiste à attribuer un poids à chacune des contraintes de l'instance. Ici, on ne cherche pas comme pour CSP à satisfaire l'intégralité des contraintes, mais plutôt à maximiser le poids total des contraintes, satisfaites. Cette variante permet, par exemple, de prendre en compte des notions de préférence ou de possibilité. Chaque affectation d'une variable se verra attribuer un coût en fonction

du poids des contraintes impliquant cette variable, l'objectif étant de privilégier les affectations à moindre coût. Ce problème est NP-difficile (COOPER et SCHIEX 2004 ; FREUDER et WALLACE 1992). Parmi les domaines d'application de ce problème, on peut citer notamment l'allocation de ressources (CABON, DE GIVRY, LOBJOIS et al. 1999), les enchères combinatoires (SANDHOLM 2002), ou encore la bio-informatique (VANDEL et GIVRY 2011 ; VANDEL, MANGIN, VIGNES et al. 2010 ; VIRICEL, GIVRY, SCHIEX et al. 2018 ; BEUVIN, GIVRY, SCHIEX et al. 2021) et le raisonnement probabiliste (PEARL 1988).

- Le *problème d'optimisation de contraintes (COP)* (ROSSI, BEEK et WALSH 2006b) :

  Une instance COP contiendra les mêmes éléments qu'une instance CSP classique auxquels on ajoute une fonction objectif f de la forme :

  $$f : D_{x_1} \times D_{x_2} \times \ldots \times D_{x_n} \rightarrow \mathbb{Q}$$

  Une solution optimale à la minimisation (resp. à la maximisation) d'une instance COP est donc une affectation minimisant (resp. maximisant) cette fonction objectif. D'une certaine manière, le but est ici de sélectionner, parmi toutes les solutions au sens CSP du terme, une solution optimisant la fonction f. Au niveau de ses applications, ce problème est utilisé pour résoudre un certain nombre de problématiques industrielles : l'optimisation des chaînes de production, de livraisons, l'ordonnancement de tâches (SYCARA, ROTH, SADEH et al. 1991) ou encore dans l'organisation de réunions simultanées (MODI et VELOSO 2004 ; PETCU et FALTINGS 2005). Il a également été utilisé afin de résoudre un problème de coordination d'un réseau de capteurs distribués (BÉJAR, DOMSHLAK, FERNÁNDEZ et al. 2005). Ce formalisme est également capable de résoudre un certain nombre de problèmes académiques tels que *Capacitated Vehicle Routing Problem (CVRP)* qui consiste à optimiser le routage de véhicules admettant des capacités ou encore *Aircraft Landing* qui consiste à optimiser les attérissages d'une série d'avions (AUDEMARD, LECOUTRE et LONCA 2022).

- Le problème *#CSP* (VALIANT 1979) :

  Cette variante a pour but de répondre à la question suivante : combien de solutions admet une instance CSP donnée ? Elle permet donc de déterminer si une instance est cohérente et, si tel est le cas, combien de solutions elle admet. Ce problème est très difficile à résoudre. En effet, il appartient à la classe de complexité #P-Complet (VALIANT 1979) même en ne considérant uniquement que des CSP binaires. Au niveau des applications, le problème #CSP est utilisé dans des domaines très variés, par exemple la planification (PALACIOS, BONET, DARWICHE et al. 2005 ; DOMSHLAK et HOFFMANN 2006), le diagnostic (KUMAR 2002), la physique (BURTON et STEIF 1994) ou encore la chimie (MANN, TACK et WILL 2007). Il a également été prouvé que l'évaluation du nombre de solutions pouvait permettre de déterminer quelles heuristiques de choix de variables/valeurs seraient les plus efficaces (KASK, DECHTER et GOGATE 2004 ; PESANT 2005 ; PESANT et ZANARINI 2012).

1.4. Comment résoudre une instance CSP ?

1.4.1. Introduction et méthodes en général

Au fil des années, un certain nombre de méthodes permettant de résoudre des instances CSP ont vu le jour. On peut diviser ces méthodes en deux grandes familles :

- Les *méthodes complètes* :

  ces méthodes consistent à déterminer si une instance CSP admet une solution. Il existe principalement deux types de méthodes complètes. Le premier, basé sur des algorithmes de recherche énumérative (ROSSI, BEEK et WALSH 2006b) combinant généralement recherche arborescente (voir la sous-section 1.4.2) et simplification du problème à l'aide de méthodes de filtrage (voir la sous-section 1.4.3). C'est ce type d'algorithmes qui est le plus souvent utilisé dans les solveurs modernes. Le second

type regroupe des méthodes basées sur des algorithmes de programmation dynamique (BERTELÈ et BRIOSCHI 1972 ; DECHTER 2006) (ici, l'idée est de résoudre un problème en combinant les résultats de plusieurs sous-problèmes, plus simples à résoudre).

- Les *méthodes incomplètes* :

  une méthode qui n'est pas complète est dite *incomplète*. Ce type de méthodes peut permettre de trouver une solution à une instance CSP, mais pas de déterminer si cette dernière n'en admet pas. On peut notamment citer des algorithmes de *recherche locale* (HOOS et STÜTZLE 2004 ; HOOS et TSANG 2006) qui consistent à n'explorer qu'une partie de l'espace de recherche. Ce type de méthode est très efficace en termes de temps. En revanche, elles ne permettent pas de déterminer si une instance n'admet pas de solution. De plus, ce type de méthode ne garantit pas de trouver une solution d'une instance s'il en existe au moins une. Autrement dit, ces méthodes vont sacrifier la complétude pour gagner en efficacité.

Dans les contributions de cette thèse, on devra dans la grande majorité des cas énumérer l'intégralité des solutions d'une instance, nous emploierons donc principalement des méthodes complètes. Un *solveur de CSP* est un programme dont le but est de résoudre des instances CSP. Pour y parvenir, il dispose de différentes briques qui pourront être combinées afin de résoudre l'instance le plus efficacement possible. Dans cette section, nous allons décrire en détails chacune d'entre elles.

1.4.2. Algorithme de parcours

Un *algorithme de parcours* est un algorithme ayant pour but de parcourir l'espace de recherche. On peut citer notamment l'algorithme *backtrack (BT)* décrit dans l'algorithme 10. L'idée générale de cet algorithme est de partir d'une affectation vide, afin de construire progressivement une affectation cohérente en explorant l'espace de recherche tout en prenant soin d'éviter les parties de cet espace menant à des affectations incohérentes. Pour réussir cela, il va effectuer un *parcours en profondeur d'abord* et va développer un *arbre de recherche*. Un arbre de recherche est une arborescence (c'est-à-dire un arbre orienté) dont les sommets vont représenter les variables du problème et dont les arcs sont étiquetés par des valeurs des domaines. Ainsi, l'arc ayant pour origine x et pour étiquette v représentera l'affectation de la valeur v à la variable x. Une branche de profondeur [1] strictement inférieure à n (c'est-à-dire au nombre de variables de l'instance CSP) correspond à une affectation partielle alors qu'une branche de profondeur n correspondra à une affectation complète. Un conflit (c'est-à-dire la violation d'au moins une contrainte) dans un tel arbre sera noté $\square$. Si l'on considère l'arbre de recherche représenté dans la figure 1.6, on a une première affectation $\{x_1 = 1, x_2 = 1, x_3 = 1\}$ qui mène à un conflit. On tente ensuite d'affecter une nouvelle valeur à x_3, ce qui donne une seconde affectation $\{x_1 = 1, x_2 = 1, x_3 = 2\}$ produisant également un conflit. Pour finir, on affecte x_2 et x_3 à de nouvelles valeurs, ce qui donne une dernière affectation $\{x_1 = 1, x_2 = 2, x_3 = 1\}$ qui se révèle cohérente.

L'algorithme BT prend en entrée une instance CSP $P = (X, D, C)$, une affectation cohérente $\mathscr{A}$ et $V \subseteq X$ l'ensemble des variables non instanciées. Une itération de cet algorithme se déroule de la manière suivante : s'il n'y a plus aucune variable à instancier, alors l'affectation $\mathscr{A}$ est une solution de l'instance (lignes 1-2). S'il reste encore des variables à instancier, alors on choisit une variable $x \in V$ (ligne 4). Tant que le domaine de x n'est pas vide, on choisit une valeur v de son domaine (ligne 7). Pour finir, si l'affectation courante $\mathscr{A}$ augmentée de l'affectation de x à v est cohérente (ligne 8), alors on met à jour l'affectation courante et on entame une nouvelle itération (ligne 9). S'il n'est pas possible d'étendre l'affectation $\mathscr{A}$ augmentée de l'affectation de x à v en une solution, BT va essayer d'affecter une autre valeur à x. Lorsque toutes les valeurs de x ont été essayées sans succès, il tente d'affecter une autre valeur à la variable précédente. Cette étape est appelée un *retour arrière chronologique*. Pour finir, si l'affectation devient vide, cela signifie que l'instance n'admet aucune solution.

En pratique, BT se révèle relativement peu efficace. Il présente en effet un certain nombre d'inconvé-

1. La profondeur d'une branche correspond au nombre de variables ayant été affectées sur cette dernière.

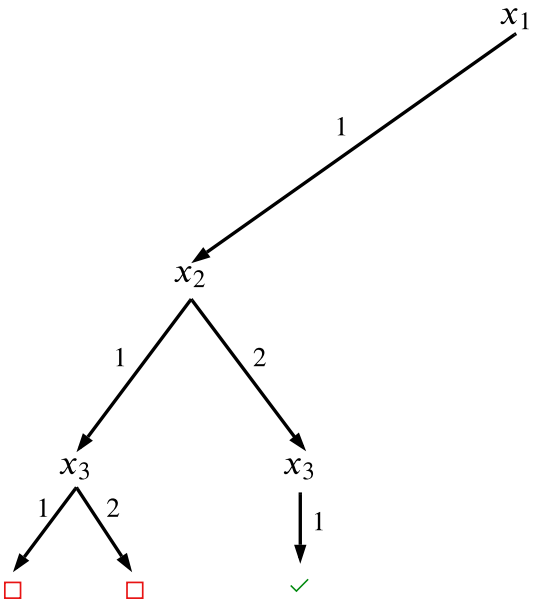


FIGURE 1.6. – Exemple d'un arbre de recherche non binaire

nients :

- L'ordre des variables à affecter est choisi arbitrairement. Nous verrons dans la sous-section 1.4.4 qu'un choix judicieux de l'ordre dans lequel instancier les variables peut réduire considérablement la taille de l'espace à parcourir.

- Comme il n'y a pas de réelle stratégie de choix de variable/valeur, les incohérences peuvent être trouvées relativement tardivement,

- À chaque fois que l'affectation courante devient incohérente, BT effectue un *retour arrière chronologique*, c'est-à-dire qu'il va tenter d'affecter une nouvelle valeur à la variable précédente dans la branche courante. Or, cette variable peut n'avoir aucune incidence sur le conflit détecté. Le fait de ne pas remonter à l'origine du conflit peut lui faire explorer un espace conséquent qui n'induira que des affectations incohérentes.

- Pour parcourir l'intégralité de l'espace de recherche, il devra faire, dans le pire des cas, d^n affectations. Pour chacune d'entre elles, il va devoir tester si aucune des m contraintes de l'instance n'est violée. Le test de validité d'une contrainte se faisant en $O(r)$, on obtient une complexité totale en $O(m.r.d^n)$.

Au fil des années, un certain nombre de méthodes ont été mises en œuvre afin d'améliorer l'efficacité de cet algorithme (chacune d'entre elles seront détaillées dans les sous-sections suivantes) :

- Appliquer des algorithmes de *filtrage* permettant de simplifier l'instance au fur et à mesure de sa résolution (voir la sous-section 1.4.3),

- utiliser des *heuristiques* permettant de choisir les variables et les valeurs à affecter de façon plus efficace (voir la sous-section 1.4.4),

- effectuer des *retours arrière non chronologiques* (ou *backjumping*) (STALLMAN et SUSSMAN 1977 ; GASCHNIG 1979 ; DECHTER 1990 ; PROSSER 1993 ; GINSBERG 1993 ; PROSSER 1995 ; CHEN 2000 ; JUSSIEN,

Algorithme 1 : *BT*

Input : $P = (X, D, C)$ une instance CSP, $\mathscr{A}$ une affectation, V l'ensemble des variables non affectées

```
if V = ∅ then
    A est une solution
else
    Choisir x ∈ V
    d ← D_x
    while d ≠ ∅ do
        Choisir v ∈ d
        d ← d\{v}
        if A ∪ {x = v} est cohérente then
            Parcourir(A ∪ {x = v}, V \{x})
```

DEBRUYNE et BOIZUMAULT 2000) : lorsque l'on arrive à une affectation incohérente, on va déterminer quelles sont les variables affectées responsables de ce conflit. Puis, au lieu de revenir simplement à la variable précédente comme dans le cas d'un retour arrière chronologique, on choisit de remettre en cause l'affectation de la variable à l'origine du conflit la plus profonde dans l'arbre de recherche.

1.4.3. Simplification et filtrage

Comme mentionné plus tôt, le problème de satisfaction de contraintes est un problème difficile à résoudre (car NP-Complet). À ce titre, il convient de tout faire pour le simplifier autant que possible, que ce soit avant ou pendant la résolution. Une manière d'y parvenir est d'utiliser des *cohérences*, qui peuvent être vues comme des propriétés locales que doit satisfaire l'instance CSP considérée. Ces cohérences sont accompagnées d'*algorithmes de filtrage*, des procédures pouvant être appliquées soit en prétraitement, soit pendant la résolution. Ces algorithmes consistent à transformer une instance $P = (X, D, C)$ en une nouvelle instance $P' = (X, D', C')$ équivalente (c'est-à-dire telle que $Sol(P') = Sol(P)$). Pour cela, ils suppriment des valeurs des domaines ou bien des tuples des relations des contraintes qui ne participent pas à une solution. En pratique, ce sont majoritairement les domaines des variables qui sont modifiés jusqu'à ce que la nouvelle instance vérifie la cohérence donnée. On parle alors de *cohérence de domaine*. Un tel nettoyage permet d'une part de limiter la taille de l'espace de recherche en réduisant la taille des domaines des variables, et d'autre part de réduire le coût de vérification de la satisfaction des contraintes en réduisant le nombre de tuples présents dans les relations de ces dernières. Lors d'un filtrage, on peut parfois observer un effet boule de neige induit par la suppression d'une valeur dont l'absence désormais permet, à son tour, de supprimer une autre valeur, et ainsi de suite jusqu'à atteindre un point fixe. Ce phénomène est appelé *propagation de contraintes*. De nombreuses notions de cohérence ont été introduites (APT 2003b ; DECHTER 2003 ; ROSSI, BEEK et WALSH 2006b ; LECOUTRE 2009 ; MACKWORTH 1977). Parmi elles, on retrouve la notion d'*arc cohérence généralisée* (MACKWORTH 1977), une cohérence de domaine qui se définit formellement de la manière suivante :

Définition 1.4.1 *Soit $P = (X, D, C)$ une instance CSP.*

- *Une valeur $v \in D_x$ est dite* arc-cohérente *si pour chaque contrainte c telle que $x \in S(c)$, il existe un tuple $t \in R(c)$ tel que $t[\{x\}] = v$ et $\forall y \in S(c), t[\{y\}] \in D_y$ (on dit que t est un support de v vis-à-vis de c).*
- *Une variable x est dite* arc-cohérente *si chaque valeur $v \in D_x$ est arc-cohérente.*
- *P est* arc-cohérente *si chaque variable de P est arc-cohérente.*

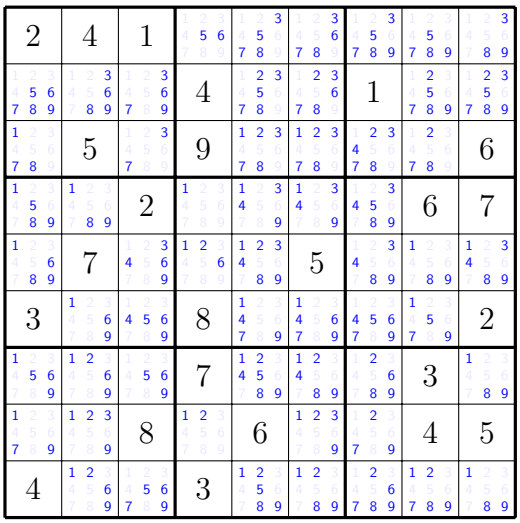


FIGURE 1.7. – État intermédiaire des domaines des variables après un filtrage par consistance d'arc

Autrement dit, une instance CSP est arc-cohérente si pour toute variable $x_i \in X$, chaque valeur $a \in D_{x_i}$ est compatible avec toute contrainte c_k portant sur x_i. Le filtrage par cohérence d'arc consiste donc à supprimer toutes les valeurs des domaines ne satisfaisant pas l'arc-cohérence. Le premier algorithme de filtrage par cohérence d'arc (AC) (WALTZ 1972) avait une complexité en temps exprimée en $O(n.m.d^2)$. Dans les années qui ont suivi, une série d'algorithmes similaires ont été présentés parmi lesquels on retrouve AC-3 (MACKWORTH 1977), AC-4 (MOHR et HENDERSON 1986), AC-6 (BESSIÈRE 1994), AC-8 (CHMEISS et JÉGOU 1998), AC-2001 (BESSIÈRE, RÉGIN, YAP et al. 2005) et AC3^{rm} (LECOUTRE, LIKITVIVATANAVONG, SHANNON et al. 2008). Chacun de ces algorithmes présente des complexités en temps et en espace différentes. Par exemple, les algorithmes AC-6 et AC-2001 présentent une complexité en temps exprimée en $O(m.d^2)$ et une complexité en espace exprimée en $O(m.d)$ dans le cas d'instances binaires.

Il faut savoir que d'autres cohérences de domaine plus fortes que la cohérence d'arc généralisée ont été présentées dans la littérature. On peut notamment citer SAC (DEBRUYNE et BESSIÈRE 1997b), Max-RPC (DEBRUYNE et BESSIÈRE 1997a) ou encore PIC (FREUDER et ELFE 1996). Ces dernières suppriment généralement plus de valeurs que la cohérence d'arc généralisée. Cependant, elles ont l'inconvénient d'avoir une complexité en temps et en espace plus grande et donc, un temps d'exécution plus important. Les algorithmes de filtrage peuvent donc avoir un coût relativement élevé. Ainsi, il convient de trouver un bon compromis entre coût du filtrage et quantité de données filtrées. Ainsi, il peut être judicieux d'utiliser une unique fois un algorithme de filtrage appliquant une cohérence forte, mais plus coûteuse telle que SAC, Max-RPC ou PIC en prétraitement, puis des algorithmes moins coûteux appliquant la cohérence d'arc pendant tout le reste de la résolution.

La figure 1.7 illustre un état intermédiaire des domaines des variables de notre instance représentant le problème du sudoku introduite dans la section 1.2 pendant un algorithme de filtrage par cohérence d'arc. On peut voir par exemple que la valeur 2 de x_4 n'est plus supportée à cause de l'affectation de x_1. Elle a donc été supprimée du domaine. Cette figure nous montre un bon exemple de la propagation de contraintes mentionnée plus haut : le domaine de x_{31} ne contenant plus que la valeur 1, cette information sera donc propagée. Cette propagation va alors supprimer notamment la valeur 1 du domaine de la variable x_{67}, qui n'aura plus que la valeur 2 dans son domaine, ce qui va, à son tour, mener à d'autres suppressions, et ainsi de suite jusqu'à l'obtention d'un point fixe.

1.4.4. Heuristiques

À chaque nouvelle itération, l'algorithme va devoir déterminer quelle nouvelle variable sera instanciée, et quelle valeur lui sera attribuée parmi celles présentes dans son domaine. Un choix judicieux peut réduire considérablement la taille de l'arbre de recherche. Néanmoins, il a été prouvé que trouver la variable induisant le plus petit arbre de recherche constitue un problème NP-Difficile (LIBERATORE 2000). Il n'est donc pas raisonnable de souhaiter déterminer le choix optimal à chaque étape de la résolution. Une solution à ce problème est de faire appel à la notion d'heuristiques. Une *heuristique* est simplement une représentation algorithmique de la notion d'intuition qu'une personne humaine utiliserait naturellement si elle devait choisir la prochaine variable à instancier. Dans cette section, nous détaillerons les différents types d'heuristiques proposées dans la littérature et nous donnerons des exemples pour chacun d'entre elles. Toutes les heuristiques qui seront présentées utilisent le principe du *first-fail* ((HARALICK et ELLIOT 1980)), une stratégie consistant à rencontrer les échecs le plus tôt possible dans l'exploration. En effet, rencontrer les conflits le plus tôt possible permet d'éviter de gaspiller du temps en explorant des branches qui ne contiendront aucune solution. Ici, nous considérons trois grandes familles d'heuristiques :

- les heuristiques de choix de variable statiques,
- les heuristiques de choix de variable dynamiques,
- les heuristiques de choix de variable dynamiques et adaptatives.

1.4.4.1. Les heuristiques de choix de variable statiques

On parle d'heuristique *statique* lorsque l'ordre d'affectation des variables est calculé en amont de la résolution et donc, indépendamment des résultats de celle-ci. Le fait que ce type d'heuristiques se base exclusivement sur l'état initial de la recherche le rend peu efficace de manière générale (DECHTER et MEIRI 1994). En effet, ces heuristiques ne tiennent pas compte des modifications apportées par un possible filtrage au cours de la résolution. Voici une liste non exhaustive des heuristiques statiques présentes dans la littérature :

- *random* : sélectionne les variables selon un ordre aléatoire construit en amont de la recherche.
- *lex* : sélectionne les variables selon l'ordre lexicographique.
- *deg* (ULLMANN 1976; DECHTER et MEIRI 1989) : trie les variables selon leur degré (c'est-à-dire le nombre de contraintes auxquelles participe la variable) décroissant et les affecte dans cet ordre.
- *dureté maximale (MT)* : La *dureté* d'une contrainte correspond au rapport du nombre de ses tuples interdits par le nombre de ses tuples possibles. Par ailleurs, la dureté d'une variable est calculée en faisant la somme des duretés des contraintes dans lesquelles elle est impliquée. Cette heuristique consiste à trier les variables par dureté décroissante et à les affecter dans cet ordre. Cette heuristique peut poser un problème, car il peut être coûteux et difficile de calculer le nombre de tuples induits par une contrainte exprimée en intention, en particulier si son arité est importante.
- *cardinalité maximale (MC)* (DECHTER et MEIRI 1989) : choisit une première variable aléatoirement, puis ordonne les autres de manière décroissante en fonction du nombre de contraintes qu'elles ont en commun avec les variables déjà choisies.
- *min-width* (FREUDER 1982) : ordonne les variables par degré croissant. À chaque nouvelle étape, on prend soin de ne pas prendre en compte les contraintes ayant déjà servi pour calculer le degré d'une variable précédente. Cette heuristique fonctionne uniquement dans le cas des CSP binaires.

1.4.4.2. Les heuristiques de choix de variable dynamiques

Contrairement aux heuristiques de choix de variables statiques, les heuristiques de choix de variable dites *dynamiques* sont capables de modifier l'ordre d'affectation des variables au cours de la résolution. Elles ne prennent néanmoins en compte que l'état courant de la résolution. Il a été prouvé que les heuristiques dynamiques étaient plus efficaces que les heuristiques statiques (DECHTER et MEIRI 1994). Parmi les heuristiques dynamiques présentes dans la littérature, on retrouve :

- *dom* (GOLOMB et BAUMERT 1965) : sélectionne une des variables non instanciées ayant le plus petit domaine.
- *ddeg* : sélectionne une des variables non instanciées ayant le plus grand degré courant. Une contrainte ayant au plus une seule variable non instanciée ne sera pas prise en compte dans le calcul, car elle n'a plus aucun impact sur le filtrage.
- *dom/deg* (BESSIÈRE et RÉGIN 1996) : sélectionne la prochaine variable en minimisant le rapport taille courante du domaine sur le degré (c'est-à-dire le nombre de contraintes auxquelles une variable participe). Cette heuristique combine les avantages de *dom* et de *deg*.
- *dom/ddeg* : sélectionne la prochaine variable en minimisant le rapport taille courante du domaine sur le degré courant. Cette heuristique combine les avantages de *dom* et de *ddeg*.

1.4.4.3. Les heuristiques de choix de variable dynamiques et adaptatives

Les heuristiques de choix de variable *dynamiques et adaptatives* (GEELEN 1992 ; REFALO 2004 ; BOUSSEMART, HEMERY, LECOUTRE et al. 2004) sont également capables de modifier l'ordre d'affectation des variables au cours de la résolution. Cependant, elles vont prendre en compte l'état courant de la recherche ainsi que l'ensemble des états précédents (contrairement aux heuristiques dynamiques qui ne considèrent que l'état courant). Ce procédé permet de faire des choix bien plus inhérents à la nature de l'instance traitée. Parmi les heuristiques de ce type, on retrouve :

- *wdeg*, *dom/wdeg* (BOUSSEMART, HEMERY, LECOUTRE et al. 2004), *dom/wdeg*$^{ca.cd}$ (WATTEZ, LECOUTRE, PAPARRIZOU et al. 2019) : pour chacune de ces heuristiques, nous associons un poids initial de 1 à chaque contrainte. À chaque fois qu'une contrainte contribue à rendre vide le domaine d'une variable, le poids de cette dernière est augmenté de 1 pour wdeg et dom/wdeg, et en fonction de l'arité courante de la contrainte et de la taille courante des domaines des variables qu'elle implique pour dom/wdeg$^{ca.cd}$. La variable x choisie correspondra à celle qui maximise la somme des poids des contraintes impliquant x et une autre variable pas encore instanciée pour wdeg. Pour les deux autres, on choisira la variable minimisant le rapport taille courante du domaine sur la somme des poids des contraintes impliquant x et une autre variable pas encore instanciée. L'idée principale de ces trois heuristiques est d'essayer d'identifier les parties difficiles de l'instance et de les traiter en priorité.
- *Conflict Search History* (CHS, (HABET et TERRIOUX 2021)) : l'idée derrière cette heuristique est de favoriser les variables étant les plus impliquées dans les conflits récents. Son fonctionnement est similaire à celui de dom/wdeg ou de dom/wdeg$^{ca.cd}$, la différence résidant principalement sur la fonction utilisée pour mettre à jour les poids.

De manière générale, ce sont les heuristiques *dom/wdeg* et *wdeg* qui sont le plus souvent utilisées quand on ne cible pas une famille d'instance spécifique. Dans le cas contraire, il peut être intéressant d'utiliser une heuristique plus générale ou encore une heuristique dédiée à cette famille.

1.4.4.4. Les heuristiques de choix de valeur

Une fois la prochaine variable à instancier sélectionnée, l'algorithme de résolution va devoir choisir la valeur à lui attribuer parmi les valeurs présentes dans son domaine courant. Pour cela, il va faire appel à une *heuristique de choix de valeur*. Ce type d'heuristiques utilise souvent le principe du *success-first* qui consiste à choisir la valeur ayant le plus de chance d'apparaître dans une solution. Ces heuristiques ont été moins étudiées que les précédentes, étant donné qu'il peut être difficile d'en trouver une qui soit générique. Voici néanmoins quelques exemples d'heuristiques de choix de valeur :

- *minVal* : sélectionne la plus petite valeur du domaine courant,
- *maxVal* : sélectionne la plus grande valeur du domaine courant,
- *randVal* : sélectionne aléatoirement une valeur du domaine courant.

1.4.5. Solveurs CSP modernes

Comme mentionné plus haut, les solveurs récents font principalement appel à des algorithmes de recherche énumératifs. Cette sous-section présentera de manière générale les méthodes utilisées par les solveurs CSP modernes. Nous aborderons entre autres les différents types de *décisions* qui peuvent être prises ainsi que les *stratégies de redémarrage* et d'*apprentissage* qui peuvent être adoptées dans le but d'optimiser le parcours de l'espace de recherche.

De manière générale, un algorithme de résolution peut prendre deux types de décision :

- des *décisions positives* de la forme $x = v$. Ici, on affecte la valeur v à la variable x,
- des *décisions négatives* de la forme $x \neq v$. Ici, on interdit l'affectation de la valeur v à la variable x.

Certains algorithmes de résolution vont uniquement exploiter des décisions positives. C'est par exemple le cas de l'algorithme Backtrack (BT) présenté dans la sous-section 1.4.2. Quand BT ajoute $\{x = v\}$ à l'affectation courante et que cela rend cette dernière incohérente, il va alors tenter d'affecter à x une autre valeur jusqu'à ce que l'affectation redevienne cohérente ou jusqu'à épuisement des valeurs. Une fois que c'est le cas, il va chercher à affecter une autre variable de la même manière. Les algorithmes de ce type sont appelés algorithmes *à branchement non binaire* et vont développer un arbre de recherche non-binaire similaire à celui décrit dans la figure 1.6.

D'autres algorithmes exploitent quant à eux à la fois des décisions positives et négatives (SABIN et FREUDER 1994 ; LECOUTRE, SAÏS, TABARY et al. 2007). De tels algorithmes vont par exemple brancher sur la décision positive $x = v$. Puis, si cela conduit à un échec, ils vont brancher sur la décision opposée, à savoir la décision négative $x \neq v$. Ces algorithmes sont appelés algorithmes *à branchement binaire* et vont développer un arbre de recherche binaire. Considérons l'arbre de recherche binaire décrit dans la figure 1.8, on branche dans un premier temps sur la décision positive $x_1 = v_i$, puis sur $x_2 = v_j$ ce qui rend l'affectation locale incohérente. On va donc ensuite brancher sur la décision négative opposée $x_2 \neq v_j$, qui va également produire une affectation incohérente. Étant donné que l'on a épuisé toutes les possibilités de décision pour la variable x_2, on va remonter et brancher sur $x_1 \neq v_i$, puis sûr $x_2 = v_j$, ce qui va donner une affectation complète et cohérente. Il a été prouvé qu'en théorie, les algorithmes à branchement binaires sont plus efficaces que les non-binaires lorsqu'il s'agit de déterminer qu'une instance CSP n'admet aucune solution (MITCHELL 2003 ; HWANG et MITCHELL 2005). Néanmoins, il reste difficile de déterminer le meilleur type d'algorithmes en pratique.

Lors de la résolution d'une instance CSP, les mauvais choix de variables peuvent conduire à explorer une partie radicalement plus importante de l'espace de recherche. Pour pallier ce problème, les solveurs récents

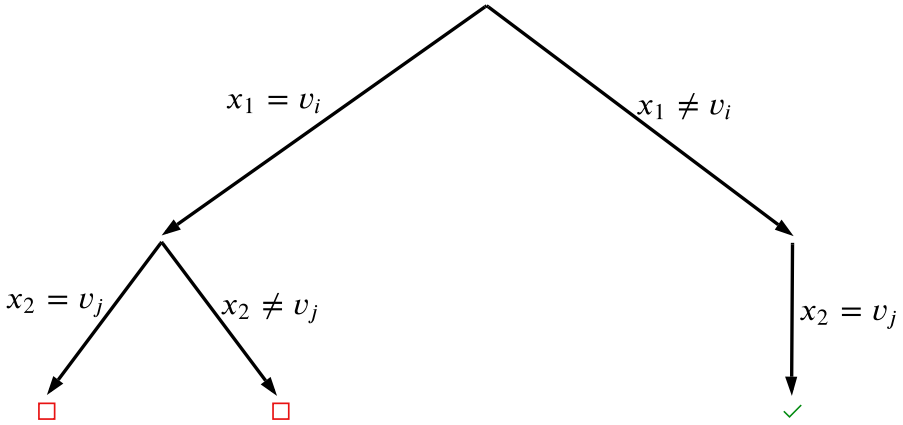


FIGURE 1.8. – Exemple d'un arbre de recherche binaire

utilisent des *redémarrages* (HARVEY 1995 ; GOMES, SELMAN, CRATO et al. 2000). Un redémarrage consiste à arrêter la recherche en cours quand le solveur détecte que le parcours ne progresse pas suffisamment. On peut choisir d'effectuer un redémarrage après un certain nombre de conflits, de nœuds visités, de temps écoulé, ... Les solveurs récents adoptent des *stratégies de redémarrage*, qui permettent de définir la fréquence à laquelle un redémarrage va avoir lieu. Ces stratégies se définissent par un critère et un seuil. À chaque fois que le critère atteint le seuil, le solveur va effectuer un redémarrage et la valeur du seuil va avoir tendance à augmenter jusqu'à ce que l'on puisse capturer l'intégralité de l'espace de recherche, ce qui va garantir la complétude de la méthode.

Parmi les stratégies de redémarrage les plus connues, on peut mentionner les suivantes :

- Stratégie arithmétique :

  le solveur effectue un redémarrage quand le critère va dépasser le seuil donné. Ce dernier va ensuite être incrémenté d'une valeur x à chaque nouveau redémarrage. On parle de *seuil fixé* si x vaut 0. Il a été démontré (LUBY, SINCLAIR et ZUCKERMAN 1993) que lorsque la distribution du temps d'exécution est connue, considérer un seuil fixé est optimal.

- Stratégie géométrique (STERGIOU et WALSH 1999) :

  le solveur effectue un redémarrage quand le critère va dépasser le seuil donné. Ce dernier va ensuite être multiplié par une valeur $x > 1$ à chaque nouveau redémarrage. Ici, on cherche à augmenter x géométriquement afin de tenter d'approcher rapidement la valeur optimale du seuil.

- Stratégie de Luby (LUBY, SINCLAIR et ZUCKERMAN 1993) :

  le seuil va prendre successivement les valeurs d'une constante x multipliée par les valeurs d'une série de nombres définie de la manière suivante :

$$s_i = \begin{cases} 2^{k-1} & \text{si } \exists k \in \mathbb{N} \text{ t.q. } i = 2^{k-1} \\ t_{i-2^{k-1}+1} & \text{si } \exists k \in \mathbb{N} \text{ t.q. } 2^{k-1} \leq i \leq 2^k - 1 \end{cases}$$

  Ainsi, les premières valeurs de la série seront les suivants : $1, 1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, 1, \ldots$

La notion d'*apprentissage* fut introduite dans (DECHTER 1986) sous le nom de *Learning While Searching*. Ce procédé consiste à ajouter de l'information à l'instance de base afin d'éviter de reprendre à nouveau certaines décisions ayant déjà mené à un conflit par le passé. Bien évidemment, la nouvelle instance doit être

équivalente à la précédente en termes d'ensemble de solutions. L'apprentissage permet d'empêcher l'exploration de certaines parties de l'espace de recherche dont on est sûr qu'elles n'induiront que des affectations incohérentes. Ainsi, effectuer un apprentissage judicieux peut permettre de réduire significativement la taille de l'espace de recherche à explorer. En général, on choisit d'apprendre des informations correspondant à des affectations ayant mené à des conflits. Ce type d'information est appelé *nogood* et est défini formellement dans (LECOUTRE, SAÏS, TABARY et al. 2007) de la manière suivante :

Définition 1.4.2 *Soient $P = (X, D, C)$ une instance CSP et Δ un ensemble de décisions. Soit $P|\Delta$ l'instance CSP (X, D', C) induite par Δ telle que $D' = \{D'_{x_1}, \ldots, D'_{x_n}\}$ avec $D'_{x_i} = \{v_i\}$ si la décision positive $x_i = v_i$ est présente dans Δ, $D'_{x_i} = D_{x_i} \setminus \{v_i\}$ si la décision négative $x_i \neq v_i$ apparaît dans Δ et $D'_{x_i} = D_{x_i}$ si aucune décision de Δ ne porte sûr x_i. On dit que Δ est un* nogood *si l'instance $P|\Delta$ est incohérente.*

On retrouve dans la littérature un très grand nombre de travaux démontrant l'intérêt de l'apprentissage de nogoods (DECHTER 1990 ; FROST et DECHTER 1994 ; FROST et DECHTER 1997 ; SCHIEX et VERFAILLIE 1994b ; SCHIEX et VERFAILLIE 1994a ; STALLMAN et SUSSMAN 1977 ; DECHTER 1986 ; GINSBERG 1993 ; RICHARDS et RICHARDS 1996 ; BAYARDO et MIRANKER 1996 ; JUSSIEN, DEBRUYNE et BOIZUMAULT 2000 ; KATSIRELOS et BACCHUS 2003 ; KATSIRELOS et BACCHUS 2005 ; LECOUTRE, SAÏS, TABARY et al. 2007). Il convient néanmoins de prendre garde à ne pas en apprendre trop. En effet, le nombre de nogoods pouvant être exponentiel, cela pourrait allonger considérablement le temps du test de cohérence des affectations. La difficulté réside donc dans le fait de trouver le bon compromis entre quantité de données enregistrées et taille de l'espace de recherche élagué. On peut par exemple mentionner (LECOUTRE, SAÏS, TABARY et al. 2007) dans lequel les auteurs choisissent de n'apprendre que les nogoods ayant une cardinalité maximale fixée arbitrairement.

Les solveurs CSP peuvent être vus comme des boites noires combinant différentes briques leur permettant de résoudre efficacement une instance CSP. Ils intègrent notamment les différentes notions mentionnées plus haut ainsi que des stratégies de redémarrage et des méthodes d'apprentissage permettant de retenir les affectations ayant conduit à un conflit afin de ne pas les refaire par la suite. Ces derniers ont grandement évolué au cours des dernières années et permettent la modélisation et la résolution de problèmes relativement difficiles. On trouve notamment ACE [2] (AbsCon Essence), Choco Solver [3] (FAGES, LORCA et PRUD'HOMME s. d.) , BTD (JÉGOU et TERRIOUX 2003) qui exploite la structure du graphe de contraintes de l'instance CSP, ou encore OR Tools [4] développé par Google.

1.5. Modélisation

Dans cette section, nous parlerons des outils disponibles permettant de faciliter la modélisation des problèmes sous la forme de CSP. Dans la sous-section 1.5.1, nous introduirons les *contraintes globales*, des patrons de contraintes pouvant impliquer un nombre quelconque de variables. La sous-section 1.5.2 présentera quant à elle des types de variables plus sophistiqués comme les *variables de graphes*, des variables CSP permettant de représenter des graphes, et les contraintes qui y sont associées.

1.5.1. Contraintes globales

Pour rappel, nous avons introduit dans la section 1.2 deux manières de représenter une contrainte : la représentation en extension (c'est-à-dire énumérer l'ensemble des tuples autorisés ou interdits par la contrainte) et la représentation en intention (c'est-à-dire représenter la contrainte par un prédicat au lieu d'un ensemble de tuples). Nous avons vu que chacune de ces représentations pouvait poser un problème en fonction de la contrainte considérée. En effet, une représentation en extension peut induire un nombre très élevé de tuples et donc, saturer l'espace mémoire. D'autre part, représenter une contrainte en intention peut

2. https://github.com/xcsp3team/ace
3. https://choco-solver.org/
4. https://developers.google.com/optimization

engendrer une augmentation du coût de son test de satisfaction.

Une solution qui s'offre à nous est de représenter la contrainte concernée sous la forme d'une *contrainte globale*. Ce type de contrainte est défini dans (LECOUTRE 2009) de la manière suivante :

Définition 1.5.1 *Une contrainte globale est un modèle de contrainte qui capture des relations ayant une sémantique précise et qui peut impliquer un nombre quelconque de variables.*

Une première contrainte globale relativement courante est la contrainte $allDifferent$ (RÉGIN 1994 ; HOEVE 2001 ; GENT, MIGUEL et NIGHTINGALE 2008) qui spécifie que toutes les variables d'un ensemble Y de variables doivent prendre des valeurs deux à deux différentes :

Définition 1.5.2 (Contrainte allDifferent (RÉGIN 1994 ; HOEVE 2001 ; GENT, MIGUEL et NIGHTINGALE 2008)) *Soit Y un sous-ensemble de variables de X. La contrainte $allDifferent(Y)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y | \forall x, y \in Y, x \neq y, t[\{x\}] \neq t[\{y\}]\}$.

On se propose maintenant de montrer un troisième exemple de modélisation de notre problème de sudoku, cette fois-ci en utilisant la contrainte globale $allDifferent$:

Exemple 1.5.1 *Modélisation $P = (X, D, C)$ du problème du sudoku avec des contraintes globales :*

- $X = \{x_1, x_2, \dots x_{81}\}$ *(on considère encore les sous-ensembles de X définis dans l'exemple 1.2.1),*
- $D = \{D_{x_1}, D_{x_2}, \dots, D_{x_{81}}\}$ *avec* $D_{x_i} = \{1, 2, \dots, 9\}, 1 \leq i \leq 81$,
  On fixe néanmoins les domaines des variables correspondantes aux cases déjà fixées comme illustré dans la figure 1.3.
- $C = C_{lignes} \cup C_{colonnes} \cup C_{régions} \cup C_{indices}$ *avec :*
  - $C_{lignes} = \{allDifferent(L_i) | 1 \leq i \leq 9\}$,
  - $C_{colonnes} = \{allDifferent(C_i) | 1 \leq i \leq 9\}$,
  - $C_{régions} = \{allDifferent(R_i) | 1 \leq i \leq 9\}$,
  - $C_{indices}$ *peut être modélisé soit comme dans l'exemple 1.2.1, soit comme dans l'exemple 1.2.2.*

Parmi les nombreuses contraintes globales définies dans la littérature, nous rappelons, à présent, celles qui seront utiles dans la suite du manuscrit.

La contrainte Count (BELDICEANU et CONTEJEAN 1994) spécifie que le nombre de variables de Y qui sont affectées avec des valeurs présentes dans un ensemble V de valeurs satisfait une condition donnée :

Définition 1.5.3 (Contrainte Count (BELDICEANU et CONTEJEAN 1994)) *Soient Y un sous-ensemble de variables de X, V un ensemble de valeurs, $\odot$ un opérateur parmi $\leq, <, =, \neq, >$ ou $\geq$, et un entier i. La contrainte $Count(Y, V, \odot, i)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y \mid |\{y \in Y | t[\{y\}] \in V\}| \odot i\}$.

La contrainte Sum spécifie que la somme des valeurs des variables de Y satisfait une condition donnée :

Définition 1.5.4 (Contrainte Sum) *Soient Y un sous-ensemble de variables de X, $\odot$ un opérateur parmi $\leq, <, =, \neq, >$ ou $\geq$, et un entier i. La contrainte $Sum(Y, \odot, i)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y \mid \sum_{y \in Y} t[\{y\}] \odot i\}$.

Par abus de notation, on désignera généralement la contrainte $Sum(Y, \odot, i)$ sous la forme $\sum_{y \in Y} y \odot i$. Il est également possible de remplacer l'entier i par une variable entière.

La contrainte $regular(Y, A)$ (CARLSSON et BELDICEANU 2004 ; PESANT 2004) spécifie que le mot formé par les variables de Y est reconnu par l'automate fini [5] A :

Définition 1.5.5 (Contrainte Regular (CARLSSON et BELDICEANU 2004 ; PESANT 2004)) *Soient Y un sous-ensemble de variables de X et A un automate. La contrainte $Regular(Y, A)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y \mid t \; est \; un \; mot \; accepté \; par \; A\}$.

La contrainte $Element(Y, v)$ (HENTENRYCK et CARILLON 1988) spécifie qu'au moins l'une des variables de Y doit avoir pour valeur v.

Définition 1.5.6 (Contrainte Element (HENTENRYCK et CARILLON 1988)) *Soient Y un sous-ensemble de variables de X et v une valeur. La contrainte $Element(Y, v)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y \mid \exists y \in Y, t[\{y\}] = v\}$.

Enfin, la contrainte globale $baseNogood(\Delta^*)$ (LECOUTRE, SAÏS, TABARY et al. 2007) permet de représenter un ensemble Δ^* de nogoods pouvant être appris durant la résolution :

Définition 1.5.7 (Contrainte baseNogood) *Soient Y un sous-ensemble de variables de X et Δ^* un ensemble de nogoods, chacun portant sur un sous-ensemble de variables de Y. La contrainte $baseNogood(\Delta^*)$ est la contrainte c définie par :*

- $S(c) = Y$ *et*
- $R(c) = \{t \in \prod_{y \in Y} D_y \mid \forall \Delta \in \Delta^*, \Delta \not\subseteq t\}$.

Un nogood Δ peut être représenté sous la forme d'une clause. Ainsi, le filtrage de la contrainte globale baseNogood peut s'appuyer sur la méthode des *watched litterals* (MOSKEWICZ, MADIGAN, ZHAO et al. 2001). Cette méthode utilisée dans les solveurs SAT récents consiste à regarder l'état d'uniquement deux littéraux par clause (on dit alors que ces littéraux sont *surveillés*). Quand un littéral est propagé, le solveur va regarder si sa négation est surveillée dans une des clauses. Si c'est le cas, il va chercher si la clause en question admet un autre littéral non affecté et non surveillé et dans le cas échéant, ce dernier va remplacer le littéral initialement surveillé. Si la clause ne contient aucun autre littéral valide à surveiller, alors le solveur va inférer le second littéral surveillé. Cette méthode a entraîné une amélioration drastique des solveurs SAT. En effet, ils devaient auparavant parcourir l'ensemble des littéraux de chaque clause à chaque nouvelle propagation alors que maintenant, il suffit de traiter deux littéraux par clause. La figure 1.9 illustre un exemple de comportement des watched litterals : nous considérons la clause $\neg x_1 \vee x_2 \vee x_3$ et initialement, les littéraux $\neg x_1$ et x_2 sont surveillés. Suite à la propagation de x_1, le littéral $\neg x_1$ n'est plus surveillé et laisse sa place à x_3 et pour finir, la propagation de $\neg x_2$ fait que le littéral x_2 n'est à son tour plus surveillé. Étant donné qu'il est désormais impossible de trouver un littéral non surveillé qui n'a pas été affecté, on en conclut que la clause est devenue unitaire. Et donc, le second littéral surveillé, à savoir x_3 sera propagé.

On peut facilement voir que les contraintes mentionnées ci-dessus peuvent porter sur un nombre quelconque de variables. Les contraintes globales possèdent des algorithmes de filtrages qui leur sont dédiés, ce qui se traduit de manière générale par un gain d'efficacité. L'utilisation de ces contraintes se révèle donc être un bon compromis entre capacité et temps de filtrage.

Les trois modélisations du problème du sudoku que nous avons présentées illustrent bien qu'un problème peut être représenté de plusieurs manières différentes. Pour chaque problème, il convient de déterminer la modélisation la plus optimale. Cette dernière peut dépendre de la nature du problème, de sa taille ou bien du solveur utilisé.

5. Un *automate fini* est un quintuplé de valeurs $A =< \Sigma, Q, I, T, E >$, avec Σ un *alphabet*, Q un ensemble fini d'*états*, $I \subseteq Q$ un ensemble d'*états initiaux*, $T \subseteq Q$ un ensemble d'*états finaux*, $E \subseteq Q \times \Sigma \times Q$ un ensemble de *transitions*. On dit que l'automate A *reconnaît* un mot s'il est possible, en partant d'un état initial, de lire l'entièreté du mot tout en terminant sur un état final. Un automate fini lit un mot caractère par caractère. Á chaque nouveau caractère, ce dernier va changer d'état en fonction des transitions qu'il admet. La figure 1.10 décrit un automate fini acceptant tous les mots du langage $0^*1^*0^*$.

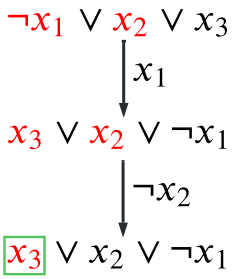


FIGURE 1.9. – Exemple de comportement des watched litterals, les littéraux en rouge correspondant aux littéraux surveillés

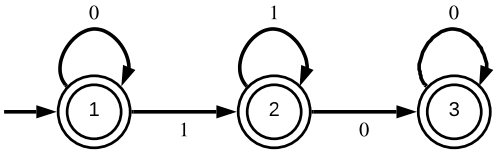


FIGURE 1.10. – Automate fini acceptant le langage $0^*1^*0^*$

1.5.2. Des variables plus sophistiquées

Les *variables de graphe* sont des variables qui vont représenter un graphe. Étant donnés deux graphes GLB et GUB représentant respectivement une borne inférieure et une borne supérieure, le domaine d'une telle variable va être défini comme étant l'ensemble des graphes g tel que $GLB \subseteq g \subseteq GUB$ [6]. Une variable de graphe peut être utilisée pour représenter aussi bien un graphe non orienté qu'un graphe orienté. La figure 1.11 illustre un exemple du domaine d'une variable de graphe. Il convient de mentionner que par soucis de performances, les graphes ne sont pas directement stockés dans le domaine. En effet, ces derniers sont simplement induits par les bornes inférieure et supérieure de la variable. Ainsi, lors d'un filtrage, on se contente de modifier ces bornes, ce qui entraînera la suppression de certains graphes du domaine. La figure 1.12 nous montre l'état des domaines de la variable de graphe après avoir forcé la présence de l'arête $\{1,3\}$. L'utilisation des variables de graphe a été d'une importance capitale pour effectuer les travaux décrits dans le chapitre 3. Il existe un grand nombre de contraintes liées à ces variables. On retrouve entre autres les suivantes :

- `nbNodes`(x_G, x) : spécifie que la valeur de la variable entière x doit correspondre au nombre de sommets de la variable de graphe x_G,
- `nbEdges`(x_G, x) : spécifie que la valeur de la variable entière x doit correspondre au nombre d'arêtes de la variable de graphe x_G,
- `nodeChanneling`(x_G, Y) : lie la présence de chaque sommet de la variable de graphe x_G à une variable booléenne de l'ensemble Y (une variable booléenne est une variable entière de domaine $\{0,1\}$. Par convention, 0 correspond à faux, et 1 à vrai). Si ce sommet est présent, alors la variable associée vaudra 1, 0 sinon,
- `edgeChanneling`(x_G, Y) : contrainte similaire à la précédente. On considère la présence des arêtes au lieu de celle des sommets,

6. Étant donné deux graphes $g = (V, E)$ et $g' = (V', E')$, on dit que $g \subseteq g'$ si $V \subseteq V'$ et $E \subseteq E'$.

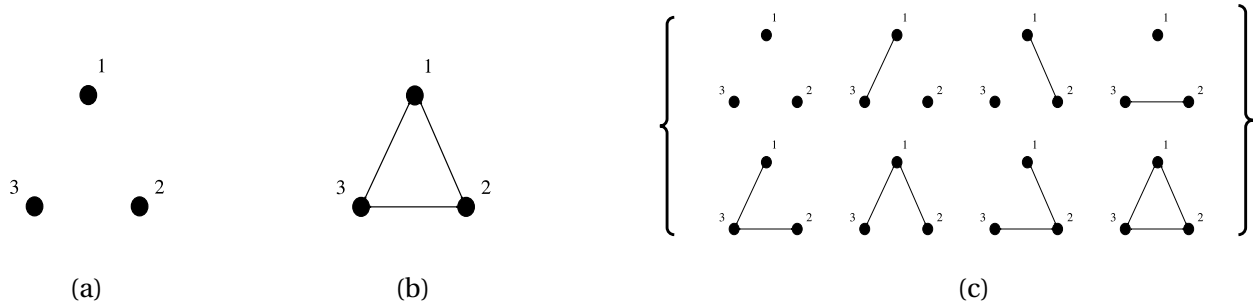


FIGURE 1.11. – Domaine d'une variable de graphe (c) induit par *GLB* (a) et *GUB* (b)

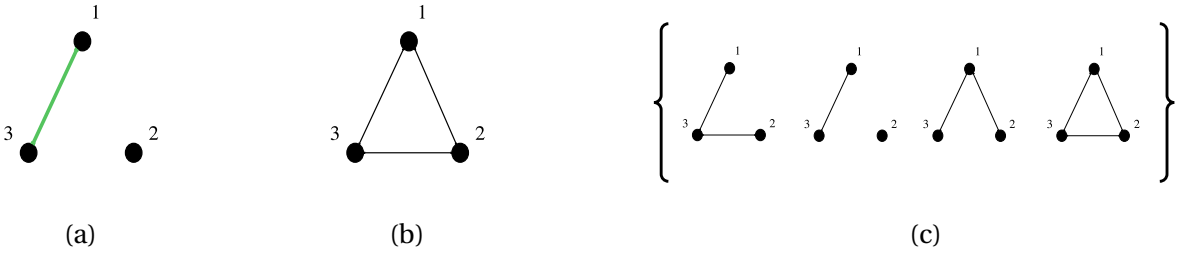


FIGURE 1.12. – Domaine d'une variable de graphe après un filtrage (c) induit par *GLB* (a) et *GUB* (b)

- `cycle`(x_G) : spécifie que le graphe non orienté décrit par x_G doit être un cycle (c'est-à-dire une suite d'arêtes successives dont les sommets extrémités sont identiques),
- `circuit`(x_G) : spécifie que le graphe orienté décrit par x_G doit être un circuit (c'est-à-dire une suite d'arcs successifs donc les sommets extrémités sont identiques),
- `subGraph`(x_G, x'_G) : spécifie que le graphe décrit par x_G doit être un sous-graphe [7] de celui décrit par x'_G.
- `connected`(x_G) : spécifie que le graphe non orienté décrit par x_G doit être connexe [8],
- `stronglyConnected`(x_G) : spécifie que le graphe orienté décrit par x_G doit être fortement connexe (la forte connexité étant une application de la connexité aux graphes orientés),
- `nbConnectedComponents`(x_G, x) : lie le nombre de composantes connexes [9] du graphe décrit par x_G à la variable entière x,
- `nbStronglyConnectedComponents`(x_G, x) : lie le nombre de composantes fortement connexes du graphe orienté décrit par la variable x_G à la variable entière x.
- `minDegree`(x_G, d) : spécifie que chaque sommet du graphe décrit par la variable de graphe x_G doit avoir un degré [10] minimal de d.

La *réification* est le fait de représenter la satisfaction d'une contrainte c sous la forme d'une variable boo-

7. Un graphe $G' = (V', E')$ est un sous-graphe du graphe $G = (V, E)$ si et seulement si $V' \subseteq V$ et $E' \subseteq E$.
8. Un graphe non orienté est connexe si est seulement si l'intégralité de ses sommets sont reliés deux à deux par un chemin (une suite d'arêtes reliant deux sommets).
9. Une composante connexe d'un graphe non orienté est un *sous-graphe* induit, maximal et connexe.
10. Le degré d'un sommet est le nombre d'arêtes issues de ce dernier

léenne v (on a alors $c \iff v$). Ce procédé est notamment utile pour impliquer une ou plusieurs contraintes dans une clause ou encore, étant donné un ensemble de contraintes, de connaître combien d'entre elles sont satisfaites.

1.5.3. Modéliser et résoudre une instance CSP

Dans cette sous-section, nous détaillerons les différents outils permettant de représenter une instance CSP et pour chacun d'entre eux, nous détaillerons les avantages et les inconvénients qui en découlent. Le premier d'entre eux consiste à la représenter sous un format spécifique. On peut notamment citer $XCSP^3$ (http://www.xcsp.org/) qui est une extension du langage XML permettant de représenter des instances CSP. Ce format est reconnu par la plupart des solveurs modernes. On peut également mentionner le format $XCSP^3 - core$ qui est une simplification d'$XCSP^3$ et qui est utilisé lors des compétitions.

$PyCSP^3$ (http://pycsp.org/) est une bibliothèque codée en python permettant de modéliser une instance CSP de manière déclarative pour ensuite l'exporter au format $XCSP^3$. L'instance pourra ensuite être résolue par l'un des nombreux solveurs acceptant le format (tels qu'ACE, Choco Solver, BTD, ...). Le principal avantage de ce procédé est que l'utilisateur est libre de choisir le solveur qui va résoudre l'instance et peut donc en choisir un qui soit le plus adapté possible à la nature de l'instance qu'il souhaite résoudre. En revanche, chaque solveur va lire l'instance générée par $PyCSP^3$ à sa manière. Ainsi, deux solveurs sont susceptibles de modéliser une même instance de deux manières différentes et l'une d'elle pourrait ne pas être optimale. De plus, $PyCSP^3$ est limité à un certain nombre de contraintes. Par exemple, il n'admet pas la réification de contraintes.

Certains solveurs (tel que Choco Solver) proposent des bibliothèques permettant de modéliser une instance de manière déclarative et de la résoudre dans la foulée. Ce procédé permet à l'utilisateur de paramétrer efficacement le solveur en fonction du type de l'instance traitée. En revanche, si l'on souhaite changer de solveur, on devra modéliser à nouveau l'instance.

L'intégralité des implémentations qui seront décrites dans cette thèse a été faite en utilisant Choco Solver. Ce choix est justifié par différents points. Premièrement, Choco admet l'utilisation de variables de graphe qui sont indispensables pour nos travaux. Deuxièmement, il permet aussi la réification de contraintes et nous avons eu plusieurs fois besoin de ce procédé. De plus, ce solveur est maintenu et évolue en permanence et à ce titre, est relativement complet et à jour en terme d'heuristiques et de méthodes de filtrage et pour finir, il laisse une grande liberté de paramétrage à l'utilisateur. Pour finir, il convient de garder à l'esprit qu'il existe plusieurs modélisations à un problème et qu'il n'est pas toujours évident de choisir la plus optimale.

1.6. Conclusion

Ce chapitre a été consacré à l'introduction du paradigme qu'est la programmation par contraintes et aux différents formalismes qu'il induit. Nous nous sommes principalement attardés sur le *problème de satisfaction de contraintes (CSP)* qui, pour rappel, consiste à modéliser un problème donné sous la forme d'un ensemble de variables dont les valeurs possibles sont définies par leurs domaines. Ces variables sont liées entre elles par des contraintes pouvant être représentées de différentes manières (en extension, en intention ou sous la forme de contraintes globales pouvant induire un nombre quelconque de variables). Bien que les variables d'un CSP soient généralement entières, il existe néanmoins des types plus exotiques telles que les variables de graphe, des variables représentant un graphe (orienté ou non) induisant un certain nombre de contraintes qui leur sont dédiées.

Nous avons vu qu'un des principaux point fort de ce formalisme était son expressivité qui lui permet d'exprimer un grand nombre de problèmes de natures variées ou encore, de représenter un problème de différentes manières (ce fut exprimé à travers l'exemple du problème du sudoku). En revanche, ce problème

est NP-Complet et donc difficile à résoudre.

On retrouve principalement deux types de méthode de résolution pour le problème CSP. Les méthodes complètes vont garantir le parcours intégral de l'espace de recherche et ainsi déterminer toutes les solutions du problème alors que les méthodes dites incomplètes ne vont en parcourir qu'une partie. Ces dernières sont généralement plus rapides, mais ne garantissent pas de trouver une solution de l'instance ou encore de déterminer si elle n'admet pas de solution. Parmi les algorithmes à méthode complète, on peut citer Backtrack (BT) qui est l'un des premiers à avoir été présenté. Ce dernier repose sur le principe d'explorer l'intégralité de l'espace de recherche en développant un arbre de recherche non binaire. Il reste néanmoins peu efficace pour diverses raisons, la première étant la taille de l'espace de recherche à parcourir ($O(d^n)$). Les autres raisons sont principalement liées à un manque de stratégie de sélection des variables/valeurs ou encore l'absence de retour arrière non chronologique. Pour pallier ce problème, différentes améliorations ont été mises en place. On peut notamment citer les méthodes de filtrage qui permettent d'élaguer l'arbre de recherche en faisant en sorte que l'instance satisfasse une cohérence donnée ou encore les heuristiques de choix de variables/valeurs qui permettent de choisir quelle valeur affecter à quelle variable de manière plus cohérente qu'un simple choix arbitraire.

Dans le prochain chapitre, nous aborderons les principales notions liées à la chimie théorique ainsi qu'aux benzénoïdes.

2. Chimie théorique

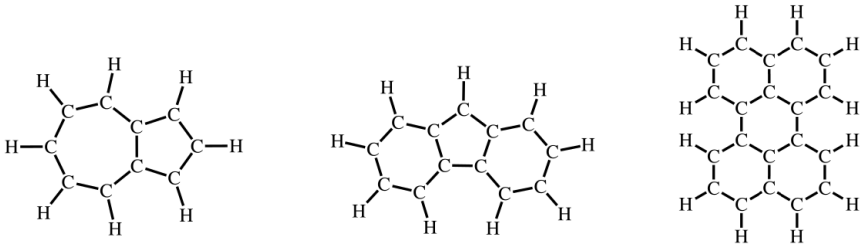
2.1. Introduction

Au cours de cette thèse, nous nous focalisons principalement sur les benzénoïdes. Plus précisément, nous nous attardons sur deux problématiques en particulier : la génération des structures de benzénoïdes répondant à des propriétés spécifiques (section 2.3) et l'estimation de l'*aromaticité* locale d'un benzénoïde donné (section 2.4). Dans la suite de ce chapitre, nous présenterons en détail ces deux problématiques et tâcherons de comprendre les enjeux qui en découlent. Dans un premier temps, nous rappelons quelques notions de base de chimie (section 2.2).

2.2. Définitions préliminaires

Une *molécule* est une structure mettant en relation plusieurs atomes. Ces relations entre atomes sont établies par l'intermédiaire de *liaisons*. L'établissement d'une liaison entre deux atomes se traduit par la mise en commun d'un nombre pair d'électrons. La *valence* d'un atome correspond au nombre de ses électrons susceptibles d'être engagés dans une liaison chimique autour de lui. Chacun de ces électrons peut se lier avec celui d'un autre atome. Les valences des atomes d'hydrogène et de carbone sont respectivement de 1 et de 4. Cela signifie donc qu'un atome d'hydrogène ne peut se lier qu'à un seul autre atome, tandis qu'un atome de carbone peut établir jusqu'à quatre liaisons. Une liaison portant sur une paire d'électrons (respectivement deux paires) est dite *simple* (resp. *double*). Dans le cas de liaisons doubles, les électrons formant la seconde paire sont appelés *électrons-π*.

Comme mentionné dans la partie I, les *hydrocarbures aromatiques polycycliques (HAP)* sont une sous-famille des hydrocarbures ayant pour particularité que leurs atomes de carbones forment des cycles de différentes tailles. Une manière de caractériser les HAP est de les classer en différentes familles, parmi lesquelles nous pouvons citer le *fluorène* et ses dérivés qui comportent un cycle à 5 carbones et au moins deux cycles à 6. On trouve également l'*azulène* et ses dérivés, qui comportent au moins un cycle à 7, un cycle à 5 et zéro ou plusieurs cycles à 6. Pour finir, on peut également citer la famille des *benzénoïdes*, qui regroupe tous les HAP possédant uniquement des cycles de taille 6 (c'est-à-dire des hexagones). C'est principalement uniquement sur cette dernière que portent les travaux présentés dans cette thèse. La figure 2.1 décrit la structure chimique de l'azulène (a), du fluorène (b) et du pérylène, un benzénoïde bien connu des chimistes (c).



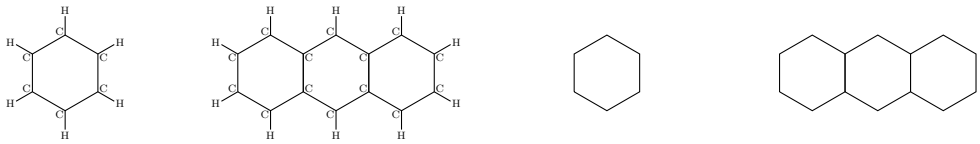
(a) (b) (c)

FIGURE 2.1. – L'azulène (a), le fluorène (b) et le pérylène (c).

Dans le cas particulier des benzénoïdes, chaque atome de carbone intervient dans au moins un cycle de six atomes de carbone. Ainsi, pour respecter les valences du carbone et de l'hydrogène, un atome de carbone sera soit lié à un hydrogène et à deux carbones, soit à trois carbones. La figure 2.2 nous montre deux benzénoïdes bien connus : le benzène de formule C_6H_6 (a) et l'anthracène de formule $C_{14}H_{10}$ (b). Les cycles de six atomes de carbone sont généralement appelés *cycles benzéniques* ou *hexagones*. Ils permettent de proposer une autre manière de caractériser les benzénoïdes. En effet, les benzénoïdes sont les HAP qui peuvent être obtenus par agrégation (ou fusion) de cycles benzéniques. Par exemple, l'anthracène est

constitué de trois cycles fusionnés.

La *taille* d'un benzénoïde est généralement exprimée en fonction du nombre de cycles benzéniques fusionnés. Pour des raisons de simplicité, il est admis d'omettre la présence des atomes d'hydrogène dans la représentation des benzénoïdes. En effet, il est possible de déduire aisément leur présence : tout atome de carbone qui n'est lié qu'à deux autres atomes de carbone est nécessairement lié à un atome d'hydrogène. Cette représentation simplifiée conduit à définir naturellement les benzénoïdes sous la forme de graphes non orientés $B = (V, E)$ tels que chaque atome de carbone correspond à un sommet de V et chaque liaison entre deux atomes de carbones est représentée par une arête de E, la nature de la liaison (simple ou double) n'étant pas ici prise en considération. Les figures 2.2(c) et 2.2(d) nous montrent les graphes associés respectivement au benzène (a) et à l'anthracène (b). Ces graphes ont la particularité d'être connexes, planaires (c'est-à-dire qu'ils peuvent être représentés dans un plan sans que deux arêtes se croisent) et bipartis. On dit qu'un graphe est *biparti* si ses sommets peuvent être partitionnés en deux sous-ensembles V_1 et V_2 de telle sorte à ce que toute arête relie un sommet de V_1 à un sommet de V_2. La figure 2.3 nous montre un partitionnement valide des sommets du graphe décrivant l'anthracène. Ce dernier est donc biparti.



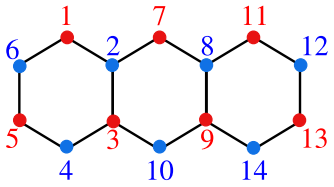
(a) (b) (c) (d)

FIGURE 2.2. – Deux exemples de benzénoïdes : le benzène (a) et l'anthracène (b) ainsi que leurs représentations graphiques (c) et (d).

Comme mentionné précédemment, la représentation sous la forme d'un graphe d'un benzénoïde ne tient pas compte de la nature des liaisons. En chimie, il peut être important de distinguer les liaisons simples des liaisons doubles. En effet, les liaisons doubles, impliquant quatre électrons, sont plus fortes que les liaisons simples. Cela implique que ces dernières sont plus difficiles à rompre, ce qui a pour conséquence une réactivité locale amoindrie. Dans le cas particulier des benzénoïdes, les liaisons doubles sont formées par les électrons-π. La notion de *structure de Kekulé* d'un benzénoïde permet de représenter une possibilité d'étiquetage des liaisons simples et doubles :

Définition 2.2.1 (KEKULÉ 1865) *On appelle* structure de Kekulé *d'un benzénoïde B une application qui, à chaque arête (liaison) de B associe son statut « simple » ou « double » de sorte que chaque atome de carbone soit impliqué dans exactement une seule liaison double. L'ensemble des structures de Kekulé d'un benzénoïde B est dénoté $\mathcal{K}(B)$. Par conséquent, $|\mathcal{K}(B)|$ désigne le nombre de structures de Kekulé de B.*

Usuellement, les liaisons doubles sont matérialisées par une barre supplémentaire entre les deux atomes liés. Par exemple, la figure 2.4 nous montre les quatre différentes structures de Kekulé de l'anthracène. Pour rappel, le benzène admet deux structures de Kekulé (voir figure 0.1), ici, dans le cas de l'anthracène, l'ajout de deux hexagones se traduit par la présence de deux structures de Kekulé supplémentaires. Le nombre de structures de Kekulé peut croître exponentiellement quand on augmente le nombre d'hexagones . De ce fait, énumérer l'intégralité des structures de Kekulé d'un benzénoïde constitue un problème difficile à résoudre (en informatique, on dit qu'un problème est difficile si ce dernier admet une forte combinatoire, ce qui est clairement le cas ici). Heureusement, il a été prouvé que le comptage du nombre de structures de Kekulé pouvait être effectué en temps polynomial dans le cas des benzénoïdes (RISPOLI 2001). En effet, on peut facilement voir qu'une structure de Kekulé d'un benzénoïde correspond à un couplage parfait (c'est-à-dire à un ensemble d'arêtes ne partageant aucun sommet et couvrant l'intégralité des sommets du graphe) du graphe associé à ce dernier. Or, il a été prouvé que compter le nombre de couplages parfaits d'un graphe planaire pouvait être réalisé en temps polynomial (KASTELEYN 1967). Étant donné que les graphes représentant les benzénoïdes sont planaires, cela s'applique également pour le comptage de leurs



(a)

	2	4	6	8	10	12	14
1	1	0	1	0	0	0	0
3	1	1	0	0	1	0	0
5	0	1	1	0	0	0	0
7	1	0	0	1	0	0	0
9	0	0	0	1	1	0	1
11	0	0	0	1	0	1	0
13	0	0	0	0	0	1	1

(b)

FIGURE 2.3. – L'anthracène avec V_1 en rouge et V_2 en bleu (a) ainsi que sa matrice de biadjacence (b)

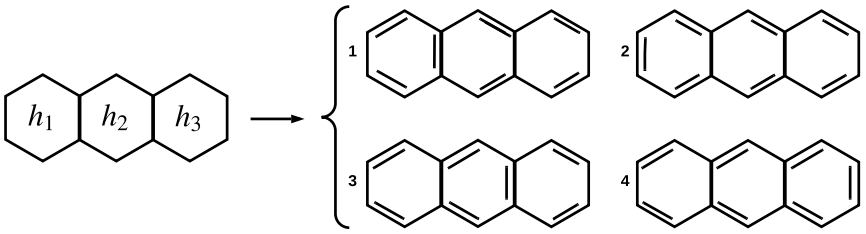


FIGURE 2.4. – Structures de Kekulé de l'anthracène

structures de Kekulé. Rispoli a proposé une méthode efficace pour effectuer ce comptage (RISPOLI 2001). Cette méthode consiste à transformer un benzénoïde $B = (V, E)$ donné en une matrice spécifique dont le déterminant correspond au nombre de structures de Kekulé de B. Cette dernière, appelée *matrice de biadjacence* de B, est obtenue en partitionnant V en deux ensembles disjoints V_1 et V_2 de telle sorte à ce que chaque arête de E lie un sommet de V_1 et un sommet de V_2. Les lignes (resp. colonnes) de cette matrice correspondront aux sommets de V_1 (resp. V_2). La valeur de cette matrice à la position (v_1, v_2) sera égale à 1 si l'arête $\{v_1, v_2\}$ existe, 0 sinon. Rispoli a ensuite démontré le théorème suivant :

Théorème 2.2.1 (RISPOLI 2001) *Le nombre de structures de Kekulé d'un benzénoïde est égal au déterminant de sa matrice de biadjacence dans le cas où* $|V_1| = |V_2|$.

Dans le cas où $|V_1| \neq |V_2|$, il est facilement démontrable que B n'admet aucune structure de Kekulé. Rispoli a également démontré que la création de la matrice et le calcul de son déterminant pouvaient se faire en temps polynomial. La figure 2.3 nous montre la matrice de biadjacence de l'anthracène. Nous pouvons voir que le déterminant de cette matrice est 4, ce qui correspond bien au nombre de structures de Kekulé de l'anthracène.

Nous allons ensuite voir que certaines problématiques de chimie théorique peuvent se traduire sous la forme de problèmes de graphes. En théorie des graphes, la coloration de graphe consiste à attribuer une couleur à chacun de ses sommets de manière que deux sommets reliés par une arête soient de couleur différente. Étant donné une coloration de graphe constituée d'une couleur c, nous notons $n^{(c)}$ le nombre de sommets à être coloriés de cette couleur. Cette notion nous permet d'introduire l'*excès de couleurs* d'un benzénoïde :

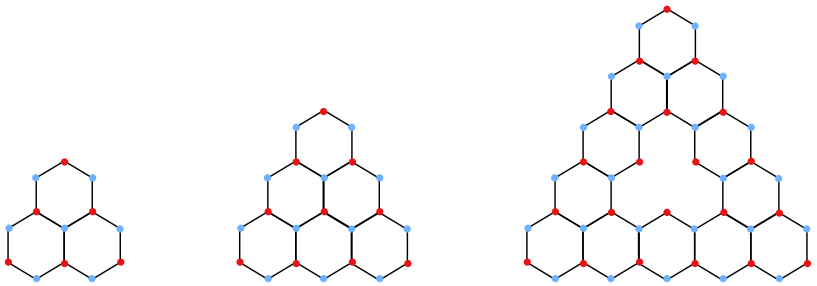
Définition 2.2.2 (Excès de couleurs (CYVIN, BRUNVOLL et CYVIN 1990)) *Étant donné un benzénoïde B et*

une de ses colorations constituée des deux couleurs c_1 *et* c_2 [1]*, son* excès de couleur *est défini par* $\Delta_B = |n^{(c_1)} - n^{(c_2)}|$.

Pour illustrer cela, la figure 2.5 nous montre trois benzénoïdes ayant respectivement un excès de couleur de 1 (a), 2 (b) et 3 (c). Nous introduisons maintenant la notion de *benzénoïdes non kekuléens implicites* (*concealed non kekulean* en anglais) qui se définissent formellement de la manière suivante :

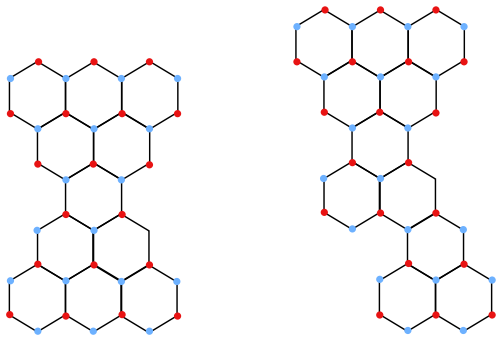
Définition 2.2.3 (Benzénoïde non kekuléen implicite (CYVIN, BRUNVOLL et CYVIN 1990)) *Un benzénoïde B est dit* non kekuléen implicite *si et seulement si* $|\mathscr{K}(B)| = 0$ *et* $\Delta_B = 0$.

Pour illustrer cela, la figure 2.6 nous montre deux des plus petits (11 hexagones) benzénoïdes non kekuléens implicites.



(a) (b) (c)

FIGURE 2.5. – Benzénoïdes ayant un excès de couleurs de 1 (a), 2 (b) et 3 (c)



(a) (b)

FIGURE 2.6. – Deux des plus petits benzénoïdes non kekuléens implicites

On se propose ensuite de définir une propriété chimique liée aux benzénoïdes appelée *paramètre d'irrégularité* (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019). Avant cela, il est nécessaire d'introduire une définition préliminaire :

Définition 2.2.4 (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) *Étant donné un benzénoïde B et h un de ses hexagones, on appelle* groupe de *cet hexagone un ensemble connexe de carbones appartenant à h tel que chaque carbone est lié à un hydrogène. On appelle* solo *(resp.* duo, trio, quatuor*) un groupe constitué d'un seul hexagone (resp. de deux, trois ou quatre). La* taille *d'un groupe est égale au nombre de carbones qu'il contient.*

1. Un graphe biparti peut toujours être colorié avec deux couleurs.

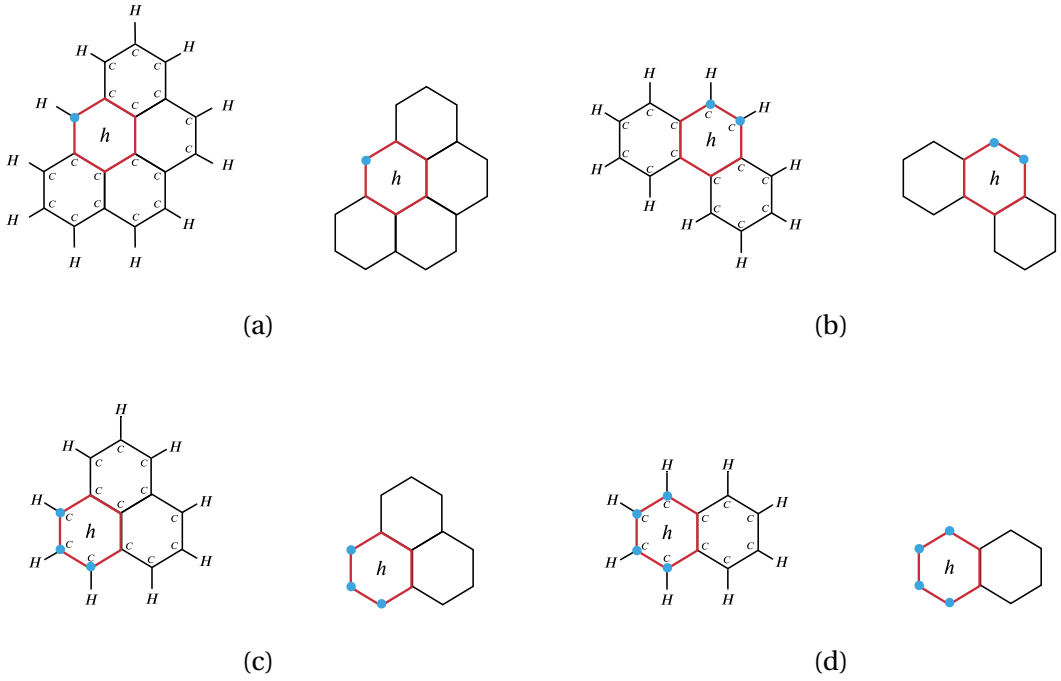


FIGURE 2.7. – Exemples de solo (a), de duo (b), de trio (c) et de quatuor (d).

Étant donné que les atomes d'hydrogènes ne sont pas présents dans notre représentation des structures de benzénoïdes, on peut aussi définir un groupe comme étant un ensemble connexe de carbones d'un hexagone h ayant un degré égal à 2. En effet, dans un benzénoïde, chaque atome de carbone est lié soit à deux carbones (et donc a un degré de 2), soit à trois autres carbones (et donc a un degré de 3). Par exemple, la figure 2.7 nous montre un exemple de chaque type de groupe, avec à chaque fois la molécule originale et sa représentation graphique associée.

Maintenant, on peut définir le *paramètre d'irrégularité* d'un benzénoïde de la manière suivante :

Définition 2.2.5 (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) *Étant donné un benzénoïde B, le* paramètre d'irrégularité *de B, noté ξ_B, est calculé de la manière suivante :*

$$\xi_B = \frac{N_3+N_4}{N_1+N_2+N_3+N_4}$$

où N_i représente le nombre de carbones appartenant à un groupe de taille i.

Par exemple, considérons le phénanthrène représenté dans la figure 2.8, on peut facilement voir qu'il possède 2 quatuors et un seul duo. On a donc $N_2 = 2$, $N_4 = 8$ (car il y a deux carbones appartenant à un duo et huit à un quatuor), et $N_1 = N_3 = 0$ (car le phénanthrène n'admet ni solo, ni trio). Par conséquent, nous avons $\xi = \frac{8}{10} = 0,8$.

Les chimistes exploitent le paramètre d'irrégularité pour caractériser l'irrégularité des HAP. L'irrégularité permet de caractériser l'isotropie d'un benzénoïde. Ainsi, un benzénoïde ayant un paramètre d'irrégularité proche de 0 aura tendance à être plus compact et à avoir un grand nombre de symétries tandis qu'un paramètre d'irrégularité proche de 1 traduira des caractéristiques opposées. Par exemple, dans (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019), Bouwman et al. ont présenté des résultats de calculs de spectres infrarouges sur un ensemble de HAP, et ces derniers étaient classés selon leur paramètre d'irrégularité. Pour information, étant donné que tous les carbones de degrés 2 d'un benzénoïde se situent dans sa bordure, l'irrégularité caractérise totalement cette dernière au même titre que les motifs.

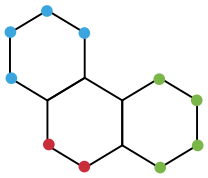


FIGURE 2.8. – Les groupes du phénanthrène.

Dans les sections suivantes, nous aborderons les travaux existants relatifs aux principales problématiques étudiées durant ma thèse. La section 2.3 sera consacrée à la génération de structures de benzénoïdes tandis que la section 2.4 sera consacrée à l'aromaticité des HAP et aux différentes manières de l'estimer.

2.3. Génération de structures de benzénoïdes

Les HAP sont connus pour exhiber une large variété de propriétés physiques et chimiques dépendant de leur taille ou encore de leur topologie. Dans la communauté d'astrophysique, il existe une théorie appelée « *hypothèse des HAP* » soutenant que l'existence des HAP dans l'espace pourrait expliquer certaines émissions non identifiées de bandes de spectres infrarouges moyens dans l'environnement astrophysique. Cette hypothèse est à l'origine de nombreuses recherches, expérimentations et investigations théoriques. Il est maintenant admis que ces ensembles de HAP de différentes tailles, formes et natures ont un rôle dans l'apparence générale des bandes de spectres infrarouges se trouvant dans le milieu interstellaire, mais la question de la sur-représentation de HAP d'une taille ou d'une forme spécifique reste ouverte. Les dernières décennies ont vu naître un grand nombre de travaux visant à déterminer l'influence de la taille, de la forme ou encore les symétries des HAP sur ces bandes de spectres infrarouges (ALLAMANDOLA, HUDGINS et SANDFORD 1999; BAUSCHLICHER, PEETERS et ALLAMANDOLA 2008; RICCA, BAUSCHLICHER, BOERSMA et al. 2012).

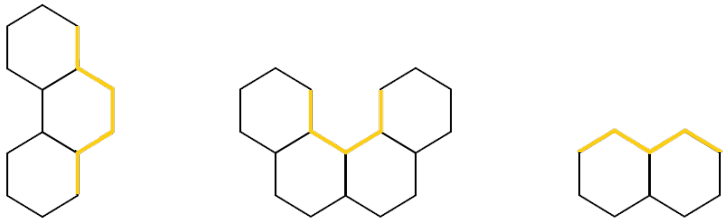
Récemment, une étude menée sur un ensemble de 328 HAPs d'au plus 150 atomes de carbone a montré que les HAP admettant des *armchair edge* (c'est-à-dire une configuration spécifique de la bordure d'un HAP, décrite en jaune dans la figure 2.9 (a). D'un autre côté, il y a déjà la figure de l'intro) ayant un nombre de Clar maximal pourraient émettre une certaine classe de source infrarouge (RICCA, ROSER, PEETERS et al. 2019). Pour ces études, les auteurs avaient besoin de générer systématiquement tous les HAPs admettant un armchair edge. Pour cela, les auteurs ont utilisé l'algorithme de Caporossi et Hansen (CAPOROSSI et HANSEN 1998).

Le contrôle de la synthèse des HAP admettant des formes bien spécifiques est un sujet très récent. Ce dernier suscite d'ailleurs beaucoup d'intérêt comme le démontrent les nombreuses publications récentes dans des journaux de chimie possédant un grand facteur d'impact (QIU, NARITA et MÜLLEN 2020; LIU et FENG 2020; AJAYAKUMAR, MA, LUCOTTI et al. 2021; CHEN, LIN, LUO et al. 2021; CHEUNG, WATANABE, SEGAWA et al. 2021; FUJISE, TSURUMAKI, WAKAMATSU et al. 2021; KANCHERLA et JØRGENSEN 2020; URYU, HIRAGA, KOGA et al. 2020; XIA, PUN, CHEN et al. 2021). Cela a donné lieu à un grand nombre d'études ayant pour objectif de mieux comprendre l'impact de la topologie d'un HAP sur ses propriétés électroniques. Ces études ont donné lieu à une amélioration de certaines propriétés optoélectroniques (DUMSLAFF, GU, PATERNÒ et al. 2020; MARTIN, HAMPEL et JUX 2020; MORI 2021; NATHUSIUS, EJLLI, ROMINGER et al. 2020; QIU, NARITA et MÜLLEN 2021; YANG, ROMINGER et MASTALERZ 2021) ou magnétiques (MISHRA, YAO, CHEN et al. 2021; ZENG, WANG, ZHANG et al. 2021) qui ont pu être combinées dans certains types de molécules admettant des bordures particulières (KEERTHI, SÁNCHEZ-SÁNCHEZ, DENIZ et al. 2020; NIU, MA, SOLTANI et al. 2020; PIZZOCHERO et KAXIRAS 2021; YAO, ZHENG, OSELLA et al. 2021).

Pour citer des exemples récents, il a été prouvé que l'ajout de deux K-régions (voir figure 2.10) supplémentaires à l'hexabenzocoronène conduit à une activité optique améliorée avec des applications potentielles en tant que matériaux laser organiques (DUMSLAFF, GU, PATERNÒ et al. 2020). De plus, il a été prouvé que les HAP admettant des *armchair edges* (voir figure 2.9(a)) étaient des semi-conducteurs ayant une grande bande interdite, tandis que ceux admettant des *zigzag bay* possédaient une meilleure conductivité, mais une stabilité moindre. De ce fait, les chimistes ont essayé de synthétiser des molécules admettant ces deux motifs simultanément. Cela a donné lieu à une meilleure dispersibilité et à des propriétés optoélectroniques améliorées (NIU, MA, SOLTANI et al. 2020).

Niu et al. (NIU, MA, SOLTANI et al. 2020) ont présenté une série de benzénoïdes admettant ces motifs que les chimistes peuvent synthétiser. L'une de ces molécules est décrite dans la figure 2.11, cette dernière contient deux instances du motif *deep bay* (en bleu, voir figure 2.9(b)) et trois instances du motif *zigzag bay* (en rouge, voir figure 2.9(c)).

Tous ces travaux démontrent l'importance d'être capable de générer des benzénoïdes admettant ou non un ou plusieurs motifs donnés.



(a) (b) (c)

FIGURE 2.9. – Les motifs *armchair edge* (a), *deep bay* (b) et *zigzag bay* (c)

FIGURE 2.10. – Le motif $K - région$

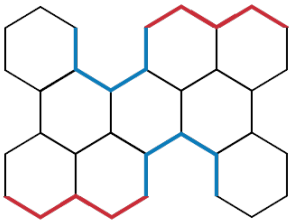


FIGURE 2.11. – Benzénoïde admettant deux occurrences du motif

La NASA possède une base de données recensant un grand nombre de HAP ainsi que certaines de leurs propriétés chimiques (https://www.astrochemistry.org/pahdb/). En 2019, des travaux utilisant cette base ont été présentés dans (BOUWMAN, LINNARTZ et TIELENS 2021). Ces derniers ont consisté premièrement à classer les HAP de cette base selon leur nombre de carbones (≤ 34 ou > 34) et leur paramètre d'irrégularité afin d'effectuer des calculs de spectres infrarouges dessus.

Tous les points évoqués ci-dessus démontrent qu'être capable de générer des benzénoïdes répondant à diverses propriétés données constitue une problématique importante en chimie théorique. Un certain nombre de travaux visant à résoudre cette problématique ont été présentés au cours des dernières années. On peut notamment citer l'algorithme de Brinkmann (BRINKMANN, CAPOROSSI et HANSEN 2002) capable de générer des structures de benzénoïdes possédant un nombre donné d'hexagones. Avant de définir ce dernier plus en détails, il est nécessaire d'introduire quelques notions de théorie des graphes. Tout d'abord, une *face* d'un graphe planaire est une zone du plan délimitée par des arêtes [2]. Le *graphe dual* d'un graphe est le graphe obtenu en remplaçant chacune de ses faces par un sommet, deux sommets seront reliés entre eux par une arête uniquement si les deux faces correspondantes sont adjacentes dans le graphe initial. Dans le cas des benzénoïdes, nous considérons donc une face par hexagone (appelées *faces internes*) ainsi qu'une *face externe* à laquelle toutes les faces correspondant à un hexagone se trouvant sur la bordure du benzénoïde seront connectées. La figure 2.12 représente le graphe dual du graphe décrit par l'anthracène, les faces internes sont notées f_1, f_2, f_3 tandis que la face externe est notée f_{ext}. Dans ses travaux, Brinkmann considère une variante appelée *graphe dual intérieur* qui correspond au graphe dual classique auquel on a retiré la face externe (ainsi, la figure 2.13 (a) représente le graphe dual interne de l'anthracène). La première problématique qui se pose est que deux benzénoïdes chimiquement différents peuvent partager le même graphe dual interne (voir figure 2.13). Pour remédier à cela, l'auteur a introduit la notion de *graphe dual intérieur étiqueté* qui consiste à attribuer une ou plusieurs étiquettes à chaque sommet du graphe dual intérieur en fonction du nombre d'arêtes de l'hexagone associé se trouvant à la périphérie du benzénoïde. La figure 2.14 décrit le graphe dual intérieur étiqueté associé aux deux benzénoïdes de la figure 2.13.

L'algorithme de Brinkmann consiste donc à construire sommet par sommet un graphe dual intérieur (ce qui correspond à construire le benzénoïde hexagone par hexagone). Ensuite, lorsqu'un nouveau graphe dual intérieur est construit, il génère l'ensemble des graphes duaux intérieurs étiquetés en prenant garde à éviter les isomorphismes. Chaque graphe trouvé de cette manière correspondra donc à un nouveau benzénoïde. Bien que très performant, cet algorithme présente deux inconvénients. Premièrement, il est incapable de générer les coronoïdes (les benzénoïdes possédant un ou plusieurs trous). Deuxièmement, le seul critère pouvant être spécifié est le nombre d'hexagones et implémenter des critères spécifiques demanderaient un lourd effort en termes de conception et de programmation, notamment à cause de la détection d'isomorphismes. De plus, les besoins des chimistes évoluent constamment et seraient intéressés par le fait de générer des structures de benzénoïdes ayant des critères plus spécifiques que le nombre d'hexagones, on peut notamment mentionner le fait de vouloir générer des structures admettant un ou plusieurs motif(s) (des sous-parties d'un benzénoïde possédant une forme spécifique) donné(s) (RIEGER et MÜLLEN 2010). Toutes les raisons énoncées font que ce type d'algorithme est difficilement adaptable aux besoins des chimistes. Au niveau des algorithmes de génération, on peut également citer celui de Caporossi et Hansen mentionné plus haut (CAPOROSSI et HANSEN 1998). Ce dernier consiste à construire l'intégralité des benzénoïdes ayant un nombre donné d'hexagones à l'aide d'un parcours en profondeur ou en largeur de l'espace de recherche.

2. On considère que le graphe est plongé sur une surface plane. Lorsque plusieurs plongements existent, on choisit celui pour lequel chaque face correspond à un hexagone.

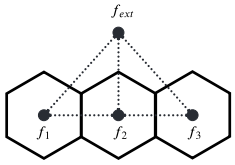
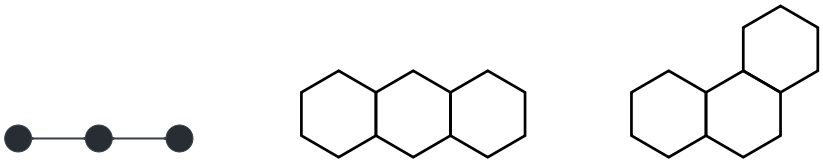


FIGURE 2.12. – Graphe dual du graphe décrivant l'anthracène



(a) (b) (c)

FIGURE 2.13. – Deux benzénoïdes (b, c) partageant le même graphe dual intérieur (a)

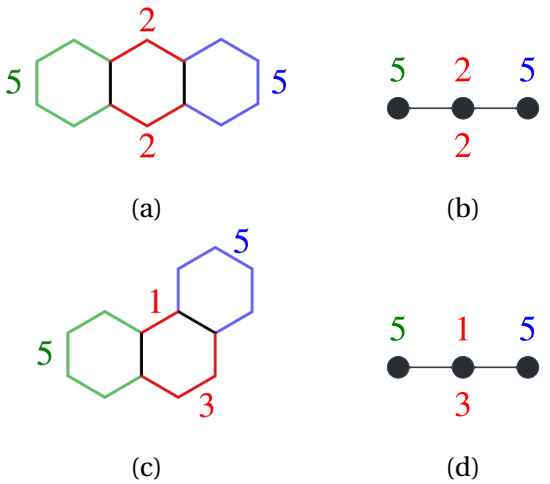
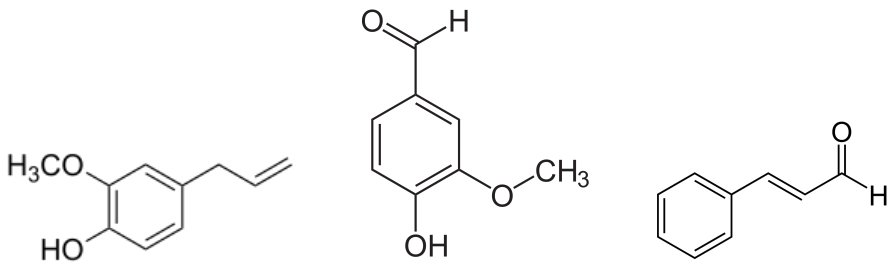


FIGURE 2.14. – Construction (a, c) et représentation (b, d) du graphe dual intérieur étiqueté des deux benzénoïdes de la figure 2.13

2.4. Estimation de l'aromaticité

2.4.1. Introduction

Comme mentionné dans la partie I, les chimistes ont longtemps eu du mal à définir l'aromaticité. Le premier critère pour déterminer si un composant était aromatique était son odeur. En effet, on s'est rendu compte que les molécules supposées aromatiques dégageaient une odeur souvent agréable. On peut citer par exemple l'eugénol, la vanilline ou encore la cinnamaldéhyde qui dégagent respectivement une odeur boisée, de vanille et de cannelle (qui ne sont pas des HAP car ils contiennent de l'oxygène). Ces molécules sont décrites dans la figure 2.15.



(a) (b) (c)

FIGURE 2.15. – Les molécules d'eugénol (a), de vanilline (b) et de cinnamaldéhyde (c)

Au final, l'*aromaticité* fut définie comme étant un phénomène induit par la délocalisation des électrons-π de la molécule considérée. Cette dernière se traduit par des propriétés moléculaires particulièrement intéressantes (stabilité thermodynamique, propriétés magnétiques, réactivité particulière ...). En d'autres termes, comme mentionné par Chen et al. (CHEN, WANNERE, CORMINBOEUF et al. 2005), l'aromaticité peut se définir de la manière suivante :

≪ Aromaticity is a manifestation of electron delocalization in closed circuits, either in two or in three dimensions. This results in energy lowering, often quite substantial, and a variety of unusual chemical and physical properties. These include a tendency toward bond length equalization, unusual reactivity, and characteristic spectroscopic features. Since aromaticity is related to induced ring currents, magnetic properties are particularly important for its detection and evaluation. ≫

Pour rappel, l'aromaticité n'est pas quelque chose de mesurable. En effet, cette dernière est seulement estimable à l'aide de méthodes reposant sur différents critères, dont :

- des critères *topologiques* : estimations liées à la topologie de la molécule considérée. On peut citer la méthode des *Ring Bond Order (RBO)* (voir la sous-sous-section 2.4.6) ou celle des *couvertures de Clar* (voir la sous-section 2.4.5).

- des critères *énergétiques* : consiste à estimer l'aromaticité de la molécule considérée en lui attribuant une ou plusieurs valeurs énergétiques. On peut par exemple citer l'*énergie de résonance* ainsi que les algorithmes qui en découlent (LIN et FAN 1999 ; LIN 2000) (voir la sous-section 2.4.4).

- des critères *magnétiques* : consiste à estimer l'aromaticité de la molécule considérée en observant la réponse de ses électrons à l'application d'un champ magnétique perpendiculaire à cette dernière. Le sens de rotation des électrons détermine alors son aromaticité. Parmi les différentes méthodes utilisant ces critères, on peut citer *Nucleus Independent Chemical Shift (NICS)* (voir la sous-section 2.4.2) ou encore *Isotropic Magnetic Shielding (IMS)* (voir la sous-section 2.4.3).

On se propose maintenant de détailler toutes les méthodes permettant d'estimer l'aromaticité d'un benzénoïde donné. La suite de cette section est organisée de la manière suivante :

- La sous-section 2.4.2 parlera de la méthode *Nucleus Independent Chemical Shift (NICS)*.

- La sous-section 2.4.3 décrira la méthode *Isotropic Magnetic Shielding (IMS)* présentée dans la sous-sous-section 2.4.3.

- La sous-section 2.4.4 détaillera le fonctionnement des algorithmes de Lin et Fan et de Lin, tous deux basés sur la méthode de (RANDIĆ 2019), consistant à calculer l'énergie de résonance d'un benzénoïde en énumérant ses circuits conjugués.

- La sous-section 2.4.5 donnera plus de détails sur la méthode permettant d'estimer l'aromaticité d'un benzénoïde à l'aide de ses couvertures de Clar.

- La sous-section 2.4.6 parlera de la méthode permettant d'estimer l'aromaticité locale d'un benzénoïde en calculant ses *Ring Bond Order (RBO)*.

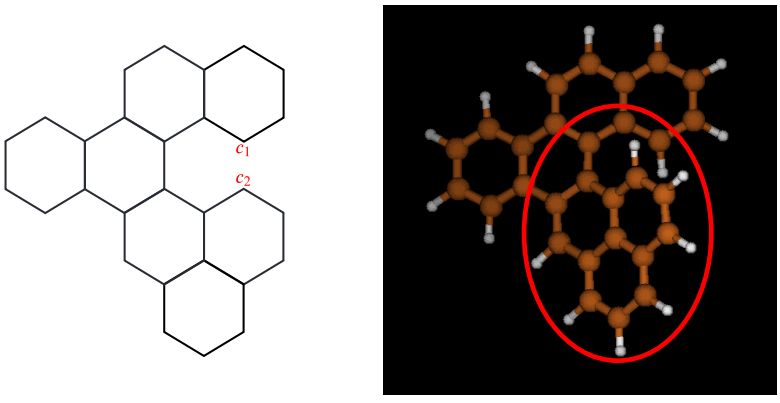
2.4.2. Nucleus Independent Chemical Shift (NICS)

Pour rappel, *Nucleus Independent Chemical Shift (NICS)* (DRANSFELD, SCHLEYER et VAN EIKEMA HOMMES 1996) est la méthode de référence des chimistes pour estimer l'aromaticité locale d'un benzénoïde. Cette méthode est dite *magnétique*. En effet, elle consiste à appliquer un champ magnétique perpendiculaire à la molécule traitée qui va induire une rotation de ses électrons. Le sens de rotation de ces derniers déterminera une estimation de l'aromaticité de la molécule. Plus formellement, NICS se divise en trois étapes :

- Optimiser la géométrie du benzénoïde traité : il faut prendre en compte le fait que la représentation graphique d'un benzénoïde peut être différente de la géométrie qu'il aura tendance à avoir naturellement. En effet, les liaisons C-C ont une longueur d'environ 1,4 Å (1/10 000 de micron) et les liaisons C-H une longueur d'environ 1,1 Å. Ces règles de liaison impliquent que certaines molécules auront tendance à se « tordre » naturellement afin de conserver une certaine distance entre ses atomes non liés. On parle alors de *géométrie optimisée*. La figure 2.16 illustre un exemple pour un benzénoïde à 7 hexagones, on peut facilement voir que la partie entourée en rouge a été déformée afin d'éviter une trop grande proximité entre les carbones C_1 et C_2 ainsi qu'entre les hydrogènes auxquels ils sont liés. La première étape de la méthode NICS consiste donc à modifier la géométrie initiale de la molécule traitée de sorte que deux atomes non liés ne soient pas trop proches.

- Déterminer les positions où seront calculées les valeurs : nous devons placer un *atome fantôme* (noté Bq, en référence à Banquo de Hamlet) à chacune des positions où l'on souhaite obtenir une valeur (ces positions doivent être différentes de celles des atomes déjà présents dans la molécule). En effet, si l'on veut déterminer la valeur d'une position donnée, il est obligatoire de considérer qu'un atome s'y trouve. Dans notre cas, nous choisissons de placer un atome fantôme sur chaque barycentre de chaque hexagone de la molécule traitée.

- Déterminer le champ magnétique induit par les atomes fantômes : ici, on simule l'application d'un champ magnétique perpendiculaire à la molécule qui va se traduire par une rotation des électrons de cette dernière. La valeur NICS de l'atome fantôme concerné dépendra du sens de cette rotation.



(a) (b)

FIGURE 2.16. – La représentation graphique d'un benzénoïde à 7 hexagones (a) et sa géométrie optimisée (b)

Le logiciel commercial Gaussian (https://gaussian.com/, (FRISCH, TRUCKS, SCHLEGEL et al. 2016)) propose une implémentation de la méthode NICS. C'est avec ce dernier que seront effectuées toutes les expérimentations qui seront présentées dans cette thèse. Notons que Gaussian n'est pas le seul logiciel à proposer une implémentation de NICS, on peut également citer Turbomole (https://www.turbomole.org/), un autre logiciel commercial. Bien que nous ayons choisi d'utiliser Gaussian pour effectuer nos travaux, ce choix est purement arbitraire. En effet, il existe potentiellement quelques différences d'implémentation entre les deux logiciels. Néanmoins, ces derniers fournissent des résultats très similaires en termes de temps et de valeurs. Notons que les valeurs NICS calculées de cette manière respectent la *convention NICS* qui consiste à considérer les valeurs proches de 0 comme étant *non-aromatiques*, les valeurs négatives et non proches de zéro comme étant *aromatiques* et les valeurs positives non proches de zéro comme étant *anti-aromatiques*.

Il faut savoir que tout calcul de chimie quantique nécessite de préciser une méthode de calcul (DFT ou autre) et une base de fonctions gaussiennes (6-31G, def2-TZVP ou autres). Tous les résultats issus de la méthode NICS présentés dans cette thèse ont été réalisés avec la fonctionnelle B3LYP et la base def2-TZVP. Pour plus d'informations à ce sujet, nous invitons le lecteur à se référer aux travaux de Cramer (CRAMER 2004).

2.4.3. Isotropic Magnetic Shielding (IMS)

La méthode *Isotropic Magnetic Shielding (IMS)* (LAMPKIN, KARADAKOV et VANVELLER 2020) permet également de fournir une estimation de l'aromaticité d'un benzénoïde. Étant donné un benzénoïde, cette dernière consiste à calculer les valeurs NICS sur un très grand nombre de ses points. Il en résulte une cartographie complète détaillant les zones du benzénoïde traité les plus aromatiques. La figure 2.17 illustre ce procédé en décrivant la carte IMS du coronène.

Le grand avantage de cette méthode est qu'elle permet de fournir une vision globale et très précise de l'aromaticité d'un benzénoïde. Le nombre de points sur lesquels on calcule les valeurs NICS est paramétrable par l'utilisateur. Plus il sera élevé, plus la carte obtenue sera précise (la méthode décrite dans la sous-section 2.4.2 se contente de calculer une valeur NICS pour chaque hexagone du benzénoïde traité). Cependant, comme mentionné dans la sous-section 2.4.2, calculer la valeur NICS d'un point requiert des calculs de chimie quantique très coûteux. Ainsi, calculer la carte IMS d'un benzénoïde de taille moyenne peut facilement prendre plusieurs heures (voire plus d'une journée). Il convient de mentionner que les cartes d'aromaticité qui seront présentées au cours de cette thèse respectent la *convention IMS*. Cette dernière considère les valeurs proches de 0 comme étant *non-aromatiques*, les valeurs positives et non proches de zéro comme étant *aromatiques* et les valeurs négatives non proches de zéro comme étant *anti-aromatiques*. Cette convention est l'opposée de la convention NICS mentionnée dans la sous-section 2.4.2. Le logiciel Gaussian présenté dans cette même sous-section est également capable de calculer la carter IMS d'un benzénoïde.

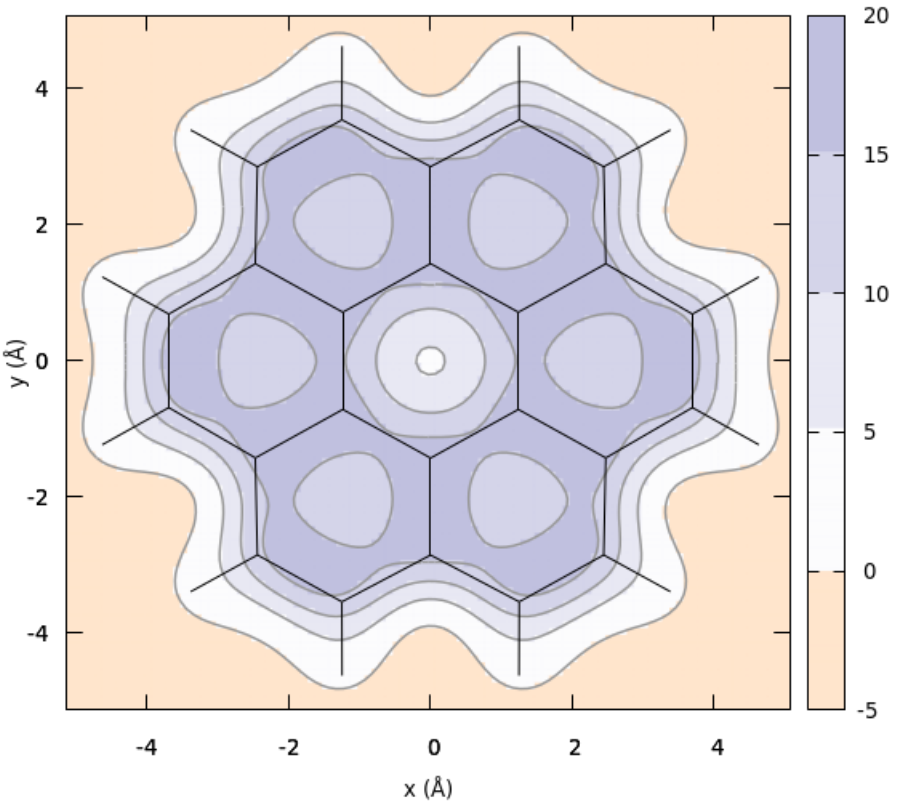


FIGURE 2.17. – Carte IMS du coronène

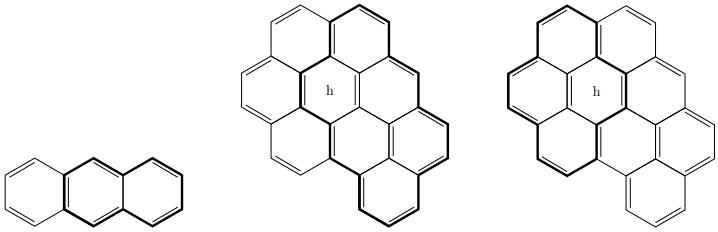
2.4.4. Énergie de résonance

L'*énergie de résonance* est une notion utilisée pour quantifier l'énergie induite par l'aromaticité d'un benzénoïde. Elle nous permet d'obtenir des informations sur sa stabilité (au plus l'énergie d'un benzénoïde est basse, au plus sa stabilité est forte). Il existe deux manières de calculer cette énergie : de manière globale en attribuant une valeur au benzénoïde (on parle alors d'*énergie de résonance globale*) ou de manière locale en attribuant une valeur à chacun des hexagones du benzénoïde (on parle alors d'*énergie de résonance locale*). Cette dernière est particulièrement intéressante. En effet, elle permet de connaître les parties les moins stables d'un benzénoïde. Étant donné que les réactions chimiques ont tendance à se produire sur ces parties-là, il est possible d'utiliser cette notion afin de prédire l'endroit où aura lieu une réaction.

Randić (RANDIĆ 2019) a présenté une méthode permettant de calculer l'énergie de résonance (locale ou globale) d'un benzénoïde consistant à énumérer l'ensemble des *circuits conjugués minimaux linéairement indépendants* (cette notion sera explicitée dans la suite de la sous-section) de chacune de ses structures de Kekulé. Avant d'aller plus loin, nous devons introduire certaines définitions :

Définition 2.4.1 (RANDIĆ 1976b) *Soient B un benzénoïde et K une de ses structures de Kekulé. Un circuit conjugué $\mathscr{C}$ de K est un cycle de B dont les arêtes alternent entre liaison simple et liaison double dans K. La taille de $\mathscr{C}$ (notée $|\mathscr{C}|$) est l'entier i tel que $\mathscr{C}$ possède $4i+2$ arêtes.*

Un circuit conjugué est donc un cycle alternant entre liaison simple et liaison double. Par exemple, le cycle représenté en gras dans la figure 2.18 (a) est un circuit conjugué de taille 2. Étant donné un benzénoïde



(a) (b) (c)

FIGURE 2.18. – Un circuit conjugué en gras (a) et un exemple de circuits redondants (b) et (c).

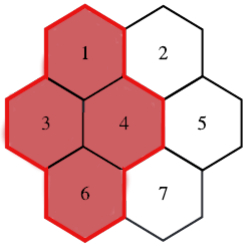


FIGURE 2.19. – Intérieur d'un circuit conjugué de taille 3

B et un de ses cycles $\mathscr{C}$, on appelle *intérieur de* $\mathscr{C}$ le sous-graphe induit par tous les sommets et arêtes se trouvant à l'intérieur de $\mathscr{C}$. La figure 2.19 décrit l'intérieur d'un circuit conjugué de taille 3. Nous pouvons maintenant aborder la notion de couverture d'un circuit conjugué :

Définition 2.4.2 (RANDIĆ et GUO 1994) *Soient B un benzénoïde et K une de ses structures de Kekulé. On dit qu'un circuit conjugué $\mathscr{C}$ de K* couvre *un hexagone h si et seulement si h se trouve entièrement à l'intérieur de $\mathscr{C}$. On parle de* circuits redondants *lorsque deux circuits conjugués couvrent le même hexagone.*

Considérons par exemple la structure de Kekulé de la figure 2.18 (b, c). L'hexagone h est couvert par deux circuits conjugués : un premier de taille 4 (figure 2.18 (b)) et un autre de taille 3 (figure 2.18 (c)). Dans ce genre de cas, on ne souhaite prendre en compte que le circuit de plus petite taille, ce qui nous permet d'introduire la notion de *minimalité* :

Définition 2.4.3 (RANDIĆ et GUO 1994) *Soient B un benzénoïde, h un de ses hexagones et K une de ses structures de Kekulé. On dit que $\mathscr{C}$ est un* circuit conjugué minimal *(h-CCM) de l'hexagone h dans K si $\mathscr{C}$ est un des circuits couvrant h ayant la plus petite taille.*

Si l'on considère les figures 2.18 (b, c), le circuit de taille 3 est un h-CCM.

Le principe de la méthode décrite par Randić est d'attribuer à chacun de ces circuits conjugués une énergie inversement proportionnelle à leurs tailles. Un circuit de petite taille se verra attribuer une énergie élevée et inversement. On note R_i l'énergie induite par un circuit conjugué de taille i. Initialement, ces valeurs sont calculées en utilisant la formule $R_i = \frac{1}{i^2}$. Néanmoins, des valeurs optimisées ont été établies par régression linéaire pour les circuits de taille au plus 4 : $R_1 = 0,869, R_2 = 0,246, R_3 = 0,100$ et $R_4 = 0,041$ (RANDIĆ 2019). Ces valeurs permettent de calculer l'énergie induite par une structure de Kekulé donnée :

Définition 2.4.4 (RANDIĆ et GUO 1994) *Soient B un benzénoïde et K une de ses structures de Kekulé. L'énergie $R(K)$ induite par les circuits minimaux de K est définie de la manière suivante :*

$$R(K) = \sum_{i \in \{1,2,\ldots\}} r_i(K) \times R_i$$

structure	hexagone	valeur du circuit	$R(K_i)$	$E(B)$
K_1	h_1 h_2 h_3	R_1 R_1 R_2	$2R_1 + R_2 = 1,984$	$\frac{6R_1+4R_2+2R_3}{4} = 1,5995$
K_2	h_1 h_2 h_3	R_1 R_2 R_3	$R_1 + R_2 + R_3 = 1,215$	
K_3	h_1 h_2 h_3	R_2 R_1 R_1	$2R_1 + R_2 = 1,984$	
K_4	h_1 h_2 h_3	R_3 R_2 R_1	$R_1 + R_2 + R_3 = 1,215$	

TABLEAU 2.1. – Détails du calcul de l'énergie de résonance globale de l'anthracène

où $r_i(K)$ est le nombre de circuits minimaux de taille i dans K.

Par extension, nous pouvons définir l'énergie induite par un benzénoïde :

Définition 2.4.5 (RANDIĆ et GUO 1994) *Soit B un benzénoïde. L'énergie $R(B)$ induite par B est définie de la manière suivante :*

$$R(B) = \sum_{K \in \mathscr{K}(B)} R(K)$$

Nous pouvons ensuite définir l'*énergie de résonance* d'un benzénoïde :

Définition 2.4.6 (RANDIĆ et GUO 1994) *Soit B un benzénoïde. L'*énergie de résonance *$E(B)$ de B est définie de la manière suivante :*

$$E(B) = \frac{R(B)}{|\mathscr{K}(B)|}$$

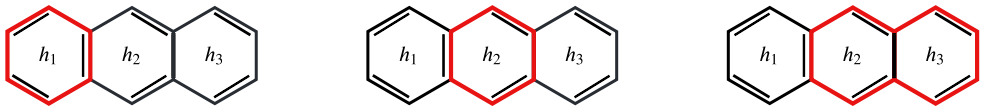
Considérons B comme étant la molécule d'anthracène (dont les structures de Kekulé sont représentées dans la figure 2.4). Le tableau 2.1 nous montre les détails du calcul de son énergie de résonance globale. Par exemple, si l'on considère K_1 comme étant sa première structure de Kekulé, nous obtenons :

- un circuit minimal de taille 1 couvrant l'hexagone h_1 (le circuit induit par l'hexagone h_1, voir la figure 2.20 (a)),
- un autre circuit minimal de taille 1 couvrant l'hexagone h_2 (le circuit induit par l'hexagone h_2, voir la figure 2.20 (b)),
- un circuit minimal de taille 2 couvrant l'hexagone h_3 (le circuit induit par les hexagones h_2 et h_3, voir la figure 2.20 (c)).

On obtient donc $R(K_1) = 2R_1 + R_2 = 1,984$. Si l'on applique le même procédé pour les trois autres structures de Kekulé, nous obtenons :

$$E(B) = \frac{6R_1 + 4R_2 + 2R_3}{4} = 1,5995$$

Pour finir, on appelle *énergie de résonance locale* d'un benzénoïde B et d'un hexagone h l'énergie obtenue en appliquant la formule précédente, mais en ne considérant uniquement que les circuits conjugués



(a) (b) (c)

FIGURE 2.20. – Les trois circuits conjugués minimaux de la première structure de Kekulé de l'anthracène

hexagone	structure	valeur du circuit	$E(B,h_i)$
h_1	K_1 K_2 K_3 K_4	R_1 R_1 R_2 R_3	$\frac{2R_1+R_2+R_3}{4} = 0,521$
h_2	K_1 K_2 K_3 K_4	R_1 R_2 R_1 R_2	$\frac{2R_1+2R_2}{4} = 0,5575$
h_3	K_1 K_2 K_3 K_4	R_2 R_3 R_1 R_1	$\frac{2R_1+R_2+R_3}{4} = 0,521$

TABLEAU 2.2. – Détails du calcul de l'énergie de résonance locale de l'anthracène

minimaux couvrant l'hexagone h :

Définition 2.4.7 *Soient B un benzénoïde et h un de ses hexagones.* L'énergie de résonance locale $E(B,h)$ *de h est définie de la manière suivante :*

$$E(B,h) = \frac{\sum_{K \in \mathcal{K}(B),\ \mathscr{C}\ un\ h-MCC\ dans\ K} R_{|\mathscr{C}|}}{|\mathcal{K}(B)|}$$

Les détails du calcul de l'énergie de résonance locale de l'anthracène sont quant à eux détaillés dans le tableau 2.2. Si l'on considère les circuits minimaux couvrant h_1 de chacune de ses structures de Kekulé, nous obtenons :

- un circuit minimal de taille 1 pour K_1 (le circuit induit par l'hexagone h_1, voir la figure 2.21 (a)),
- un autre circuit minimal de taille 1 pour K_2 (le circuit induit par l'hexagone h_1, voir la figure 2.21 (b)),
- un circuit minimal de taille 2 pour K_3 (le circuit induit par les hexagones h_1 et h_2, voir la figure 2.21 (c)),
- un circuit minimal de taille 3 pour K_4 (le circuit induit par les hexagones h_1, h_2 et h_3, voir la figure 2.21 (d)).

Nous avons donc $E(B,h_1) = \frac{2R_1+R_2+R_3}{4} = 0,521$. Pour obtenir les valeurs relatives aux autres hexagones, il suffit de répéter le même processus. On peut facilement voir que l'énergie de résonance locale d'un hexagone est comprise entre 0 et 1. Une valeur proche de 1 est synonyme de forte aromaticité et inversement. Pour terminer, nous pouvons noter que pour un benzénoïde donné, la somme des aromaticités locales de chacun de ses hexagones est égale à son aromaticité globale.

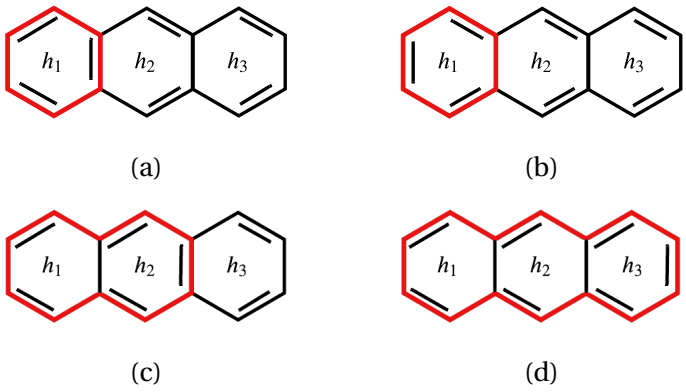


FIGURE 2.21. – Les circuits conjugués minimaux du premier hexagone de l'anthracène pour chacune de ses structures de Kekulé

Dans la suite de cette sous-section, nous présenterons les différents algorithmes permettant de calculer l'énergie de résonance d'un benzénoïde donné. À partir de maintenant, nous considérons un hexagone comme étant un tuple de 6 sommets $h = (v_1, v_2, v_3, v_4, v_5, v_6)$. Les sommets d'un tel hexagone sont listés dans le sens horaire, en partant du sommet se situant tout en haut. Dans d'autres termes, v_1 se réfère au sommet situé en haut de l'hexagone, v_2 à celui situé en haut à droite, etc.

2.4.4.1. Calculer l'énergie de résonance d'un benzénoïde en énumérant ses structures de Kekulé

En 1999, Lin et Fan (LIN et FAN 1999) ont présenté une méthode capable d'estimer l'aromaticité d'un benzénoïde donné basée sur le calcul de son énergie de résonance. Étant donné un benzénoïde B, cette méthode (qui est décrite dans l'algorithme 2) va tout d'abord énumérer tous les circuits conjugués minimaux (c'est-à-dire calculer un circuit conjugué minimal pour chaque hexagone h) pour chacune de ses structures de Kekulé. Ensuite, on peut déduire l'énergie induite par chacun de ces circuits et les additionner. Pour finir, il suffit de diviser cette somme par le nombre de structures de Kekulé de B pour obtenir son énergie de résonance globale. Cette méthode utilise donc les travaux de Randić présentés ci-dessus. La contribution principale des auteurs réside dans la manière de calculer les h-CCM.

Soient B un benzénoïde, K une de ses structures de Kekulé et h un hexagone de B. Lin et Fan vont identifier le h-CCM en fonction de la position des doubles liaisons sur l'hexagone h dans K. Pour chacune des configurations de doubles liaisons de h, ils vont calculer les chemins de plus petite longueur dont les arêtes alternent entre liaison simple et double entre tous les sommets de h (que l'on appellera *chemin conjugué minimal*). Notons qu'un chemin conjugué minimal entre deux sommets de l'hexagone h ne doit contenir aucune arête de ce dernier. La figure 2.22 liste toutes les configurations de doubles liaisons possibles pour un hexagone h identifiées par Lin et Fan ainsi que les h-CCM qui en découlent. Par exemple, si $h = (v_1, v_2, v_3, v_4, v_5, v_6)$ possède la configuration de doubles liaisons décrite dans la configuration 2 de la figure 2.22, alors l'algorithme de Lin et Fan va construire le circuit conjugué minimal couvrant h en trouvant le chemin conjugué minimal entre v_1 et v_4 et en l'ajoutant au chemin décrit par $v_1 - v_2 - v_3 - v_4$ (ou $v_1 - v_6 - v_5 - v_4$). Ici, $v_1 - v_2 - v_3 - v_4$ représente le chemin obtenu en considérant les arêtes $\{v_1, v_2\}$, $\{v_2, v_3\}$ et $\{v_3, v_4\}$. L'opération de concaténation de chemins est notée $\oplus$ dans l'algorithme 2.

L'algorithme 2 décrit cette méthode. Premièrement, pour chaque structure de Kekulé K du benzénoïde passé en entrée (ligne 2) et chaque hexagone h (ligne 3), il détermine la configuration de liaisons doubles de h dans K à l'aide de la fonction $configuration\_liaisons\_doubles$ (ligne 4) en accord avec la figure 2.22. Dans le cas de la première configuration, la méthode $chemin\_droit$ (ligne 7) sert à déterminer si on se situe dans le cas (a) ou (b) décrit dans la figure 2.22. De plus, les configurations 3 et 4 peuvent également induire deux circuits de tailles différentes, il convient donc de déterminer lequel d'entre eux est le plus petit à

Algorithme 2 : *Calcul_Énergie_Résonance_Lin_Fan (CER-LF)*

Input : un benzénoïde B
Output : l'énergie de résonance globale $E(B)$

1 $énergie \leftarrow 0$
2 **foreach** $K \in \mathcal{K}(B)$ **do**
3 **foreach** $h = (v_1, v_2, \ldots, v_6) \in hexagones(B)$ **do**
4 $config \leftarrow configuration\_liaisons\_doubles(K, h)$
5 **if** $config = 1$ **then**
6 $P \leftarrow chemin\_minimal(h, v_5, v_6)$
7 **if** $chemin\_droit(P)$ **then**
8 $\mathscr{C} \leftarrow v_6 - v_1 - v_2 - v_3 - v_4 - v_5 \oplus chemin\_minimal(h, v_5, v_6)$
9 **else** $\mathscr{C} \leftarrow v_6 - v_5 \oplus chemin\_minimal(h, v_5, v_6)$
10 **else if** $config = 2$ **then**
11 $\mathscr{C} \leftarrow v_1 - v_2 - v_3 - v_4 \oplus chemin\_minimal(h, v_4, v_1)$
12 **else if** $config = 3$ **then**
13 $\mathscr{C}_1 \leftarrow v_1 - v_2 \oplus chemin\_minimal(h, v_2, v_5) \oplus v_5 - v_6 \oplus chemin\_minimal(h, v_6, v_1)$
14 $\mathscr{C}_2 \leftarrow chemin\_minimal(h, v_1 - v_2) \oplus v_2 - v_3 - v_4 - v_5 \oplus chemin\_minimal(h, v_5, v_6)$
15 $\mathscr{C} \leftarrow plus\_petit\_circuit(C_1, C_2)$
16 **else if** $config = 4$ **then**
17 $\mathscr{C}_1 \leftarrow chemin\_minimal(h, v_1, v_2) \oplus v_2 - v_3 \oplus chemin\_minimal(h, v_3, v_4) \oplus v_4 - v_5 \oplus chemin\_minimal(h, v_5, v_6) \oplus v_6 - v_1$
18 $\mathscr{C}_2 \leftarrow v_1 - v_2 \oplus chemin\_minimal(h, v_2, v_3) \oplus v_3 - v_4 \oplus chemin\_minimal(h, v_4, v_5) \oplus v_5 - v_6 \oplus chemin\_minimal(h, v_6, v_1)$
19 $\mathscr{C} \leftarrow plus\_petit\_circuit(C_1, C_2)$
20 $énergie \leftarrow énergie + R_{|\mathscr{C}|}$
21 **return** $\frac{énergie}{|\mathcal{K}(B)|}$

l'aide de la fonction $plus\_petit\_circuit$ (lignes 15 et 19). Ensuite, il utilise la méthode $chemin\_minimal$ (qui va calculer le chemin conjugué minimal entre u et v) afin de construire un circuit minimal en accord avec sa configuration de doubles liaisons (lignes 5-19). À chaque nouvel h-CCM calculé, il va ajouter sa contribution énergétique (ligne 20). Pour finir, la somme des contributions de chaque circuit minimal est divisée par le nombre de structures de Kekulé de B afin d'obtenir l'énergie de résonance de ce dernier (ligne 21).

Le principal inconvénient de cette méthode est qu'elle nécessite d'énumérer l'intégralité des structures de Kekulé d'un benzénoïde. Étant donné que ce nombre peut être exponentiel, cette méthode est clairement inefficace en pratique et ne peut être utilisée que sur des benzénoïdes possédant un faible nombre de structures de Kekulé.

2.4.4.2. Calculer l'énergie de résonance d'un benzénoïde sans énumérer ses structures de Kekulé

Afin de résoudre les problèmes liés à la méthode précédente, Lin a présenté en 2000 une autre méthode capable de faire une approximation efficace de l'énergie de résonance d'un benzénoïde donné sans avoir à énumérer l'ensemble de ses structures de Kekulé. En contrepartie, cette méthode ne considère que les circuits conjugués de taille au plus 4. Cela n'est pas forcément problématique, car les circuits de taille supérieure apportent une énergie moindre qui peut être négligeable. Afin de décrire cette méthode, nous devons introduire différentes définitions :

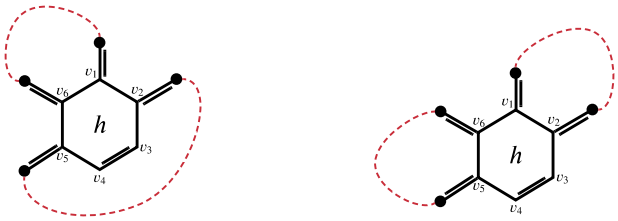
Définition 2.4.8 *Soient B un benzénoïde et $\mathscr{C}$ un de ses cycles (avec $4i+2$ arêtes, $i \in \mathbb{N}$). $M(\mathscr{C})$ est le nombre*

(a) (b)

Configuration 1

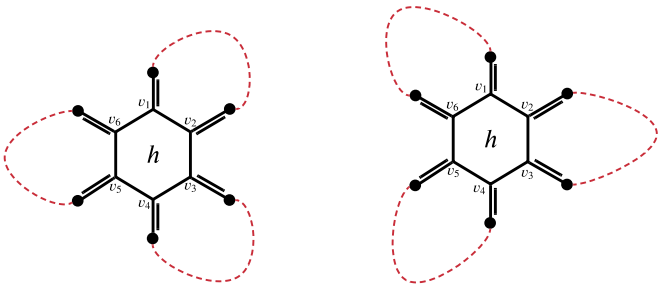
(c) (d)

Configuration 2



(e) (f)

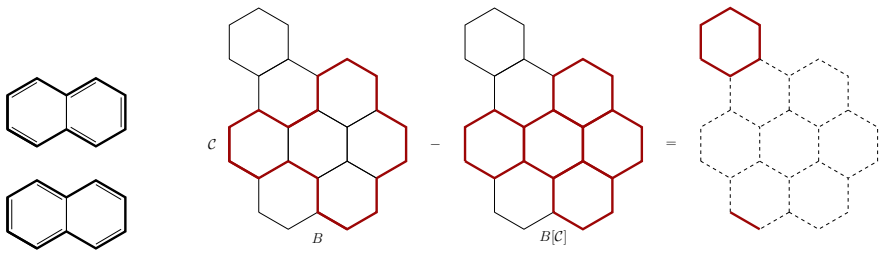
Configuration 3



(g) (h)

Configuration 4

FIGURE 2.22. – Toutes les configurations de liaisons doubles possibles pour un hexagone (LIN et FAN 1999).



(a) (b)

FIGURE 2.23. – Un cycle $\mathscr{C}$ tel que $M(\mathscr{C}) = 2$ (a) et un exemple du calcul du nombre d'occurrences d'un cycle (b).

de couplages parfaits de $\mathscr{C}$ et de son intérieur induisant un circuit minimal pour au moins un des hexagones qu'il couvre.

Par exemple, considérons $\mathscr{C}$ comme étant le cycle de taille 2 représenté dans la figure 2.23(a). $\mathscr{C}$ peut induire deux circuits conjugués différents qui sont tous les deux clairement minimaux et couvrent deux hexagones. Nous avons donc $M(\mathscr{C}) = 2$.

Définition 2.4.9 (RANDIĆ, GUO et KLEIN 1996) *Soient $B = (V, E)$ un benzénoïde et $\mathscr{C}$ un de ses cycles (avec $4i+2$ arêtes, $i \in \mathbb{N}$). $B[\mathscr{C}]$ est le sous-graphe de B induit par $\mathscr{C}$ et son intérieur.*

La méthode proposée par Lin (LIN 2000) se base sur le théorème suivant :

Théorème 2.4.1 (RANDIĆ, GUO et KLEIN 1996) *Soient B un benzénoïde et $\mathscr{C}$ un de ses cycles (avec $4i+2$ arêtes, $i \in \mathbb{N}$). $\mathscr{C}$ est un h-CCM dans $|\mathscr{K}(B - B[\mathscr{C}])| \times M(\mathscr{C})$ structures de Kekulé de B où $B - B[\mathscr{C}]$ est le sous-graphe induit par la suppression des sommets appartenant à $\mathscr{C}$ et son intérieur.*

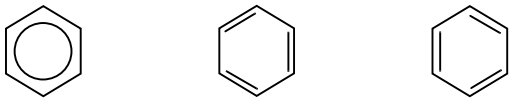
Considérons maintenant le benzénoïde B décrit dans la figure 2.23(b) et le cycle $\mathscr{C}$ décrit par la ligne rouge. $B[\mathscr{C}]$ correspond ainsi à tous les hexagones présents dans l'intérieur de $\mathscr{C}$ (c'est-à-dire les hexagones surlignés en rouge dans la figure du milieu). Pour calculer le nombre d'occurrences de $\mathscr{C}$ en tant que h-CCM, nous devons calculer le nombre de couplages parfaits du sous-graphe induit par $B - B[\mathscr{C}]$. Dans cet exemple, ce sous-graphe (décrit par des lignes rouges dans la figure de droite) possède deux couplages parfaits. De plus, nous avons $M(\mathscr{C}) = 1$ car si nous considérons les deux structures de Kekulé de $\mathscr{C}$ qui admettent un circuit conjugué, il n'y en a qu'une seule des deux pour laquelle $\mathscr{C}$ est minimal pour au moins un de ses hexagones. On peut donc conclure que $\mathscr{C}$ apparaît deux fois en tant que h-CCM dans toutes les structures de Kekulé de B.

Pour résumer, la méthode présentée par Lin (LIN 2000) décrite dans l'algorithme 3 prend en entrée un benzénoïde B et une base contenant l'intégralité des cycles de taille au plus 4 pouvant induire au moins un h-CCM ainsi qu'une autre base contenant tous les circuits redondants de taille identique. Premièrement, l'algorithme va générer l'ensemble des cycles de B correspondant à un des éléments de la première base (ligne 1). Cet ensemble sera noté $\mathscr{C}^*$. Ensuite, pour chaque cycle de $\mathscr{C}^*$, il va compter combien de h-MCC sont induits par ce cycle dans l'ensemble des structures de Kekulé de B (lignes 3-4), comme décrit dans la figure 2.23 (b). Ensuite, il va devoir trouver tous les couples de cycles de B pouvant produire un des circuits redondants présents dans la seconde base afin de s'assurer que le cycle ayant la plus grande taille ne soit pas comptabilisé (lignes 5-8).

Algorithme 3 : *Approximation_Énergie_Résonance*

Input : un benzénoïde B, une base de h-CCM, une base de circuits redondants
Output : un approximation de l'énergie de résonance globale $E(B)$

$\mathscr{C}^* \leftarrow générer\_circuits(B,1,4)$
$énergie \leftarrow 0$
foreach $\mathscr{C} \in \mathscr{C}^*$ **do**
  $énergie \leftarrow énergie + R_{|\mathscr{C}|} \times |K(B - B[\mathscr{C}])| \times M(\mathscr{C})$
foreach $(\mathscr{C}_1, \mathscr{C}_2) \in \mathscr{C}^* \times \mathscr{C}^*$ **do**
  if $circuits\_redondants(\mathscr{C}_1, \mathscr{C}_2)$ **then**
    $taille \leftarrow \max(|\mathscr{C}_1|, |\mathscr{C}_2|)$
    $énergie \leftarrow énergie - R_{taille} \times |K(B - B[\mathscr{C}_1 \cup \mathscr{C}_2])|$
return $\frac{énergie}{|\mathscr{K}(B)|}$



(a) (b) (c)

FIGURE 2.24. – Hexagone admettant un rond de Clar (a), et les deux configurations de liaisons doubles que cela implique (b, c)

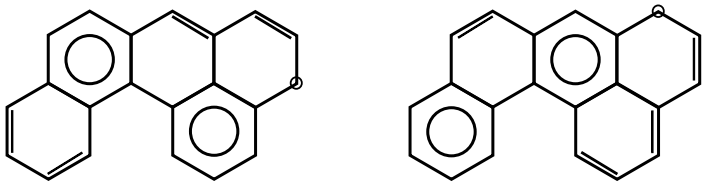
L'intérêt principal de cette méthode est qu'elle ne requiert pas d'énumérer l'ensemble des structures de Kekulé du benzénoïde traité, contrairement à la méthode de Lin et Fan décrite précédemment. Le seul problème étant de compter le nombre de couplages parfaits dans différents sous-graphes de la molécule. Dans le cas général, compter le nombre de couplages parfaits d'un graphe est un problème #P-Complet (donc très fortement combinatoire et difficile à résoudre) même dans le cas de graphes bipartis. Heureusement, il a été prouvé que cette tâche pouvait être accomplie en temps polynomial pour les graphes planaires (KASTELEYN 1967) et une méthode efficace uniquement dédiée aux benzénoïdes a été présentée en 2001 (RISPOLI 2001).

2.4.5. Couvertures de Clar

Il existe une autre méthode que les structures de Kekulé pour représenter la position des électrons d'un benzénoïde. Elle consiste à calculer une de ses *couvertures de Clar* (CHOU, CHOU, LI et al. 2012 ; WITEK et LANGNER 2020). Avant de définir formellement cette notion, il est nécessaire d'introduire plusieurs définitions :

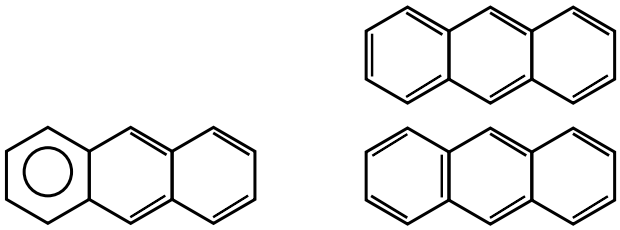
Définition 2.4.10 (Rond de Clar) *Un* rond de Clar *symbolise la présence de trois liaisons doubles au sein d'un même hexagone (les deux configurations possibles sont décrites dans la figure 2.24). Les carbones appartenant à un hexagone admettant un rond de Clar ne peuvent pas induire de liaison double avec un carbone n'appartenant pas à cet hexagone. Un rond de Clar est représenté graphiquement par un cercle à l'intérieur de l'hexagone concerné.*

Définition 2.4.11 (Électron célibataire) *Un* électron célibataire *est un électron dont l'atome ne participe ni à une liaison double, ni à un rond de Clar (on parle également d'électron* radicalaire*). Un électron célibataire est représenté graphiquement par un cercle autour de l'atome concerné.*



(a) (b)

FIGURE 2.25. – Deux des couvertures de Clar d'un benzénoïde à 5 hexagones



(a) (b)

FIGURE 2.26. – Une des couvertures de Clar de l'anthracène (a) et les deux structures de Kekulé qu'elle induit (b).

Définition 2.4.12 (Liaison fixe) *On dit qu'une liaison d'un benzénoïde est* fixe *si elle apparaît soit toujours simple, soit toujours double dans l'intégralité des structures de Kekulé de ce dernier.*

Nous pouvons maintenant définir formellement une couverture de Clar d'un benzénoïde de la manière suivante :

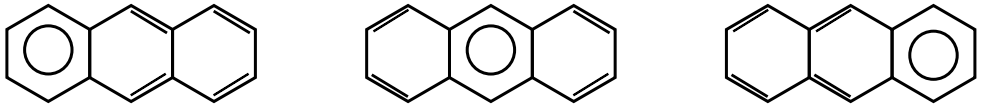
Définition 2.4.13 (Couverture de Clar) *Étant donné un nombre d'électrons radicalaires donnés, une* couverture de Clar *d'un benzénoïde désigne une manière valide de placer des ronds de Clar et des liaisons doubles de manière à maximiser le nombre de ronds de Clar. On appelle* nombre de Clar *d'un benzénoïde B (noté $\gamma(B)$) le nombre de ronds présents dans l'une de ses couvertures de Clar.*

Notons qu'à partir de maintenant, lorsque l'on parlera d'une couverture de Clar sans mentionner son nombre d'électrons célibataires, on considère que ce dernier est égal à 0. La figure 2.25 décrit l'une des couvertures de Clar pour un benzénoïde de 5 hexagones. Nous pouvons voir les couvertures de Clar d'un benzénoïde comme une factorisation de ses structures de Kekulé. En effet, une couverture de Clar d'un benzénoïde B n'admettant aucun électron radicalaire représente $2^{\gamma(B)}$ de ses structures de Kekulé (chaque rond admettant 2 configurations de doubles liaisons) (voir la figure 2.26). La valeur d'un hexagone est comprise entre 0 et 1. Ici, il est évident que plus cette valeur est élevée, plus ce dernier est considéré comme aromatique et inversement.

Une autre manière d'estimer l'aromaticité d'un benzénoïde est d'énumérer l'intégralité de ses couvertures de Clar, et d'attribuer à chaque hexagone une valeur dépendante du nombre de ronds qu'ils admettent. Pour illustrer cela, nous pouvons regarder la figure 2.27 qui liste toutes les couvertures de Clar de l'anthracène ainsi que la figure 2.28 qui décrit l'estimation de son aromaticité obtenue de cette manière.

2.4.6. Calculer les Ring Bond Order (RBO) d'un benzénoïde

La notion de *bond order* introduite par Pauling et al. (PAULING, BROCKWAY et BEACH 1935) est un concept bien connu au sein de la communauté des chimistes. Elle consiste à déterminer à quelle fréquence une



(a) (b) (c)

FIGURE 2.27. – Les trois couvertures de Clar de l'anthracène

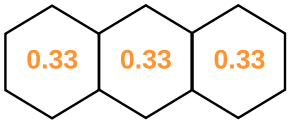


FIGURE 2.28. – Estimation de l'aromaticité de l'anthracène avec la méthode des couvertures de Clar

liaison d'un benzénoïde sera double dans l'ensemble de ses structures de Kekulé. Cette notion apparaît donc comme une autre façon de décrire la dynamique des électrons présents dans une molécule et fournir une estimation de son aromaticité. En 2018, Randić introduit la notion de *ring bond order*, qui est une généralisation des bond orders aux hexagones des benzénoïdes (RANDIĆ et BALABAN 2018). Ces deux notions peuvent se définir de la manière suivante :

Définition 2.4.14 (Bond Order) *Soient $B = (V, E)$ un benzénoïde et $e \in E$ une de ses liaisons. On appelle* bond order *de e le ratio du nombre de structures de Kekulé de B dans lesquelles la liaison e est double sur le nombre total de structures de Kekulé de B.*

Définition 2.4.15 (Ring Bond Order (RBO)) *Soient B un benzénoïde et h un de ses hexagones. On appelle* ring bond order (RBO) *de h la somme des bond order associés aux six liaisons définissant h.*

Afin d'illustrer cette notion, penchons-nous tout d'abord sur la figure 2.29 (a). Cette dernière décrit les bond orders de l'anthracène. On peut facilement voir que l'arête décrite en rouge n'est double qu'une seule fois parmi les quatre structures de Kekulé de l'anthracène (voir la figure 2.4), cette dernière a donc un bond order de $\frac{1}{4}$. Pour obtenir tous les bond orders, il suffit de répéter cette opération pour chacune des arêtes de la molécule considérée. Une fois cette étape terminée, il suffit pour chaque hexagone, d'additionner les bonds order de ses six arêtes afin d'obtenir ses ring bond orders (voir la figure 2.29 (b) pour l'anthracène). Il est important de mentionner que les RBO d'une molécule sont uniques, alors que cette dernière peut admettre plusieurs structures de Clar. La valeur du RBO de l'hexagone d'un benzénoïde est comprise entre 0 et 3. Cette dernière vaudra 0 si aucune des liaisons de cet hexagone n'est double dans toutes les structures de Kekulé de la molécule. Au contraire, cette valeur vaudra 3 si l'hexagone admet trois liaisons doubles dans chacune des structures de Kekulé. Plus une valeur est proche de 3 et plus l'hexagone concerné est considéré comme aromatique et inversement.

(a) (b)

FIGURE 2.29. – Les bond orders (a) et les ring bond orders (b) de l'anthracène.

Troisième partie

Travaux effectués

3. Génération de structures de benzénoïdes

3.1. Introduction

Comme mentionné dans la section 2.3, être capable de générer des structures de benzénoïdes satisfaisant un ensemble de propriétés données constitue une problématique importante en chimie. En effet, cela permettrait d'effectuer des études sur des familles de benzénoïdes bien spécifiques, on peut citer à nouveau Ricca (RICCA, ROSER, PEETERS et al. 2019) qui a exhibé le fait que les HAP admettant des *armchair edge* (voir figure 3.1) pouvaient émettre une certaine classe de source infrarouge. De plus, il existe un certain nombre de bases de données accessibles en ligne répertoriant des PAH (on peut notamment citer la *NASA PAH Ames Database* disponible ici : https://www.astrochemistry.org/pahdb/). Bien que très fournies, certaines de ces bases restent incomplètes et, à ce titre, être capable de générer des ensembles exhaustifs de structures de benzénoïdes répondant à diverses propriétés pourrait permettre de les compléter. Par exemple, Bouwmann et al (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) ont présenté en 2019 une étude des spectres infrarouges de certains HAP récupérés via la NASA PAH Ames Database. Durant nos travaux, nous nous sommes rendus compte qu'ils avaient considéré des ensembles de molécules incomplets. Être capable de générer des ensembles exhaustifs de benzénoïdes permettrait de palier ce problème.

Nous avons également vu que les méthodes de génération existantes (BRINKMANN, CAPOROSSI et HANSEN 2002 ; BRUNVOLL, CYVIN et CYVIN 1990) ont beau être efficaces en termes de performances, elles présentent néanmoins certains inconvénients. Premièrement, elles peuvent ne pas garantir l'exhaustivité des ensembles de structures générés. C'est le cas notamment de la méthode de Brinkmann qui est incapable de gérer les coronoïdes (les benzénoïdes admettant des trous). Deuxièmement, elles ne permettent pas de générer des molécules satisfaisant des propriétés spécifiques (pour reprendre le même exemple, la méthode de Brinkmann se limite au nombre d'hexagones). Faire en sorte que ces méthodes admettent de nouvelles propriétés demanderait un lourd effort en termes de conception et de programmation. Pour toutes ces raisons, les méthodes présentées jusqu'à présent manquent cruellement de flexibilité et sont difficilement adaptables aux besoins des chimistes qui sont en constante évolution.

Dans ce contexte, nous cherchons à mettre en place un outil flexible permettant de générer des structures de benzénoïdes de manière exhaustive. Ce dernier devra pouvoir admettre un grand nombre de propriétés (qu'elles soient chimiques ou structurelles, locales ou globales). Ces dernières devront être facilement combinables et l'outil devra faire preuve d'une grande flexibilité, en particulier du point de vue de l'ajout de nouvelles propriétés. Plus formellement, nous nous proposons de résoudre le *Problème de Génération de Benzénoïdes (BGP)* défini de la manière suivante :

Définition 3.1.1 (Problème de Génération de Benzénoïdes (PGB)) *Étant donné un ensemble de propriétés structurelles $\mathscr{P}$, générer l'ensemble des benzénoïdes satisfaisant toutes les propriétés de $\mathscr{P}$.*

Pour résoudre ce problème, notre choix s'est naturellement porté sur une approche exploitant la programmation par contraintes. La raison principale de ce choix est qu'en exploitant la grande flexibilité de la PPC, nous serons capables de modéliser rapidement et efficacement un grand nombre de propriétés et de les combiner. De plus, la programmation par contraintes nous garantit l'exhaustivité des ensembles de

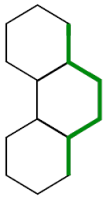


FIGURE 3.1. – Le motif *armchair edge*

molécules générés. Nous considérons en premier lieu un modèle CSP général capable de générer de manière exhaustive les structures de benzénoïdes ayant un nombre donné d'hexagones. Ensuite, nous considérons un ensemble de variables et de contraintes (appelé *module*) pour chaque propriété additionnelle que nous souhaitons modéliser. Chacun de ces modules pourra ensuite être greffé ou non au modèle général en fonction des besoins de l'utilisateur. Ainsi, si nous voulons modéliser une nouvelle propriété, nous avons juste à implémenter le module correspondant et à l'ajouter au modèle général. Ce procédé permet d'ajouter/-combiner facilement des propriétés et fournit donc la flexibilité qui faisait défaut aux méthodes présentées dans l'état de l'art. De plus, l'utilisation de la programmation par contraintes avec ses méthodes complètes garantit l'exhaustivité des ensembles de structures générés.

Cette partie est organisée de la manière suivante : dans un premier temps, nous présenterons toutes les définitions préliminaires dans la partie 3.2. Ensuite, nous décrirons notre modèle général dans la section 3.3 avant de détailler les différentes manières d'étendre ce modèle pour représenter de nouvelles propriétés dans la section 3.4. Pour finir, nous conclurons et donnerons quelques perspectives dans la section 3.10.

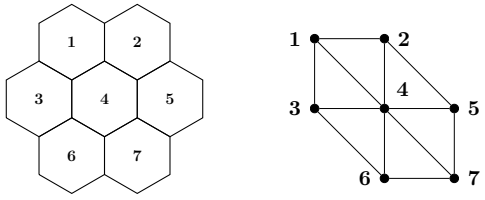
3.2. Définitions préliminaires

Dans cette section, nous introduirons certaines notions liées aux benzénoïdes qui seront utilisées dans les sections suivantes. Tout d'abord, un benzénoïde peut être vu comme l'agrégation de cycles benzéniques (c'est-à-dire d'hexagones), il est donc utile de pouvoir raisonner sur ces hexagones. Pour cela, nous définissons le *graphe d'hexagones* d'un benzénoïde comme suit :

Définition 3.2.1 (Graphe d'hexagones) *Le* graphe d'hexagones *d'un benzénoïde $B = (V, E)$ est le graphe non orienté $B_h = (V_h, E_h)$ tel que V_h possède un sommet v_h pour chaque face hexagonale h de B (la face externe ainsi que les « trous » sont donc exclus). Il y a une arête $\{v_h, v_{h'}\}$ dans E_h si et seulement si les faces hexagonales correspondantes h et h' sont adjacentes.*

Cette notion de graphe d'hexagones nous permet d'exprimer les interactions (sous la forme de partage de liaisons) entre les hexagones d'un benzénoïde. La figure 3.2 nous montre le coronène (a) ainsi que son graphe d'hexagones (b). Le graphe d'hexagones d'un benzénoïde peut être vu comme l'équivalent du graphe dual intérieur utilisé par Brinkmann et al. défini dans la section 2.3 (Brinkmann, Caporossi et Hansen 2002). On se propose ensuite d'introduire la notion de *diamètre* d'un benzénoïde :

Définition 3.2.2 (Diamètre) *Le diamètre d'un benzénoïde correspond au nombre d'arêtes du plus long chemin minimal entre deux des sommets de son graphe d'hexagones*[1].



(a) (b)

FIGURE 3.2. – Le coronène (a) et son graphe d'hexagones (b).

La figure 3.4 décrit un benzénoïde de diamètre 4. Avant de décrire notre modélisation du problème PGB, on se propose de rappeler ou d'introduire certaines notions. Un coronénoïde de taille k est la molécule de

1. Un chemin minimal (ou plus court chemin) entre deux sommets u et v est la plus petite suite d'arêtes consécutives reliant u à v.

benzène (c'est-à-dire un hexagone, voir la figure 2.2) auquel l'on ajoute $k-1$ couronnes d'hexagones. Le benzène correspond donc au coronénoïde de taille 1 et la figure 3.3 décrit les coronénoïdes de taille 2, 3, et 4. Notons qu'un coronénoïde de taille donnée admettra toujours un diamètre spécifique :

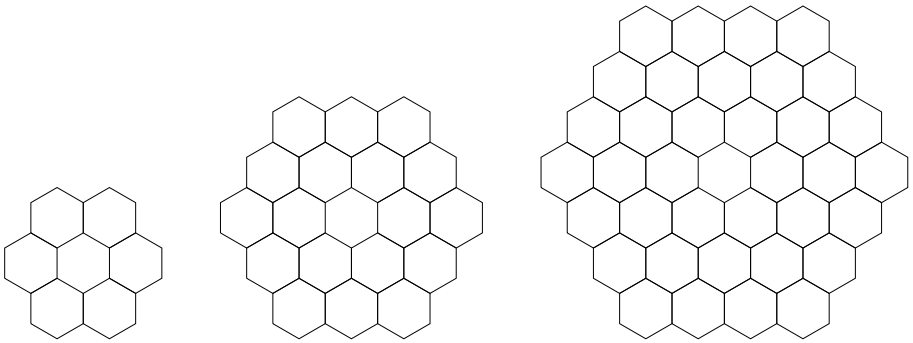


FIGURE 3.3. – Les coronénoïdes de taille 2 (a), 3 (b) et 4 (c)

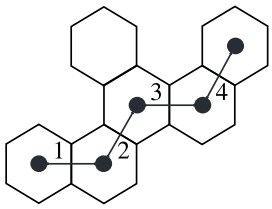


FIGURE 3.4. – Exemple de benzénoïde de diamètre 4

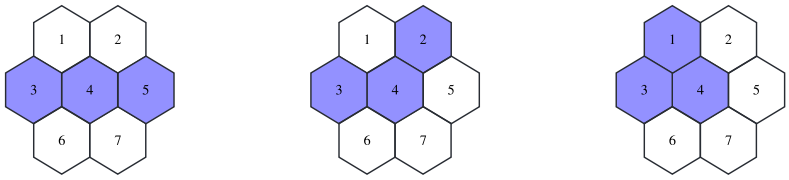
Propriété 3.2.1 *Le diamètre d'un coronénoïde de taille k est* $2 \times k - 1$.

Pour finir, la principale raison qui nous a poussé à nous intéresser aux coronénoïdes réside dans leur capacité à « contenir » l'ensemble des benzénoïdes ayant un nombre donné d'hexagones :

Propriété 3.2.2 *N'importe quel benzénoïde possédant n hexagones peut être contenu dans un coronénoïde de taille* $k(n) = \lfloor \frac{n}{2} + 1 \rfloor$.

De plus, un autre avantage est que peu importe le benzénoïde de n hexagones considéré, ce dernier restera toujours dans le coronénoïde de taille $k(n)$ après avoir subi une rotation.

La figure 3.5 illustre le fait que les trois benzénoïdes possédant trois hexagones sont bien contenus dans le coronénoïde de taille $k(3) = 2$. Ainsi, si nous raisonnons en termes de graphe d'hexagones, obtenir l'ensemble des benzénoïdes de n hexagones équivaut à trouver tous les sous-graphes connexes du graphe d'hexagones du coronénoïde de taille $k(n)$ (que nous noterons $B_h^{c(k(n))}$). Le modèle général que nous allons présenter repose sur cette propriété.



(a) (b) (c)

FIGURE 3.5. – L'intégralité des benzénoïdes ayant 3 hexagones sont contenus dans le coronénoïde de taille $k(3) = 2$

3.3. Un modèle CSP général

Étant donné un entier n, nous souhaitons définir un modèle général $\mathscr{M}$ capable de résoudre le problème PGB avec $\mathscr{P}$ ne contenant uniquement que la propriété « *posséder au plus n hexagones* » sous la forme d'une instance CSP $I = (X, D, C)$. Notons qu'il est impératif de borner supérieurement la taille des benzénoïdes générés afin de ne générer uniquement que des ensembles finis de molécules. Pour cela, nous pouvons borner supérieurement un certain nombre de critères tels que le nombre d'hexagones, de carbones, d'hydrogènes ou encore le diamètre. Nous avons néanmoins opté pour le nombre d'hexagones, car ce critère est le plus naturel pour les chimistes. En effet, ces derniers ont tendance à caractériser un benzénoïde par son nombre d'hexagones. Premièrement, on considère une variable de graphe x_G qui va représenter le graphe d'hexagones du benzénoïde généré. Son domaine est défini par l'ensemble des sous-graphes du graphe d'hexagones du coronénoïde de taille $k(n)$ (voir la figure 3.6). Ainsi, la borne inférieure de x_G correspondra à un graphe vide et sa borne supérieure à $B_h^{c(k(n))}$. Ensuite, nous introduisons un ensemble de n_c variables booléennes $X_V = \{x_1, \ldots, x_{n_c}\}$ (n_c étant le nombre de sommets de $B_h^{c(k(n))}$). La variable x_i a pour valeur 1 si le i-ème hexagone du graphe d'hexagones du coronénoïde de taille $k(n)$ est présent dans x_G, 0 sinon. Pour des raisons de simplicité, nous choisissons de numéroter les hexagones de haut en bas, puis de gauche à droite. La figure 3.7 détaille les variables associées à chaque hexagone d'un coronénoïde de taille 3. De la même manière, on considère un second ensemble X_E de m_c variables booléennes $y_{i,j}$ (m_c étant le nombre d'arêtes de $B_h^{c(k(n))}$). La variable $y_{i,j}$ vaudra 1 si l'arête $\{i, j\}$ du graphe d'hexagones de $B_h^{c(k(n))}$ est présente dans x_G. Pour finir, nous considérons une variable entière τ ayant pour domaine l'ensemble des entiers compris entre 1 et n (inclus). Cette dernière représentera le nombre d'hexagones du benzénoïde généré.

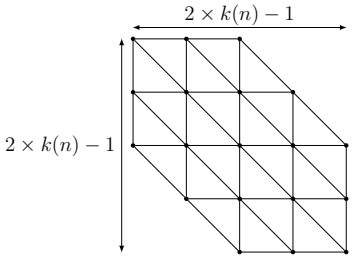


FIGURE 3.6. – Borne supérieure de la variable de graphe x_G.

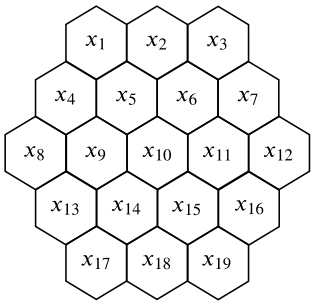


FIGURE 3.7. – Variables de X_V associées aux hexagones d'un coronénoïde de taille 3

Pour finir, nous modélisons les propriétés suivantes sous la forme de contraintes :

- *Lien entre les sommets de la variable de graphe x_G et les variables de X_V* : Comme mentionné ci-dessus, la variable x_i vaut 1 si le i-ème hexagone de $B_h^{c(k(n))}$ est présent dans x_G. Nous appliquons donc une contrainte `nodeChanneling` pour chaque variable $x_i \in X_V$ qui implique que x_i vaut 1 si et seulement si x_G contient le i-ème sommet du graphe d'hexagones du coronénoïde de taille $k(n)$.

- *Lien entre les arêtes de la variable de graphe x_G et les variables de X_E* : De manière similaire au point précédent, nous appliquons une contrainte `edgeChanneling` pour chaque variable $y_{i,j} \in X_E$ qui implique que $y_{i,j}$ vaut 1 si et seulement si x_G contient l'arête $\{i, j\}$ du graphe d'hexagones de $B_h^{c(k(n))}$.

- *x_G est un sous-graphe induit du graphe d'hexagones du coronénoïde englobant* : Toute valeur de x_G ne constitue pas nécessairement un graphe d'hexagones valide. Par exemple, considérons le graphe d'hexagones du coronène décrit dans la figure 3.2 (b). Ici, enlever uniquement l'arête $\{1,2\}$ ne produira pas un graphe d'hexagones valide, car les sommets 1 et 2 sont présents et adjacents, ces derniers doivent donc être impérativement reliés par une arête. Pour s'assurer que x_G décrive bien un graphe d'hexagone valide, il faut spécifier que pour tout couple de sommets du graphe d'hexagones du coronénoïde englobant représentant deux hexagones adjacents, si ces deux sommets sont présents dans x_G, alors ils doivent impérativement être reliés par une arête et inversement. Ainsi, pour chaque arête $\{i, j\}$ du graphe d'hexagone du coronénoïde, nous ajoutons la contrainte $x_i = 1 \wedge x_j = 1 \Leftrightarrow y_{i,j} = 1$.

- *Le benzénoïde généré possède au plus n hexagones* : Cette contrainte peut être facilement modélisée en appliquant une contrainte `nbNodes` sur x_G et τ : `nbNodes`(x_G, τ). Le nombre d'hexagones du benzénoïde généré sera ainsi borné par le domaine de τ qui est lui-même compris entre 1 et n.

- *Le graphe d'hexagones du benzénoïde généré est connexe* : Pour cette propriété, nous appliquons la contrainte `connected` à la variable x_G. Cette dernière spécifie que x_G doit être connexe.

- *Il n'y a pas de trous constitués d'un unique hexagone* : Quand six hexagones forment un cycle, la face se situant à l'intérieur de ce cycle n'est pas un trou, mais un autre hexagone. Par exemple, le cycle formé par les hexagones 1, 2, 5, 7, 6 et 3 du coronène (cf. figure 3.2) implique la présence de l'hexagone 4. Pour garantir cette propriété, nous ajoutons un ensemble de contraintes spécifiant que x_G ne peut pas avoir de trou formé d'un seul hexagone. Pour chaque hexagone u, on considère l'ensemble $N(u)$ des voisins de u (c'est-à-dire, l'ensemble des hexagones adjacents à u) dans le graphe d'hexagones. Ensuite, pour chaque sommet u ayant six voisins, nous considérons une contrainte entre x_u et les variables correspondant à ses voisins qui spécifie : $\sum_{v \in N(u)} x_v = 6 \Rightarrow x_u = 1$.

Pour résumer, chaque sommet présent dans x_G correspond à un hexagone du benzénoïde généré. Les contraintes de ce modèle permettent de ne générer que des benzénoïdes possédant une structure valide (à

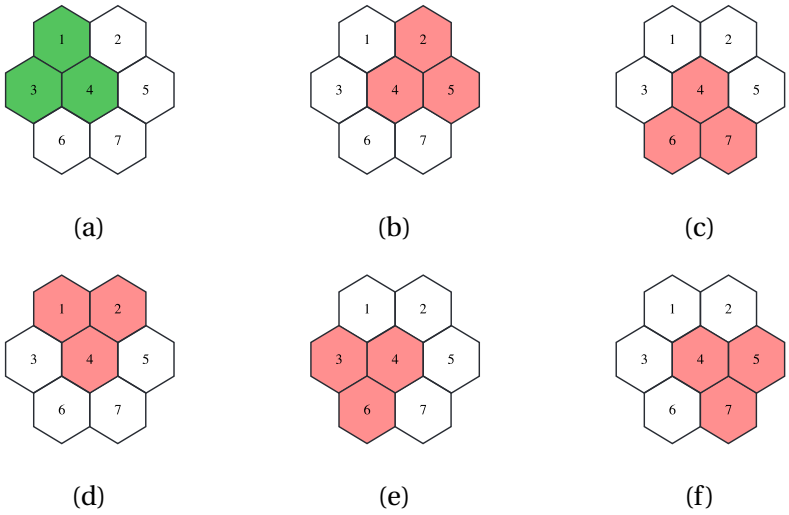


FIGURE 3.8. – Intégralité des images du benzénoïde induit par la solution $\{x_1 = 1, x_3 = 1, x_4 = 1\}$ dans un coronénoïde de taille 2

savoir n'être qu'en une seule partie et ne pas admettre de trous constitués d'un unique hexagone) ainsi que le nombre d'hexagones voulu par l'utilisateur. La dernière problématique à traiter est de prendre garde à ne pas générer de benzénoïdes redondants. En effet, rien n'empêche actuellement la génération de deux benzénoïdes identiques par translation dans le coronénoïde englobant, par rotation ou par symétrie axiale.

Pour éviter cette redondance, lorsqu'une solution est générée, on se propose d'apprendre des clauses ayant pour but d'interdire les images de la solution par rotation, translation ou symétrie axiale. Nous faisons cela à l'aide de la contrainte `baseNogood` (définie dans la sous-section 1.5.1). Cette contrainte permet d'ajouter des clauses contenant des variables booléennes au modèle. Ainsi, à chaque fois qu'un benzénoïde de n_h hexagones $\{h_1, h_2, \ldots, h_{n_h}\}$ (qui correspond à l'affectation partielle $S = \{x_{h_1} = 1, x_{h_2} = 1, \ldots, x_{h_{n_h}} = 1\}$) est renvoyé par le solveur, nous calculons ensuite la totalité des images par translation, symétrie axiale, et rotation. Leur nombre étant en $O(|c(k(n))|) = O(n^2)$, ce calcul peut être fait en temps polynomial. La figure 3.8 nous montre toutes les images du benzénoïde induit par la solution $\{x_1 = 1, x_3 = 1, x_4 = 1\}$ dans un coronénoïde de taille 2. Ensuite, pour chaque image $\{h'_1, h'_2, \ldots, h'_{n_h}\}$ contenant l'hexagone central du coronénoïde englobant calculée (qui correspond donc à l'affectation partielle $S = \{x_{h'_1} = 1, x_{h'_2} = 1, \ldots, x_{h_{n'_h}} = 1\}$), nous ajoutons la clause suivante au modèle :

$$\neg x_{h'_1} \vee \neg x_{h'_2} \vee \ldots \neg x_{h'_{n_h}} \vee \neg(\tau = n_h)$$

Notons que durant cette thèse, nous nous autorisons, pour des soucis de clarté, à intégrer des contraintes directement dans des clauses. En pratique, il est possible d'ajouter des contraintes dans une clause grâce à un procédé appelé *réification*. La réification consiste à exprimer la satisfaisabilité d'une contrainte sous la forme d'une variable booléenne : si la contrainte est satisfaite, la variable vaudra vrai, et inversement. Ici, nous ajoutons $\neg(\tau = n_h)$ à la clause apprise dans le but d'éviter que la génération d'un benzénoïde ayant plus de n_h hexagones, mais possédant tous les hexagones de l'une des images de taille inférieure ne soit bloquée. Il y a des petites subtilités sur les stratégies d'apprentissage en fonction des modules considérés, ces dernières seront détaillées dans la section 3.7.

3.4. Extensions du modèle général

Le principal avantage de notre méthode est qu'elle nous permet de pouvoir générer n'importe quel benzénoïde, y compris ceux possédant des trous (contrairement à la méthode de Brinkmann (BRINKMANN, CAPOROSSI et HANSEN 2002) présentée dans la section 2.3. De plus, l'utilisation de la programmation par contraintes rend plus facile l'ajout de propriétés additionnelles demandées par les chimistes. Pour chaque nouvelle propriété souhaitée, il nous suffit de partir du modèle général et d'ajouter les variables ou contraintes nécessaires pour l'exprimer.

3.4.1. Génération de benzénoïdes ayant une forme arborescente ou étant catacondensés

Dans cette sous-section, nous nous pencherons sur deux catégories de benzénoïde très proches : les benzénoïdes *arborescents* et les *catacondensés*. Un benzénoïde est dit arborescent si son graphe d'hexagones est un arbre [2] et catacondensé s'il n'admet aucun triplet d'hexagones deux à deux adjacents. La figure 3.9 décrit l'intégralité des benzénoïdes catacondensés de cinq hexagones (ces derniers sont aussi arborescents). À première vue, ces deux catégories pourraient sembler identiques, mais la famille des catacondensés admet quelques molécules qui ne sont pas arborescentes, on peut par exemple citer le benzénoïde décrit dans la figure 3.10 (qui est également le plus petit benzénoïde admettant un trou).

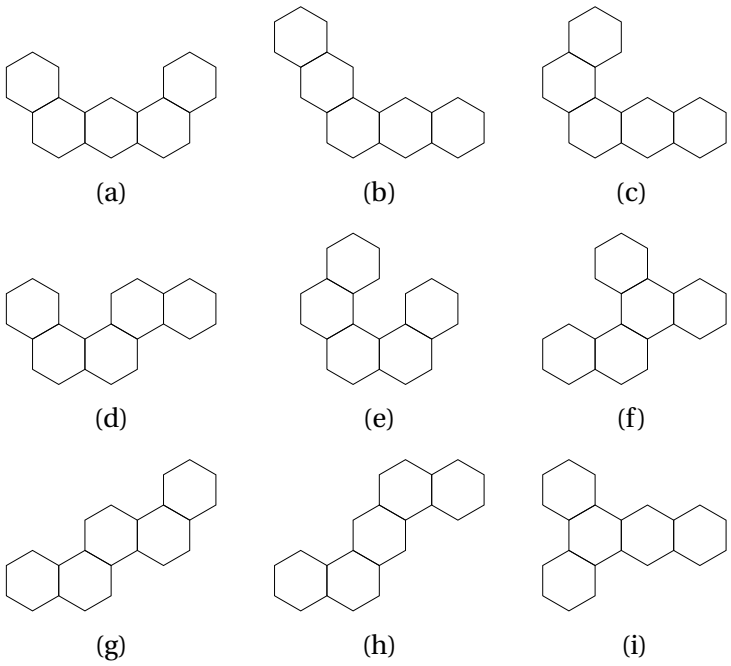


FIGURE 3.9. – L'ensemble des benzénoïdes catacondensés possédant 5 hexagones

On se propose maintenant de décrire deux modèles capables respectivement de générer des benzénoïdes arborescents et catacondensés. Pour chacun de ces deux modèles, nous partirons du modèle général $\mathscr{M}$ décrit dans la section 3.3. Pour générer des benzénoïdes arborescents, une simple contrainte $\texttt{tree}(x_G)$ spécifiant que le graphe d'hexagones du benzénoïde généré doit être un arbre est suffisante. Pour ce qui est des catacondensés, il suffit d'énumérer tous les triplés de sommets (u, v, w) du graphe d'hexagones du coronénoïde englobant et pour chacun d'entre eux, ajouter la clause suivante afin de garantir qu'au moins un des trois sommets ne sera pas présent dans x_G :

$$\neg x_u \vee \neg x_v \vee \neg v_w$$

2. Un arbre est un graphe non orienté et connexe qui n'admet aucun cycle.

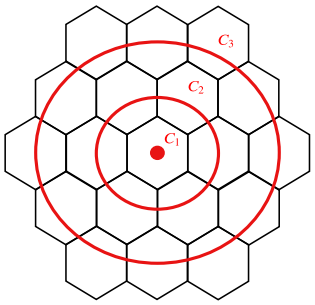


FIGURE 3.11. – Numérotation des couronnes pour un coronénoïde de taille 3

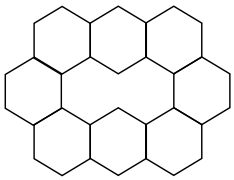


FIGURE 3.10. – Un benzénoïde catacondensé non arborescent

3.4.2. Génération de coronénoïdes

Comme mentionné dans la section 3.2, un coronénoïde de taille k est la molécule de benzène à laquelle on ajoute $k-1$ couronnes d'hexagones. Il est utilisé dans le modèle général $\mathscr{M}$ décrit dans la section 3.3 afin d'englober les différents benzénoïdes qui seront générés. Dans cette section, on se propose de décrire un module capable de générer de telles molécules.

Ici, nous numérotons les couronnes du coronénoïde englobant en partant du centre jusqu'à la périphérie comme illustré dans la figure 3.11. On note donc C_c la c-ème couronne en partant du centre.

Nous partirons encore une fois du modèle général $\mathscr{M}$. On y ajoute une variable entière ϱ qui aura pour domaine l'ensemble des entiers entre 1 et $k(n)$ (inclus). Cette variable va correspondre au nombre de couronnes du coronénoïde généré. Ensuite, pour chaque couronne C_c du coronénoïde englobant, et pour chaque hexagone h présent dans cette dernière, nous ajoutons les clauses suivantes au modèle :

$$x_h \iff (\varrho \geq c) \equiv \begin{cases} \neg x_h \vee (\varrho \geq c) \\ \neg(\varrho \geq c) \vee x_h \end{cases}$$

En d'autres termes, ces clauses permettent de spécifier qu'un hexagone h de la couronne C_c est présent dans le benzénoïde généré si et seulement si le coronénoïde généré est au moins de taille c. L'équivalence permet également de garantir que si un sommet d'une couronne est présent, alors tous les autres hexagones de cette dernière le seront aussi. Comme pour le modèle général $\mathscr{M}$ (section 3.3), nous nous autorisons à introduire des contraintes dans les clauses par le biais de la réification. Pour rappel, cela consiste à exprimer la satisfaisabilité d'une contrainte sous la forme d'une variable booléenne. Pour finir, il ne reste plus qu'à prendre en compte les contraintes sur la taille des coronénoïdes générés spécifiées par l'utilisateur. Ces dernières seront simplement des contraintes en intention portant sur la variable ϱ.

3.4.3. Génération de benzénoïdes ayant une forme rectangulaire

Les benzénoïdes *rectangulaires* désignent tous les benzénoïdes dont la forme représente un rectangle plein. Du fait de leurs fortes densités, ils admettent des propriétés intéressantes telles qu'un nombre élevé

de structures de Kekulé. La figure 3.12 décrit l'ensemble des benzénoïdes rectangulaires possédant au plus 6 hexagones. Dans cette section, nous présentons un module permettant de générer les structures de benzénoïdes ayant une forme rectangulaire. Le modèle obtenu en ajoutant ce module au modèle général $\mathscr{M}$ sera dénoté $\mathscr{M}_r$. Notons que nous avons présenté un autre modèle capable de générer des benzénoïdes rectangulaires (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2022). Néanmoins, ce dernier était clairement moins performant que $\mathscr{M}_r$. Par exemple, l'ancien modèle mettait presque cinq heures afin de générer l'intégralité des benzénoïdes de hauteur 5 et de largeur variant entre 1 et 5, tandis que $\mathscr{M}_r$ est capable de le faire en moins d'une seconde.

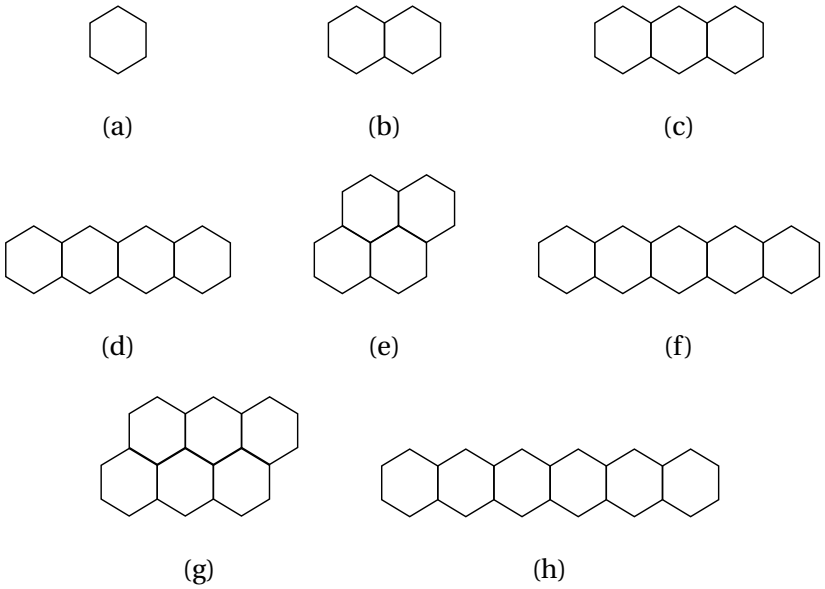
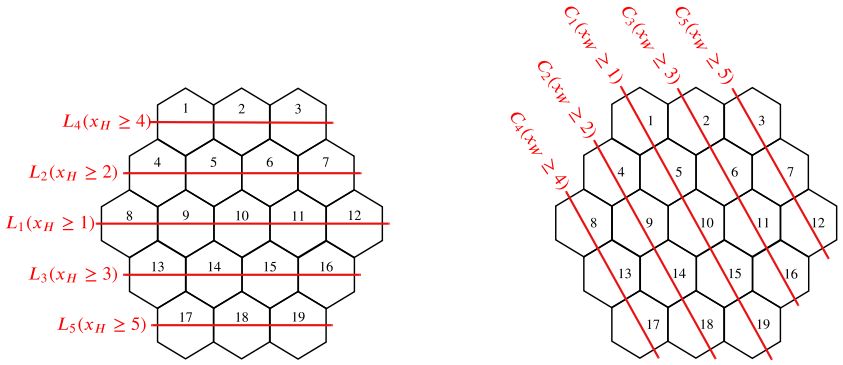


FIGURE 3.12. – Ensemble des benzénoïdes rectangulaires ayant au plus 6 hexagones

Pour ce module, on se propose de numéroter les lignes et les colonnes du coronénoïdes englobant en partant du centre jusqu'aux extrémités. La figure 3.13 décrit ce procédé dans le cas d'un coronénoïde englobant de taille 3. La première ligne correspond donc à la ligne centrale, la seconde à celle se trouvant au-dessus de la ligne centrale, la troisième à celle en dessous, etc. Ici, l'idée générale est qu'un rectangle de i lignes et j colonnes sera généré sur l'ensemble des hexagones appartenant à l'intersection des hexagones des i premières lignes et des j premières colonnes. Pour illustrer cela, la figure 3.14 nous montre comment sont placés les benzénoïdes rectangulaires de dimension au plus 3×3 dans un coronénoïde de taille 3.



(a) (b)

FIGURE 3.13. – Représentations des lignes et des colonnes pour le coronénoïde de taille 3

Encore une fois, nous partons du modèle général décrit dans la section 3.3. Nous ajoutons à ce modèle deux variables entières x_W et x_H ayant toutes deux pour domaine l'ensemble des entiers compris entre 1 et le diamètre du coronénoïde englobant. Ces deux variables représenteront respectivement la largeur (c'est-à-dire le nombre de colonnes) et la hauteur (c'est-à-dire le nombre de lignes) du benzénoïde généré.

Ensuite, pour chaque hexagone i du coronénoïde englobant, nous définissons deux entiers l_i (resp. c_i) correspondant au numéro de la ligne (resp. de la colonne) qui contient l'hexagone i. Ainsi, si l'on regarde à nouveau la figure 3.13, dans le cas ou $i = 5$, nous aurions $l_i = 2$ et $c_i = 1$. Ensuite, pour chaque hexagone i du coronénoïde englobant, nous ajoutons les clauses suivantes au modèle général :

- *Si l'hexagone i est présent dans la solution, alors $x_H \geq l_i$:*

$$x_i \implies (x_H \geq l_i) \equiv \neg x_i \vee (x_H \geq l_i)$$

- *Si l'hexagone i est présent dans la solution, alors $x_W \geq c_i$:*

$$x_i \implies (x_W \geq c_i) \equiv \neg x_i \vee (x_W \geq c_i)$$

- *Si $x_H \geq l_i$ et $x_W \geq c_i$ alors l'hexagone i est présent dans la solution :*

$$((x_H \geq l_i) \wedge (x_W \geq c_i)) \implies x_i \equiv \neg(x_H \geq l_i) \vee \neg(x_W \geq c_i) \vee x_i$$

3.4.4. Génération de benzénoïdes ayant une forme de losange

Les benzénoïdes ayant une forme en losange constituent une sous-famille des benzénoïdes rectangulaires qui regroupe l'ensemble de ces molécules ayant le même nombre de lignes et de colonnes. La figure 3.15 nous montre les trois plus petits losanges. Pour pouvoir générer ces molécules, il suffit d'ajouter au modèle général les variables/contraintes de $\mathscr{M}_r$ (voir la sous-section 3.4.3) ainsi qu'une contrainte spécifiant que x_W doit être égal à x_H.

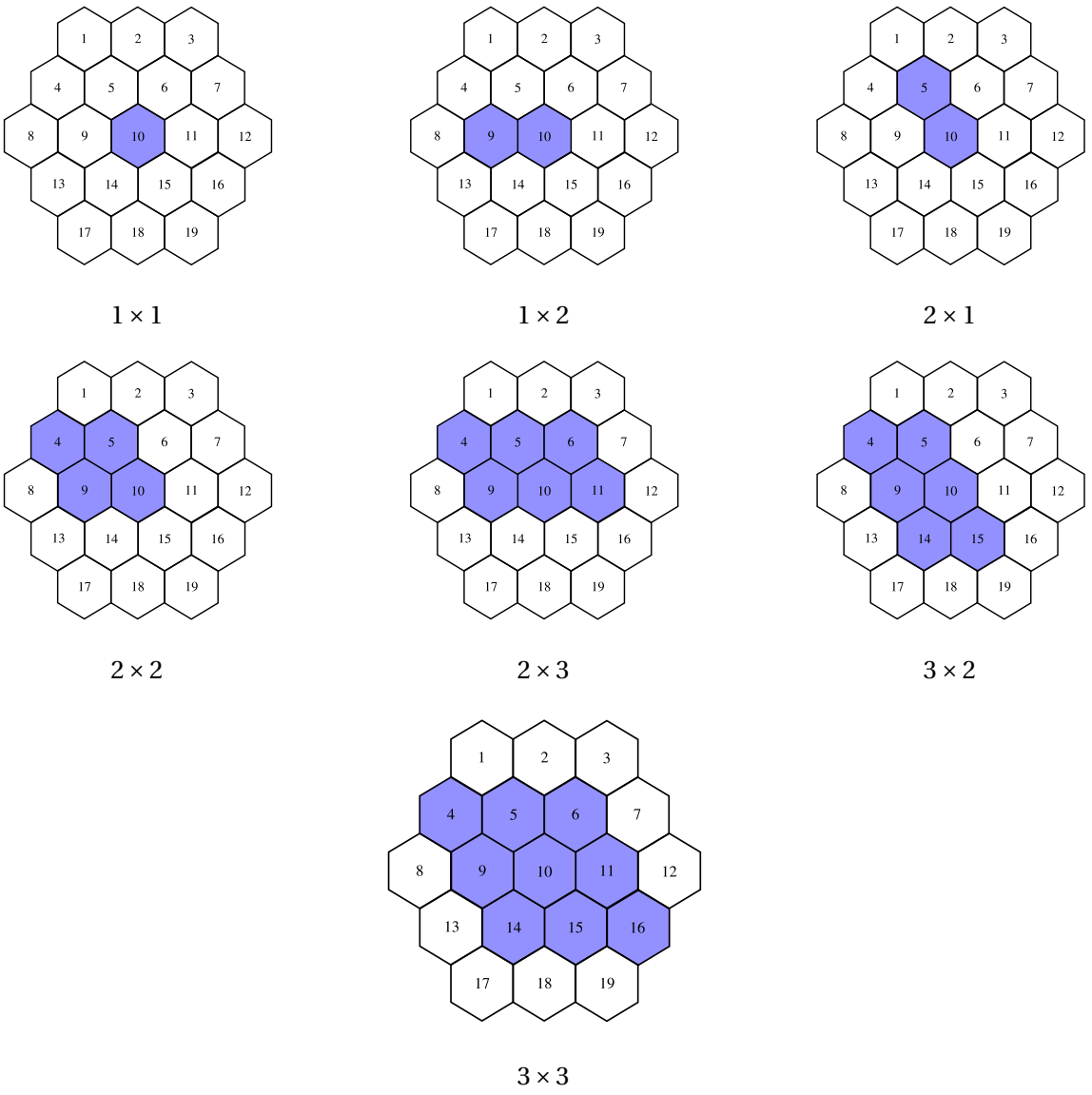


FIGURE 3.14. – Exemple de benzénoïdes rectangulaires pouvant être inclus dans un coronénoïde de taille 3

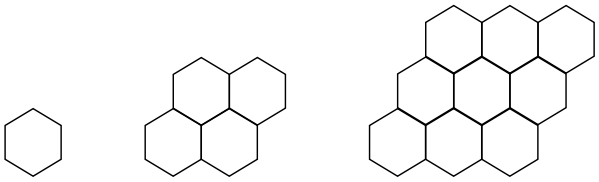


FIGURE 3.15. – Les trois plus petits losanges

3.4.5. Génération de benzénoïdes possédant des trous

Les chimistes définissent les benzénoïdes admettant au moins un trou comme étant des *coronoïdes* (à ne pas confondre avec coronénoïdes). Les propriétés de certaines de ces molécules (notamment les structures de graphènes avec des trous correctement placés) intéressent fortement les chimistes (DI GIOVANNANTONIO, YAO, EIMRE et al. 2020; BESER, KASTLER, MAGHSOUMI et al. 2016; DIAS 2008). La figure 3.16 décrit les cinq coronoïdes admettant 9 hexagones. Il y a déjà eu des travaux liés à la génération et à l'énumération de ces molécules. Par exemple, les travaux présentés dans (BRUNVOLL, CYVIN et CYVIN 1990) énumèrent les coronoïdes possédant deux trous et génèrent les plus petits coronoïdes de 18 et 19 hexagones admettant trois trous). Malheureusement, ces méthodes de génération sont soit pas assez efficaces, soit bien trop spécifiques pour être utilisées de manière plus générale. Le premier type d'approche consiste à essayer de construire des coronoïdes spécifiques en partant de cycles d'hexagones et en essayant d'ajouter des hexagones autour tout en prenant soin de maintenir au moins un trou. Le second type d'approche consiste à générer tous les benzénoïdes ayant un nombre donné d'hexagones, puis de détecter ceux qui contiennent des trous. Une autre solution consisterait à générer des ensembles de benzénoïdes ne contenant pas de coronoïdes (voir la méthode de Brinkmann et al. (BRINKMANN, CAPOROSSI et HANSEN 2002)), puis d'essayer de creuser des trous dans ces molécules. Cependant, on peut facilement remarquer que ces approches peuvent être coûteuses en temps assez rapidement comparé à une approche qui générerait simplement des coronoïdes. En effet, quand on augmente le nombre d'hexagones requis de 1, le nombre de benzénoïdes obtenus est multiplié par 5 environ.

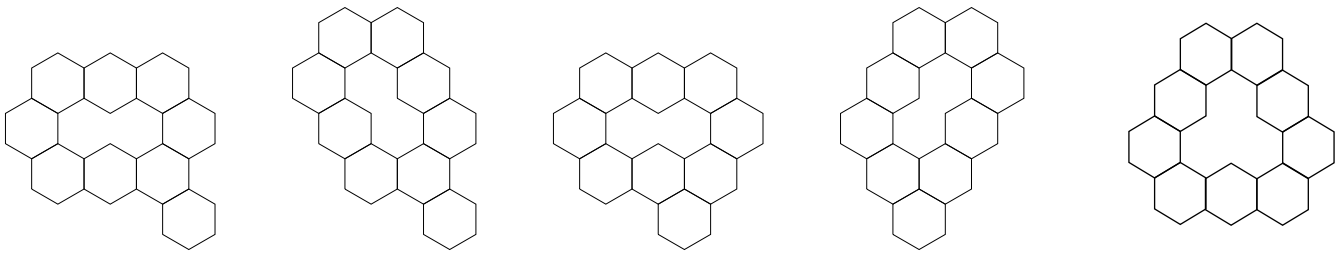
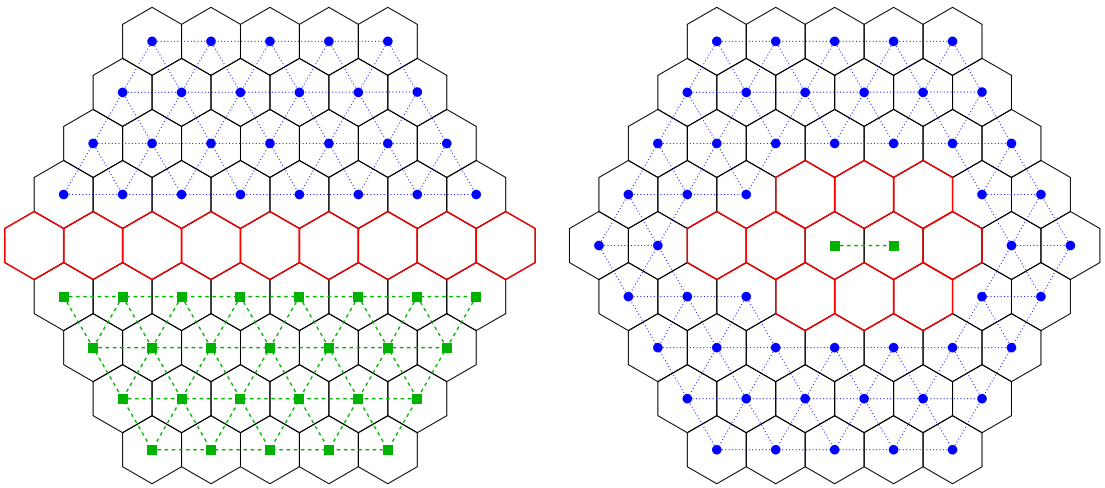


FIGURE 3.16. – Ensemble des coronoïdes de 9 hexagones

Dans cette partie, nous présenterons une modélisation capable de résoudre le problème de génération de coronoïdes. L'intérêt de cette dernière est qu'elle permet de spécifier le nombre de trous présents dans les benzénoïdes générés. De ce fait, elle permet également de ne générer que les benzénoïdes n'admettant aucun trou (en spécifiant un nombre de trous égal à 0). Plus formellement, nous souhaitons modéliser la propriété : *"tous les benzénoïdes générés ont h trous et n hexagones"*.



(a) (b)

FIGURE 3.17. – Deux benzénoïdes de neuf hexagones (en rouge) inclus dans le coronénoïde de taille 5 et son complémentaire composé, pour chacun, de deux composantes connexes décrites respectivement en bleu et en vert).

n	#coronoïdes	temps (s)	#benzénoïdes sans trous
8	1	0.986	585
9	5	1.051	1 261
10	43	81.270	30 086
11	283	279.981	141 229
12	1 954	30 611.139	3 198 256

TABLEAU 3.1. – Nombre de coronoïdes obtenus en creusant des trous depuis tous les benzénoïdes de n hexagones.

Ici, nous partons de l'idée qu'un trou est constitué de plusieurs hexagones du coronénoïde englobant qui ne participent pas à la structure décrite par x_G. Étant donné le graphe d'hexagones $B_h^{c(k(n))}$ du coronénoïde de taille $k(n)$, nous définissons $\overline{B_h}$ comme étant le complémentaire du graphe B_h dans $B_h^{c(k(n))}$. En d'autres termes, les sommets de ce graphe sont les sommets de $B_h^{c(k(n))}$ qui n'appartiennent pas aux sommets de B_h (décrit par x_G), les arêtes sont également conservées. Il est clair que toute composante connexe de $\overline{B_h}$ ne comportant aucun hexagone appartenant au contour de $B_h^{c(k(n))}$ correspond à un trou. Par conséquent, le nombre de trous peut être contraint en utilisant la contrainte globale `nbConnectedComponents`. Malheureusement, nous ne pouvons pas l'appliquer directement à $\overline{B_h}$ car il peut admettre un nombre quelconque de composantes connexes indépendamment de l'existence de trou. Par exemple, la figure 3.17 présente deux benzénoïdes de neuf hexagones dont le graphe $\overline{B_h}$ possède deux composantes connexes décrites respectivement en bleu et vert. Le premier benzénoïde (figure 3.17(a)) ne possède aucun trou et n'est donc pas un coronoïde, contrairement à celui de la figure 3.17(b). Il est donc important de faire en sorte que les composantes connexes de $\overline{B_h}$ touchant la bordure de $B_h^{c(k(n))}$ soient considérées comme une unique composante connexe. Pour cela, on peut facilement remarquer que toutes les composantes connexes de $\overline{B_h}$ touchant la bordure de $B_h^{c(k(n))}$ sont connectées à la face externe de $B_h^{c(k(n))}$. On se propose donc d'ajouter à $\overline{B_h}$ la face externe de $B_h^{c(k(n))}$. Ce procédé nous donne un nouveau graphe dénoté $B_{he}^{c(k(n))}$. La figure 3.18 présente ce graphe pour le coronoïde de la figure 3.17(b). Maintenant, toutes les composantes connexes problématiques sont réunies en une seule. Nous pouvons donc appliquer la contrainte `nbConnectedComponents`$(B_{he}^{c(k(n))}, h+1)$ qui va spécifier que le nombre de composantes connexes de $B_{he}^{c(k(n))}$ doit être égal à $h+1$.

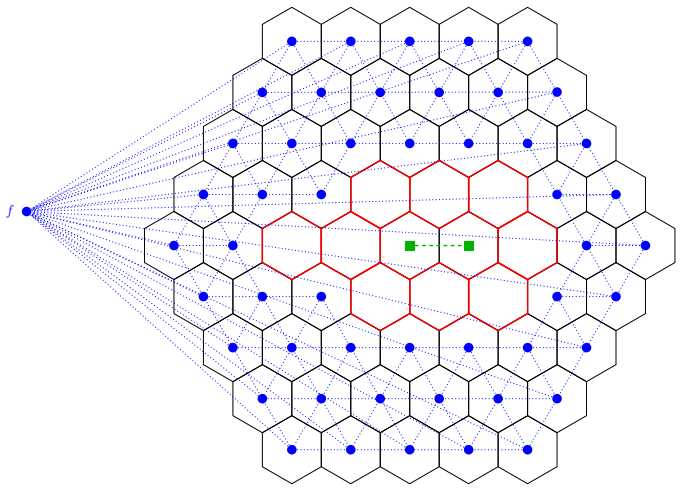


FIGURE 3.18. – Un coronoïde de neuf hexagones (en rouge) inclus dans le coronénoïde de taille 5 et son complémentaire (en vert et en bleu). Le complémentaire possède deux composantes connexes (décrites respectivement en bleu et en vert). Les points bleus et les arêtes bleues en pointillés correspondent respectivement aux sommets et aux arêtes de la première composante. Le sommet f correspond à la face externe du complémentaire. La seconde composante ne contient pas la face externe et représente donc un trou.

Plus formellement, nous ajoutons les variables suivantes au modèle général afin de pouvoir fixer le nombre de trous :

- Une nouvelle variable de graphe x_T qui va représenter $\overline{B_h}$. L'ensemble de ses sommets sera constitué de la face externe et des hexagones non présents dans x_G. x_T aura comme domaine l'ensemble des sous-graphes du graphe obtenu en ajoutant un sommet au graphe d'hexagones du coronénoïde englobant. Ce dernier va représenter la face externe et sera lié à tous les sommets représentant un hexagone se trouvant sur la bordure du coronénoïde.

- Un ensemble de n_c variables booléennes $\{x_1^{T_2},\dots,x_{n_c}^{T_2}\}$ (n_c étant le nombre d'hexagones du coronénoïde de taille $k(n)$). Comme pour x_i avec x_G, la variable $x_i^{T_2}$ vaudra 1 si le i-ème hexagone du coronénoïde de taille $k(n)$ est utilisé dans le graphe décrit par x_T, 0 sinon. De la même manière, nous ajoutons un ensemble de m_c variables booléennes $\{y_{i,j}^{T_2}|\{i,j\}$ est une arête du graphe d'hexagones du coronénoïde de taille $k(n)\}$. La variable $y_{i,j}^{T_2}$ vaudra 1 si l'arête $\{i,j\}$ du coronénoïde de taille $k(n)$ est présente dans le graphe décrit par x_T, 0 sinon.

Comme pour x_G, nous utilisons des contraintes `nodeChanneling` et `edgeChanneling` afin de lier $x_i^{T_2}$, $y_{i,j}^{T_2}$ et x_T et pour finir, nous considérons les contraintes suivantes afin de garantir que x_T et x_C admettent les bonnes propriétés :

- *x_T possède h+1 composantes connexes* : comme indiqué précédemment, nous appliquons la contrainte `nbConnectedComponents`$(x_T, h+1)$.

- *Un trou doit être constitué d'au moins deux hexagones* : pour cela, nous appliquons la contrainte `minDegrees`$(x_T,1)$. Cette dernière spécifie que tous les sommets du graphe décrit par x_T doivent avoir un degré minimal de 1.

- *Un hexagone du coronénoïde englobant est soit dans x_G, soit dans x_T* : dans ce cas, nous ajoutons une clause XOR entre x_i^G et $x_i^{T_2}$ (c'est-à-dire $x_i \oplus x_i^{T_2}$) pour tout $i \in \{1,\dots,n_c\}$.

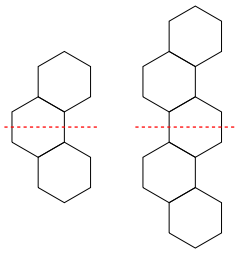
- *Si un hexagone de x_T se situe sur la bordure du coronénoïde englobant, alors il doit exister une arête liant cet hexagone et la face externe* : nous ajoutons une clause $\overline{x_i^{T_2}} \vee y_{i,f}^{T_2}$, où f est l'indice de la face externe, pour tout $i \in \{1, \ldots, n_c\}$ dont l'hexagone appartient à la bordure du coronénoïde englobant.

3.4.6. Génération de benzénoïdes admettant des symétries

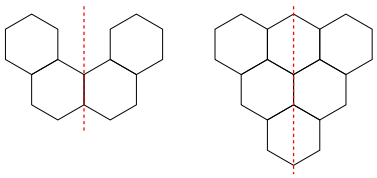
La notion de symétrie est un concept central de la chimie, que ce soit au niveau atomique, moléculaire ou supramoléculaire. En chimie organique, la recherche de composants hautement symétriques admettant des squelettes de polyèdres réguliers (tétraèdres, cubes, dodécaèdres), les cages pseudo-sphériques (fullerènes), les hélices (Roy, Berezhnaia, Villa et al. 2020), les tubes, ... ont été la source de beaucoup de recherches (Bauschlicher, Peeters et Allamandola 2008 ; Cocchi, Prezzi, Ruini et al. 2014 ; Bouwman, Linnartz et Tielens 2021). Par exemple, le tétraédrane, un HAP hypothétique, n'a toujours pas été synthétisé (en 2021), alors que la synthèse des cubanes ou des dodécaèdres date respectivement des années 1960 et 1980 (Eaton et Cole 1964 ; Ternansky, Balogh et Paquette 1982).

Les molécules symétriques ne sont pas uniquement fascinantes par leur « beauté », elles admettent également un certain nombre de propriétés très intéressantes (électroniques, spectroscopiques, magnétiques, ...) (Kastler, Schmidt, Pisula et al. 2006 ; Konishi, Horii, Shiomi et al. 2019). Dans le domaine de la chimie théorique, l'importance des symétries est reconnue depuis longtemps dans le développement de méthodes de structures de l'électronique moderne. L'utilisation de la théorie des groupes permet une simplification drastique de certaines étapes de la résolution de l'équation de Schrödinger, ce qui induit une économie de temps d'exécution très élevée (Longuet-Higgins 1963 ; Taylor 1992).

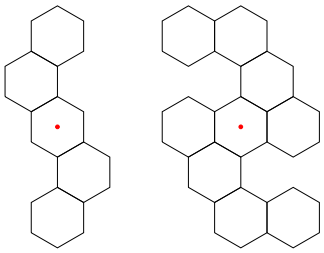
Les benzénoïdes sont classés par les chimistes selon 13 différentes classes de symétrie (symétries laissant un benzénoïde invariant par une rotation de 60, 120 ou 180 degrés et/ou par miroir). Chacune d'entre elles est définie dans les figures 3.19 et 3.20. Nous pouvons facilement générer les benzénoïdes appartenant à ces classes en considérant des contraintes qui vont forcer ces symétries. Plus précisément, pour chaque symétrie σ désirée, il suffit de poster la contrainte $x_h \Leftrightarrow x_{\sigma(h)}$ pour tous les hexagones h de x_G, où $\sigma(h)$ est l'image de h par la symétrie σ. Les axes de symétries sont naturellement placés au centre du coronénoïde englobant.



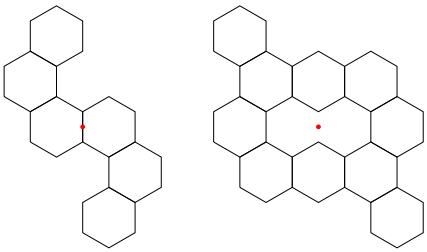
symétrie face-mirror (C_2v(a))



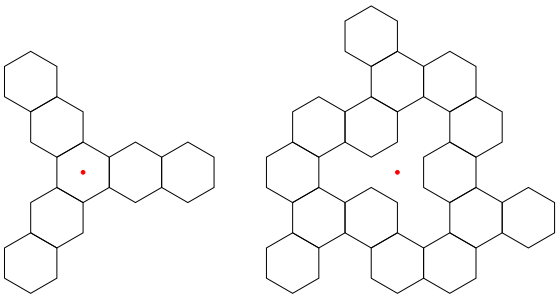
symétrie edge-mirror (C_2v(b))



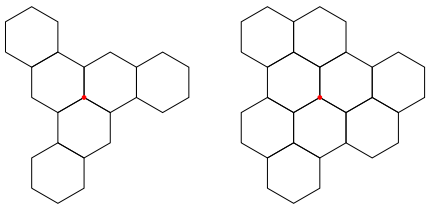
symétrie vertex-180°-rotation (C_2h(i))



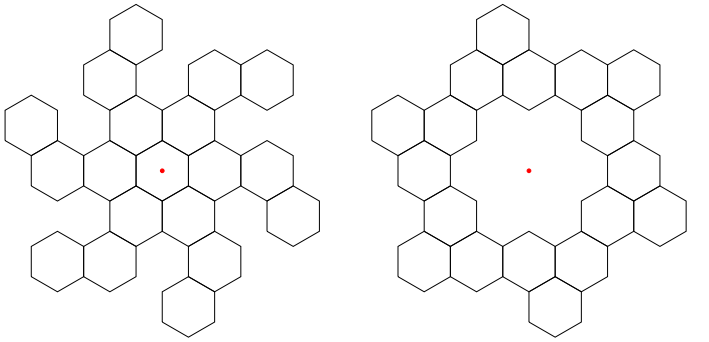
symétrie edge-180°-rotation (C_2h(ii))



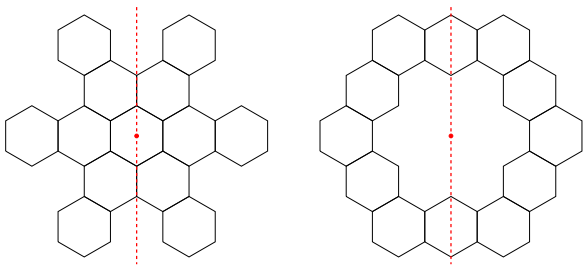
symétrie face-120°-rotation (C_3h(i))



symétrie vertex-120°-rotation (C_3h(ii))

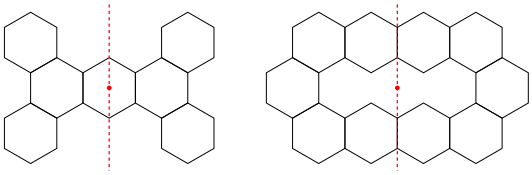


symétrie face-60°-rotation (C_6h)

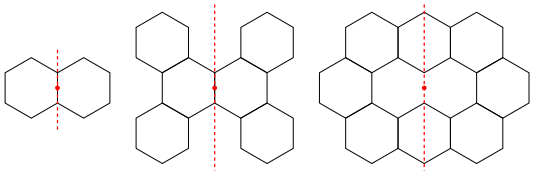


symétrie vertex-60°-rotation + edge-mirror (D_6h)

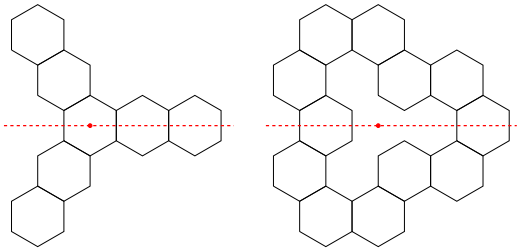
FIGURE 3.19. – Les 13 classes de symétries admises par les chimistes (1/2)



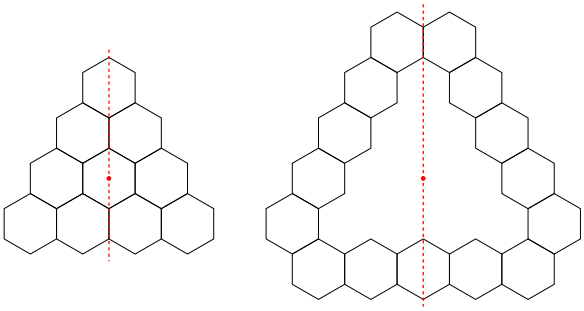
symétrie face-180°-rotation + edge-mirror (D_2h(i))



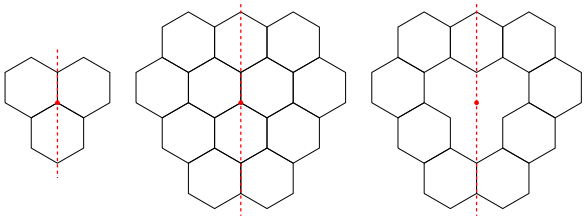
symétrie edge-180°-rotation + edge-mirror (D_2h(ii))



symétrie face-120°-rotation + face-mirror (D_3h(ia))



symétrie face-120°-rotation + edge-mirror (D_3h(ib))



symétrie vertex-120°-rotation + edge-mirror (D_3h(ii))

FIGURE 3.20. – Les 13 classes de symétries admises par les chimistes (2/2)

3.4.7. Génération de benzénoïdes ayant un facteur d'irrégularité donné

Pour rappel, le *paramètre d'irrégularité* ξ_B (définition 2.2.5) d'un benzénoïde B est calculé de la manière suivante :

$$\xi_B = \frac{N_3 + N_4}{N_1 + N_2 + N_3 + N_4}$$

Où N_1, N_2, N_3 et N_4 correspondent respectivement aux nombres de carbones de B impliqués dans un solo, duo, trio ou quatuor (définition 2.2.4). Dans cette sous-section, on souhaite définir un modèle CSP permettant de générer des structures de benzénoïdes ayant un paramètre d'irrégularité donné (voir définition 2.2.5). Dans (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019), Bouwman et al. ont présenté des résultats de calculs de spectres infrarouges sur un ensemble de PAH, et ces derniers étaient classés selon leur paramètre d'irrégularité. Être capable de générer des structures de benzénoïdes ayant un paramètre d'irrégularité donné (ou inclus dans un intervalle) serait donc intéressant d'un point de vue chimie. Par exemple, cela pourrait permettre de compléter les travaux de Bouwman et al. en considérant des ensembles exhaustifs de benzénoïdes ayant un paramètre d'irrégularité donné.

Avant d'expliquer comment cette propriété peut être modélisée sous la forme d'une instance CSP, il faut ajouter quelques précisions. Premièrement, on appelle *voisinage* d'un hexagone h l'ensemble des hexagones adjacents à ce dernier. Il est facile de voir qu'un hexagone a forcément entre 1 et 6 autres hexagones dans son voisinage (et 0 dans le cas du benzène). Le modèle qui sera décrit dans cette sous-section repose sur le fait que l'existence d'un solo, d'un duo, d'un trio ou d'un quatuor dans h dépend uniquement de son voisinage. En d'autres termes, les hexagones présents dans le voisinage de h dans le graphe d'hexagones nous permettent de déterminer les groupes induits par l'hexagone h. Pour pouvoir exploiter formellement

cette propriété, on se propose de numéroter les 6 voisins possibles de h de la manière suivante : la position 1 est associée au voisin situé en haut à droite tandis que les autres positions sont numérotés de 2 à 6 (dans le sens des aiguilles d'une montre). La figure 3.21 nous montre cette numérotation. Ensuite, on appelle *configuration de voisinage de h* un tuple de 6 valeurs tel que sa i-ème valeur est égale à 1 si h admet un voisin à la position i, et 0 dans le cas contraire. Par exemple, la figure 3.21 (a) nous montre un hexagone h ayant $(1,0,1,0,1,0)$ comme configuration de voisinage. Si l'on considère cette configuration, on voit clairement que h n'admet aucun groupe, quelle que soit sa taille.

Deuxièmement, on peut facilement voir que si on passe d'une configuration à une autre en effectuant une rotation de 60°, le nombre de solos, duos, trios et quatuors de ces deux configurations sera identique. Ainsi, étant donné une configuration de voisinage N_h d'un hexagone h, on note N_h^* l'ensemble des configurations de voisinage qui peuvent être obtenues en appliquant des rotations successives de 60° sur N_h. Par exemple, la figure 3.21 nous montre l'ensemble de configurations $(1,0,1,0,1,0)^*$. On remarque également qu'appliquer une rotation de 60° sur une configuration est équivalent à effectuer une permutation cyclique sur le tuple associé à cette configuration. Ainsi, on peut facilement identifier toutes les configurations N_h^* possibles et pour chacune d'entre elles, déterminer les groupes induits. La figure 3.22 présente les 13 configurations N_h^* possibles et les groupes qu'elles induisent. Notons que le cas (a) est particulier, car h admet un groupe de taille supérieure à 4. Comme on ne s'intéresse qu'aux groupes de taille au plus 4, on va considérer qu'il n'admet aucun groupe. De plus, ce cas n'est pas très intéressant, car il ne concerne qu'une seule molécule : le benzène (constitué d'un unique hexagone).

On se propose maintenant de définir un modèle $\mathscr{M}_\xi$ capable de générer des structures de benzénoïdes ayant un paramètre d'irrégularité donné. Comme pour les modèles précédents, on part du modèle général et on va y ajouter plusieurs variables et contraintes dans le but d'être capable de générer des structures de benzénoïdes ayant un paramètre d'irrégularité donné. Premièrement, on ajoute les variables suivantes au modèle général :

- $X_Z = \{z_1, \ldots, z_{n_c}\}$: un ensemble de variables booléennes ($\{0,1\}$ comme domaine). La variable z_h vaudra 1 si l'hexagone h n'induit aucun groupe, 0 sinon.

- $X_S = \{s_1, \ldots, s_{n_c}\}$: un ensemble de variables entières ayant pour domaine $\{0,1,2\}$. La variable s_h aura pour valeur le nombre de carbones de l'hexagone h appartenant à un solo.

- $X_D = \{d_1, \ldots, d_{n_c}\}$: un ensemble de variables entières ayant pour domaine $\{0,2\}$. La variable d_h aura pour valeur le nombre de carbones de l'hexagone h appartenant à un duo.

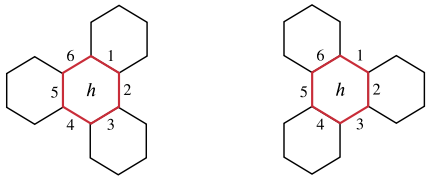
- $X_T = \{t_1, \ldots, t_{n_c}\}$: un ensemble de variables entières ayant pour domaine $\{0,3\}$. La variable t_h aura pour valeur le nombre de carbones de l'hexagone h appartenant à un trio.

- $X_Q = \{q_1, \ldots, q_{n_c}\}$: un ensemble de variables entières ayant pour domaine $\{0,4\}$. La variable q_h aura pour valeur le nombre de carbones de l'hexagone h appartenant à un quatuor.

- N_0 : une variable entière ayant pour domaine $\{0, \ldots, n_c\}$ qui représente le nombre d'hexagones n'induisant aucun groupe.

- N_1, N_2, N_3, N_4 : quatre variables entières qui représentent respectivement le nombre de carbones impliqués dans un solo, duo, trio et quatuor du benzénoïde. Chacune de ces variables a pour domaine $\{0, \ldots, n_c\}$.

- x_ξ : une variable entière de domaine $\{0,1,\ldots,100\}$ qui représente le paramètre d'irrégularité du benzénoïde généré multiplié par 100. On doit procéder de la sorte, car Choco n'autorise pas les variables réelles. On a donc :
$$x_\xi = 100 \times \xi = \frac{100 \times (N_3 + N_4)}{N_1 + N_2 + N_3 + N_4}$$



(a) (b)

FIGURE 3.21. – L'ensemble de configuration de voisinage $(1,0,1,0,1,0)^*$.

En considérant la valeur 100, on suppose qu'une précision de l'ordre de 0,01 est suffisante. Si ce n'est pas le cas, il est bien évidemment possible d'adapter cette valeur à la précision voulue.

Au niveau des contraintes, la première tâche consistera à exprimer la relation entre l'existence de l'hexagone h (donc de la variable x_h) et les variables associées à cet hexagone z_h, s_h, d_h, t_h et q_h. Si l'hexagone h n'existe pas dans le benzénoïde généré (et donc si x_h est égal à 0), alors par définition, cet hexagone ne devra induire aucun groupe. Pour modéliser cela, on considère les contraintes suivantes :

$$\begin{cases} x_h = 0 \Rightarrow z_h = 0 \\ x_h = 0 \Rightarrow s_h = 0 \\ x_h = 0 \Rightarrow d_h = 0 \\ x_h = 0 \Rightarrow t_h = 0 \\ x_h = 0 \Rightarrow q_h = 0 \end{cases}$$

En revanche, si l'hexagone h existe (et donc que x_h vaut 1), alors on doit calculer les valeurs de z_h, s_h, d_h, t_h et q_h en fonction de la configuration de voisinage de h. Pour faire ça, on considère une contrainte de table dont les tuples autorisés sont construits à partir de tous les cas possibles listés dans la figure 3.22. En d'autres termes, nous avons un tuple autorisé pour chaque configuration de voisinage possible et chacun d'entre eux fait le lien avec le nombre de groupes induits par la configuration. Par exemple, la table 3.2 présente une partie de la contrainte de table associée à l'ensemble de configurations de voisinage $(1,0,0,0,0,0)^*$ (représenté dans la figure 3.22(b) pour un hexagone h). Penchons-nous sur la portée de cette contrainte, elle englobe toutes les variables x_i correspondant aux hexagones de la configuration de voisinage de h ainsi que les variables z_h, s_h, d_h, t_h, q_h et x_h. La présence de x_h dans la table s'explique par le fait que nous voulons que cette dernière soit égale à 1 pour que la contrainte de table soit active. En effet, dans le cas contraire, cela signifierait que l'hexagone h n'admet aucun groupe et la contrainte de table serait inutile. Notons que les hexagones situés sur le bord du graphe d'hexagones du coronénoïde de taille $k(n)$ ne peuvent pas avoir six voisins. Dans ce cas, on ne va considérer que les voisins existant dans la portée de la contrainte, et la table sera calculée en ne sélectionnant que les lignes ayant la valeur 0 pour les hexagones manquants et en projetant la relation obtenue sur la portée de la contrainte.

Configuration de voisinage de h						z_h	s_h	d_h	t_h	q_h	x_h
1	0	0	0	0	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	1	1

TABLEAU 3.2. – Une partie de la contrainte de table associée à l'ensemble de configurations $(1,0,0,0,0,0)^*$ représenté dans la figure 3.22(b) pour l'hexagone h.

Avec ces contraintes, nous sommes capables de connaître tous les groupes de chaque hexagone en accord avec leur configuration de voisinage. Maintenant, nous ajoutons une série de contraintes `sum` ayant pour but de fixer les valeurs de N_0, N_1, N_2, N_3 et de N_4 :

- N_0 *est égal au nombre d'hexagones n'induisant aucun groupe* :

$$N_0 = \sum_{z_h \in X_Z} z_h$$

- N_1 *est égal au nombre de solos* :

$$N_1 = \sum_{s_h \in X_S} s_h$$

- N_2 *est égal au nombre de duos* :

$$N_2 = \sum_{d_h \in X_D} d_h$$

- N_3 *est égal au nombre de trios* :

$$N_3 = \sum_{t_h \in X_T} t_h$$

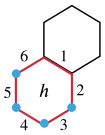
- N_4 *est égal au nombre de quatuors* :

$$N_4 = \sum_{q_h \in X_Q} q_h$$

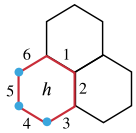
Ensuite, nous devons fixer la valeur de x_ξ à $\frac{100 \times (N_3+N_4)}{N_1+N_2+N_3+N_4}$. Cela peut être facilement fait en appliquant une série de contraintes spécifiant que $x_\xi = 100 \times (N_3 + N_4) \div (N_1 + N_2 + N_3 + N_4)$ où $\div$ fait référence au quotient de la division euclidienne.

Pour conclure, il est possible d'ajouter des contraintes arithmétiques $(\leq, =, \geq)$ sur $x_\xi, N_0, N_1, N_2, N_3$ et N_4 en fonction des souhaits de l'utilisateur. Cela permet de générer des structures de benzénoïdes ayant un paramètre d'irrégularité donné, et/ou un nombre donné de solos/duos/trios/quatuors.

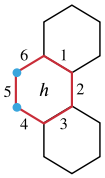
(a) $(0,0,0,0,0,0)^* \Longrightarrow$ aucun groupe



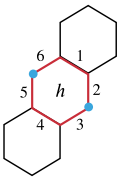
(b) $(1,0,0,0,0,0)^* \Longrightarrow$ un quatuor



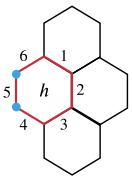
(c) $(1,1,0,0,0,0)^* \Longrightarrow$ un trio



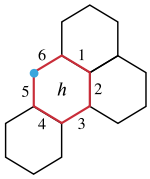
(d) $(1,0,1,0,0,0)^* \Longrightarrow$ un duo



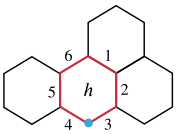
(e) $(1,0,0,1,0,0)^* \Longrightarrow$ deux solos



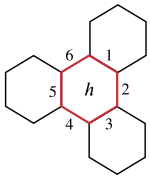
(f) $(1,1,1,0,0,0)^* \Longrightarrow$ un duo



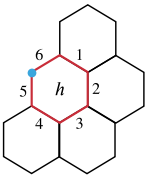
(g) $(1,1,0,1,0,0)^* \Longrightarrow$ un solo



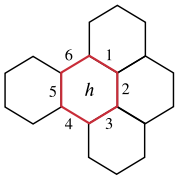
(h) $(1,1,0,0,1,0)^* \Longrightarrow$ un solo



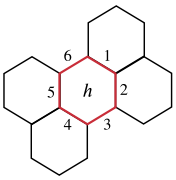
(i) $(1,0,1,0,1,0)^* \Longrightarrow$ aucun groupe



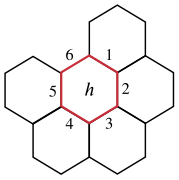
(j) $(1,1,1,1,0,0)^* \Longrightarrow$ un solo



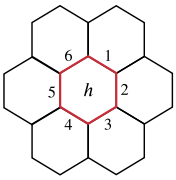
(k) $(1,1,1,0,1,0)^* \Longrightarrow$ aucun groupe



(l) $(1,1,0,1,1,0)^* \Longrightarrow$ aucun groupe



(m) $(1,1,1,1,1,0)^* \Longrightarrow$ aucun groupe



(n) $(1,1,1,1,1,1)^* \Longrightarrow$ aucun groupe

FIGURE 3.22. – Toutes les configurations de voisinages possibles et les groupes qu'elles induisent.

3.4.8. Génération de benzénoïdes ayant un nombre donné de carbones et d'hydrogènes

Dans cette partie, on se propose de décrire un modèle CSP que l'on notera $\mathscr{M}_{CH}$ capable de générer des structures de benzénoïdes ayant un nombre donné d'atomes de carbone et d'hydrogène. Être capable de générer des molécules ayant de telles propriétés est intéressant du point de vue des chimistes. En effet, ils ont l'habitude de définir les molécules par leur *formule moléculaire*. Cette formule spécifie le nombre d'atomes de chaque élément apparaissant dans la molécule. Par exemple, la formule moléculaire de l'anthracène représenté à la figure 2.2 (b) est $C_{14}H_{10}$, ce qui signifie que cette molécule est constituée de 14 carbones et 10 hydrogènes. Pour illustrer cela davantage, dans le contexte des HAP, les chimistes utilisent cette formule afin de classifier les HAP (par exemple dans la *NASA Ames database* (BAUSCHLICHER, RICCA, BOERSMA et al. 2018)). De plus, ils leur arrivent régulièrement d'avoir affaire à des molécules dites *isomères* qui sont des molécules ayant la même formule moléculaire, mais des structures différentes.

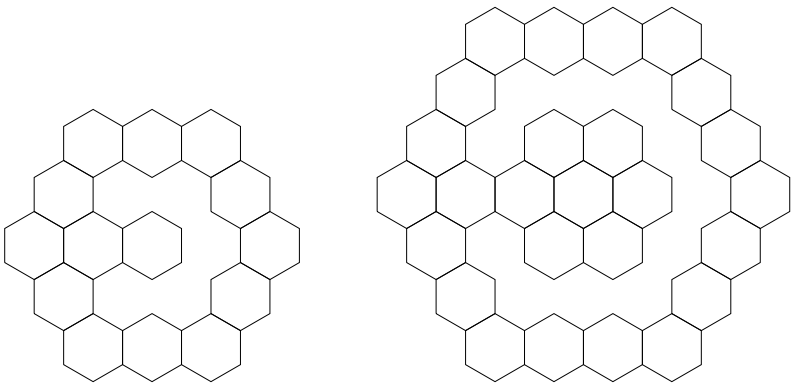
Étant donné deux entiers v et μ représentant respectivement le nombre de carbones et d'hydrogènes voulus, on démarre encore une fois du modèle général et on va y ajouter des variables et des contraintes dans le but d'être capable de générer des structures de benzénoïdes ayant un nombre donné d'atomes de carbone et d'hydrogène. Dans les deux sous parties suivantes, nous allons décrire comment est-ce qu'on peut contraindre le nombre de carbones et d'hydrogènes.

3.4.8.1. Contraindre le nombre d'atomes d'hydrogène

Afin de pouvoir contraindre le nombre d'atomes d'hydrogène, on doit tout d'abord l'exprimer dans notre modèle. Il est facile de voir que le nombre d'hydrogènes d'un benzénoïde est égal au nombre de carbones appartenant à un solo, un duo, un trio ou un quatuor excepté pour le cas du benzène qui possède six hydrogènes. Il est donc possible de modéliser cette propriété sous la forme d'un modèle similaire à celui présenté dans la sous-section 3.4.7. Les seules différences avec ce modèle sont d'une part que nous n'avons plus à traiter la présence des différents groupes (étant donné que l'on ne ce soucie que du nombre d'hydrogènes) et que nous devons prendre en compte le cas où un hexagone ne possède aucun voisin (pour pouvoir générer le benzène correctement). Nous introduisons donc une variable entière N_H qui représentera le nombre d'hydrogènes du benzénoïde généré. On sait que le plus petit nombre d'hydrogènes d'un benzénoïde est 6 (atteint par le benzène). La borne inférieure du domaine sera donc 6. Pour ce qui est du nombre d'hydrogènes le plus élevé, il est atteint par les structures de benzénoïdes construites de la manière suivante. Si $k(n)$ est impair, on démarre du benzène, sinon du coronénoïde de taille 2. Ensuite, on va successivement ajouter à notre construction une couronne vide suivie d'une couronne remplie d'hexagones jusqu'à obtenir $k(n)$ couronnes. Comme un benzénoïde est toujours connexe, on ajoute un hexagone dans chacune de ses couronnes vides afin de garantir sa connexité. La figure 3.23 présente les benzénoïdes avec le plus d'hydrogènes quand $k(n)$ est égal à 3 ou 4. On peut facilement voir qu'une couronne de taille c a $6 \times c$ hydrogènes dans sa bordure extérieure et en a $(c-1) \times c$ dans sa bordure intérieure. Par conséquent, en considérant les hexagones ajoutés pour assurer la connexité, on peut définir la borne supérieure du domaine de N_H avec la formule suivante :

$$\begin{cases} 6 & & & \text{si } k(n) = 1 \\ 4 & + \sum_{1<c\leq k(n),\ c\ est\ impaire} & 6\times c + (c-1)\times c - 2 & \text{si } k(n) \text{ est impair et supérieur à 1} \\ 10 & + \sum_{2<c\leq k(n),\ c\ est\ pair} & 6\times c + (c-1)\times c - 2 & \text{sinon} \end{cases}$$

Nous considérons un ensemble $H^* = \{H_1, \ldots, H_{n_c}\}$ de variables entières. Chacune de ces variables aura pour domaine l'ensemble des entiers compris entre 1 et 6 (inclus). La valeur de la variable H_h correspondra au nombre d'hydrogènes présents sur l'hexagone h du benzénoïde généré. Nous considérons ensuite une contrainte de table dont les tuples sont construits à partir de toutes les configurations de voisinage d'un hexagone (représentées dans la figure 3.24) et chacun d'entre eux fait le lien avec le nombre d'hydrogènes induits par la configuration. Intéressons-nous maintenant à la portée de cette contrainte. Pour un hexagone h,



(a) (b)

FIGURE 3.23. – Les structures de benzénoïdes maximisant le nombre d'hydrogènes et qui peuvent être incluses dans un coronénoïde de taille 3(a) et 4 (b).

cette dernière englobe toutes les variables x_i correspondant aux hexagones de la configuration de voisinage de h ainsi que les variables H_h et x_h. Comme pour le modèle décrit dans la sous-section 3.4.7, la présence de x_H dans la table s'explique par le fait que l'on veut que cette dernière soit égale à 1 pour que la contrainte de table soit active. Car dans le cas contraire, cela signifierait que l'hexagone h n'est pas présent dans le benzénoïde généré et que par conséquent, il n'admet aucun hydrogène. La table 3.3 décrit une partie de la contrainte de table associée à l'ensemble de configurations $(1,0,0,0,0,0)^*$ représentée dans la figure 3.24 (b). Ensuite, il ne reste plus qu'à poser la contrainte suivante afin de spécifier que la valeur de N_H soit égale au nombre d'hydrogènes du benzénoïde généré :

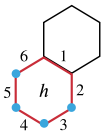
$$N_H = \sum_{H_h \in H^*} H_h$$

Pour terminer, nous pouvons, comme pour l'irrégularité, contraindre le nombre d'hydrogènes en fonction des souhaits de l'utilisateur à l'aide de contraintes arithmétiques. Par exemple, si on veut générer toutes les structures de benzénoïdes possédant μ atomes d'hydrogène, on va ajouter la contrainte $N_H = \mu$.

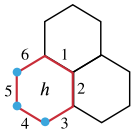
Configuration de voisinage de h						h_h	x_h
1	0	0	0	0	0	4	1
0	1	0	0	0	0	4	1
0	0	1	0	0	0	4	1
0	0	0	1	0	0	4	1
0	0	0	0	1	0	4	1
0	0	0	0	0	1	4	1

TABLEAU 3.3. – Une partie de la contrainte de table associée à l'ensemble de configurations $(1,0,0,0,0,0)^*$ représenté dans la figure 3.22(b) pour l'hexagone h.

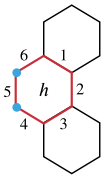
(a) $(0,0,0,0,0,0)^* \Longrightarrow$ 6 hydrogènes



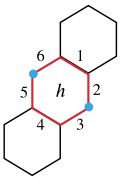
(b) $(1,0,0,0,0,0)^* \Longrightarrow$ 4 hydrogènes



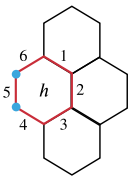
(c) $(1,1,0,0,0,0)^* \Longrightarrow$ 3 hydrogènes



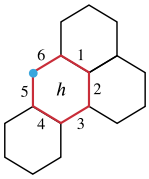
(d) $(1,0,1,0,0,0)^* \Longrightarrow$ 2 hydrogènes



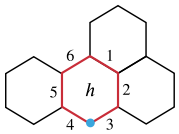
(e) $(1,0,0,1,0,0)^* \Longrightarrow$ 2 hydrogènes



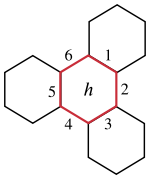
(f) $(1,1,1,0,0,0)^* \Longrightarrow$ 2 hydrogènes



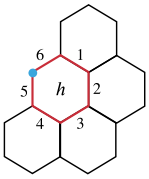
(g) $(1,1,0,1,0,0)^* \Longrightarrow$ 1 hydrogène



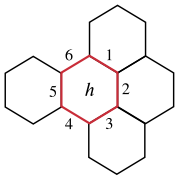
(h) $(1,1,0,0,1,0)^* \Longrightarrow$ 1 hydrogène



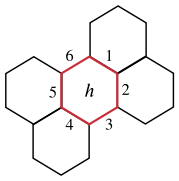
(i) $(1,0,1,0,1,0)^* \Longrightarrow$ aucun hydrogène



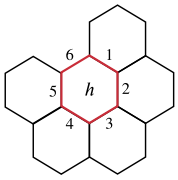
(j) $(1,1,1,1,0,0)^* \Longrightarrow$ 1 hydrogène



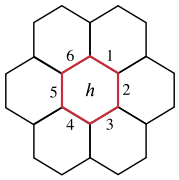
(k) $(1,1,1,0,1,0)^* \Longrightarrow$ aucun hydrogène



(l) $(1,1,0,1,1,0)^* \Longrightarrow$ aucun hydrogène



(m) $(1,1,1,1,1,0)^* \Longrightarrow$ aucun hydrogène



(n) $(1,1,1,1,1,1)^* \Longrightarrow$ aucun hydrogène

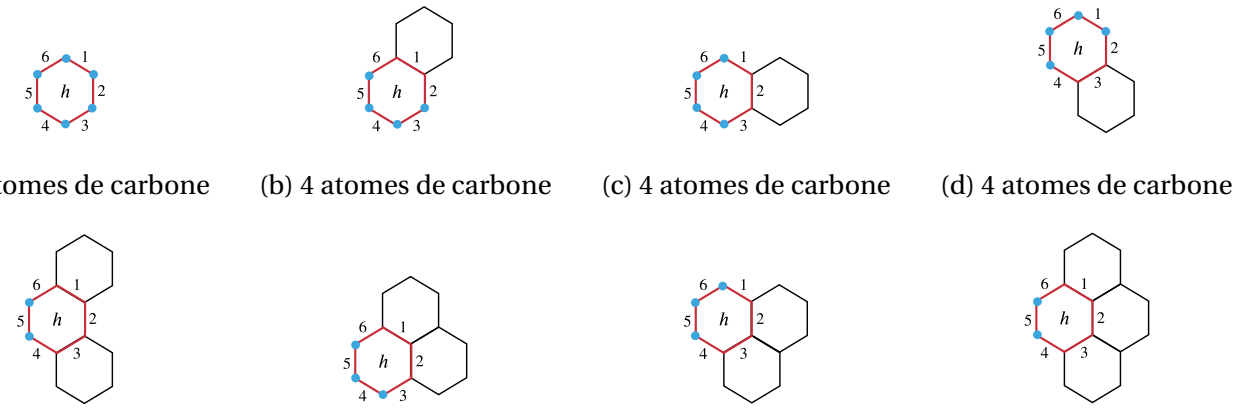
FIGURE 3.24. – Toutes les configurations de voisinages possibles et le nombre d'hydrogènes qu'elles induisent.

3.4.8.2. Contraindre le nombre d'atomes de carbones

Étant donné que notre modèle général repose principalement sur les hexagones, les atomes de carbone (et d'hydrogène) ne sont pas explicitement représentés. De plus, certains atomes de carbone peuvent être partagés par deux ou trois hexagones. Par conséquent, quand on va vouloir modéliser le nombre de carbones, il faudra faire attention à ne pas compter plusieurs fois le même atome de carbone. Pour éviter ce problème, nous allons exploiter une partition de l'ensemble des carbones basée sur les hexagones. Un atome de carbone est attaché à un hexagone h s'il apparaît uniquement dans h, ou s'il apparaît dans au moins deux hexagones (h inclus) et qu'il se situe le plus à gauche possible. Cela peut facilement être implémenté en considérant, pour chaque hexagone, la partie de son voisinage située à droite et en utilisant les configurations de voisinage d'une manière similaire à laquelle elles ont été exploitées dans la partie 3.4.7. En d'autres termes, la configuration de voisinage est restreinte aux hexagones 1, 2 et 3 (ce qui correspond aux hexagones haut-droit, droit et bas-droit). La figure 3.25 nous montre toutes les configurations possibles pour un hexagone h et le nombre associé de carbones en accord avec notre partition. Par exemple, la figure 3.26 décrit un nombre d'atomes de carbone pour chaque hexagone du coronène en considérant la partition définie plus haut.

On part encore une fois du modèle général. D'abord, on introduit une variable entière N_C qui représentera le nombre de carbones du benzénoïde généré. Son domaine est $\{6,\dots,\nu_{k(n)}\}$ avec $\nu_{k(n)}$ le nombre d'atomes de carbone du coronénoïde de taille $k(n)$. On peut facilement prouver par induction sur $k(n)$ que $\nu_{k(n)}$ est égal à $6.k(n)^2$. Ensuite, on ajoute également au modèle une série de variables entières $N_C^1,\dots,N_C^{n_c}$, chacune de domaine $\{2,3,4,6\}$. La variable N_C^h correspondra au nombre d'atomes de carbone liés à l'hexagone h en accord avec la partition définie plus haut. Au niveau des contraintes, on considère une contrainte de table décrite dans le tableau 3.4. Chaque tuple de cette table représente une des configurations décrites dans la figure 3.25. Pour un hexagone h, la portée de la contrainte englobera toutes les variables x_i correspondant aux hexagones voisins de h situés à sa droite ainsi que N_C^h. Comme précédemment, cette contrainte de table sera active uniquement quand $x_h = 1$ (on utilise encore une fois la contrainte `ifThen`). Évidemment, si h est situé sur la bordure supérieure, droite ou inférieure du coronénoïde de taille $k(n)$, on adaptera la contrainte comme dans la partie précédente. Pour finir, on définit N_C comme étant la somme des atomes de carbones associés à chaque hexagone du benzénoïde en appliquant la contrainte $N_C = \sum_{i\in\{1,\dots n_c\}} N_c^i$.

Encore une fois, il est possible d'appliquer des contraintes arithmétiques sur N_C afin de pouvoir générer des benzénoïdes ayant un nombre donné de carbones. Par exemple, si on veut générer les structures de benzénoïdes possédant ν atomes de carbones, nous pouvons ajouter la contrainte $N_C = \nu$.



(a) 6 atomes de carbone (b) 4 atomes de carbone (c) 4 atomes de carbone (d) 4 atomes de carbone

(e) 2 atomes de carbone (f) 3 atomes de carbone (g) 3 atomes de carbone (h) 2 atomes de carbone

FIGURE 3.25. – Toutes les configurations de voisinage possibles pour un hexagone h et le nombre de carbones associés en accord avec la partition définie.

Configuration de h			c_h
0	0	0	6
0	0	1	4
0	1	0	4
0	1	1	3
1	0	0	4
1	0	1	2
1	1	0	3
1	1	1	2

TABLEAU 3.4. – La contrainte de table correspondant aux possibles configurations de voisinages et leurs contributions en carbone associées.

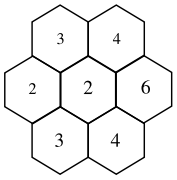


FIGURE 3.26. – Le nombre d'atomes de carbone de chaque hexagone du coronène en accord avec la partition définie.

3.5. Optimisation des modèles

En pratique, nous avons mis en place un certain nombre d'optimisations des modèles qui ont pour but d'accélérer la génération en élaguant l'espace de recherche. Ces optimisations diffèrent en fonction du ou des modules pris en compte. Elles ont volontairement été omises dans les sections précédentes afin d'assurer au lecteur une meilleure compréhension. Cette section présentera trois de ces optimisations et sera organisée de la manière suivante : la sous-section 3.7 détaillera les cas de figure dans lesquels nous pouvons réduire la quantité de nogoods appris. Pour finir, la sous-section 3.8 décrira les cas de figure où l'on peut réduire la taille du coronénoïde englobant.

3.6. Fixer les benzénoïdes générés sur le bord du coronénoïde englobant

Dans cette sous-section, nous présenterons une méthode permettant de spécifier que la structure du benzénoïde généré touche au moins un des hexagones de la bordure gauche du coronénoïde englobant (voir figure 3.27). Cette action a pour but de limiter l'espace de recherche à parcourir en bloquant un certain nombre de symétries par translation. Pour y parvenir, nous considérons X_{bord}, un sous ensemble de X_V (un ensemble de variables booléennes du modèle général permettant de déterminer la présence ou non d'un sommet dans la variable de graphe x_G). Si l'on reprend l'exemple de la figure 3.27, nous obtenons $X_{bord} = \{x_1, x_4, x_8, x_{13}, x_{17}\}$. Il suffit ensuite d'ajouter la contrainte $\texttt{sum}(X_{bord}, \geq, 1)$ afin de spécifier qu'au moins une des variables de X_{bord} soit égale à 1 (ce qui revient à dire que toute structure de benzénoïde générée doit avoir au moins un hexagone appartenant à la bordure gauche du coronénoïde englobant).

Cette optimisation n'est toutefois pas utilisable dans tous les cas de figure. En effet, nous ne pouvons pas l'utiliser dans le cas ou l'on souhaite générer des benzénoïdes rectangulaires avec le second module (voir

sous-section 3.4.3) car nous spécifions que l'hexagone central doit être présent, ce qui implique que certains benzénoïdes générés ne pourront pas atteindre la bordure gauche. Nous n'appliquons pas non plus cette contrainte dans le cas où l'on souhaite générer des benzénoïdes admettant des symétries, car les axes de symétries sont placés au centre du coronénoïde englobant.

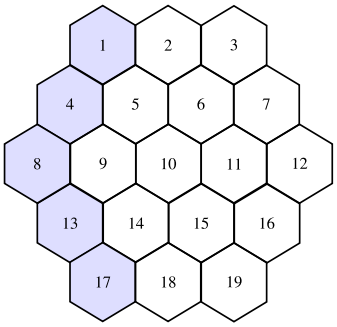


FIGURE 3.27. – Bordure gauche du coronénoïde de taille 3

3.7. Limiter l'apprentissage des nogoods

Dans la section 3.3, nous avons mentionné qu'à chaque fois qu'un nouveau benzénoïde était généré par le solveur, nous apprenions les nogoods correspondant à l'intégralité de ses images par translation/rotation dans le coronénoïde englobant. Cependant, bien que cette approche se révèle très efficace pour éviter la génération de benzénoïdes redondants, le nombre de nogoods appris peut vite devenir élevé, notamment quand nous considérons des coronénoïdes englobants de grandes tailles. C'est dans cette optique que nous avons déterminé différents cas de figures ou nous pouvions réduire le nombre de nogoods appris :

- *Dans le cas où la contrainte de bord est active* :

  Dans ce cas, nous choisissons de n'apprendre que les nogoods correspondant aux images par translation/rotation du benzénoïde dont au moins l'un des hexagones appartient à la bordure gauche du coronénoïde englobant. Cela se justifie par le fait que les autres images sont déjà interdites par la contrainte de bord. La figure 3.28 nous montre un exemple d'apprentissage dans un coronénoïde englobant de taille 2.

- *Dans le cas où l'on veut générer des benzénoïdes admettant certaines symétries* :

  Dans le cas où l'on souhaite générer des benzénoïdes admettant les symétries face-mirror (C_2v(a)) ou edge-mirror (C_2v(b)) (voir sous-section 3.4.6), nous n'apprenons que les nogoods correspondant à l'image par translation du benzénoïde symétrique selon l'axe considéré (les autres images sont, dans tous les cas, interdites par les contraintes spécifiant les symétries). Étant donné que nous ne considérons qu'un seul axe pour ces symétries, les nogoods correspondant aux rotations ne sont pas utiles. La figure 3.29 nous montre un exemple d'apprentissage de nogoods dans le cas de la symétrie face-mirror (C_2v(a)) tandis que la figure 3.30 fait de même pour la symétrie edge-mirror (C_2v(b)). Pour les symétries n'incluant ni face-mirror (C_2v(a)), ni edge-mirror (C_2v(b)), nous conservons la stratégie d'apprentissage de base.

- *Dans les cas où l'on souhaite générer coronénoïdes ou des benzénoïdes en forme de rectangle ou de losanges* :

  Dans ce cas, il n'est pas nécessaire d'apprendre de nogoods. En effet, grâce aux contraintes des modèles permettant de générer ces molécules, chacune des molécules générées n'admettra aucune

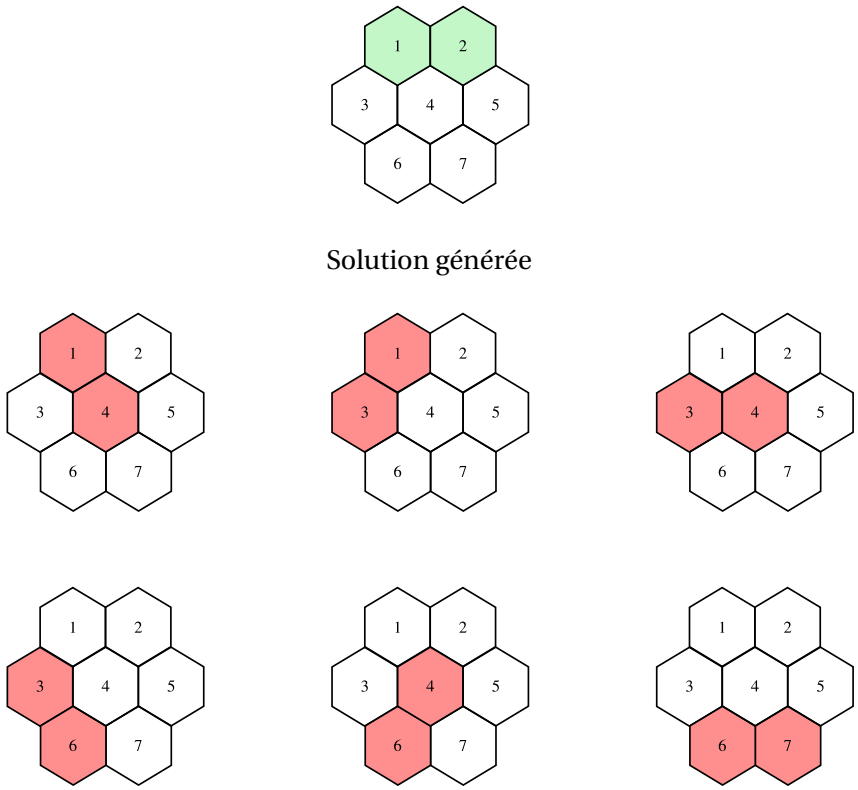
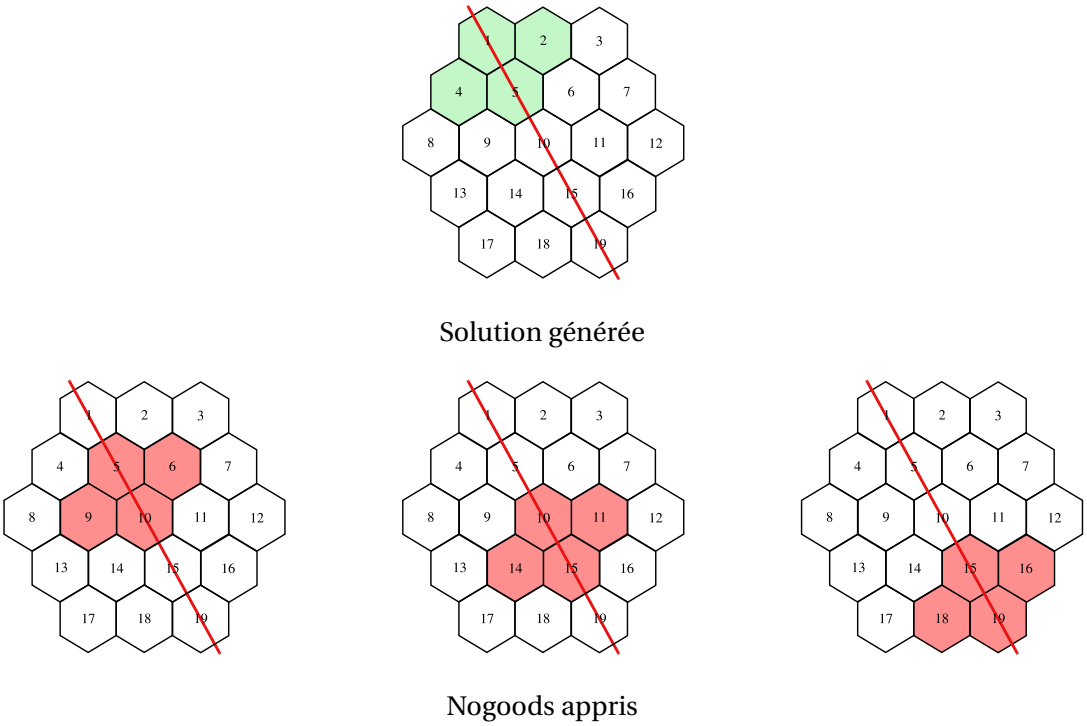


FIGURE 3.28. – Exemple d'apprentissage de nogoods lorsque la contrainte de bord est active

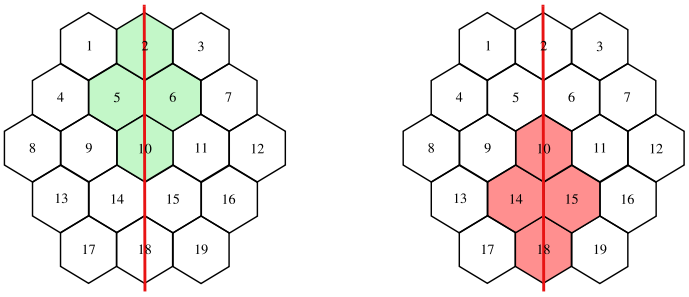
image au sein de son coronénoïde englobant.



Solution générée

Nogoods appris

FIGURE 3.29. – Exemple d'apprentissage de nogoods lorsque l'on souhaite générer des benzénoïdes admettant la symétrie face-mirror (C_2v(a))



Solution générée Nogood appris

FIGURE 3.30. – Exemple d'apprentissage de nogoods lorsque l'on souhaite générer des benzénoïdes admettant la symétrie edge-mirror (C_2v(b))

3.8. Réduire la taille du coronénoïde englobant

Jusqu'à maintenant, il a été convenu que lorsque nous souhaitons obtenir des structures de benzénoïdes dont le nombre d'hexagones est borné par n, nous considérions un coronénoïde englobant de taille $k(n) = \lfloor \frac{n}{2} + 1 \rfloor$ en accord avec la propriété 3.2.2. Néanmoins, les seuls benzénoïdes possédant n hexagones ne pouvant être contenus que dans un coronénoïde de taille minimale $k(n)$ sont les benzénoïdes de diamètre $n-1$ tels que les *acènes* (l'ensemble des benzénoïdes constitués d'hexagones en ligne droite) ou encore des benzénoïdes admettant des *coudées*. La figure 3.31 illustre cela en décrivant trois benzénoïdes de 5 hexagones ne pouvant être contenus que dans au coronénoïde de taille au moins $k(5) = 3$. Ainsi, lorsque l'on a la garantie qu'un ensemble de benzénoïdes d'au plus n hexagones que l'on souhaite générer n'admet aucune molécule de diamètre $n-1$, il n'est pas nécessaire de considérer un coronénoïde englobant de taille

$k(n)$. Il est donc possible de réduire la taille de ce dernier afin de raccourcir l'espace de recherche, et ce, avec la garantie qu'aucune solution ne soit perdue. Par exemple, les benzénoïdes rectangulaires admettent des acènes (l'intégralité de molécules ayant une largeur ou une hauteur égale à 1), il est donc impossible de réduire la taille du coronénoïde englobant dans ce cas-là. En revanche, l'ensemble des benzénoïdes de n hexagones satisfaisant la symétrie (face)-60-rotation n'admet aucun benzénoïde de diamètre $n-1$. Dans ce cas, il est donc possible de réduire la taille du coronénoïde englobant.

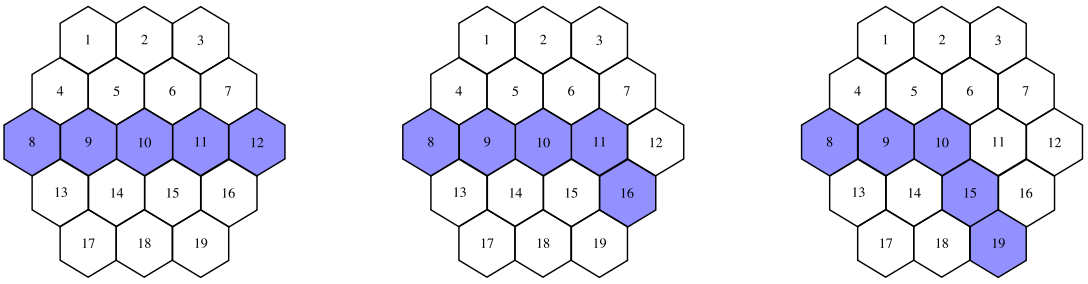


FIGURE 3.31. – Exemples de benzénoïdes de 5 hexagones ne pouvant être contenu que dans un coronénoïde de taille minimale $k(5) = 3$

La table 3.5 détaille toutes les optimisations possibles pour la borne supérieure du nombre d'hexagones. Lorsque deux critères permettant de réduire cette borne sont spécifiés, nous considérons la borne supérieure la plus petite. Par exemple, si on souhaite générer les structures de benzénoïdes ayant au plus 10 hexagones, et qui sont des rectangles de dimension au plus 3×3, alors nous considérons que la borne supérieure du nombre d'hexagones sera 9. Nous notons h_{UB} la borne supérieure optimale du nombre d'hexagones obtenue de cette manière.

Une fois la valeur de h_{UB} calculée, il est possible de réduire la taille du coronénoïde englobant en fonction des critères de génération sélectionnés. La table 3.6 décrit certains des cas de figures possibles. Si aucun des critères mentionnés dans cette table n'est sélectionné, alors on considérera un coronénoïde englobant de taille $k(h_{UB})$. Dans le cas où l'on souhaite générer des structures de benzénoïdes possédant plusieurs des critères décrits dans cette table, on convient de choisir la plus petite borne obtenue. Prenons par exemple la symétrie C_6h (face)-60-rotation pour laquelle le nombre optimisé de couronnes est $\lfloor \frac{h_{UB}+10}{6} \rfloor$. Les benzénoïdes admettant cette symétrie en demandant la plus grande taille du coronénoïde englobant ont une forme d'étoile à 8 branches (la figure 3.32 décrit cela pour un coronénoïde englobant de taille 3). La formule $\lfloor \frac{h_{UB}+10}{6} \rfloor$ décrit le nombre de couronnes du plus petit coronénoïde englobant pouvant contenir ce type de molécule.

critère spécifié	taille du coronénoïde englobant
C_6h (face)-60-rotation D_6h (vertex)-60-rotation+(edge)-mirror	$\lfloor \frac{h_{UB}+10}{6} \rfloor$
C_3h (i) face-120-rotation C_3h (ii) vertex-120-rotation D_3h (ia) face-120-rotation+(edge)-mirror D_3h (ib) face-120-rotation+edge-mirror D_3h (ii) face-120-rotation+(edge)-mirror	$\frac{h_{UB}+4}{3}$
coronoïdes	$\frac{h_{UB}+2-4\times trous_{max}}{4}$

TABLEAU 3.6. – Optimisation de la taille du coronénoïde

critère spécifié	h_{UB}
nombre de carbones	$\left\lceil \frac{carbones_{max}-6}{4}+1 \right\rceil$
nombre d'hydrogènes	$\left\lceil \frac{hydrogenes_{max}-8}{2}+1 \right\rceil$
coronénoïdes	$6\times \frac{couronnes_{max}\times(couronnes_{max}-1)}{2}+1$
rectangles	$hauteur_{max}\times largeur_{max}$

TABLEAU 3.5. – Optimisation de la borne supérieure du nombre d'hexagones

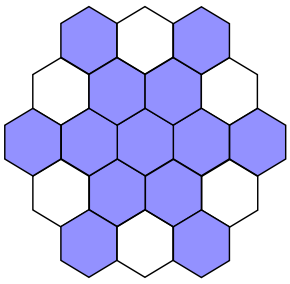


FIGURE 3.32. – Le plus grand benzénoïde admettant la symétrie C_6h (face)-60-rotation pouvant être contenu dans un coronénoïde de taille 3

3.9. Générer des structures de benzénoïdes en pratique

Dans cette section, on se propose de tester la capacité de notre modèle à générer des structures de benzénoïdes admettant différentes propriétés. Notre modèle a été implémenté à l'aide de la librairie Choco Solver (v. 4.10.8), nous avons conservé ses paramètres par défaut. Les expérimentations ont été réalisées sur un Dell PowerEdge M610 avec un processeur Intel Xeon E5620 2,4 GHz et 24GB de RAM. Chaque calcul a été lancé avec un unique thread, une limite de mémoire de 12GB ainsi qu'une limite de temps de 24 heures.

Considérons dans un premier temps notre modèle général $\mathscr{M}$ (défini dans la section 3.3). Nous avons fait varier le nombre d'hexagones entre 1 et 10. La table 3.7 détaille les résultats obtenus en fournissant le nombre de variables et de contraintes du modèle, le nombre de structures de benzénoïdes générées (qui correspond donc au nombre de solutions) ainsi que le temps d'exécution en secondes. La première observation que l'on peut faire est que le nombre de variables et de contraintes sera toujours le même pour deux entiers n et $n+1$ quand n est pair. Cela s'explique, car ces valeurs induiront toutes les deux un coronénoïde englobant de taille $k(n)$ (donc par définition, $k(n) = k(n+1)$ si n est pair). De plus, Choco Solver stocke toutes les clauses dans une unique contrainte qui sera manipulée comme une base de clauses (donc, toutes les clauses liées à l'apprentissage de nogoods et aux différents modèles correspondront à une unique contrainte). Ensuite, on peut également remarquer que résoudre ce problème pour $n+1$ hexagones sera toujours plus couteux que de le résoudre pour n hexagones, même si les deux problèmes associés sont relativement proches en termes de variables et de contraintes. En effet, la principale différence réside dans une base de clauses plus conséquente et d'une contrainte sur le nombre d'hexagones plus relaxée qui requiert le parcours d'un plus grand espace de recherche. On peut remarquer que générer toutes les structures de benzénoïde de 10 hexagones prend un temps conséquent (presque 8 heures). Ce n'est pas forcément un problème, car pour les chimistes, la génération exhaustive de structures de benzénoïdes n'a de sens que pour un nombre d'hexagones au plus égal à 9. Au-delà, le nombre de structures commence à devenir trop important (il est multiplié par environ 4,5 pour chaque hexagone ajouté) pour être exploitable en pratique. De plus, en pratique, ils s'intéressent davantage à la génération de structures satisfaisant des propriétés spécifiques. Comme mentionné plus tôt, le principal avantage de notre approche est sa grande flexibilité qui nous permet facilement d'ajouter/combiner diverses propriétés. Ce procédé va ajouter des variables et des contraintes au modèle et ces dernières contribueront à réduire la taille de l'espace de recherche qui pourra être exploré en un temps raisonnable.

La table 3.8 présente les résultats obtenus quand on génère toutes les structures des benzénoïdes catacondensés. Nous pouvons voir que le simple fait d'ajouter au modèle général des contraintes interdisant la formation de triangles d'hexagones réduit significativement le temps d'exécution. Si nous jetons un coup d'œil à la table 3.9, nous voyons que c'est également le cas pour la génération de coronoïdes. De plus, l'optimisation du nombre de couronnes (voir la section 3.5), nous permet de générer les coronoïdes ayant 12 hexagones en moins de 24 heures, contrairement aux deux modèles cités précédemment pour lesquels on s'était arrêté à 10 hexagones.

Pour finir, nous nous intéressons à la génération de benzénoïdes admettant des symétries. Les tables 3.10, 3.11 et 3.12 nous montrent respectivement les résultats obtenus en générant les benzénoïdes admettant les symétries face-60°-rotation (C_6h), symétrie face-120°-rotation (C_3h(i)) et symétrie vertex-180°-rotation (C_2h(i)). L'optimisation du nombre de couronnes dans le cas des deux premières symétries (voir la sous-section 3.5) combinée aux contraintes imposant la symétrie désirée permet de générer des molécules admettant jusqu'à 17 hexagones en moins de 24h. Par exemple, générer les benzénoïdes de 17 hexagones admettant la symétrie face-60°-rotation (C_6h) requiert uniquement un coronénoïde englobant de taille 4, alors qu'il en faudrait un de taille 9 dans le cas général. Cela est d'autant plus intéressant, car les chimistes sont souvent à la recherche de molécules admettant des symétries (Roy, Berezhnaia, Villa et al. 2020; Bauschlicher, Peeters et Allamandola 2008; Cocchi, Prezzi, Ruini et al. 2014; Bouwman, Linnartz et Tielens 2021).

n	#couronnes	$\|X\|$	$\|C\|$	#solutions	temps
3	2	113	18	3	0,11
4	3	287	48	7	0,19
5	3	287	48	22	0,28
6	4	551	96	81	0,80
7	4	551	96	331	2,15
8	5	905	162	1 436	28,72
9	5	905	162	6 510	427,02
10	6	1 349	246	30 129	28 326,55

TABLEAU 3.7. – La taille du coronénoïde englobant, le nombre de variables et de contraintes, le nombre de benzénoïde générés et le temps d'exécution (en secondes) pour le modèle général avec un nombre n d'hexagones variant de 3 à 10.

n	#couronnes	$\|X\|$	$\|C\|$	#solutions	temps
3	2	113	18	2	0,09
4	3	287	48	5	0,12
5	3	287	48	12	0,16
6	4	551	96	36	0,46
7	4	551	96	118	0,96
8	5	905	162	412	7,95
9	5	905	162	1 492	34,85
10	6	1 349	246	5 587	1 296,86

TABLEAU 3.8. – La taille du coronénoïde englobant, le nombre de variables et de contraintes, le nombre de benzénoïde générés et le temps d'exécution (en secondes) pour le modèle des benzénoïdes catacondensés avec un nombre d'hexagones n variant de 3 à 10.

n	#couronnes	$\|X\|$	$\|C\|$	#solutions	temps
8	3	377	115	1	0,99
9	3	377	115	5	1,05
10	4	731	229	43	81,27
11	4	731	229	283	279,98
12	5	1 214	392	1 954	30 611,14

TABLEAU 3.9. – La taille du coronénoïde englobant, le nombre de variables et de contraintes, le nombre de benzénoïdes générés et le temps d'exécution (en secondes) pour le modèle des coronoïdes avec un nombre d'hexagones n variant de 8 à 12.

n	#couronnes	$\|X\|$	$\|C\|$	#solutions	temps
3	2	111	17	0	0,59
4	2	111	17	0	0,59
5	2	111	17	0	0,60
6	2	111	17	0	0,63
7	2	113	18	1	0,91
8	3	285	47	0	0,86
9	3	285	47	0	0,86
10	3	285	47	0	0,87
11	3	285	47	0	0,88
12	3	287	48	1	0,13
13	3	287	48	2	0,14
14	4	549	95	0	0,13
15	4	549	95	0	0,14
16	4	549	95	0	0,14
17	4	549	95	0	0,14

TABLEAU 3.10. – La taille du coronénoïde englobant, le nombre de variables et de contraintes, le nombre de benzénoïdes générés et le temps d'exécution (en secondes) pour le modèle des benzénoïde symétriques (rotation 60°) pour un nombre d'hexagones variant de 3 à 17.

n	#couronnes	$\|X\|$	$\|C\|$	#solutions	temps
3	2	111	17	0	0,06
4	2	113	18	1	0,10
5	3	285	47	0	0,09
6	3	285	47	0	0,10
7	3	287	48	3	0,15
8	4	549	95	0	0,15
9	4	549	95	0	0,17
10	4	551	96	9	0,24
11	5	903	161	0	0,35
12	5	905	162	2	0,42
13	5	905	162	32	0,65
14	6	1 347	245	0	1,35
15	6	1 349	246	7	1,27
16	6	1 349	246	130	3,10
17	7	1 881	347	0	7,36

TABLEAU 3.11. – La taille du coronénoïde englobant, le nombre de variables et de contraintes, le nombre de benzénoïdes générés et le temps d'exécution (en secondes) pour le modèle des benzénoïdes symétriques (rotation 120°) pour un nombre d'hexagones variant de 3 à 17.

3.10. Conclusions et perspectives

Dans ce chapitre, nous nous sommes penchés sur la résolution du *Problème de Génération de Benzénoïdes (BGP)* qui, selon un ensemble de propriétés (structurelles ou chimiques) $\mathscr{P}$ consiste à générer l'intégralité des benzénoïdes satisfaisant chacune de ces dernières. Nos préoccupations principales étaient de mettre au point une approche flexible qui nous permette de facilement pouvoir ajouter de nouvelles propriétés et de les combiner entre elles. Nous recherchions également une garantie d'exhaustivité sur les ensembles de molécules générés. Pour finir, la méthode mise au point devait être capable de générer les ensembles de structures de benzénoïde le plus efficacement possible. La grande flexibilité de la programmation par

n	#couronnes	$\|X\|$	$\|C\|$	#solutions	temps
3	2	113	18	1	0,10
4	3	285	48	0	0,09
5	3	287	48	3	0,15
6	4	549	95	0	0,16
7	4	551	96	11	0,26
8	5	903	161	0	0,37
9	5	905	162	43	0,69
10	6	1 349	246	1	1,29
11	6	1 349	246	191	3,60
12	7	1 883	348	7	10,22
13	7	1 883	348	885	32,57
14	8	2 507	468	37	107,55
15	8	2 507	468	4203	482,47
16	9	3 221	606	210	1 301,95
17	9	3 221	606	20 253	8 690,37

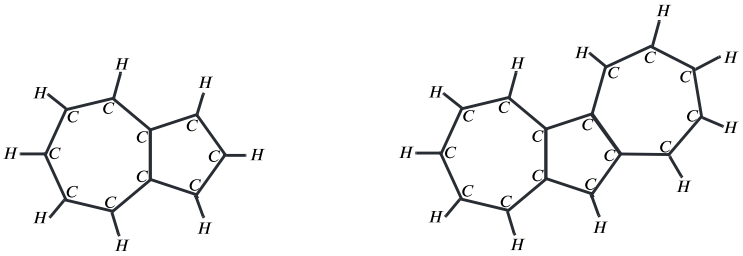
TABLEAU 3.12. – La taille du coronénoïde englobant, le nombre de variables et de contraintes, le nombre de benzénoïdes générés et le temps d'exécution (en secondes) pour le modèle des benzénoïde symétriques (rotation 180°) pour un nombre d'hexagones variant de 3 à 17.

contraintes et sa capacité à résoudre efficacement des problèmes difficiles ont fait d'elle la candidate idéale pour cette tâche.

Ainsi, nous avons présenté un modèle basé sur le formalisme CSP structuré de la manière suivante : tout d'abord, nous considérons un modèle dit *général* capable de générer l'intégralité des benzénoïdes possédant au plus un nombre donné d'hexagones. Ce dernier contient toutes les variables et les contraintes nécessaires à la génération des benzénoïdes. Afin de modéliser de nouvelles propriétés, nous avons mis en place des *modules* qui sont des ensembles de variables et de contraintes que nous pouvons greffer au modèle général, chaque module correspondant à une propriété additionnelle spécifique. En procédant de la sorte, nous avons été capables de modéliser un nombre conséquent de propriétés structurelles (benzénoïdes catacondensés, coronénoïdes, coronoïdes, benzénoïdes rectangulaires ou rhombiques, benzénoïdes admettant des symétries, ...) et chimiques (nombre de carbones/hydrogènes, paramètre d'irrégularité, ...). De plus, l'utilisation des modules permet à l'utilisateur de sélectionner précisément les propriétés qu'il veut appliquer aux benzénoïdes qu'il génère. Cela permet également de pouvoir facilement ajouter de nouvelles propriétés. Ce procédé nous a ainsi permis de fournir une approche flexible, pouvant parfaitement s'adapter aux besoins des chimistes qui l'utiliseront. Globalement, les temps obtenus sont satisfaisants et notre modèle est capable de générer rapidement des molécules intéressantes pour les chimistes. De plus, nous fournissons une approche bien plus flexible que les méthodes de génération déjà existantes (BRINKMANN, CAPOROSSI et HANSEN 2002 ; CAPOROSSI et HANSEN 1998) qui ont pour seul critère le nombre d'hexagones des benzénoïdes générés. De manière générale, nous pouvons dire que l'objectif énoncé initialement a été atteint. Les travaux détaillés dans ce chapitre ont été présentés à la conférence CP2020 (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2020) ainsi que dans le journal Constraints (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2022).

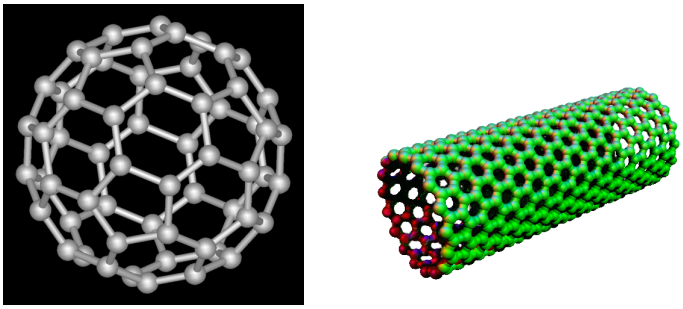
Au niveau des perspectives, la première était d'abord d'être capable de générer des benzénoïdes admettant ou excluant une ou plusieurs formes spécifiques (appelées *motifs*). Les travaux relatifs à cette problématique sont présentés dans le chapitre 4. Une autre perspective pourrait être de s'inspirer des travaux de Matsuoka et al. (MATSUOKA, ITO, SARLAH et al. 2021) pour générer de nouvelles molécules. Ces derniers ont présenté une méthode nommée *growth-from-template* consistant à partir d'une molécule, puis à ajouter successivement un motif donné sur différentes parties des motifs admis par cette dernière. Un autre point intéressant à aborder serait de générer des HAP ne contenant pas uniquement des hexagones. On peut notamment citer les *azulénoïdes* qui sont des HAP admettant exclusivement des cycles de taille 5 ou 7 (voir

figure 3.33). Un autre point intéressant à aborder pourrait être la génération de molécules non-planaires telles que les *fullerènes* (voir la figure 3.34). Néanmoins, résoudre de telles problématiques est loin d'être trivial et pourrait nécessiter de repenser entièrement les modèles que nous avons mis au point.



(a) (b)

FIGURE 3.33. – Deux exemples d'azulénoïdes



(a) (b)

FIGURE 3.34. – Buckminsterfullerène (ou footballène) (a) et un exemple de nanotube (b)

4. Génération de structures de benzénoïdes avec prise en compte de motifs

4.1. Introduction

Nous avons vu dans la section 2.3 que l'étude de la topologie des benzénoïdes avait donné lieu, ces dernières années, à de nombreuses études publiées dans des journaux de chimie réputés. On peut citer notamment (Qiu, Narita et Müllen 2020 ; Liu et Feng 2020 ; Ajayakumar, Ma, Lucotti et al. 2021 ; Chen, Lin, Luo et al. 2021 ; Cheung, Watanabe, Segawa et al. 2021 ; Fujise, Tsurumaki, Wakamatsu et al. 2021 ; Kancherla et Jørgensen 2020 ; Uryu, Hiraga, Koga et al. 2020 ; Xia, Pun, Chen et al. 2021). Il a également été démontré que la forme d'un benzénoïde avait une influence directe sur ses propriétés optoélectroniques (Dumslaff, Gu, Paternò et al. 2020 ; Martin, Hampel et Jux 2020 ; Mori 2021 ; Nathusius, Ejlli, Rominger et al. 2020 ; Qiu, Narita et Müllen 2021 ; Yang, Rominger et Mastalerz 2021) ou magnétiques (Mishra, Yao, Chen et al. 2021 ; Zeng, Wang, Zhang et al. 2021). L'intégralité des publications citées ont été présentées après le début de l'année 2020, ce qui démontre que ce sujet est très étudié en ce moment même. C'est dans ce contexte que nous avons abordé cette problématique.

Dans ce chapitre, nous définissons les *motifs* comme étant une partie d'un benzénoïde possédant une forme spécifique. Ce chapitre est organisé de la manière suivante : premièrement, la section 4.2 décrira de manière formelle toutes les notions liées à ces motifs. Ensuite, nous aborderons les problématiques suivantes et pour chacune d'entre elles, nous présenterons un ou plusieurs modèles tous issus du modèle général $\mathscr{M}$ présenté dans la section 3.3 capable de les résoudre :

- générer les structures de benzénoïdes admettant un motif donné (section 4.3),
- générer les structures de benzénoïdes admettant plusieurs motifs donnés (section 4.4),
- générer les structures de benzénoïdes excluant un motif donné (sous-section 4.5.1),
- générer les structures de benzénoïdes admettant plusieurs fois un même motif (sous-section 4.5.2).

Pour finir, la section 4.6 détaillera les résultats expérimentaux et présentera les modèles les plus performants et la section 4.7 conclura ce chapitre et présentera quelques perspectives futures.

4.2. Définitions préliminaires

Le modèle général $\mathscr{M}$ présenté dans la section 3.3 ainsi que toutes ses extensions présentées dans la section 3.4 représentent uniquement des propriétés structurelles concernant l'ensemble de la molécule. Ces propriétés peuvent être ainsi qualifiées de *globales*. Néanmoins, il peut être utile dans certains cas de raisonner en termes de propriétés locales qui ne peuvent être satisfaites que par certaines parties (appelées fragments) des structures générées. Plus particulièrement, il est important d'être capable de pouvoir traiter les propriétés relatives aux bordures des structures de benzénoïdes.

Dans la section précédente, nous avons défini un motif, de manière générale, comme étant une partie d'un benzénoïde possédant une forme spécifique. Dans la pratique, bien que la forme d'un benzénoïde soit

importante, les chimistes sont davantage intéressés par la bordure des benzénoïdes qu'ils étudient, car cette dernière peut induire des propriétés chimiques intéressantes (DUMSLAFF, GU, PATERNÒ et al. 2020 ; MARTIN, HAMPEL et JUX 2020 ; MORI 2021 ; NATHUSIUS, EJLLI, ROMINGER et al. 2020 ; QIU, NARITA et MÜLLEN 2021 ; YANG, ROMINGER et MASTALERZ 2021 ; MISHRA, YAO, CHEN et al. 2021 ; ZENG, WANG, ZHANG et al. 2021). Ainsi, même si tous les modèles présentés dans ce chapitre raisonnent en termes d'hexagones, ce sont les bordures qui nous intéressent vraiment.

Les propriétés locales que nous considérons dans cette partie peuvent être définies en « dessinant » une forme dont les briques de base sont des hexagones. Ces hexagones peuvent être de trois natures différentes :

(i) Les hexagones *positifs* dont la présence est requise dans la propriété,

(ii) Les hexagones *négatifs* dont l'absence est requise dans la propriété,

(iii) Les hexagones *neutres* dont la présence ou l'absence n'ont aucune influence sur la propriété.

Bien que l'utilité des hexagones positifs soit évidente, on peut se poser la question de l'intérêt des hexagones négatifs et neutres. Les hexagones négatifs (resp. neutres) peuvent par exemple être utilisés pour indiquer qu'il n'y a rien entre deux hexagones positifs, ou encore pour définir la bordure du benzénoïde (resp. pour garantir un certain espace entre deux hexagones positifs). Pour les chimistes, la *bordure* d'un motif est un sous-ensemble des arêtes situées à l'intersection de ses hexagones positifs et négatifs. La figure 4.1 illustre cela en décrivant, en jaune, les bordures des motifs *deep bay* et *zigzag bay*. Nous pouvons voir que la bordure du motif deep bay n'englobe par toutes les arêtes présentes sur l'intersection de ses hexagones positifs et négatifs, alors que c'est le cas pour zigzag bay. Pour pouvoir représenter ces formes, nous introduisons la notion de graphe d'hexagones étendu :

Définition 4.2.1 (graphe d'hexagones étendu) *Un* graphe d'hexagones étendu *est un graphe d'hexagones dont les sommets et les arêtes sont étiquetés par les symboles + (pour les hexagones positifs), − (pour les hexagones négatifs) et ∘ (pour les hexagones neutres) de telle sorte que :*

(i) Chaque sommet est étiqueté avec la nature de l'hexagone qu'il représente.

(ii) Une arête est étiquetée − si au moins un de ses sommets est étiqueté −, ∘ si au moins un des sommets est étiqueté ∘. Sinon, elle sera étiquetée +.

Pour finir, la propriété locale que nous souhaitons définir peut être définie par un *motif* :

Définition 4.2.2 (motif) *Un* motif P *est défini par un triplet* $(P_+, P_-, P_\circ)$ *ainsi que par un graphe d'hexagones étendu* P_h *de telle sorte que :*

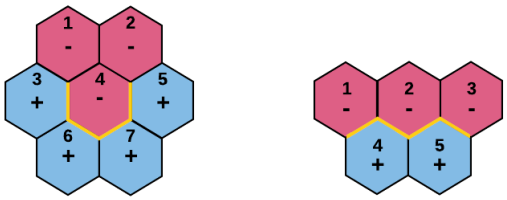
(i) P_+, P_- *et* $P_\circ$ *représentent respectivement les ensembles d'hexagones positifs, négatifs et neutres,*

(ii) Ces trois ensembles sont deux à deux disjoints,

(iii) P_h *est un graphe connexe et ses sommets sont définis par* $P_+ \cup P_- \cup P_\circ$.

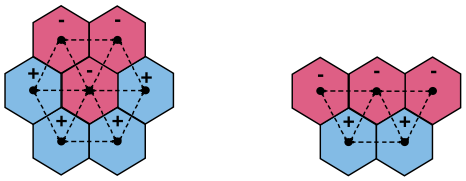
Son ordre k_P *est défini par la longueur maximale (exprimée en nombre d'arêtes) des plus court chemins de* P_h *séparant n'importe quel hexagone négatif ou neutre d'un hexagone positif.*

En d'autres termes, un motif est défini par une collection d'hexagones positifs, négatifs et neutres dont l'arrangement est décrit par un graphe d'hexagones étendu. Au cours de ce chapitre, lorsque nous



(a) (b)

FIGURE 4.1. – Bordures des motifs *deep bay* (a) et *zigzag bay* (b)



(a) (b)

FIGURE 4.2. – Graphes d'hexagones étendus associés aux motifs *deep bay* (a) et *zigzag bay* (b)

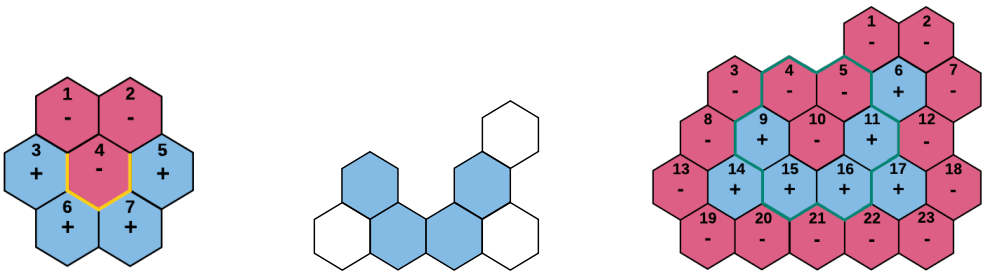
définissons un motif, on se réserve le droit de définir également sa bordure. Pour rappel, la bordure d'un motif est caractérisée par un sous-ensemble de l'intersection entre les hexagones positifs et négatifs du motif. La figure 4.1 illustre cela en décrivant les motifs *deep bay* (WOHNER, LAM et SATTLER 2015) (a) et *zigzag bay* (b), tous les deux d'ordre 1 ainsi que leurs bordures respectives (représentées en vert). Le premier est composé de quatre hexagones positifs et de trois hexagones négatifs tandis que le second contient deux hexagones positifs et trois hexagones négatifs. La figure 4.2 décrit, quant à elle, les graphes d'hexagones étendus associés à ces deux motifs. Pour finir, nous définissons la notion d'inclusion de motif :

Définition 4.2.3 *Étant donné k, un entier positif, B_h^k est le graphe d'hexagones étendu représentant le benzénoïde B entouré par k couches d'hexagones négatifs (c'est-à-dire le graphe d'hexagone de B auquel nous ajoutons tous les hexagones négatifs situés à une distance au plus k d'un hexagone de B). Un* fragment *F^k d'ordre k d'un benzénoïde B est un sous-ensemble d'hexagones de B_h^k dont le graphe d'hexagones étendu est connexe. On dit qu'il* satisfait *le motif P si $k = k_P$ et s'il existe une bijection qui, à chaque hexagone positif de F^k, associe un hexagone positif ou neutre de M et, à chaque hexagone négatif de F^k, associe un hexagone négatif ou neutre de P. Un benzénoïde B* contient *(ou* inclut*) le motif P s'il possède un fragment d'ordre k_P satisfaisant P.*

Le fait de considérer B_h ou B_h^k ne change pas la nature du benzénoïde B. B_h^1 représente simplement le vide autour du benzénoïde, ce qui est nécessaire pour certaines propriétés. Par exemple, le benzénoïde de la figure 4.3(b) ainsi que celui de la figure 4.4 satisfont le motif *deep bay* (représenté dans la figure 4.3(a)). Pour cela, on doit prendre en compte l'absence d'hexagones sur la bordure du benzénoïde afin d'identifier un fragment valide. Cela peut être fait grâce au graphe d'hexagone étendu représenté dans la figure 4.3(c).De plus, nous pouvons facilement voir que si un benzénoïde inclut un motif, alors il satisfait également sa bordure.

Dans cette partie, on se propose de générer des structures de benzénoïdes satisfaisant des propriétés locales pouvant être exprimées grâce aux motifs. On souhaite modéliser les propriétés suivantes :

(i) Inclusion d'un motif donné,



(a) (b) (c)

FIGURE 4.3. – Le motif *deep bay* (a), un benzénoïde incluant ce motif (b) et le benzénoïde « étendu » lié à son graphe d'hexagones étendu B_h^1 avec le motif encadré en vert (c)

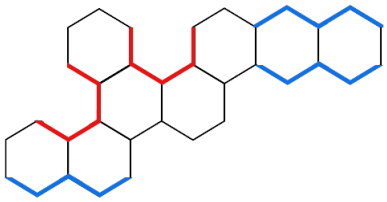


FIGURE 4.4. – Benzénoïde admettant deux occurrences du motif *deep bay* (en rouge) et trois occurrences du motif *zigzag bay* (en bleu)

(ii) Inclusion d'un ensemble de motifs donné,

(iii) Exclusion d'un motif donné,

(iv) Inclusions multiples d'un motif donné.

Pour chacune de ces propriétés, l'idée est de partir du modèle général $\mathscr{M}$ et d'y ajouter différents ensembles de variables et de contraintes. En procédant de la sorte, il est possible de combiner les propriétés globales de $\mathscr{M}$ et celles représentées par les autres extensions présentées dans cette partie avec les propriétés locales liées aux motifs. L'intégralité des motifs qui seront utilisés dans ce chapitre sont décrits dans la figure 4.5 (e-g). Les éléments (h,i) de cette figure décrivent deux des benzénoïdes de 4 hexagones admettant une occurrence du motif *armchair edge* et l'élément (k) représente, quant à lui, un benzénoïde de 4 hexagones excluant ce même motif.

Même si cette partie est à l'origine motivée par les travaux des chimistes sur la forme des bordures, il va au-delà. En effet, ce problème pourrait également être utile pour modéliser certaines des propriétés décrites dans la section 3.4. Par exemple, générer des benzénoïdes admettant les motifs décrits dans la figure 4.6 permettent respectivement de générer les benzénoïdes non-catacondensés (a) et les benzénoïdes admettant des trous constitués de deux hexagones (b). Notons que le motif décrit pour le premier cas n'admet pas de bordure, étant donné qu'il ne possède pas d'hexagones négatifs. De plus, le deuxième cas peut facilement être généralisé à des trous de n'importe quelle taille.

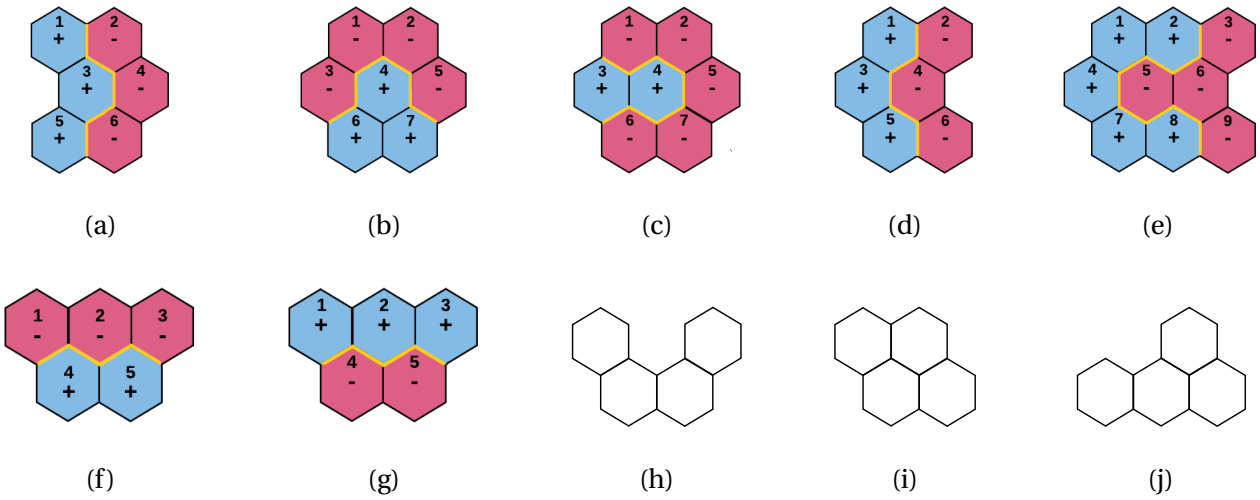


FIGURE 4.5. – Les motifs utilisés en tant que benchmarks ainsi que le motif *deep bay* (appellé *cove* dans (NIU, MA, SOLTANI et al. 2020)) : *armchair edge* (a), C_3H_3 *protusion* (b), C_4H_4 *protusion* (c), *shallow armchair bay* (d), *ultra deep bay* (e), *zigzag bay* (f), renommé en *zigzag* dans (NIU, MA, SOLTANI et al. 2020), et *zigzag edge* (g). Deux des quatre structures de benzénoïdes de quatre hexagones admettant une occurrence du motif *armchair edge* (h) et (i). Une des trois structures de benzénoïde de quatre hexagones ne contenant aucune occurrence du motif *armchair edge* (j).

4.3. Génération de structures incluant un motif donné

Soit P un motif possédant n_P hexagones (pouvant être soit positifs, soit négatifs, soit neutres). Nous numérotons arbitrairement chaque hexagone de P par une valeur entre 1 et n_P. Les ensembles d'hexagones P_+, P_- et $P_\circ$ sont définis en accord avec la section précédente.

Dans cette section, on se propose d'essayer de résoudre le problème consistant à générer l'ensemble complet des benzénoïdes possédant au plus n hexagones et incluant le motif P. Trois modèles capables de résoudre ce problème seront présentés dans cette partie. Pour chacun d'entre eux, nous partirons du modèle général $\mathscr{M}$ présenté dans la partie 3.3, puis nous y ajouterons différents ensembles de variables et de contraintes dans le but de représenter les propriétés additionnelles désirées. L'idée générale du premier modèle consistera à identifier toutes les positions possibles d'un fragment satisfaisant le motif P. Le second modèle considérera l'existence d'un fragment en analysant les configurations de voisinage de chaque hexagone. Pour finir, l'idée générale du troisième modèle sera de représenter un problème d'isomorphisme de sous-graphe.

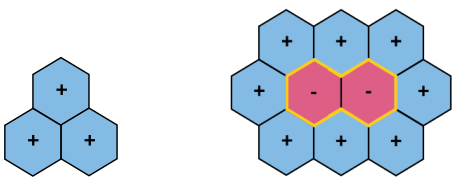


FIGURE 4.6. – Motifs dont l'inclusion permet de générer les benzénoïdes non catacondensés (a) et les coronoïdes possédant des trous de deux hexagones (b).

4.3.1. Premier modèle

Comme mentionné ci-dessus, nous partons du modèle général $\mathscr{M}$ avec un coronénoïde de taille $k(n)$. Dans ce premier modèle (noté $\mathscr{M}_{i_1}$), on se propose dans un premier temps d'identifier tous les fragments possibles du motif P dans ce coronénoïde. Leur nombre étant en $O(|c(k(n))|) = O(n^2)$, cette identification peut être faite de manière efficace en appliquant successivement des rotations, ainsi que des symétries axiales et des translations. Ensuite, pour chacun de ces fragments F_i, nous définissons les ensembles d'hexagones F_{i+}, $F_{i\circ}$ et F_{i-} qui correspondent respectivement à ses hexagones positifs, négatifs et neutres. La figure 4.7 nous montre trois fragments parmi l'ensemble des fragments possibles du motif deep bay au sein d'un coronénoïde englobant de taille 3. Nous associons ensuite à chaque fragment F_i une variable booléenne e_i de telle sorte que le fragment F_i soit présent dans la solution si e_i est vraie. Cette propriété peut être garantie par la contrainte suivante :

$$e_i = 1 \Rightarrow \bigwedge_{j \in F_{i-}} x_j = 0 \wedge \bigwedge_{j \in F_{i+}} x_j = 1$$

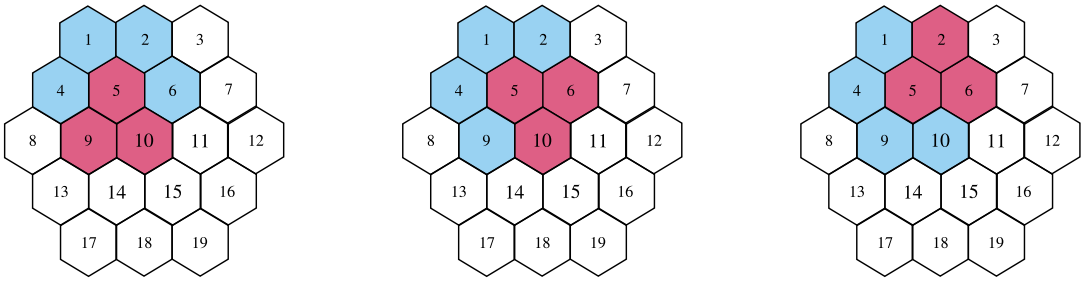


FIGURE 4.7. – Trois des fragments du motif deep bay dans un coronénoïde de taille 3. Les hexagones de F_{i+} (resp. F_{i-}) sont représentés en bleu (resp. en rouge)

Notons que pour les motifs d'ordre strictement positif, il n'est pas nécessaire de considérer un coronénoïde de taille supérieure. En effet, on s'autorise à placer des hexagones négatifs ou neutres à l'extérieur du coronénoïde. Par conséquent, ces hexagones seront considérés comme absents. Dans ce cas, les hexagones placés de cette manière ne seront pas placés dans F_{i-} ni dans $F_{i\circ}$, mais dans un nouvel ensemble F_{i*}. Pour finir, nous appliquons la contrainte de somme suivante pour garantir l'existence d'au moins un fragment satisfaisant le motif P :

$$\sum_j e_j = 1$$

Ici, nous avons choisi d'appliquer la contrainte $\sum_j e_j = 1$ et non $\sum_j e_j \geq 1$ qui semblerait, au premier abord, plus naturelle. Imposer l'égalité nous permet, une fois une des variables e_j affectée à 1, de fixer automatiquement toutes les autres à la valeur 0. Par ailleurs, pour une structure de benzénoïde donnée (c'est-à-dire pour une affectation donnée des variables x_i), autoriser plusieurs variables e_j à prendre la valeur 1 ne produirait pas de nouvelles structures, et conduirait à énumérer inutilement différentes valeurs possibles pour les e_j. Imposer l'égalité permet donc un meilleur filtrage et améliore ainsi le temps de résolution de l'instance.

4.3.2. Deuxième modèle

Dans ce second modèle (noté $\mathscr{M}_{i_2}$), nous souhaitons exprimer l'existence d'un fragment correspondant à un motif P en considérant le voisinage de chaque hexagone. Pour faire cela, nous partons du modèle général

$\mathscr{M}$ (nous considérons un coronénoïde de taille $k(n)$) et nous ajoutons une variable entière f_i pour chaque hexagone du coronénoïde de taille $k(n)$. Chaque variable f_i possède le domaine $\{0, 1, \ldots, n_P\}$ et prendra la valeur positive j si l'hexagone i participe au fragment recherché comme étant le j-ème hexagone de P, et 0 s'il ne participe pas au fragment. En d'autres termes, la variable f_i spécifie si l'hexagone i participe à un fragment, et si oui, à quel hexagone de P il correspond.

Maintenant, la problématique suivante est de s'assurer que les hexagones positifs (resp. négatifs) soient présents (resp. absents) dans la structure générée. Pour rappel, la structure est représentée par la variable de graphe x_G et les variables booléennes x_i. Pour chaque hexagone i du coronénoïde, nous ajoutons les contraintes suivantes :

1. $x_i = 1 \Rightarrow f_i \in P_+ \cup P_\circ \cup \{0\}$ (Si l'hexagone i est présent dans x_G, alors, soit il participe au fragment en tant qu'hexagone positif ou neutre, soit il ne participe pas au fragment),

2. $x_i = 0 \Rightarrow f_i \in P_- \cup P_\circ \cup \{0\}$ (si l'hexagone i n'est pas présent dans x_G, alors, soit il participe au fragment en tant qu'hexagone négatif ou neutre, soit il ne participe pas au fragment),

3. $f_i \in P_+ \Rightarrow x_i = 1$ (si l'hexagone i participe au fragment en tant qu'hexagone positif, alors il doit être présent dans x_G),

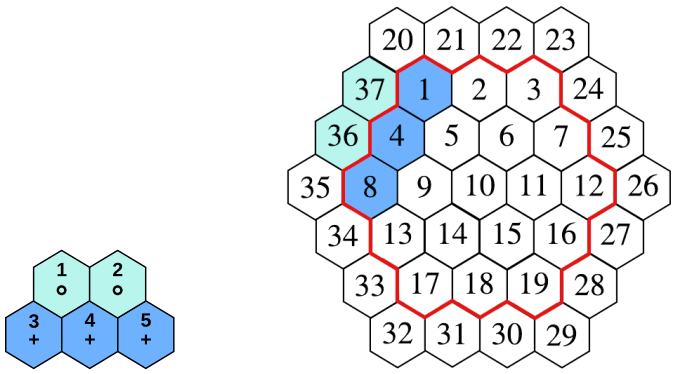
4. $f_i \in P_- \Rightarrow x_i = 0$ (si l'hexagone i participe au fragment en tant qu'hexagone négatif, alors il doit être absent de x_G.

Veuillez noter qu'en pratique, seule la moitié des conditions est nécessaire pour que la modélisation soit valide. Ainsi, une implémentation de ce modèle pourrait ne contenir que les conditions 1 et 2 (ou 3 et 4).

Ensuite, on doit définir la bijection établissant que le fragment construit satisfait le motif P. En d'autres termes, on doit garantir qu'exactement n_P hexagones de la structure doivent correspondre aux n_P hexagones du motif P. Ainsi, pour chaque hexagone j du motif, on ajoute une contrainte globale $\texttt{Count}(\{f_1, \ldots, f_{n_c}\}, \{j\}) = 1$ si $j \in P_+$ (≤ 1 sinon). La valeur 0 est obtenue dans le cas où un hexagone négatif ou neutre est placé à l'extérieur du coronénoïde de taille $k(n)$. Le fait de permettre aux hexagones négatifs et neutres de déborder du coronénoïde nous permet d'éviter d'ajouter au modèle des variables additionnelles (et donc de nouvelles contraintes) pour représenter les k_P couches d'hexagones absents mentionnés dans la définition d'un fragment (voir la définition 4.2.3).

La dernière étape consiste à définir le motif. Pour cela, nous considérons les liens de voisinage entre chaque hexagone du motif. Un hexagone peut posséder jusqu'à 6 hexagones voisins. Pour un hexagone h donné, nous considérons ses voisins potentiels notés $\{v_1, ..., v_6\}$ (les voisins sont numérotés dans le sens horaire, en démarrant par celui situé en haut à droite). Ensuite, nous listons les différentes configurations de voisinage de chaque hexagone h du motif en fonction de s'il participe ou non au fragment. Chaque configuration est définie par un tuple de 6 valeurs entières (une pour chaque voisin de h, la i-ème valeur du tuple correspondant au voisin v_i). Chacune de ces valeurs est un entier j différent de 0 si le voisin associé participe au fragment en tant qu'hexagone j du motif P, 0 sinon. Pour chaque position de l'hexagone h dans le motif P, nous considérons 6 différentes configurations possibles dans le but de prendre en compte toutes les rotations de 60° du motif. Pour information, calculer les rotations de 60° d'une configuration équivaut à effectuer des permutations circulaires successives du tuple associé.

Par exemple, dans la table 4.1, nous listons toutes les configurations de voisinage possibles quand l'hexagone est à la première position du motif *deep bay* (on considère la numérotation décrite dans la figure 3.2). Par soucis d'encombrement, nous ne donnons qu'une configuration pour les autres positions. Pour définir le motif, nous allons utiliser ces relations afin de définir une contrainte de table (VERHAEGHE, LECOUTRE et SCHAUS 2017). Nous considérons une contrainte de table pour chaque hexagone h du coronénoïde de



(a) (b)

FIGURE 4.8. – Un motif de 5 hexagones (a) et un de ses fragments inclus dans le coronénoïde de taille 3 auquel nous rajoutons une couronne (b)

TABLEAU 4.1. – La contrainte de table compacte décrivant le voisinage pour le motif *deep bay*.

f_i	f_{v_1}	f_{v_2}	f_{v_3}	f_{v_4}	f_{v_5}	f_{v_6}
0	*	*	*	*	*	*
1	0	2	4	3	0	0
1	2	4	3	0	0	0
1	4	3	0	0	0	2
1	3	0	0	0	2	4
1	0	0	0	2	4	3
1	0	0	2	4	3	0
2	0	0	5	4	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮

f_i	f_{v_1}	f_{v_2}	f_{v_3}	f_{v_4}	f_{v_5}	f_{v_6}
3	1	4	6	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
4	2	5	7	6	3	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
5	0	0	0	7	4	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮
6	4	7	0	0	0	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	5	0	0	0	6	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮

taille $k(n)$ dont la portée englobe f_h ainsi que toutes les variables f_i associées à un voisin de h dans $B_h^{c(k(n))}$. Pour les hexagones se situant en bordure du coronénoïde, on ne considère uniquement que les lignes de la table dont les voisins participant au fragment correspondent à des hexagones appartenant au coronénoïde (peu importe leur nature) ou à des hexagones négatifs ou neutres placés à l'extérieur du coronénoïde. Pour finir, nous effectuons une projection de ces lignes sur les voisins présents de la variable f_h. Par exemple, considérons le motif d'ordre 1 décrit par la figure 4.8 (a) et un fragment de ce motif dans un coronénoïde englobant de taille 3, auquel nous avons ajouté une couronne supplémentaire d'hexagones externes. Ici, nous considérons uniquement les lignes de la table où les variables f_1, f_4 et f_8 valent respectivement 3, 4 et 5.

4.3.3. Troisième modèle

Dans cette section, nous décrivons un modèle qui est basé sur le problème d'*isomorphisme de sous-graphe*. Avant de détailler ce dernier, il est nécessaire d'introduire cette notion :

Définition 4.3.1 (Isomorphisme de graphes) *Soient G et H deux graphes. Un* isomorphisme *f entre G et H est une bijection entre les sommets de ces deux graphes de telle sorte qu'une paire de sommets $\{u, v\}$ est une arête de G si et seulement si $\{f(u), f(v)\}$ est une arête de H. On dit que deux graphes sont* isomorphes *s'il existe un isomorphisme entre ces derniers.*

Nous définissons ensuite le *problème d'isomorphisme de sous-graphe* :

Définition 4.3.2 (Problème d'isomorphisme de sous-graphe) *Le* problème d'isomorphisme de sous-graphe *est un problème de décision se définissant de la manière suivante :*

***entrée** : deux graphes G et H*
***question** : existe-t-il un sous-graphe G_0 de G qui soit isomorphe à H ?*

Un fragment d'ordre k d'un benzénoïde B correspond à un sous-graphe connexe de B_h^k. De ce fait, tester s'il existe un fragment satisfaisant un motif P dans un benzénoïde B est, en quelque sorte, la même chose que de déterminer s'il existe un sous-graphe isomorphe à P_h dans $B_h^{k_P}$. Cependant, ce n'est pas exactement le classique problème d'isomorphisme de sous-graphe, mais une de ses variantes qui considère un étiquetage sur les sommets et les arêtes. Regardons à nouveau la figure 4.3 qui, pour rappel, décrit le motif *deep bay* (a), un benzénoïde l'incluant (b) ainsi que le benzénoïde lié à son graphe d'hexagones étendu B_h^1 (c). Nous considérons ensuite la bijection f entre le graphe d'hexagone étendu du motif deep bay et le sous-graphe induit par les sommets 4, 5, 9, 10, 11, 15 et 16 (ainsi que toutes les arêtes liant ce sommet) du graphe d'hexagones étendu du benzénoïde incluant ce motif (voir la figure 4.3 (c)) :

$$\begin{cases} f(1) = 4 \\ f(2) = 5 \\ f(3) = 9 \\ f(4) = 10 \\ f(5) = 11 \\ f(6) = 15 \\ f(7) = 16 \end{cases}$$

Dans ce cas, il est facile de voir que f est un isomorphisme entre ces deux graphes et donc que le benzénoïde en question inclut le motif deep bay.

Notre modèle $\mathscr{M}_{i_3}$ est le suivant. Encore une fois, nous partons du modèle général $\mathscr{M}$ auquel nous ajoutons une variable s_i pour chaque hexagone du motif P (peu importe sa nature). Chaque variable s_i aura pour domaine $\{1,\ldots,n'_c\}$ avec n'_c le nombre d'hexagones du coronénoïde de taille $k(n)+k_P$. Nous considérons un coronénoïde de cette taille au lieu de celui de taille $k(n)$ car nous avons besoin d'entourer le coronénoïde de taille $k(n)$ avec k_P couronnes d'hexagones absents. Notons que cela n'a aucun impact sur la variable de graphe x_G ou sur les variables x_i car nous ajoutons uniquement des hexagones que nous savons absents de la structure. La variable s_i aura pour valeur j si le i-ème hexagone du motif P est lié au j-ème hexagone du coronénoïde de taille $k(n)+k_P$. Par convention, on considère que les valeurs de j entre 1 et n_c correspondent aux hexagones présents dans le coronénoïde de taille $k(n)$. Nous ajoutons ensuite les contraintes suivantes pour exprimer la notion d'isomorphisme :

- *Injectivité* : Les hexagones participant au fragment doivent être deux à deux différents. Cette propriété est spécifiée grâce à la contrainte globale `allDifferent`$(\{s_1,\ldots,s_{n_P}\})$. Cette contrainte s'assure également que n_P hexagones de x_G participent au fragment.

- *Préservation des arêtes* : Nous devons garantir que deux sommets voisins de P_h soient obligatoirement associés à deux sommets voisins du coronénoïde de taille $k(n)$. Ainsi, pour chaque arête $\{i,i'\}$ de P_h (peu importe sa nature), nous posons une contrainte de table sur s_i et $s_{i'}$ dont la relation contient toutes les paires (j,j') telles que $\{j,j'\}$ est une arête du graphe d'hexagone du coronénoïde de taille $k(n)+k_P$.

Cette partie du modèle est inspirée du modèle du problème d'isomorphisme de sous-graphe présenté dans (Lecoutre et Roussel 2018). Cependant, il faut noter que, dans notre cas, le graphe dans lequel le sous-graphe est recherché n'est pas connu à l'avance étant donné que nous sommes en train de le construire. De ce fait, nous considérons à la place le graphe d'hexagones du coronénoïde englobant de taille $k(n)+K_P$. Les étiquettes des arêtes sont déterminées en fonction de celles des sommets qui eux correspondent aux variables s_i.

Maintenant, il ne reste plus qu'à exprimer l'adéquation entre les étiquettes des sommets et l'existence des hexagones à l'aide des contraintes suivantes :

1. $\forall i \in P_+, \forall j \in \{1,\dots,n_c\}, s_i = j \Rightarrow x_j = 1$ (si un hexagone positif i de P est lié à l'hexagone j de x_G, alors j doit être présent dans x_G),

2. $\forall i \in P_+, \forall j \in \{1,\dots,n_c\}, x_j = 0 \Rightarrow s_i \neq j$ (si l'hexagone j de x_G est absent, il ne peut pas être lié à un hexagone positif i de P),

3. $\forall i \in P_-, \forall j \in \{1,\dots,n_c\}, s_i = j \Rightarrow x_j = 0$ (si l'hexagone négatif i de P est lié à un hexagone j de x_G, alors j doit être absent dans x_G), et

4. $\forall i \in P_-, \forall j \in \{1,\dots,n_c\}, x_j = 1 \Rightarrow s_i \neq j$ (si un hexagone j de x_G est présent, alors il ne peut pas être lié à un hexagone négatif i de P).

Veuillez noter qu'en pratique, seule la moitié de ces conditions est nécessaire. Ainsi, une implémentation de ce modèle peut ne contenir que les contraintes issues des conditions 1 et 3 (ou 2 et 4).

Pour finir, nous aborderons les limites de cette approche du point de vue de la chimie. La figure 4.9(a)-(b) nous montre deux motifs différents, tous les deux constitués de trois hexagones positifs et dont les graphes d'hexagones sont isomorphes. On voit clairement que les deux molécules associées à ces motifs ne possèdent pas les mêmes propriétés chimiques. Cependant, si on demande à Choco de produire les structures incluant séparément ces deux motifs (en utilisant le modèle $\mathscr{M}_{i_3}$), nous obtiendrions les mêmes solutions. Pour régler ce problème, nous ajoutons une étape de pré-traitement avant de générer l'instance à résoudre. Cette étape consiste à ajouter des hexagones neutres au motif de telle sorte à ce que chaque arête de son graphe d'hexagone apparaisse dans au moins un triangle (c'est-à-dire une clique de taille 3), un triangle représentant trois hexagones deux à deux adjacents. Appliquer cette méthode permet de différencier les deux motifs mentionnés ci-dessus. Dans notre cas, nous avons simplement utilisé un algorithme glouton qui ajoute des hexagones neutres au motif initial jusqu'à ce que chaque arête de ce dernier participe à au moins un triangle. Les figures 4.9(c)-(d) présentent les motifs obtenus une fois que l'on a appliqué cette méthode aux motifs décrits dans les figures 4.9(a)-(b). Pour finir, notons que ce pré-traitement n'est pas toujours nécessaire. Par exemple, il ne modifierait pas le motif deep bay car chaque arête de son graphe d'hexagone est déjà impliquée dans au moins un triangle.

FIGURE 4.9. – Les limites du raisonnement en termes d'isomorphismes de sous-graphe avec deux motifs différents (a) et (b) ayant des graphes d'hexagones isomorphes et les motifs obtenus après application du pré-traitement (c)-(d).

4.4. Génération de structures incluant plusieurs motifs

Dans cette section, nous nous intéressons à la génération de structures incluant plusieurs motifs simultanément. Soit $E_P = \{P^1,\dots,P^\ell\}$ l'ensemble de ces motifs. La considération de plusieurs motifs soulève la question suivante : comment deux motifs peuvent interagir entre eux? Nous listons ici quatre de ces

interactions qui sont logiques d'un point de vue chimie :

(1) Deux motifs peuvent partager n'importe quel type d'hexagones. Cette interaction peut avoir du sens quand l'on souhaite, par exemple, générer des benzénoïdes à la fois non catacondensés et admettant des trous constitués de deux hexagones (obtenables en demandant l'inclusion des deux motifs décrits dans la figure 4.6.

(2) Deux motifs ne peuvent partager que des hexagones absents (c'est-à-dire qu'ils ne peuvent partager que du vide). Cette interaction permet d'éviter les chevauchements entre deux hexagones positifs.

(3) Deux motifs sont entièrement disjoints. Cette interaction permet d'exclure complètement les chevauchements.

(4) Deux motifs peuvent partager des hexagones, mais aucune arête appartenant à leurs bordures respectives. Cette interaction constitue une alternative moins stricte que la précédente. Pour que celle-ci puisse être appliquée, la bordure de chaque motif considérée doit être spécifiée.

Une première approche naïve pour résoudre ce problème de motifs multiples serait de résoudre une collection de problèmes de motifs simples. Cela demanderait d'énumérer tous les motifs simples pouvant être construits en combinant tous les motifs de E_P. Cependant, étant donné la forte combinatoire, il n'est pas question de considérer cette approche. Par conséquent, on se propose d'adapter les modèles que nous avons présentés dans la partie précédente.

4.4.1. Premier modèle

Ce modèle (noté $\mathscr{M}^1_{m_1}$) est une adaptation du modèle $\mathscr{M}_{i_1}$ décrit dans la sous-section 4.3.1 prenant en compte plusieurs motifs au lieu d'un seul. Pour chaque motif $P^j \in E_P$, nous notons $F^j = \{F^j_1, \ldots, F^j_{n_{F^j}}\}$ l'ensemble des fragments qu'il admet au sein du coronénoïde englobant (n_{F^j} correspondant au nombre de fragments admis). Le fragment F^j_i sera représenté par un triplet $(F^j_{i+}, F^j_{i-}, F^j_{i\circ})$ dont les composantes représentent respectivement ses hexagones positifs, négatifs et neutres. Nous associons ensuite à chaque fragment F^j_i une variable booléenne e^j_i de sorte que le fragment soit présent dans la solution uniquement si la variable est vraie. Cette propriété peut être garantie par la contrainte suivante :

$$e^j_i = 1 \implies \bigwedge_{k \in F^j_{i-}} x_k = 0 \wedge \bigwedge_{k \in F^j_{i+}} x_k = 1$$

De la même manière que pour $\mathscr{M}_{i_1}$, il n'est pas nécessaire de considérer un coronénoïde englobant de taille supérieure dans le cas où l'on traite des motifs d'ordre strictement positif, car on s'autorise à placer des hexagones négatifs ou neutre à l'extérieur du coronénoïde. Pour finir, pour chaque motif $P^j \in E_P$, nous appliquons une contrainte de somme afin de garantir l'existant d'au moins un fragment satisfaisant le motif P^j :

$$\sum_{i \leq n_{F^j}} e^j_i = 1$$

Intéressons-nous maintenant au cas où l'on autorise le fait que les motifs puissent partager du vide, c'est-à-dire uniquement des hexagones absents (modèle $\mathscr{M}^2_{m_1}$). Dans ce cas, nous ajoutons au modèle $\mathscr{M}^1_{m_1}$ des contraintes de la forme $e^j_i = 0 \vee e^{j'}_{i'} = 0$ pour tout couple de fragments $\{F^j_i, F^{j'}_{i'}\}$ partageant un hexagone positif (c'est-à-dire si $(F^j_{i+} \cap F^{j'}_{i'+}) \cup (F^j_{i+} \cap F^{j'}_{i'\circ}) \cup (F^j_{i\circ} \cap F^{j'}_{i'+}) \neq \emptyset$). Sinon, s'ils partagent des hexagones neutres, ces derniers devront être absents de la structure. Cela peut être assuré en considérant la contrainte

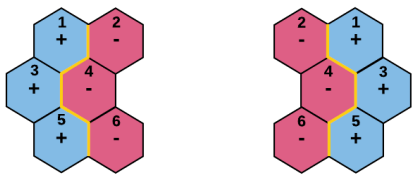
$(e_i^j = 1 \wedge e_{i'}^{j'} = 1) \Rightarrow x_h = 0$ pour chaque hexagone $h \in F_{i'\circ}^j \cap F_{i'\circ}^{j'}$.

Si l'on souhaite obtenir des motifs deux à deux disjoints (modèle $\mathscr{M}_{m_1}^3$), nous devons ajouter à $\mathscr{M}_{m_1}^1$ les clauses d'exclusion mutuelle $e_i^j = 0 \vee e_{i'}^{j'} = 0$ pour chaque couple de fragments $\{F_i^j, F_{i'}^{j'}\}$ se chevauchant (c'est-à-dire tous les fragments tels que $(F_{i+}^j \cup F_{i-}^j \cup F_{i\circ}^j \cup F_{i*}^j) \cap (F_{i'+}^{j'} \cup F_{i'-}^{j'} \cup F_{i'\circ}^{j'} \cup F_{i'*}^{j'})$).

Pour finir, si l'on souhaite obtenir des motifs pouvant partager des hexagones, mais aucune arête de leurs bordures respectives (modèle $\mathscr{M}_{m_1}^4$), alors il suffit d'appliquer les clauses d'exclusion mutuelle uniquement quand les deux fragments considérés partagent des arêtes de leurs bordures respectives.

4.4.2. Second modèle

Ce modèle (noté $\mathscr{M}_{m_2}^1$) est une généralisation du modèle $\mathscr{M}_{i_2}$ décrit dans la sous-section 4.3.2 capable de considérer plusieurs motifs au lieu d'un seul. Pour chaque motif P^j de E_P, nous considérons un ensemble de variables f_i^j équivalentes aux variables f_i de $\mathscr{M}_{i_2}$. Bien évidemment, nous considérons également les contraintes associées à ces variables. Cependant, pour chaque contrainte de table utilisée pour définir le motif P^j, nous introduisons une variable booléenne t^j dans sa portée. Cette variable vaudra 1 si la configuration associée est obtenue après avoir appliqué une symétrie axiale sur P^j, 0 sinon. Cette table listera donc toutes les configurations valides du motif P^j ou de son image par symétrie axiale. La prise en compte des symétries axiales dans le cas de l'inclusion de plusieurs motifs est nécessaire pour pouvoir lister toutes les manières possibles de combiner plusieurs motifs entre eux. Il existe plusieurs axes de symétrie possibles. Cependant, il n'est nécessaire que d'en considérer un seul étant donné que les autres peuvent être obtenus en appliquant des rotations de 60°. L'utilisation de la variable t^j pour chaque contrainte de table définissant P^j garantit que P^j sera considéré si t^j vaut 0. Dans le cas contraire, c'est son image par symétrie axiale qui sera considérée. Ce procédé empêche la formation de fragments erronés. En effet, sans cette variable, il serait possible de considérer une partie de P^j et une autre partie de son image par symétrie axiale. La figure 4.10 décrit le motif *shallow armchair bay* (a) ainsi que son image par symétrie axiale (b). Notons que cette variable n'est pas utile dans le cas de $\mathscr{M}_{i_2}$, car elle entraînerait la génération de structures équivalentes. Le modèle que nous venons de décrire correspond au premier cas de figure d'interaction entre motifs (c'est-à-dire que deux motifs peuvent partager n'importe quel type d'hexagones).



(a) (b)

FIGURE 4.10. – Le motif *shallow armchair bay* (a) et son image par symétrie axiale (b)

Pour traiter le second cas de figure qui autorise uniquement deux motifs à partager du vide (c'est-à-dire des hexagones absents), nous ajoutons au modèle $\mathscr{M}_{m_2}^1$ décrit ci-dessus des contraintes d'exclusion mutuelle pour les hexagones présents. Pour chaque hexagone h du coronénoïde englobant, nous ajoutons la contrainte suivante :

$$x_h = 1 \Rightarrow \texttt{Count}(\{f_h^1, \ldots, f_h^\ell\}, \{1 \ldots, n_{E_P}\}) \leq 1 \text{ avec } n_{E_P} = \max_{P^j \in E_P} n_{P^j}$$

En d'autres termes, si l'hexagone h est présent dans la structure générée, alors il doit participer à au plus un fragment. Ce nouveau modèle sera dénoté $\mathscr{M}_{m_2}^2$.

Ensuite, on se propose de traiter le troisième cas de figure où deux motifs doivent être entièrement disjoints. Ici, nous devons considérer le fait que des hexagones peuvent être partagés à l'extérieur du coronénoïde englobant. Pour cela, nous définissons l'ordre k_{E_P} de l'ensemble E_P comme étant l'ordre maximal d'un motif P^j de E_P. Ensuite, nous considérons le modèle $\mathscr{M}^1_{m_2}$, mais avec un coronénoïde englobant ayant k_{E_P} couronnes de plus (nous avons donc un coronénoïde englobant de taille $k(n)+k_{E_P}$). En d'autres termes, nous ajoutons au modèle $\mathscr{M}^1_{m_2}$ une variable f^j_i pour chaque hexagone situé à l'extérieur du coronénoïde de taille $k(n)$ pour chaque motif $P_j \in E_P$. Étant donné que tous les hexagones sont explicitement représentés, les contraintes de table doivent prendre en compte ces nouvelles variables. De plus, les contraintes `Count` de $\mathscr{M}^1_{m_2}$ pour les hexagones négatifs ou neutres j' du motif P^j sont maintenant de la forme $\texttt{Count}(\{f^j_1,\dots,f^j_{n'_c}\},\{j'\})=1$. Pour finir, nous ajoutons une contrainte d'exclusion mutuelle $\texttt{Count}(\{f^1_h,\dots,f^\ell_h\},\{1\dots,n_{E_P}\})=1$ pour chaque hexagone h du coronénoïde de taille $k(n)+k_{E_P}$. Ce nouveau modèle sera noté $\mathscr{M}^3_{m_2}$.

Pour finir, si l'on souhaite que les bordures respectives des motifs ne se chevauchent pas, alors pour tout hexagone h du coronénoïde englobant et tout couple d'hexagones i et i' appartenant respectivement aux motifs P^j et $P^{j'}$ partageant une arête, nous appliquons la contrainte suivante :

$$x_h = 1 \Rightarrow f^j_h \neq i \vee f^{j'}_h \neq i'$$

Ce nouveau modèle sera noté $\mathscr{M}^4_{m_2}$.

4.4.3. Troisième modèle

Le principe est le même que pour les deux modèles précédents. Pour chaque motif P^j de E_P, nous ajoutons au modèle $\mathscr{M}$ un ensemble de variables s^j_i équivalent aux variables s_i utilisées dans le modèle $\mathscr{M}_{i_3}$ (décrit dans la sous-section 4.3.3). Bien évidemment, nous considérons aussi toutes les contraintes associées à ces variables. En procédant de la sorte, nous obtenons un modèle noté $\mathscr{M}^1_{m_3}$ correspondant au premier cas de figure d'interaction entre deux motifs (deux motifs peuvent partager n'importe quel type d'hexagones). Étant donné que le modèle $\mathscr{M}_{i_3}$ dépend de l'ordre du motif considéré, nous devons utiliser un coronénoïde de taille $k(n)+k_{E_P}$. Nous devons également appliquer à chaque motif le même traitement que le motif P dans le modèle $\mathscr{M}_{i_3}$ afin d'éviter toute ambiguïté.

Nous pouvons ensuite étendre ce modèle au modèle $\mathscr{M}^3_{m_3}$ dans le but de traiter le cas de figure où deux motifs doivent être entièrement disjoints. Pour cela, nous ajoutons la contrainte d'exclusion mutuelle $\texttt{allDifferent}(\{s^1_1,\dots,s^1_{n_{p1}}\}\cup\dots\cup\{s^\ell_1,\dots,s^\ell_{n_{p\ell}}\})$.

Ensuite, le modèle $\mathscr{M}^2_{m_3}$ correspondra au second cas de figure. Il est obtenu en partant du modèle $\mathscr{M}^1_{m_3}$ et en y ajoutant les contraintes suivantes :

- *Les hexagones positifs sont disjoints deux à deux dans x_G* :

$$\texttt{allDifferent}(\{s^j_i \mid j\in\{1,\dots,\ell\}, i\in P^j_+\})$$

- *Si deux hexagones neutres désignent le même hexagone de x_G, alors le sommet correspondant doit être absent de x_G* :

$$\forall j,j'\in\{1,\dots,\ell\}, j<j', \forall i\in P^j_\circ, \forall i'\in P^{j'}_\circ, s^j_i = s^{j'}_{i'} \Rightarrow s^j_i > n_c \vee \texttt{Element}(\{x_z \mid z\in\{1,\dots,n_c\}\}, s^j_i)=0)$$

- *Un hexagone du coronénoïde englobant ne doit pas être lié à un hexagone neutre et un hexagone positif de deux motifs différents* :

$$\forall j, j' \in \{1, \dots \ell\}, j \neq j', \forall i \in P_\circ^j, \forall i' \in P_+^{j'}, s_i^j \neq s_{i'}^{j'}$$

Pour le quatrième cas de figure, pour chaque couple P^j et $P^{j'}$ partageant une bordure au niveau des hexagones i et i' respectivement, nous posons la contrainte suivante :

$$s_i^j \neq s_{i'}^{j'}$$

Le modèle ainsi obtenu sera noté $\mathscr{M}_{m_3}^4$.

4.5. Autres problématiques liées aux motifs

Dans cette section, nous abordons d'autres problématiques liées aux motifs des benzénoïdes. La sous-section 4.5.1 décrit un modèle capable de générer des structures de benzénoïdes n'incluant pas un motif donné, tandis que la sous-section 4.5.2 présente un modèle capable de contraindre le nombre d'occurrences d'un motif donné au sein d'un benzénoïde.

4.5.1. Génération de structures excluant un motif donné

Nous souhaitons maintenant générer l'intégralité des structures possédant au plus n hexagones et ne contenant pas un motif P donné. Les raisonnements suivis pour les modèles $\mathscr{M}_{i_2}$ et $\mathscr{M}_{i_3}$ ne semblent pas applicables ici, car nous serions obligés de garantir qu'il n'existe pas d'affectation appropriée des variables f_i ou de sous-graphe isomorphe. Par conséquent, nous considérons une approche similaire à celle utilisée dans le modèle $\mathscr{M}_{i_1}$ (voir la sous-section 4.3.1). Plus précisément, nous partons encore une fois du modèle général $\mathscr{M}$ auquel nous ajoutons une variable e_i pour chaque fragment possible dans le coronénoïde englobant (de taille $k(n)$ ici). Chaque variable e_i sera vraie si la contrainte $\bigwedge_{j \in F_{i-}} x_j = 0 \wedge \bigwedge_{j \in F_{i+}} x_j = 1$ est satisfaite (en d'autres termes, si le fragment considéré est présent dans la structure). Pour finir, nous posons une contrainte `Sum` afin de spécifier que $\sum_j e_j = 0$. Ce modèle est noté $\mathscr{M}_{e_1}^1$. Une formulation équivalente consiste à représenter directement chaque fragment F_i sous la forme d'un nogood $\bigvee_{j \in F_{i-}} x_j = 1 \vee \bigvee_{j \in F_{i+}} x_j = 0$, ce qui donne un second modèle noté $\mathscr{M}_{e_1}^2$.

4.5.2. Génération de structures incluant plusieurs fois un même motif

Il existe différents moyens de contraindre le nombre d'occurrences d'un motif P donné et certaines d'entre elles sont facilement modélisables. Par exemple, si l'on souhaite générer des structures admettant au moins k occurrences deux à deux disjointes du motif P, nous pouvons partir d'un modèle parmi $\mathscr{M}_{m_1}^3$, $\mathscr{M}_{m_2}^3$ et $\mathscr{M}_{m_3}^3$ et considérer que E_P comme un (multi-)ensemble contenant k fois P. Il existe néanmoins d'autres problématiques moins évidentes par exemple, la prise en compte d'exactement k exemplaires du motif. Dans le but de faciliter la compréhension, nous définissons une variable n_e qui va représenter le nombre d'occurrences du motif P contenu dans la structure générée. Cette dernière aura pour domaine $\{0, \dots, k_{max}\}$ avec k_{max} étant le nombre maximal d'occurrences désirées par l'utilisateur. Si l'utilisateur ne spécifie aucune valeur de k_{max}, on considérera par défaut $k_{max} = \left\lfloor \frac{n}{|P_+|} \right\rfloor$.

Une fois de plus, l'approche suivie dans le modèle $\mathscr{M}_{i_1}$ (voir la sous-section 4.3.1) semble être la plus appropriée. Nous partons donc de ce modèle et nous y ajoutons la variable n_e. En plus des variables e_i permettant de définir tous les fragments possibles, nous ajoutons une variable e'_i pour chacun d'entre eux. La variable e'_i sera vraie si le fragment F_i est présent dans la structure générée. Cela est garanti en ajoutant pour chaque fragment F_i, la contrainte $\bigwedge_{j \in F_{i-}} x_j = 0 \wedge \bigwedge_{j \in F_{i+}} x_j = 1 \Rightarrow e'_i = 1$. Ensuite, étant donné que des fragments peuvent partager des hexagones, nous ajoutons les contraintes d'exclusion mutuelle : pour chaque hexagone

h, nous considérons la contrainte $\sum_{i|h\in F_i} e'_i \geq 1 \Rightarrow \sum_{i|h\in F_i} e_i = 1$ (étant donné que $F_i = F_{i+} \cup F_{i-} \cup F_{io} \cup F_{i*}$). Ainsi, cela garantit que si un hexagone participe simultanément à plusieurs fragments, alors seul l'un d'entre eux sera considéré comme présent. Pour finir, la contrainte $\sum_j e_j = n_e$ nous permet de s'assurer que n_e correspond bien au nombre d'occurrences du motif P dans la structure générée. Des contraintes arithmétiques pourront alors être posées sur cette variable en fonction des besoins de l'utilisateur. Le modèle ainsi obtenu est noté $\mathscr{M}_o^1$. Notons qu'il est également possible d'utiliser cette variable afin de trouver les structures maximisant le nombre d'occurrences de P. Dans ce cas, nous devrons considérer le formalisme *COP* (*Constraint Optimisation Problem*, défini dans la section 1.3). Pour rappel, une instance COP contient les mêmes éléments qu'une instance CSP, auxquels on rajoute une fonction objectif dépendant des variables de cette dernière. Il est ensuite possible de demander de minimiser ou de maximiser la valeur de cette fonction.

4.6. Comparaisons expérimentales

Dans cette section, on se propose de comparer empiriquement les différents modèles décrits dans les sections précédentes. Pour cela, nous considérons les 8 motifs représentés dans les figures 4.5(a)-(g) et 4.3(a) issus de (Wohner, Lam et Sattler 2015). Nous faisons varier le nombre n d'hexagones présents dans les structures du nombre d'hexagones du motif à 9. Cela nous permet de produire 55 instances (resp. 135) du problème de génération de structures admettant/excluant un motif donné (resp. admettant deux motifs donnés). Précisons que nous recherchons ici les structures ayant exactement n hexagones. Pour rappel, notre implémentation est basée sur Choco Solver (v. 4.10.8). Nous considérons quatre heuristiques de choix de variables présentes dans l'état de l'art : $dom/wdeg$ (Boussemart, Hemery, Lecoutre et al. 2004), $dom/wdeg^{ca.cd}$ (Wattez, Lecoutre, Paparrizou et al. 2019), CHS (Habet et Terrioux 2019) et dom (chacune d'entre elles est définie dans la sous-section 1.4.4). Cette dernière consiste à choisir comme prochaine variable à affecter, la prochaine variable selon l'ordre lexicographique ayant le plus petit domaine. Au niveau des heuristiques de choix de valeur, nous avons choisi d'utiliser inc et $desc$ qui vont respectivement choisir en priorité la valeur la plus petite et la plus grande. Ces expérimentations ont été effectuées sur un serveur DELL PowerEdge R440 avec un processeur Intel Xeon 4112 2,6 GHz et 32 Go de mémoire. Le temps limite de calcul a été fixé à 2 heures pour chacune des instances. Notons que nous ne comparons pas notre approche avec une méthode existante, car, à notre connaissance, aucune n'a été proposée jusqu'à maintenant. Cela est probablement dû au fait que cette problématique est très récente.

	$\mathscr{M}_{i_1}$				$\mathscr{M}_{i_2}$				$\mathscr{M}_{i_3}$			
	inc		$desc$		inc		$desc$		inc		$desc$	
	#I	Temps	#I	Temps	#I	Temps	#I	Temps	#I	Temps	#I	Temps
dom	55	0,56	55	0,92	55	0,59	55	0,92	55	0,61	55	0,98
$dom/wdeg$	49	16,67	54	11,06	48	16,78	55	9,35	50	15,75	55	8,94
$dom/wdeg^{ca.cd}$	55	5,66	55	2,09	52	11,00	54	7,40	51	13,39	53	8,26
CHS	47	22,23	47	20,70	48	20,33	48	18,93	52	15,43	55	8,51

Tableau 4.2. – Le nombre d'instances ayant été traitées avec succès (#I) et les temps cumulés associés en heures (Temps) pour chaque heuristique de choix de variable et de valeur et pour les modèles $\mathscr{M}_{i_1}$, $\mathscr{M}_{i_2}$ et $\mathscr{M}_{i_3}$.

Pour commencer, regardons la table 4.2, nous pouvons observer que, peu importe le modèle (parmi $\mathscr{M}_{i_1}$, $\mathscr{M}_{i_2}$ et $\mathscr{M}_{i_3}$) , l'heuristique de choix de valeur inc semble être plus efficace quand on utilise l'heuristique de choix de variable dom tandis que $desc$ semble être plus efficace avec les heuristiques $dom/wdeg$, $dom/wdeg^{ca.cd}$ et CHS. Maintenant, intéressons-nous à l'heuristique de choix de variable. On observe que ce ne sont pas les plus sophistiquées d'entre elles qui produisent les meilleurs résultats. En effet, peu importe le modèle, l'heuristique dom semble être la plus efficace pour résoudre notre problème. De plus, comme

montré dans la figure 4.11(a), dom produit de meilleurs résultats que $dom/wdeg^{ca.cd}$ sur presque toutes les instances considérées. Pour finir, étant donné une même heuristique de choix de variable et de valeur, nous pouvons remarquer que les modèles obtiennent souvent des résultats similaires. Si nous focalisons notre attention sur dom et $desc$ (voir les figures 4.11(b)-(d)), $\mathscr{M}_{i_1}$ se révèle légèrement meilleur que $\mathscr{M}_{i_2}$ et $\mathscr{M}_{i_3}$ qui, eux, obtiennent des résultats très similaires. Cette similarité s'explique peut-être par le fait que tous ces modèles sont basés sur le modèle général $\mathscr{M}$ et partagent donc un grand nombre de variables et de contraintes en commun. En effet, ces modèles ont un nombre similaire de contraintes, mais le modèle $\mathscr{M}_{i_1}$ en possède un peu moins que les autres. Pour finir, l'approche la plus optimale semble être d'utiliser le modèle $\mathscr{M}_{i_1}$ avec les heuristiques dom et inc.

Intéressons-nous maintenant à la génération de structures contenant deux motifs donnés. Nous pouvons noter que les tendances observées dans la table 4.3 sont relativement similaires à celles obtenues en voulant générer des structures ne contenant qu'un seul motif. Encore une fois, l'heuristique de choix de variable dom produit les meilleurs résultats. Une légère différence par rapport au cas précédent est que l'efficacité des autres heuristiques de choix de variable semble dépendre du modèle considéré. De plus, nous observons une différence plus marquée entre les performances des différents modèles, et ce, peu importe les heuristiques considérées. Globalement, le modèle $\mathscr{M}^3_{m_1}$ se révèle être le plus performant, suivi par $\mathscr{M}^3_{m_3}$ puis par $\mathscr{M}^3_{m_2}$ qui se révèle être le modèle le moins performant. Cela est clairement visible sur les figures 4.11(e)-(g) quand on considère les heuristiques dom et $desc$. Ce résultat semble être en phase avec le nombre de contraintes de chaque modèle. En effet, il apparaît $\mathscr{M}^3_{m_2}$ possède environ deux fois plus de contraintes que les modèles $\mathscr{M}^3_{m_1}$ et $\mathscr{M}^3_{m_3}$.

Concernant l'exclusion d'un motif donné, nous observons encore une fois des tendances similaires. Chacun des deux modèles parvient à traiter toutes les instances. Toutefois, si nous comparons les deux modèles $\mathscr{M}^1_{e_1}$ et $\mathscr{M}^2_{e_1}$ (voir la figure 4.11(h) et la table 4.4), il apparaît que le second est plus efficace et que l'heuristique dom est encore une fois la plus efficace. Cette différence de performance peut s'expliquer par le fait que $\mathscr{M}^2_{e_1}$ ne considère aucune autre variable que celles du modèle général. De plus, au niveau des contraintes, il ne considère que quelques clauses supplémentaires.

Le point le plus frappant dans ces résultats est que l'heuristique dom produise de bien meilleurs résultats que des heuristiques adaptatives telles que $dom/wdeg$, $dom/wdeg^{ca.cd}$ ou encore CHS. Pour essayer de comprendre ce phénomène, nous avons, dans un premier temps, comparé le nombre de conflits obtenus avec les modèles $\mathscr{M}_{i_1}$, $\mathscr{M}_{i_2}$ et $\mathscr{M}_{i_3}$ pour les heuristiques dom et $dom/wdeg^{ca.cd}$. Nous avons choisi de comparer ces deux heuristiques, car $dom/wdeg^{ca.cd}$ est l'heuristique adaptative ayant obtenu les meilleures performances. La figure 4.13 détaille ces comparaisons pour la génération du motif deep bay avec les trois modèles mentionnés précédemment. Le nombre d'hexagones a été fixé à 6. Nous pouvons voir, que, peu importe le modèle considéré, l'heuristique dom est toujours celle qui produit le moins de conflits.

Nous pensons que ce phénomène est lié à la nature du problème. En effet, les modèles considérés admettent des variables dont les domaines sont fortement homogènes et, en plus, la plupart d'entre elles sont des variables booléennes (c'est d'autant plus vrai pour le modèle M_{i_1}, ce qui pourrait expliquer ses meilleures performances). De plus, quand nous comparons l'ordre d'affectation des variables, il apparaît que l'utilisation de l'heuristique dom implique l'affectation prioritaire des variables x_i correspondant aux premiers hexagones du coronénoïde englobant. Cela peut se justifier par le fait que quand deux domaines sont de tailles identiques, l'heuristique dom choisira la première selon l'ordre lexicographique. Cela est bénéfique pour notre problème, car en choisissant en priorité ces variables, nous avons plus de chances d'affecter rapidement un x_i correspondant à un hexagone de la bordure gauche du coronénoïde englobant et ainsi satisfaire la contrainte de bord (voir la section 3.6). En revanche, l'heuristique $dom/wdeg^{ca.cd}$ ne semble pas prioriser ces variables. Nous pensons que cela est dû au fait que les x_i correspondant à des hexagones des bordures du coronénoïde englobant admettent un degré moins important que les autres. En effet, ces variables ne seront concernées que par trois ou quatre contraintes `edgeChanneling` (voir section 3.3) étant donné qu'elles n'ont que trois ou quatre voisins, tandis que les autres variables x_i seront concernées par six contraintes. L'heuristique dom se révèle donc plus adaptée à nos modèles.

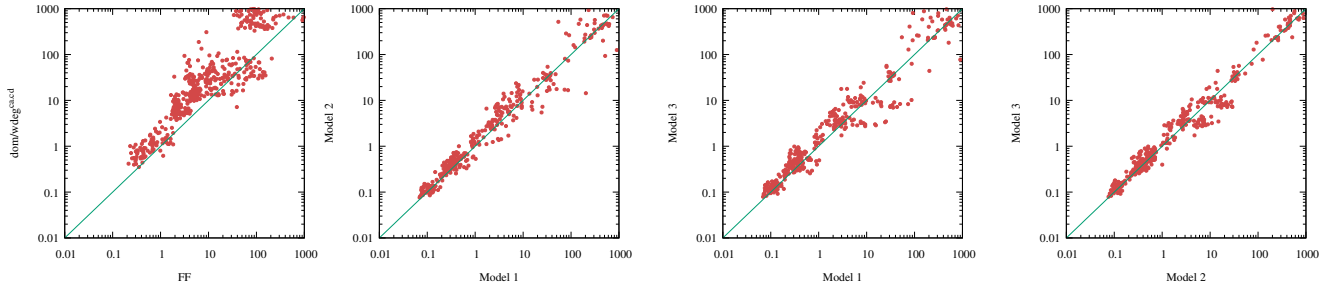
Pour finir, dans la figure 4.12(b), nous comparons le nombre de structures de benzénoïdes totales, le nombre de structures admettant un et deux motifs donnés, ainsi que le nombre de structures en excluant un en fonction du nombre n d'hexagones. D'une part, cette figure permet d'observer que le nombre de structures augmente en même temps que n. Nous pouvons ensuite remarquer que nous avons généré un grand nombre de structures. Par exemple, dans le cas de l'inclusion d'un motif simple, nous avons dû générer environ 70% du nombre total de structures.

	$\mathscr{M}^3_{m_1}$				$\mathscr{M}^3_{m_2}$				$\mathscr{M}^3_{m_3}$			
	inc		*desc*		*inc*		*desc*		*inc*		*desc*	
	#I	Temps	#I	Temps	#I	Temps	#I	Temps	#I	Temps	#I	Temps
dom	135	4,77	135	6,12	127	53,51	135	23,81	135	5,34	135	6,40
$dom/wdeg$	113	81,43	126	61,03	112	73,49	118	69,48	120	67,33	135	31,60
$dom/wdeg^{ca.cd}$	133	37,18	135	14,31	101	95,71	110	71,01	101	93,58	117	65,94
CHS	92	105,38	96	111,03	98	96,50	69	139,50	123	58,98	134	43,15

TABLEAU 4.3. – Le nombre d'instances qui ont été calculées avec succès (#I) et les temps associés cumulés en heure (Temps) pour chaque heuristique de choix de variable/valeur possible et les modèles $\mathscr{M}^3_{m_1}$, $\mathscr{M}^3_{m_2}$ et $\mathscr{M}^3_{m_3}$.

	$\mathscr{M}^3_{e_1}$				$\mathscr{M}^3_{e_2}$			
	inc		*desc*		*inc*		*desc*	
	#I	Temps	#I	Temps	#I	Temps	#I	Temps
dom	54	2,15	54	2,52	54	1,95	54	2,18
$dom/wdeg$	50	7.56	51	6.49	50	6.25	52	6.70
$dom/wdeg^{ca.cd}$	53	3.57	53	3.48	53	3.50	34	3.78
CHS	47	6.03	50	10.29	50	6.45	52	6.85

TABLEAU 4.4. – Le nombre d'instances qui ont été calculées avec succès (#I) et les temps associés cumulés en heure (Temps) pour chaque heuristique de choix de variable/valeur possible et les modèles $\mathscr{M}^3_{e_1}$ et $\mathscr{M}^3_{e_2}$.



(a) dom vs $dom/wdeg^{ca.cd}$ (b) $\mathscr{M}_{i_1}$ vs $\mathscr{M}_{i_2}$ (c) $\mathscr{M}_{i_1}$ vs $\mathscr{M}_{i_3}$ (d) $\mathscr{M}_{i_2}$ vs $\mathscr{M}_{i_3}$

(e) $\mathscr{M}^3_{m_1}$ vs $\mathscr{M}^3_{m_2}$ (f) $\mathscr{M}^3_{m_1}$ vs $\mathscr{M}^3_{m_3}$ (g) $\mathscr{M}^3_{m_2}$ vs $\mathscr{M}^3_{m_3}$ (h) $\mathscr{M}^1_{e_1}$ vs $\mathscr{M}^2_{e_1}$

FIGURE 4.11. – Comparaison des heuristiques de choix de variable dom et $dom/wdeg^{ca.cd}$ basée sur le temps d'exécution (en secondes) pour $\mathscr{M}_{i_1}$ et l'heuristique de choix de valeur $desc$. Comparaison deux à deux des modèles basées sur le temps d'exécution (en secondes) quand on utilise l'heuristique de choix de variable dom et l'heuristique de choix de valeur $desc$ (b)-(h).

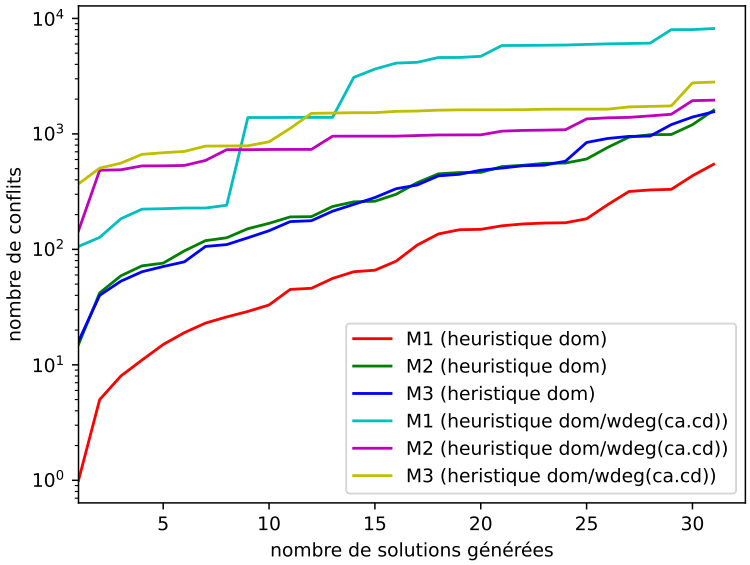


FIGURE 4.13. – Le nombre de conflits cumulés rencontrés par M_{i_1}, M_{i_2} et M_{i_3} à chaque solution générée pour les heuristiques dom et $dom/wdeg^{ca.cd}$.

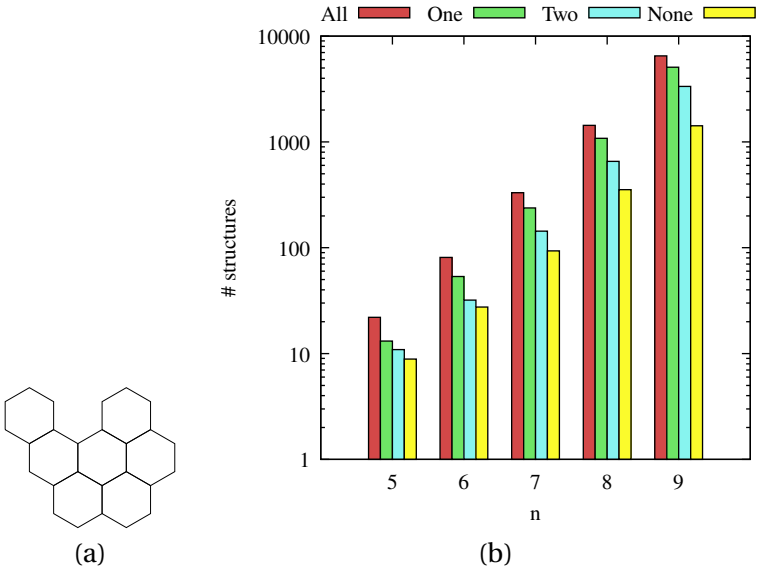


FIGURE 4.12. – Une des 25 structures de benzénoïdes de 7 hexagones contenant les motifs *armchair edge* et *deep bay* (a). Nombre de structures de benzénoïdes (all) et le nombre moyen de structures contenant un motif donné (un), deux motifs donnés (two) ou excluant un motif donné (none) (b).

4.7. Conclusions et perspectives

Dans ce chapitre, nous avons présenté une approche basée sur la programmation par contraintes capable de générer des benzénoïdes satisfaisant diverses propriétés liées à la présence ou non de motifs. Pour cela, nous avons mis au point différents modèles, chacun d'entre eux basées sur le modèle général décrit dans la section 3.3. Les approches considérées consistaient respectivement à calculer en amont toutes les possibles occurrences du ou des motifs voulus au sein du coronénoïde englobant, à lier chaque hexagone du coronénoïde englobant participant au motif à l'hexagone correspondant de ce dernier ou à résoudre une variante du problème d'isomorphisme de sous-graphe. Après comparaison des différents temps d'exécution sur un ensemble de huit motifs bien connus des chimistes, la première approche se révèle être la plus performante. À notre connaissance, il n'existait aucun outil capable de générer des benzénoïdes satisfaisant de telles propriétés. Cela pourrait donc annoncer de futures retombées prometteuses, étant donné que l'étude de la topologie des bordures des benzénoïdes est un sujet extrêmement étudié depuis quelques années (QIU, NARITA et MÜLLEN 2020; LIU et FENG 2020; AJAYAKUMAR, MA, LUCOTTI et al. 2021; CHEN, LIN, LUO et al. 2021; CHEUNG, WATANABE, SEGAWA et al. 2021; FUJISE, TSURUMAKI, WAKAMATSU et al. 2021; KANCHERLA et JØRGENSEN 2020; URYU, HIRAGA, KOGA et al. 2020; XIA, PUN, CHEN et al. 2021; MISHRA, YAO, CHEN et al. 2021; ZENG, WANG, ZHANG et al. 2021). L'intégralité des travaux détaillés dans ce chapitre ont été présentés à la conférence CP2021 (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2021). Il faut également mentionner que ces résultats diffèrent de ceux que nous avons présentés dans cet article. En effet, nous utilisions à l'époque une contrainte permettant de spécifier que le benzénoïde généré soit collé à la bordure gauche du coronénoïde englobant afin d'éliminer des symétries par translation. Cela donnait lieu à des résultats différents, mais les tendances observées restaient néanmoins similaires.

Au niveau des perspectives, une première pourrait naturellement être l'étude des répercussions de cet outil d'un point de vue chimie. De plus, les modèles présentés dans ce chapitre admettent une forte combinatoire et générer des molécules admettant des motifs de grande taille peut rapidement s'avérer coûteux en temps. Une autre perspective pourrait donc être d'améliorer l'efficacité en pratique de notre approche.

5. Calcul d'énergie de résonance

5.1. Introduction

Ce chapitre présentera en détails les travaux effectués dans le but de calculer l'énergie de résonance (définie dans la sous-section 2.4.4) des benzénoïdes à l'aide de la programmation par contraintes. Ce dernier est organisé de la manière suivante : la section 5.2 présentera une implémentation de l'algorithme de Lin & Fan qui consiste à calculer l'énergie de résonance d'un benzénoïde donné en énumérant l'intégralité de ses structures de Kekulé. La section 5.3 présentera quant à elle une amélioration de l'algorithme de Lin. Pour rappel, cet algorithme consiste à calculer une approximation de l'énergie de résonance d'un benzénoïde en ne considérant que les circuits conjugués minimaux de taille au plus 4. Contrairement à l'algorithme de Lin & Fan, il n'est pas nécessaire d'énumérer l'intégralité des structures de Kekulé du benzénoïde traité, mais juste de compter le nombre de structures admis par certains de ses sous-graphes, tâche qui peut s'effectuer en temps polynomial. Pour plus de détails sur ces deux algorithmes, nous invitons le lecteur à se référer à la sous-section 2.4.4. Pour finir, la section 5.4 présentera les différents résultats expérimentaux obtenus et la section 5.5 conclura ce chapitre.

5.2. Implémentation de l'algorithme de Lin et Fan à l'aide de la PPC

Lin et Fan ont développé une implémentation de leur méthode (présentée dans (LIN et FAN 1999)). Malheureusement, cette dernière n'est plus disponible. Par conséquent, nous présenterons dans cette sous-section une implémentation de cette méthode utilisant la programmation par contraintes. Comme mentionné dans la sous-section 2.4.4, cette méthode est inefficace en pratique quand la taille des benzénoïdes considérés augmente. Cela est dû au fait que cette méthode requiert d'énumérer l'ensemble des structures de Kekulé du benzénoïde traité, bien que ce nombre soit potentiellement exponentiel. Dans notre cas, la raison principale qui nous a motivé à proposer cette implémentation est d'avoir un point de comparaison avec notre seconde méthode qui sera présentée dans la sous-section 5.3.

Notre implémentation (baptisée LFCP) exploite la programmation par contraintes dans le but d'énumérer l'intégralité des structures de Kekulé du benzénoïde traité (ligne 2 de l'Algorithme 2). Pour cela, nous considérons le modèle $\mathscr{M}_1^{\mathscr{K}}$ qui consiste à modéliser ce problème sous la forme d'une instance CSP dont chacune des solutions correspondra à une structure de Kekulé. Comme n'importe quel benzénoïde $B = (V, E)$ peut être vu comme un graphe biparti, nous pouvons diviser V en deux sous-ensembles de sommets V_1 et V_2 de telle sorte que chaque arête de E soit liée à un sommet de V_1 et à un sommet de V_2. Nous considérons une variable y_v pour chaque sommet v de V_1. Cette variable aura pour domaine l'ensemble des sommets w de V_2 tel que $\{v, w\} \in E$. En procédant de la sorte, si la variable y_v est affectée à la valeur w, cela signifie que l'arête $\{v, w\}$ correspondra à une liaison double. Par définition d'une solution (voir section 1.3), cela assure que chaque carbone de V_1 participera à exactement une double liaison. Il reste maintenant à assurer cette propriété pour les sommets de V_2. Cela peut être facilement fait en considérant une contrainte globale `allDifferent` (voir la sous-section 1.5.1) impliquant toutes les variables de X. Nous obtenons donc le modèle $\mathscr{M}_1^{\mathscr{K}}$ suivant :

$$\begin{cases} X = \{y_v | v \in V_1\} \\ D = \{D_{y_v} | v \in V_1\} \text{ avec } D_{y_v} = \{w | w \in V_2,\ \{v, w\} \in E\} \\ C = \{\texttt{allDifferent}(X)\} \end{cases}$$

Les solutions de ce modèle correspondent clairement à l'ensemble des structures de Kekulé (et donc des couplages parfaits) de B. Par rapport au filtrage de la contrainte `allDifferent`, Régin a présenté un algorithme efficace basé sur les couplages d'un graphe particulier, appelé *graphe de valeur* (RÉGIN 1994). Il s'avère que dans le cas de notre modèle, ce graphe de valeur correspond exactement à B. De plus, n'importe quel solveur ayant implémenté ce filtrage est capable d'énumérer efficacement les structures de Kekulé de B à n'importe quelle étape de la recherche. Notons qu'un autre modèle (que nous noterons $\mathscr{M}_2^{\mathscr{K}}$) a été proposé (MANN et THIEL 2013). Ce dernier consiste à considérer une variable booléenne $x_{v,w}$ pour chaque arête $\{v, w\}$ de B. La valeur de la variable $x_{v,w}$ sera vraie si l'arête $\{v, w\}$ correspond à une liaison double

dans la solution générée, faux sinon. Ensuite, pour chaque sommet v de B, on doit appliquer une contrainte globale `Sum` (définie dans la sous-section 1.5.1) spécifiant que la somme des variables représentant les arêtes impliquant v doit être égale à 1. Une variable vaudra ainsi 1 si et seulement si l'arête qu'elle représente correspond à une liaison double. Plus formellement, obtenons le modèle $\mathcal{M}_2^{\mathcal{K}}$ suivant :

$$\begin{cases} X = \{x_{v,w} | \{v,w\} \in E\} \\ D = \{D_{v,w} | \{v,w\} \in E\} \text{ avec } D_{v,w} = \{0,1\} \\ C = \{ \sum\limits_{\{v,w\} \in E} x_{v,w} = 1 | v \in V\} \end{cases}$$

Afin de sélectionner un modèle pour notre implémentation de l'algorithme de Lin et Fan, nous comparons les temps d'exécution des deux modèles $\mathcal{M}_1^{\mathcal{K}}$ et de $\mathcal{M}_2^{\mathcal{K}}$. Pour cela, nous les appliquons aux coronénoïdes de taille 2 à 8, car ils possèdent un nombre de structures de Kekulé qui croît rapidement avec leur taille. La table 5.1 présente les différents temps d'exécution des deux modèles et fournit, à titre indicatif, les temps de la méthode de Rispoli. Pour rappel, cette dernière permet uniquement de dénombrer le nombre de couplages sans avoir besoin de les énumérer. Ces expérimentations ont été effectuées sur un serveur DELL PowerEdge R440 avec un processeur Intel Xeon 4112 2,6 GHz et 32 Go de mémoire. Le temps limite de calcul a été fixé à 24 heures pour chacune des instances. Nous pouvons observer que le modèle $\mathcal{M}_2^{\mathcal{K}}$ s'avère plus rapide que le modèle $\mathcal{M}_1^{\mathcal{K}}$ dès que le nombre de structures devient conséquent. Il est intéressant de mentionner qu'il n'y a pas d'échecs rencontrés durant la résolution des instances par ces deux modèles. Cela laisse à penser qu'ils bénéficient tous deux d'une bonne propriété. Pour l'implémentation, notre choix s'est donc porté sur le modèle $\mathcal{M}_2^{\mathcal{K}}$. Par ailleurs, on peut également remarquer que le temps de calcul, quel que soit le modèle, devient rapidement prohibitif quand le nombre de structures de Kekulé devient grand. L'approche de Lin et Fan semble donc difficilement exploitable dans un tel cas de figure. Nous pouvons enfin noter que la méthode de Rispoli est très rapide pour dénombrer le nombre de structures de Kekulé, toutes les instances étant résolues en moins d'un dixième de secondes. Malheureusement, nous ne pouvons pas l'utiliser ici, car nous avons besoin d'énumérer l'intégralité des structures de Kekulé du benzénoïde traité et pas uniquement de les compter.

nombre de couronnes	nombre de structures	temps (s)		
		$\mathcal{M}_1^{\mathcal{K}}$	$\mathcal{M}_2^{\mathcal{K}}$	Rispoli
2	20	0,034	0,017	0,004
3	980	0,163	0,257	0,005
4	232 848	8,3	4,0	0,010
5	$2,67 \times 10^8$	13 703	6 459	0,016
6	$1,47 \times 10^{12}$	TIMEOUT	TIMEOUT	0,021
7	$3,94 \times 10^{16}$	TIMEOUT	TIMEOUT	0,031
8	$9,22 \times 10^{18}$	TIMEOUT	TIMEOUT	0,041

TABLEAU 5.1. – Nombre de structures de Kekulé de coronénoïdes de taille 2 à 8 et temps d'exécution des modèles $\mathcal{M}_1^{\mathcal{K}}$, $\mathcal{M}_2^{\mathcal{K}}$ et de la méthode de Rispoli (en secondes) avec un temps limite fixé à 24 heures

5.3. Amélioration de l'algorithme de Lin à l'aide de la PPC

Dans cette section, nous présentons une nouvelle méthode améliorant celle proposée par Lin (LIN 2000) et utilisant la programmation par contraintes. Pour rappel, l'algorithme de Lin calcule l'énergie de résonance globale d'un benzénoïde alors qu'ici, le but est de fournir une approximation de son énergie de résonance locale. Pour rappel, l'énergie de résonance locale est plus utile que sa variante globale. En effet, elle permet de prédire sur quelles parties du benzénoïde traité les réactions chimiques ont le plus de chance d'avoir lieu.

Avant de présenter cette méthode plus en détails, nous devons introduire certaines définitions.

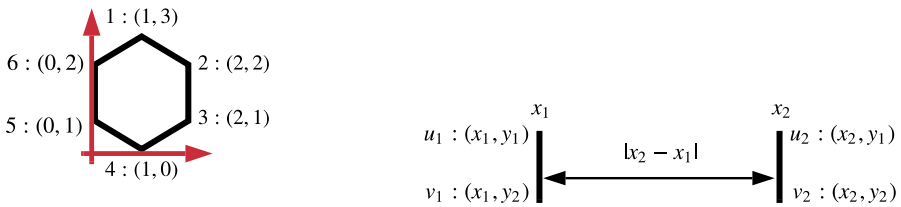
5.3.1. Définitions préliminaires

Premièrement, on se propose d'introduire la notion de *coordonnées* d'un benzénoïde :

Définition 5.3.1 *Soit $B = (V, E)$ un benzénoïde. Une* fonction de coordonnées $c : V \to \mathbb{Z}^2$ *de B est une fonction liant un couple unique d'entiers $(c(v).x, c(v).y)$ (c'est-à-dire une abscisse et une ordonnée dans le plan cartésien) à chaque sommet v de B de telle sorte que si $(v_1, v_2, v_3, v_4, v_5, v_6)$ correspondent aux sommets d'un hexagone (dans le sens horaire) avec v_1 le sommet ayant l'ordonnée la plus grande, alors nous avons :*

$$\begin{cases} c(v_2) = (c(v_1).x + 1, c(v_1).y - 1) \\ c(v_3) = (c(v_1).x + 1, c(v_1).y - 2) \\ c(v_4) = (c(v_1).x, c(v_1).y - 3) \\ c(v_5) = (c(v_1).x - 1, c(v_1).y - 2) \\ c(v_6) = (c(v_1).x - 1, c(v_1).y - 1) \end{cases}$$

La figure 5.1(a) décrit un exemple simple de coordonnées pour le cas du benzène.



(a) (b)

FIGURE 5.1. – Un exemple de coordonnées pour le benzène (a) et un exemple d'intervalle (b).

Nous nous intéressons maintenant à la notion d'*arête verticale* :

Définition 5.3.2 *Soient $B = (V, E)$ un benzénoïde et c une fonction de coordonnées. Une arête $e = \{u, v\} \in E$ est une* arête verticale *de B si et seulement si $c(u).x = c(v).x$.*

Ainsi, les arêtes verticales du benzénoïde décrit dans la figure 5.1(a) sont $\{v_2, v_3\}$ et $\{v_5, v_6\}$. Nous introduisons maintenant la notion d'intervalle liée aux arêtes verticales :

Définition 5.3.3 *Soient B un benzénoïde et c une fonction de coordonnées. Un* intervalle *I de B est un couple $I = (e_1, e_2)$ de telle sorte que :*

$$\begin{cases} e_1 = \{u_1, v_1\} \in E \\ e_2 = \{u_2, v_2\} \in E \\ c(u_1).y = c(u_2).y \\ c(v_1).y = c(v_2).y \end{cases}$$

De plus :

$$\begin{cases} I.x_1 = c(u_1).x \\ I.y_1 = c(u_1).y \\ I.x_2 = c(u_2).x \\ I.y_2 = c(v_1).y \end{cases}$$

*On note $|I| = |I.x_2 - I.x_1|$ la **taille** de I.*

En d'autres termes, on peut dire qu'un intervalle va représenter l'espace contenu entre deux arêtes verticales ayant la même ordonnée. La figure 5.1(b) nous montre un exemple d'intervalle. Maintenant que les notions de coordonnées, d'arêtes verticales et d'intervalles ont été correctement introduites, nous pouvons détailler le fonctionnement de notre algorithme.

5.3.2. Description générale de l'algorithme

On se propose maintenant de décrire en détails notre algorithme. Nous l'avons baptisé CERCP (pour Calcul_Énergie_Résonance_CP) et l'avons détaillé dans l'algorithme 4. Notre méthode prend en entrée un benzénoïde $B = (V, E)$, une fonction de coordonnées c, une base contenant tous les cycles de taille au plus 4 pouvant induire au moins un h-CCM ainsi qu'une seconde base contenant tous les couples de circuits de la première base pouvant induire un circuit redondant. Les cycles de ces bases sont stockés sous la forme d'un ensemble d'intervalles. Ce procédé sera expliqué en détails dans la sous-section 5.3.5. Notre méthode retourne ensuite une approximation de l'énergie de résonance locale $E(B,h)$ de chaque hexagone h de B.

Dans un premier temps, nous calculons l'ensemble $\mathscr{C}^*$ de tous les cycles de B de taille au plus 6 (ligne 3). Nous choisissons de générer les cycles de taille au plus 6 car la taille d'un cycle induit par deux circuits redondants de taille au plus 4 sera au plus égale à 6. Ensuite, pour chaque cycle $\mathscr{C}$ de $\mathscr{C}^*$ (ligne 4), nous commençons par représenter le cycle selon une collection d'intervalles (ligne 5) et, si $\mathscr{C}$ apparaît dans la base des circuits minimaux (ligne 6), alors on énumère tous les circuits minimaux qu'il induit. Ensuite, pour chacun de ces circuits (ligne 7), nous ajoutons sa contribution à l'énergie de résonance locale de l'hexagone pour lequel le circuit est minimal (lignes 8-9). Notons que, pour chaque cycle $\mathscr{C}$, nous traitons séparément chacun des circuits minimaux qu'il induit et à ce titre, nous n'avons plus besoin de considérer la notion $M(\mathscr{C})$ (voir la définition 2.4.8). Cependant, $\mathscr{C}$ peut également correspondre au contour de l'union de deux circuits redondants sur un hexagone h (lignes 10-11). Si c'est le cas, on doit s'assurer de ne pas prendre en compte la contribution du circuit ayant la plus grande taille. Pour cela, nous supprimons la contribution du circuit ayant la plus grande taille (lignes 12-13) car il a déjà été comptabilisé (ligne 9). Par exemple, considérons le cycle $\mathscr{C}$ comme étant l'union des deux circuits représentés dans la figure 5.2 (b). Il peut être obtenu soit par l'union des deux cycles de la figure de gauche (de taille 3 et 4), soit par ceux de la figure de droite (également de taille 3 et 4), nous devons donc supprimer la contribution de deux circuits de taille 4. Pour terminer, on divise la contribution de chaque hexagone par le nombre de structures de Kekulé de B (ligne 14-15).

5.3.3. Énumération des cycles

Nous allons maintenant voir comment s'y prendre afin de générer tous les cycles correspondant soit à un h-CCM de taille au plus 4 (ligne 4 de l'algorithme 4) ou à une union de deux h-CCM redondants (ligne 6). Étant donné que le cycle induit par l'union de deux h-CCM de taille au plus 4 est de taille au plus 6, nous devons énumérer tous les cycles de taille au plus 6. Cela permet d'obtenir l'intégralité des circuits (minimaux ou redondants) en une unique énumération. On se propose de modéliser ce problème sous la forme d'une instance CSP. Nous considérons tout d'abord une variable de graphe x_G ayant pour domaine l'ensemble des sous-graphes compris entre un graphe vide et B. À chaque nouvelle solution, cette variable représentera le cycle qui aura été généré. Afin de garantir que la valeur de cette variable corresponde à un cycle, nous appliquons une contrainte `cycle` (décrite dans la sous-section 1.5.2) sur cette dernière. Nous devons maintenant nous assurer que les cycles générés soient de taille au plus 6. Pour cela, nous introduisons une variable booléenne x_e pour chaque arête e de B. La valeur de x_e vaudra 1 si et seulement si l'arête e apparaît dans le graphe décrit par x_G (et donc 0 dans le cas contraire). Cela peut être fait en appliquant une contrainte `edgeChanneling` (également décrite dans la sous-section 1.5.2) sur x_G et cet ensemble de variables booléennes. Pour finir, nous posons une contrainte `Sum` sur l'ensemble des variables x_e afin d'imposer $\sum_{x_e \mid e \in E} x_e \in \{6, 10, 14, 18, 22, 26\}$ car nous cherchons à générer les circuits de taille au plus 6 (pour rappel, un circuit de taille i admet $4i+2$ arêtes). Plus formellement, nous considérons le modèle $\mathscr{M}_{\mathscr{C}^*}$:

Algorithme 4 : *Calcul_Énergie_Résonance_CP (CERCP)*

Input : un benzénoïde B, une fonction de coordonnées c, une base de a h-MCC, une base de circuits redondants
Output : l'énergie de résonance locale $E(B,h)$ pour chaque hexagone h de B

1 **foreach** h *de* B **do**
2 $\quad énergie[h] \leftarrow 0$
3 $\mathscr{C}^* \leftarrow générer\_cycles\_choco(B,1,6)$
4 **foreach** $\mathscr{C} \in \mathscr{C}^*$ **do**
5 $\quad id \leftarrow identifier\_cycle(\mathscr{C})$
6 $\quad$ **if** $dans\_base\_circuits\_minimaux(id)$ **then**
7 $\quad\quad$ **foreach** $\mathscr{C}_m \in circuits\_minimaux(\mathscr{C})$ **do**
8 $\quad\quad\quad h \leftarrow$ hexagone t.q. $\mathscr{C}_m$ est un h-MCC
9 $\quad\quad\quad énergie[h] \leftarrow énergie[h] + R_{|\mathscr{C}_m|} \times |\mathscr{K}(B - B[\mathscr{C}_m])|$
10 $\quad$ **else if** $dans\_base\_circuits\_redondants(id)$ **then**
11 $\quad\quad$ **foreach** $(\mathscr{C}_1,\mathscr{C}_2)$ *t.q.* $\mathscr{C} = \mathscr{C}_1 \bigcup \mathscr{C}_2$ *et* $redondant(\mathscr{C}_1,\mathscr{C}_2,h)$ **do**
12 $\quad\quad\quad \mathscr{C}' \leftarrow argmax(|\mathscr{C}_1|,|\mathscr{C}_2|)$
13 $\quad\quad\quad énergie[h] \leftarrow énergie[h] - |\mathscr{K}(B - B[\mathscr{C}])| \times R_{|\mathscr{C}'|}$
14 **foreach** h *de* B **do**
15 $\quad énergie[h] \leftarrow énergie[h]/|\mathscr{K}(B)|$
16 **return** $énergie$

$$\begin{cases} X = \{x_G\} \cup \{x_e | e \in E\} \\ D = \{D_{x_G}\} \cup \{D_{x_e} | e \in E\} \text{ avec } D_{x_G} = \{g | \emptyset \subseteq g \subseteq B\} \; et \; D_{x_e} = \{0,1\} \\ C = \{cycle(x_G), \sum_{x_e | e \in E} x_e \in \{6,10,14,18,22,26\}\} \cup \{\texttt{edgeChanneling}(x_e, x_G) | e \in E\} \end{cases}$$

Étant donné que le solveur Choco que nous utilisons implémente les variables de graphes ainsi que toutes les contraintes mentionnées, il peut très facilement exprimer ce modèle.

5.3.4. Comptage du nombre de structures de Kekulé

Maintenant, nous cherchons à compter le nombre de structures de Kekulé (ligne 12) ou de $B - B[\mathscr{C}]$ pour différents cycles $\mathscr{C} \in \mathscr{C}^*$ (lignes 7 et 11). Dans (RISPOLI 2001), Rispoli présente une méthode permettant de compter le nombre de structures de Kekulé d'un benzénoïde (pour rappel, cette méthode a été expliquée en détails dans la section 2.2) en temps polynomial. C'est donc cette méthode qui est utilisée à la ligne 12 afin de compter le nombre de structures de Kekulé.

Maintenant, jetons de nouveau un coup d'œil aux lignes 7 et 12. On cherche à calculer le nombre de structures de Kekulé de $B - B[\mathscr{C}]$. Pour rappel, $B - B[\mathscr{C}]$ est une partie du benzénoïde B (et donc un sous-graphe) induit par la suppression des sommets appartenant au cycle $\mathscr{C}$ et à son intérieur. Dans certains cas, ce sous-graphe peut ne pas correspondre à un benzénoïde et donc la méthode de Rispoli risque de ne pas pouvoir s'appliquer correctement. Si l'on prend par exemple la figure 2.23(b), le sous-graphe correspondant possède deux composantes connexes dont une (celle du bas) qui n'est pas un benzénoïde, la méthode de Rispoli n'est donc pas directement applicable dans ce cas.

Pour résoudre ce problème, une solution consiste à supprimer tous les sommets ayant un degré égal à 1 ainsi que tous leurs voisins et à répéter ce processus jusqu'à l'obtention d'un point fixe, c'est-à-dire un sous-graphe qui ne contient plus aucun sommet de degré 1. En effet, un sommet de degré 1 n'a qu'un

seul voisin et donc, par définition, d'une structure de Kekulé, ces deux sommets seront forcément liés par une liaison double. On peut donc les supprimer sans que cela affecte le nombre de structures de Kekulé du sous-graphe restant. Ainsi, à la fin de cette procédure, tous les sommets restants auront un degré égal à 0, 2 ou 3. Dans le premier cas, la composante connexe en question ne contiendra qu'un seul sommet, et donc n'induira aucune structure de Kekulé.

On se propose maintenant d'étendre le théorème 2.2.1 énoncé par Rispoli (RISPOLI 2001) afin qu'il puisse s'appliquer aux composantes connexes ne contenant que des sommets de degrés 2 ou 3 obtenues après avoir appliqué notre procédure de nettoyage :

Théorème 5.3.1 *Soit $G = (V, E)$ un graphe connexe dont tous les cycles sont de taille $4n+2$ $(\forall\, n \geq 0)$, alors le nombre de couplages parfaits de B est égal au déterminant de sa matrice de biadjacence.*

Prouver ce théorème est relativement simple : le théorème de Rispoli repose sur un lemme (RISPOLI 2001) spécifiant que tous les cycles d'un benzénoïde sont de taille $4n+2$ $(\forall\, n \geq 0)$. Dans le cas de notre procédure, nous partons d'un benzénoïde (qui ne contient donc que des cycles de taille $4n+2$) et nous lui retirons des sommets et des arêtes. Le sous-graphe ainsi obtenu ne contiendra donc que des composantes connexes restreintes à un sommet isolé ou des composantes connexes ayant plusieurs sommets et pour lesquelles tous les cycles seront nécessairement de taille $4n+2$. Le lemme étant applicable dans ce dernier cas de figure, cela garantit la validité du théorème 5.3.1. Aussi, nous pouvons exploiter le calcul du déterminant pour compter le nombre de structures de Kekulé pour n'importe quel benzénoïde B (ligne 12) ou celle de n'importe quel $B - B[\mathscr{C}]$ (après avoir appliqué la procédure de nettoyage) [1].

5.3.5. Identification des cycles

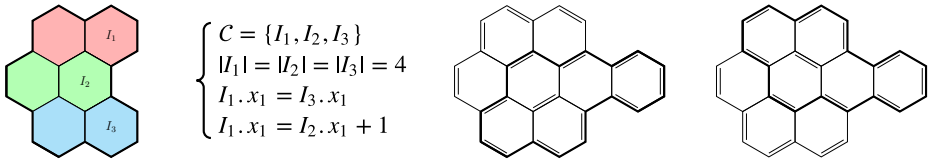
Une fois que Choco a généré un cycle, nous devons maintenant déterminer si ce dernier est présent dans la base des h-CCM (ligne 4) ou des circuits redondants (ligne 6). Pour cela, on se propose de représenter un cycle par un ensemble d'intervalles (cette notion a été définie dans la partie 5.3.1 ainsi que par un ensemble de relations liant ces intervalles). Ici, le but de ces relations est de représenter la distance entre arêtes verticales de chaque couple d'intervalles (pour chaque couple, nous considérons soit les deux arêtes gauches, soit les deux arêtes droites). Ainsi, étant donné un ensemble d'intervalles et de telles relations, nous sommes capables de construire le circuit représenté et inversement. La figure 5.2 (a) nous montre un exemple d'une telle représentation.

Les deux bases que nous considérons décrivent donc l'ensemble des cycles (h-CCM ou cycle redondant) identifiés par Lin (LIN 2000) par un ensemble d'intervalles liés par des relations. Maintenant, à chaque fois qu'un nouveau cycle est renvoyé par Choco, il nous suffit de le représenter de la manière décrite précédemment et de vérifier s'il appartient bien à une des deux bases. Nous sommes ainsi capables de déterminer si un cycle renvoyé par Choco peut induire un h-CCM ou être obtenu par l'union de deux circuits redondants.

5.4. Résultats expérimentaux

Actuellement, la méthode NICS (décrite dans la sous-section 2.4.2 et reposant sur des calculs de chimie quantique) est la méthode de référence des chimistes quand il s'agit d'estimer l'aromaticité d'un benzénoïde. Bien que cette dernière produise des résultats de bonne qualité, elle est relativement lourde et le traitement d'un benzénoïde de grande taille peut prendre rapidement plusieurs jours. L'objectif de ce chapitre était

1. Dans (CARISSAN, DIM, HAGEBAUM-REIGNIER et al. 2020), cette partie de l'algorithme est accomplie en énumérant les structures de Kekulé à l'aide du modèle $\mathscr{M}_1^{\mathscr{K}}$ présenté dans la sous-section 5.2. En effet, à cause d'une première bibliothèque peu efficace, le temps de calcul du déterminant de matrice était relativement lent. De plus, nous n'avions pas encore établi l'extension de la méthode de Rispoli et nous ne pouvions donc pas traiter les sous-graphes. Notons que nous avons comparé expérimentalement les temps de la méthode de Rispoli, de la résolution du modèle $\mathscr{M}_1^{\mathscr{K}}$ et de celle du modèle décrit par Mann et Thiel (MANN et THIEL 2013) (voir le tableau 5.1) et sans surprises, la première méthode est de loin la plus efficace.



(a) (b)

FIGURE 5.2. – Un exemple de cycle et des relations entre ses intervalles (a), et un exemple de circuits redondants (b).

donc d'essayer de proposer une méthode produisant des résultats de qualité similaire tout en étant plus rapide que NICS. Nous avons donc présenté dans ce chapitre deux méthodes supplémentaires : LFCP, une implémentation de l'algorithme de Lin et Fan, et CERCP, une amélioration de l'algorithme de Lin, toutes deux utilisant la programmation par contraintes.

Dans cette section, nous présentons les comparaisons expérimentales entre ces méthodes qui permettent toutes les trois d'estimer l'aromaticité d'un benzénoïde donné. Ici, la principale problématique sera de comparer la qualité des résultats obtenus par CERCP/LFCP et par NICS. En effet, ces trois méthodes ne produisent pas le même type de résultat, les deux premières attribuant une valeur énergétique à chaque hexagone en fonction des circuits conjugués minimaux qu'ils admettent (dans le cas de CERCP, on ne considère que les circuits de taille au plus 4) alors que les valeurs attribuées par NICS reposent sur des calculs de chimie quantique consistant à simuler l'application d'un champ magnétique perpendiculaire à la molécule, la valeur attribuée à chaque hexagone correspondra au sens de rotation de ses électrons. Pour plus d'informations sur la méthode NICS, nous invitons le lecteur à se référer à la section 2.4.2 de l'état de l'art. Cette section est organisée de la manière suivante : la sous-section 5.4.1 détaillera le protocole expérimental utilisé, la sous-section 5.4.2 comparera les méthodes CERCP et LFCP et pour finir, la sous-section 5.4.3 comparera les méthodes CERCP et NICS.

5.4.1. Protocole expérimental

Les méthodes CERCP et LFCP sont implémentées en Java et compilées avec Java SE 1.8. Elles sont disponibles dans notre logiciel BenzAI (https://benzai-team.github.io/BenzAI/). Ces deux implémentations font appel à la bibliothèque Choco Solver (v. 4.10.8) afin d'effectuer les calculs liés à la programmation par contraintes. Notons que nous avons gardé les paramètres de base de Choco. Pour information, comme mentionné dans (LIN 2000), la méthode proposée par Lin a déjà été implémentée. Malheureusement, cette implémentation ne semble pas être disponible. Nous considérons $R_i = \frac{1}{i^2}$ à la place des valeurs optimisées, car ces dernières sont uniquement définies pour des circuits de taille au plus 4. Or, dans le cas de LFCP, nous énumérons les circuits ayant une taille supérieure à 4. Cependant, utiliser ou non les valeurs optimisées n'influence pas les comparaisons entre CERCP et NICS. Pour le cas de NICS, nous exploitons l'implémentation présente dans le logiciel commercial Gaussian (http://gaussian.com/). Les valeurs obtenues avec NICS respectent la *convention NICS* (voir la sous-section 2.4.2), c'est-à-dire que les valeurs négatives traduisent une aromaticité, les valeurs nulles ou très proches de zéro une non-aromaticité et les valeurs positives une anti-aromaticité. Toutes les méthodes mentionnées ci-dessus ont été lancées sur des serveurs avec des processeurs Intel Xeon Gold de 2,20 GHz et 256 GB sous CentOS Linux 8.1.1911. Le temps d'exécution de chaque méthode a été limité à 24 heures par benzénoïde.

On se propose maintenant de décrire l'ensemble de molécules que nous considérons pour nos expérimentations. Ce dernier est divisé en trois sous-ensembles :

- $\mathscr{B}_1$ est un ensemble de 48 molécules dont le nombre d'hexagones varie entre 1 et 109. Il est composé de molécules bien connues des chimistes et qui sont souvent utilisées dans des comparaisons similaires (RANDIĆ 2019 ; RUIZ-MORALES 2004). Il correspond à l'ensemble utilisé dans (CARISSAN, DIM,

HAGEBAUM-REIGNIER et al. 2020 ; RANDIĆ 2019) auquel nous avons ajouté les molécules utilisées dans (RUIZ-MORALES 2004). Toutes les molécules considérées sont décrites dans les annexes (voir les figures A.1 à A.8).

- $\mathscr{B}_2$ est une sélection de 50 benzénoïdes catacondensés ayant entre 5 et 10 hexagones. Pour chacune des valeurs n considérées, nous avons sélectionné aléatoirement 10 molécules parmi l'ensemble des catacondensés de n hexagones. Les benzénoïdes catacondensés sont connus pour induire des circuits conjugués de petite taille.

- $\mathscr{B}_3$ contient les benzénoïdes de forme rectangulaires d'une hauteur de 5 hexagones et d'une largeur allant de 1 à 20. Ce genre de benzénoïdes sont connus pour posséder des circuits de tailles variées, et particulièrement de grandes tailles. Notons que nous avons également considéré des molécules de hauteur différentes, mais les résultats obtenus étaient identiques. Nous nous concentrons donc ici sur les molécules ayant une hauteur de 5.

Pour ces comparaisons, nous nous intéressons à deux critères : le temps d'exécution et la qualité de l'estimation. Pour le premier, nous considérons le temps CPU requis pour l'exécution de chacune des méthodes. Bien qu'il soit tout aussi important que le premier d'un point de vue chimie, la caractérisation du second est plus difficile. En effet, rappelons-nous que l'aromaticité ne peut pas être mesurée. Il n'y a donc aucune valeur de référence que nous pourrions utiliser pour estimer la qualité de nos résultats. Les méthodes CERCP et LFCP sont différentes de NICS et ne renvoient pas de valeurs similaires, même si chacune d'entre elles permettent d'estimer l'aromaticité locale d'un benzénoïde donné. Plus précisément, CERCP et LFCP renvoient des valeurs entre 0 et 1 tandis que pour NICS, les valeurs ne sont pas bornées par un intervalle spécifique. En effet, CERCP et LFCP ont pour but de décrire le comportement de la structure électronique de la molécule considérée tandis que l'approche NICS consiste à calculer l'intensité de la distorsion de sa structure électronique par un champ magnétique extérieur. Ces deux types de méthodes ont néanmoins tendance à avoir des variations proportionnelles, il est donc possible de les comparer. Ainsi, peu importe les méthodes M_1 et M_2 considérées, ces dernières sont comparées en calculant le coefficient de corrélation de Pearson sur les valeurs renvoyées par les méthodes. Plus précisément, étant donné un benzénoïde B, nous appliquons M_1 et M_2 sur B. Nous obtenons en retour deux valeurs $v_h^{M_1}$ et $v_h^{M_2}$ pour chaque hexagone h. Chacune de ces valeurs correspondra à l'aromaticité de h selon les méthodes M_1 et M_2. Le coefficient de corrélation de Pearson du benzénoïde B varie entre 0 et 1, une valeur proche de 1 que l'ordre d'aromaticité des hexagones coïncide et inversement. On le calcule selon la formule suivante :

$$r^B_{M_1,M_2} = \left| \frac{n\sum_h v_h^{M_1} v_h^{M_2} - \sum_h v_h^{M_1} \sum_h v_h^{M_2}}{\sqrt{n\sum_h (v_h^{M_1})^2 - \left(\sum_h v_h^{M_1}\right)^2}\sqrt{n\sum_h (v_h^{M_2})^2 - \left(\sum_h v_h^{M_2}\right)^2}} \right|$$

où n est le nombre d'hexagones de B.

5.4.2. Comparaisons entre CERCP et LFCP

Dans un premier temps, on se propose de comparer les temps d'exécution de CERCP et LFCP. Si l'on observe l'ensemble $\mathscr{B}_1$ (voir la table 5.2 et la figure 5.3 (a)), on remarque que les deux méthodes présentent des temps d'exécution similaires pour la majorité des molécules. Néanmoins, on voit que CERCP devient plus performante quand le nombre de structures de Kekulé de la molécule considérée augmente. On peut également observer que LFCP dépasse la limite de temps de 24 heures (TO) pour traiter les trois molécules de $\mathscr{B}_1$ admettant le grand nombre de structures de Kekulé. Cela peut facilement s'expliquer par le fait que CERCP doit juste compter le nombre de structures de Kekulé du benzénoïde traité (ainsi que de certains de ses sous-graphes), tâche qui se fait en temps polynomial alors que LFCP doit énumérer l'intégralité des

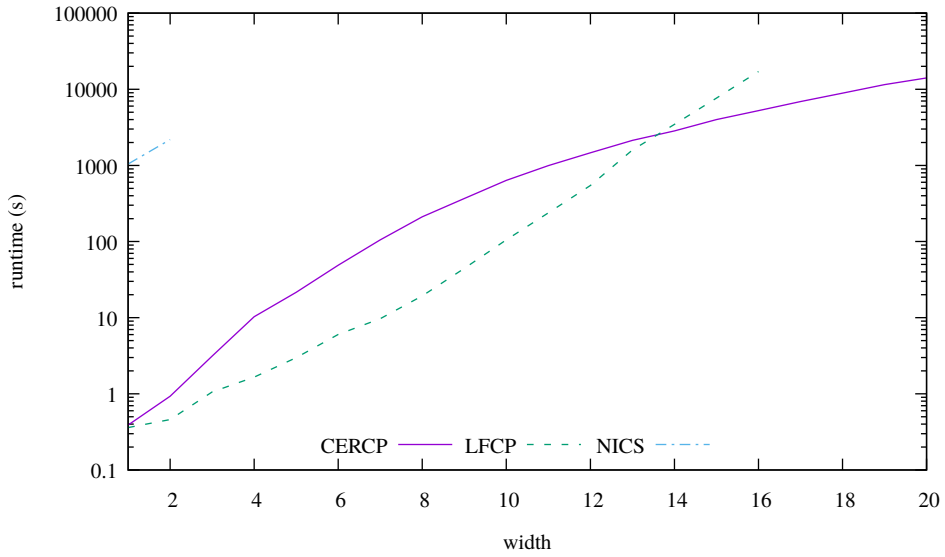


FIGURE 5.4. – Les temps d'exécution (en secondes) de CERCP, LFCP et NICS pour les benzénoïdes rectangulaires de hauteur 5 et de largeur variant entre 1 et 20.

structures de Kekulé. On remarque également que le nombre de structures de Kekulé d'un benzénoïde tend à augmenter quand ce dernier est plus compact. En parallèle, le nombre de circuits augmente plus lentement, ce qui rend CERCP encore utilisable. Si l'on considère maintenant l'ensemble B_2, nous remarquons que les temps d'exécution des deux méthodes sont similaires. Cela peut encore une fois s'expliquer par le fait que les catacondensés sont des molécules peu compactes et, à ce titre, ils admettent peu de structures de Kekulé. En revanche, les benzénoïdes de forme rectangulaire (ensemble $\mathscr{B}_3$) sont par définition des molécules très compactes. Pour ces derniers, CERCP semble être plus lent que LFCP dans un premier temps (voir la figure 5.4). Cependant, après une certaine taille, CERCP devient beaucoup plus rapide que LFCP. Cela peut être expliqué par le fait que le nombre de structures de Kekulé des benzénoïdes rectangulaires de grande taille devient très élevé. Comme pour l'ensemble $\mathscr{B}_1$, l'énumération des structures de Kekulé devient beaucoup trop coûteuse à partir d'un certain point. Par exemple, LFCP a eu besoin d'environ 23 300 s pour traiter le benzénoïde rectangulaire de dimensions 5×14 tandis que CERCP a pu traiter celui de dimensions 5×16 dans un délai similaire (22 516 s). De plus, sur l'ensemble des molécules considérées, CERCP n'a jamais dépassé la limite de temps de 24 heures que nous avions fixée.

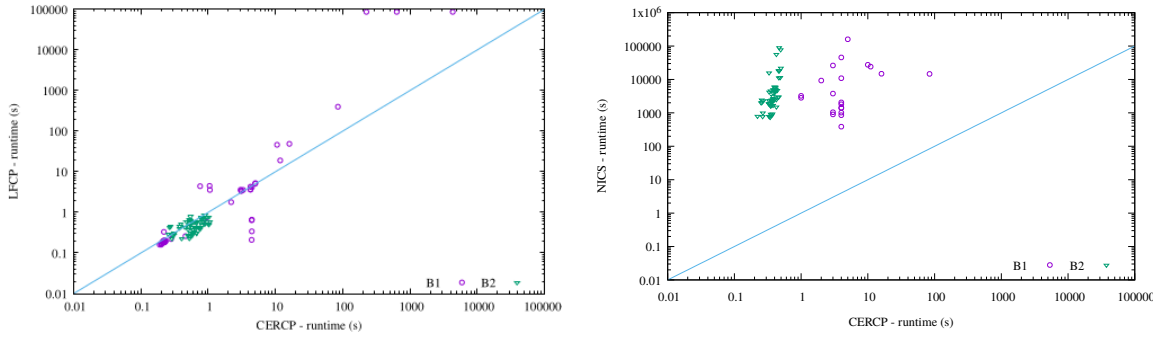


FIGURE 5.3. – Comparaisons des temps d'exécution sur les ensembles $\mathscr{B}_1$ et $\mathscr{B}_2$: CERCP vs LFCP (a) et CERCP vs NICS (b).

Intéressons-nous maintenant à la qualité des estimations de l'aromaticité pour CERCP et LFCP. On se

Id	n	Nombre de structures de Kekulé	Temps			coef.
			CERCP	LFCP	NICS	
1	1	2	0,19	0,16	51,8	-
2	2	3	0,20	0,16	148,0	-
3	3	5	0,20	0,17	1 110,0	1,0
4	3	4	0,20	0,17	293,6	1,0
5	4	5	0,21	0,18	480,4	1,0
6	4	8	0,21	0,18	1 746,0	1,0
7	4	9	0,21	0,19	1 483,0	1,0
8	4	6	0,22	0,18	571,2	1,0
9	4	7	0,21	0,18	2 150,8	0,74
10	5	6	0,22	0,33	734,2	1,0
11	5	9	0,22	0,19	5 276,6	0,51
12	5	10	0,22	0,19	3 936,0	0,98
13	5	12	0,22	0,20	4 521,8	0,90
14	5	11	0,22	0,20	9 714,8	0,69
15	5	13	0,22	0,20	1 0974,6	1,0
16	5	13	0,23	0,20	3 294,4	0,80
17	5	14	0,23	0,20	2 9483,2	0,95
18	5	11	0,23	0,19	1 873,2	0,86
19	5	9	0,23	0,19	3 992,0	0,89
20	5	9	0,23	0,19	867,2	1,0
21	7	20	0,28	0,22	4 562,4	1,0
22	13	250	0,95	0,58	5 890,0	0,91
23	20	3 250	4,47	4,15	10 910,2	0,90
24	19	3 100	3,04	3,60	26 270,6	0,84
25	24	16 100	11,78	18,97	24 152,0	0,90
26	25	34 560	10,57	46,10	27 462,4	0,75
27	11	25	0,46	0,26	3 545,8	0,97
28	17	1 320	2,20	1,78	9 299,0	0,88
29	8	34	4,47	0,66	1 399,0	0,81
30	8	31	4,43	0,64	45 715,2	0,98
31	8	19	4,43	0,21	1 425,4	0,95
32	8	16	4,46	0,34	386,2	1,0
33	9	40	1,06	3,57	3 238,6	0,92
34	9	20	3,10	3,38	901,4	0,98
35	9	30	3,10	3,40	3 774,8	0,98
36	10	50	3,31	3,56	1 047,2	0,98
37	14	175	4,98	5,06	2 056,2	0,99
38	8	45	0,76	4,40	2 044,0	0,94
39	10	101	1,05	4,48	2 836,8	0,94
40	6	14	4,26	4,29	1 019,8	0,96
41	6	10	4,24	3,64	854,0	0,98
42	6	14	4,27	3,65	1 807,4	0,88
43	13	432	5,04	5,23	TO	-
44	61	267 227 532	635,58	TO	TO	-
45	49	126 672 896	225,23	TO	TO	-
46	30	27 508	16,10	48,60	14 798,8	0,47
47	37	232 848	84,99	399,15	14 683,8	1,0
48	109	53 930 238 785 494 000	4 324,50	TO	TO	-

TABLEAU 5.2. – Nombre d'hexagones, nombre de structures de Kekulé, temps d'exécution (en secondes) de CERCP, LFCP et NICS, et valeur du coefficient de corrélation de Pearson entre CERCP et NICS pour les molécules de $\mathscr{B}_1$.

propose de calculer le coefficient de corrélation de Pearson entre les résultats obtenus par les deux méthodes. Pour n'importe quel benzénoïde des ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$, le coefficient a une valeur comprise entre 0,9997 et 1. En d'autres termes, la relation linéaire est quasiment parfaite. On peut donc considérer que les deux méthodes produisent des résultats équivalents. Ces résultats étaient néanmoins prévisibles. En effet, les deux méthodes consistent à évaluer l'énergie de résonance de chaque hexagone du benzénoïde traité. L'unique différence entre les deux méthodes est que LFCP considère tous les circuits tandis que CERCP se restreint aux circuits de taille au plus 4. La contribution d'un circuit étant d'autant plus basse que sa taille est grande, les circuits de taille supérieure à 4 n'ont qu'une faible influence sur le résultat final.

5.4.3. Comparaisons entre CERCP et NICS

Étant donné que CERCP et LFCP ont obtenu une qualité de résultats quasiment identique, nous ne comparons ici que CERCP et NICS. Au niveau du temps d'exécution, la balance penche très clairement en faveur de CERCP qui se révèle bien plus efficace quel que soit le benzénoïde et le benchmark considéré (voir les figures 5.3(b) et 5.4). Par exemple, CERCP a traité la molécule 26 de $\mathscr{B}_1$ en 10,57 s, contre 27 462 s pour NICS. Pour certains benzénoïdes, notamment les plus benzénoïdes rectangulaires, NICS dépasse la limite de temps de 24h que nous avions fixée et peut avoir besoin de plusieurs jours de calcul. D'une certaine manière, ces résultats étaient également prévisibles, car NICS repose sur des calculs de chimie quantique bien plus coûteux que les algorithmes utilisés par CERCP.

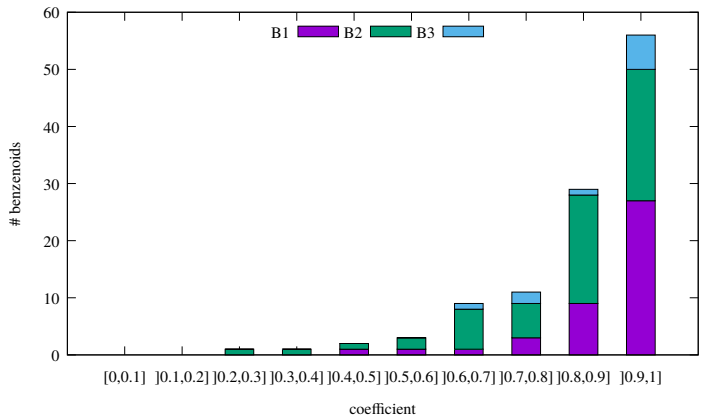


FIGURE 5.5. – Le coefficient de corrélation de Pearson entre CERCP et NICS pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

On se propose maintenant de se pencher sur la qualité de l'estimation de l'aromaticité des deux méthodes. Ici, nous considérons encore une fois le coefficient de corrélation de Pearson, mais cette fois-ci entre les résultats obtenus par CERCP et NICS (voir la figure 5.5 et la table 5.2).La figure 5.5 fournit le nombre de benzénoïdes pour différents intervalles de valeurs du coefficient pour les trois ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$. Sur les 112 instances comparables, 56 (50%) ont un coefficient supérieur ou égal à 0,9, 85 (environ 76%) en ont un supérieur ou égal à 0,8 et 96 d'entre elles (environ 86%) admettent un coefficient supérieur ou égal à 0,7. Si l'on regarde les résultats plus en détails, on observe que la valeur du coefficient n'est pas liée à la taille du benzénoïde. En revanche, la forme semble avoir une certaine influence comme on peut le voir dans la figure 5.5. Pour conclure, on peut dire que CERCP et NICS produisent des résultats similaires dans la grande majorité des cas et donc que les chimistes peuvent tout à fait exploiter CERCP pour estimer l'aromaticité d'un benzénoïde bien plus rapidement qu'en utilisant NICS, le tout en ayant une estimation satisfaisante.

Notons que les résultats présentés ici sont différents de ceux présents dans l'article qui a été présenté au journal Constraints (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2022). En effet, nous nous sommes rendus compte que les coefficients de corrélation entre CERCP et NICS chutaient considérablement pour

les benzénoïdes rectangulaires (ensemble $\mathscr{B}_3$) quand la largeur de ces derniers augmentait. Cela était dû à la base utilisée par la méthode NICS (B3LYP-631G). Cette dernière permet des calculs plus rapides, mais produit en contrepartie des résultats de précision moindre. Pour éviter cela, nous avons choisi d'utiliser la base def2TZVP dans ces travaux, qui fournit des résultats bien plus précis, mais, accroît le coût des calculs. Les méthodes CERCP et NICS semblent maintenant avoir la même tendance sur les benzénoïdes de $\mathscr{B}_3$ comme le montre l'histogramme décrit dans la figure 5.5. En revanche, seuls les deux plus petits ont pu être calculés en moins de 24 heures (à titre d'information, le benzénoïde rectangulaire de taille 5×10 a nécessité presque deux mois de temps CPU). Cela explique l'absence de temps pour la méthode NICS dans la figure 5.4.

5.5. Conclusions et perspectives

Dans ce chapitre, nous nous sommes intéressés à un problème fondamental de chimie théorique, à savoir l'estimation de l'aromaticité d'un benzénoïde donné. Nous avons présenté LFCP, une implémentation de la méthode de Lin&Fan (LIN et FAN 1999) et CERCP, une méthode basée sur celle proposée par Lin (LIN 2000) utilisant la programmation par contraintes. Ces deux méthodes servent à calculer l'énergie de résonance d'un benzénoïde donné. Dans notre cas, nous nous sommes intéressé principalement à l'énergie de résonance locale. Pour rappel, cette dernière est plus utile que sa variante globale, car elle permet de prédire sur quelles parties du benzénoïde traité les réactions chimiques ont le plus de chance d'avoir lieu. Nous avons effectué une série d'expérimentations sur un large ensemble de molécules afin de comparer cette méthode à NICS, la méthode de référence actuelle des chimistes théoriciens et nous avons obtenu des résultats intéressants. En effet, CERCP est d'une part bien plus rapide que NICS en termes de temps d'exécution. D'autre part, nous avons vu que les deux méthodes proposaient des résultats de qualité similaire et donc qu'il était tout à fait cohérent de les comparer. Ces travaux ont permis de mettre en lumière un nouveau domaine d'application de la programmation par contraintes et ont donné lieu à la publication de deux articles : le premier dans la conférence internationale CP en 2020 (CARISSAN, DIM, HAGEBAUM-REIGNIER et al. 2020) et le second dans le journal Constraints en 2022 (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2022).

Pour ce qui est des perspectives, la sous-section 7.3.1 explique qu'il est possible d'utiliser le calcul d'énergie de résonance afin d'obtenir des benzénoïdes entièrement aromatiques à partir de benzénoïdes qui le sont partiellement. Étant donné que ces benzénoïdes possèdent des propriétés intéressantes (MISHRA, BEYER, EIMRE et al. 2019 ; DUMSLAFF, GU, PATERNÒ et al. 2020 ; MATSUOKA, ITO, SARLAH et al. 2021), creuser cette piste constitue une perspective intéressante. Une autre perspective intéressante serait de ne calculer l'énergie de résonance locale que d'un hexagone de chacune des classes de symétries du benzénoïde considéré. Au cours de cette thèse, nous avons essayé de faire cela pour améliorer CERCP mais les résultats obtenus n'étaient pas du tout satisfaisants. Cela était dû au fait qu'une multitude cycles inutiles était générée pour chaque hexagone. Pour finir, une perspective plus générale serait d'étendre ces expérimentations à d'autres ensembles de benzénoïdes.

6. Outils implémentés

Dans ce chapitre, nous présentons les différents outils mis au point pendant cette thèse qui ont été mis à disposition de la communauté des chimistes. Ici, le premier objectif est de regrouper tous les travaux qui ont été effectués (liés à la génération de structures de benzénoïdes ainsi qu'à l'estimation de l'aromaticité) sous la forme d'un logiciel proposant une interface intuitive, et qui soit facilement utilisable par les chimistes, et ce, même dans le cas où ces derniers n'auraient aucune connaissance en programmation par contraintes. Le second objectif est de compléter les données présentes dans la *NASA Ames PAH IR Spectroscopic Database* mentionnée dans les travaux de Bouwman et al. (BOUWMAN, LINNARTZ et TIELENS 2021). Tout ceci a donné lieu à la création de deux outils distincts :

- le logiciel *BenzAI*, capable de générer des ensembles exhaustifs de structures de benzénoïdes selon les différents critères énoncés dans les chapitres 3 ou 4, de les visualiser et d'effectuer différents calculs dessus (notamment l'estimation de leur aromaticité via les méthodes décrites dans la section 2.4 et la méthode CERCP détaillée dans le chapitre 5.

- la base de données *BenzDB*, qui répertorie un certain nombre d'informations sur l'ensemble de tous les benzénoïdes possédant au plus neuf hexagones. On y retrouve des données permettant de les identifier facilement tels que leur identifiant chimique international (International Chemical Identifier / InChI), ou encore des informations issues de calculs coûteux comme leur carte IMS (définie dans la sous-section 2.4.3), leurs spectres infrarouges ou encore, les valeurs NICS (voir la sous-section 2.4.2) de leurs hexagones. Le fait de stocker ces informations permet aux chimistes de pouvoir les consulter facilement sans avoir à refaire ces calculs chronophages.

Ce chapitre sera donc consacré à la présentation de ces deux outils. La section 6.1 décrira le logiciel BenzAI tandis que la section 6.2 présentera la base de données BenzDB.

6.1. Le logiciel BenzAI

6.1.1. Introduction

Dans cette section, nous présentons BenzAI (VARET, PRCOVIC, TERRIOUX et al. 2022), un logiciel que nous avons développé au cours de cette thèse. Ce dernier inclut l'intégralité des travaux énoncés précédemment, à savoir la génération de structures de benzénoïdes selon tous les critères listés dans les chapitres 3 et 4 ainsi que le calcul de l'énergie de résonance d'un benzénoïde donné selon les méthodes CERCP et LFCP décrites dans le chapitre 5. En plus de cela, il inclut également un certain nombre d'autres fonctionnalités qui seront détaillées dans les sous-sections suivantes. Le but principal de BenzAI est de proposer aux chimistes théoriciens, mais aussi aux étudiants, une manière simple d'utiliser les outils que nous avons mis en œuvre. En effet, l'intégralité de l'aspect programmation par contraintes est caché derrière une interface intuitive. Ainsi, utiliser correctement ce logiciel ne requiert aucune compétence dans ledit domaine. Ce logiciel a été entièrement développé en Java, l'interface graphique a, quant à elle, été implémentée à l'aide de la bibliothèque JavaFX (disponible ici : https://openjfx.io/). BenzAI est disponible à l'adresse suivante : https://benzai-team.github.io/BenzAI/. Le site met à disposition des utilisateurs le code source du logiciel, des versions pré-compilées multiplateformes (Windows/Linux/MacOS) ainsi que de plus amples informations sur ses fonctionnalités. Une chaîne YouTube proposant différents tutoriels est également disponible à l'adresse https://www.youtube.com/channel/UCjJ19k3zEIBWdQDTPhQ-i3Q. La suite de cette section sera organisée de la manière suivante : la sous-section 6.1.2 décrira comment fonctionne le logiciel de manière générale et la sous-section 6.1.3 détaillera les différentes fonctionnalités de ce dernier.

6.1.2. Fonctionnement général de BenzAI

BenzAI permet de traiter des ensembles de benzénoïdes appelés *collections*. Ces dernières sont visualisables dans l'interface portant le même nom. Cette interface peut être vue comme le tableau de bord et la

partie principale du logiciel. La figure 6.1 nous montre une collection contenant l'intégralité des benzénoïdes de 5 hexagones. BenzAI propose différentes manières d'ajouter de nouveaux benzénoïdes à une collection :

- En utilisant le générateur, le processus complet de génération est détaillé dans la sous-sous-section 6.1.2.1.
- En les important depuis la base de données BenzDB que nous avons mise en ligne (voir la section 6.2). Cette dernière contient l'intégralité des benzénoïdes ayant au plus 9 hexagones ainsi qu'un certain nombre d'informations à leurs sujets qui seront détaillées dans la section 6.2.
- En les dessinant via l'interface prévue à cet effet (voir figure 6.2).
- En les important sous la forme d'un fichier .graph. Ce format texte que nous avons mis au point est inspiré du format DIMACS qui est utilisé pour représenter des graphes. Ce format respecte la syntaxe suivante. La première ligne est de la forme :

  ```
  p DIMACS nc nl nh
  ```

  Ici, `nc`, `nl` et `nh` correspondent respectivement au nombre de carbones, de liaisons C-C et d'hexagones du benzénoïde représenté. Cette ligne est ensuite suivie d'une série de lignes de la forme :

  ```
  e x1_y1 x2_y2
  ```

  Chacune de ces lignes spécifie l'existence d'une liaison entre les atomes situés aux positions (x_1, y_1) et (x_2, y_2). Pour finir, le fichier se termine par une série de lignes de la forme :

  ```
  h x1_y1 x2_y2 x3_y3 x4_y4 x5_y5 x6_y6
  ```

  Chacune de ces lignes spécifie, quant à elle, qu'un hexagone est formé par les sommets se trouvant aux positions $(x_1, y_1), (x_2, y_2), \ldots, (x_6, y_6)$. La figure 6.3 nous montre la manière dont BenzAI considère la molécule d'anthracène et la figure 6.4 décrit sa représentation dans le format .graph.
- Il est possible de déplacer ou copier des benzénoïdes d'une collection à une autre. BenzAI permet aussi de réaliser des opérations ensemblistes (union, intersection ou différence) entre deux collections pour produire une troisième collection.

Il est également possible d'effectuer des opérations de tri ou de filtrage. Une collection peut être triée selon les critères suivants (de manière croissante ou décroissante) :

- le nombre d'hexagones,
- le nombre de carbones,
- le nombre d'hydrogènes,
- le paramètre d'irrégularité,
- le nombre de structures de Kekulé.

L'opération de filtrage consiste à créer une nouvelle collection ne contenant que les molécules d'une collection initiale satisfaisant un critère spécifique. BenzAI propose de nombreux critères de filtrage :

- avoir un nombre d'hexagones donné,

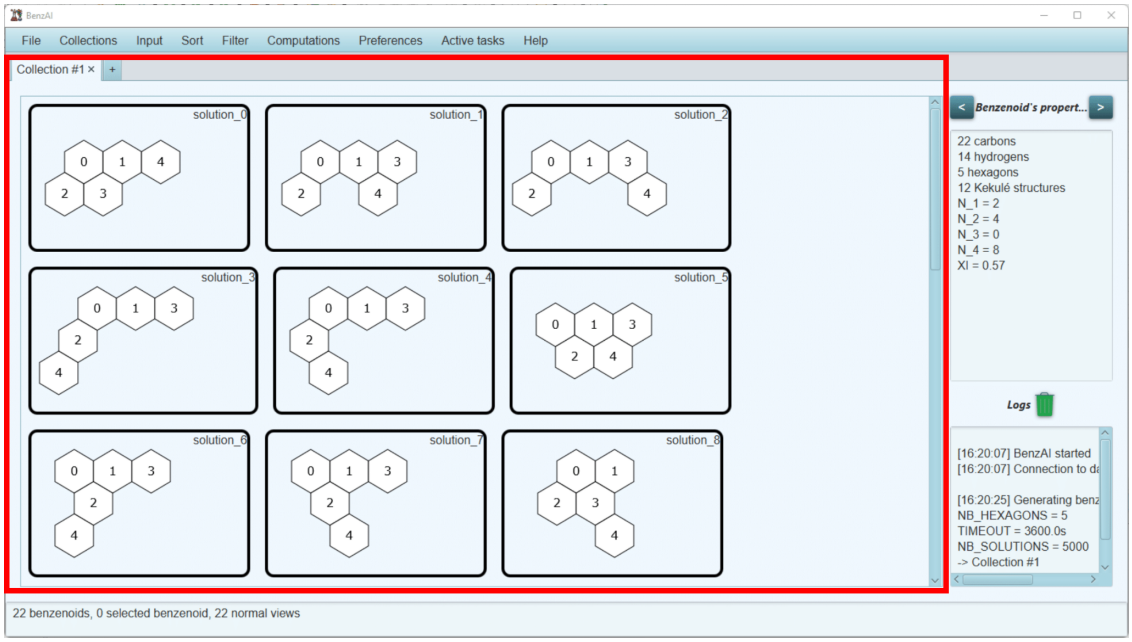


FIGURE 6.1. – Exemple de collection contenant l'intégralité des benzénoïdes de 5 hexagones (seuls les neufs premiers sont visibles ici)

- avoir un nombre de carbones donné,
- avoir un nombre d'hydrogènes donné,
- avoir un paramètre d'irrégularité donné,
- avoir un nombre de structures de Kekulé donné,
- avoir un diamètre donné,
- être un coronénoïde,
- être un coronoïde,
- être un benzénoïde catacondensé,
- avoir une forme rectangulaire ou rhombique,
- admettre une symétrie donnée,
- admettre une ou plusieurs propriétés de motif,
- être entièrement aromatiques (c'est-à-dire avoir chaque hexagone qui admet une énergie de résonance locale non nulle),
- être partiellement aromatiques (c'est-à-dire avoir au moins un hexagone admettant une énergie de résonance nulle),
- être un benzénoïde non kekuléen implicite (voir la définition 2.2.3).

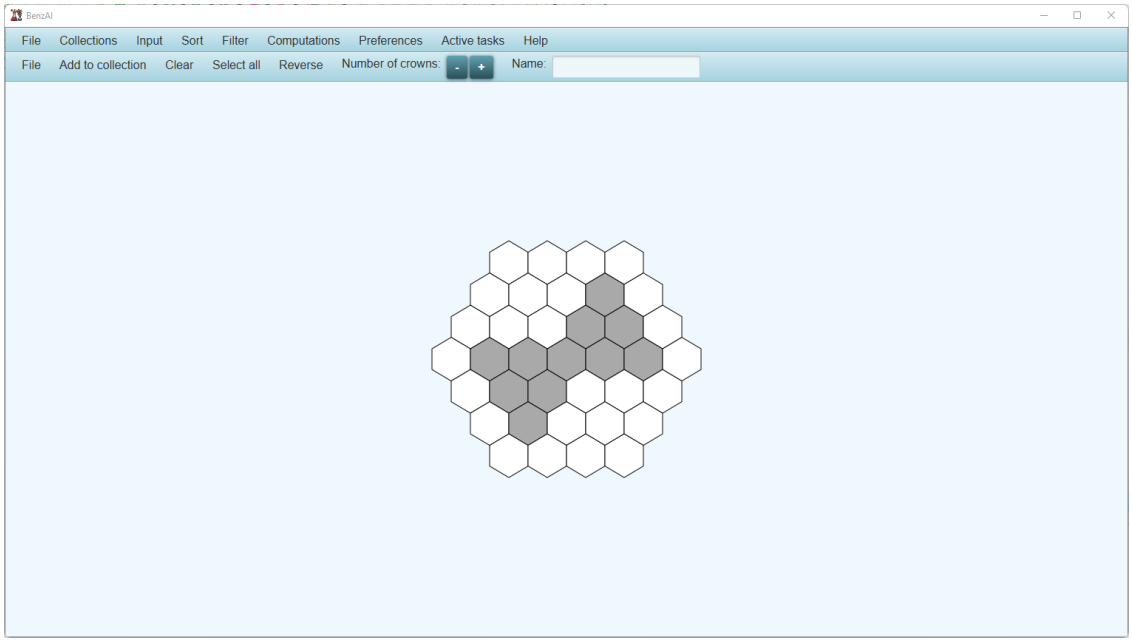


FIGURE 6.2. – Interface de dessin de BenzAI.

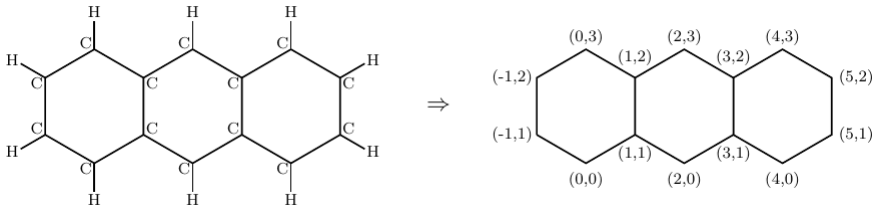


FIGURE 6.3. – Structure de l'anthracène interprétée par BenzAI

6.1.2.1. Génération de structures de benzénoïdes

Ici, nous présentons en détail le processus de génération de notre logiciel BenzAI. Ce dernier peut être schématisé sous la forme du diagramme de flux décrit dans la figure 6.5. On se propose maintenant de détailler chacune des différentes opérations décrites dans ce diagramme :

1. L'utilisateur saisit les différents critères de génération dans l'interface prévue à cet effet.

2. BenzAI crée l'instance CSP correspondant aux critères spécifiés par l'utilisateur. L'instance CSP est créée selon les différents modèles décrits dans les sections 3.3 et 3.4.

3. L'instance créée est résolue par Choco Solver.

4. À la fin de la résolution, Choco Solver renvoie un ensemble de solutions S correspondant à l'ensemble des structures de benzénoïdes générées.

5. BenzAI vérifie s'il y a besoin d'effectuer un post-traitement. Ce cas de figure arrive quand l'utilisateur spécifie un critère contraignant le nombre de structures de Kekulé des molécules générées (par exemple, avoir un nombre de structures de Kekulé donné ou être un benzénoïde non-kékuléen implicite). Considérons, par exemple, la figure 6.6 qui décrit les différents benzénoïdes obtenables en partant d'un benzénoïde de taille 5. Le nombre de structures de Kekulé de chacun d'entre eux est

```
p DIMACS 14 16 3
e 0_0 1_1
e 0_0 -1_1
e 1_1 1_2
e 1_1 2_0
e -1_1 -1_2
e 1_2 0_3
e 1_2 2_3
e 2_0 3_1
e 0_3 -1_2
e 2_3 3_2
e 3_1 3_2
e 3_1 4_0
e 3_2 4_3
e 4_0 5_1
e 4_3 5_2
e 5_1 5_2
h 0_0 1_1 1_2 0_3 -1_2 -1_1
h 2_0 3_1 3_2 2_3 1_2 1_1
h 4_0 5_1 5_2 4_3 3_2 3_1
```

FIGURE 6.4. – Fichier .graph représentant l'anthracène.

spécifié. Nous pouvons facilement voir que l'ajout d'un unique hexagone peut totalement modifier le nombre de structures de Kekulé de la molécule obtenue. Il est donc relativement compliqué de contraindre le nombre de structures de Kekulé d'un benzénoïde directement pendant la génération, le fait que cette propriété soit globale la rendant détectable tardivement (ce qui limite grandement l'efficacité d'un propagateur spécifique). De plus, la méthode de Rispoli décrite dans la section 2.2 permettant de compter le nombre de structures de Kekulé d'un benzénoïde semble difficilement transposable sous forme de contrainte. L'utilisation d'un post-traitement semble donc être préférable ici.

6. Si un post-traitement est nécessaire, BenzAI va l'effectuer en supprimant de l'ensemble de solutions S tous les benzénoïdes ne satisfaisant pas les critères.

7. À la fin du post-traitement, un nouvel ensemble de solutions S' est obtenu. Ce dernier correspond à l'ensemble de molécules satisfaisant tous les critères spécifiés par l'utilisateur.

8. BenzAI affiche la collection correspondant à l'ensemble de solutions S (où S' si un post-traitement a été effectué).

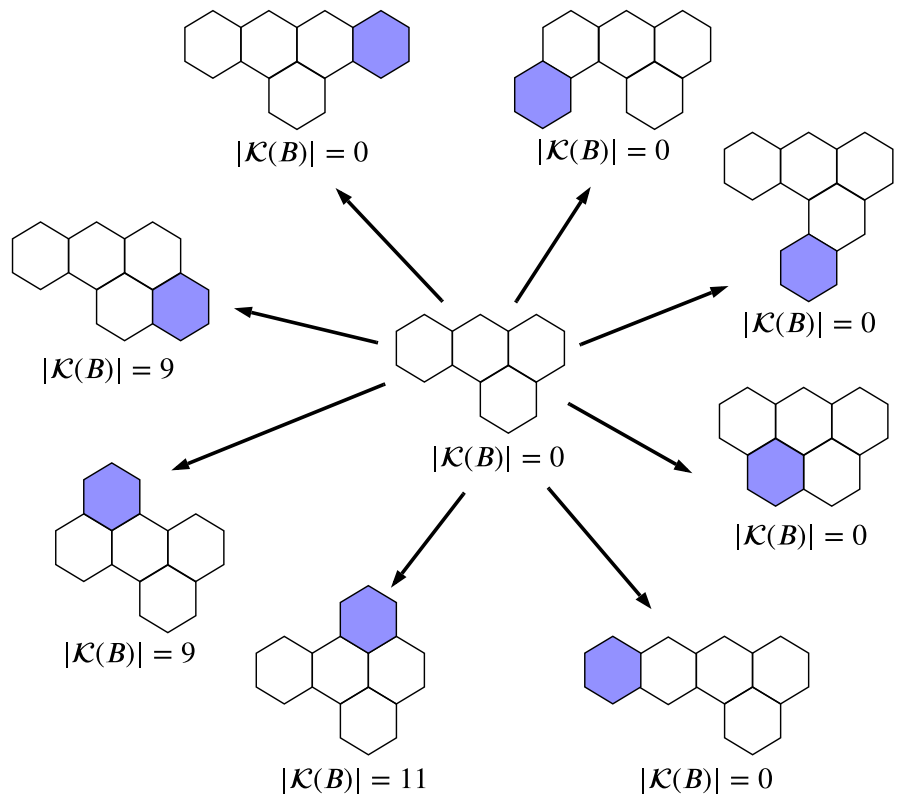


FIGURE 6.6. – Évolution du nombre de structures de Kekulé pendant la génération

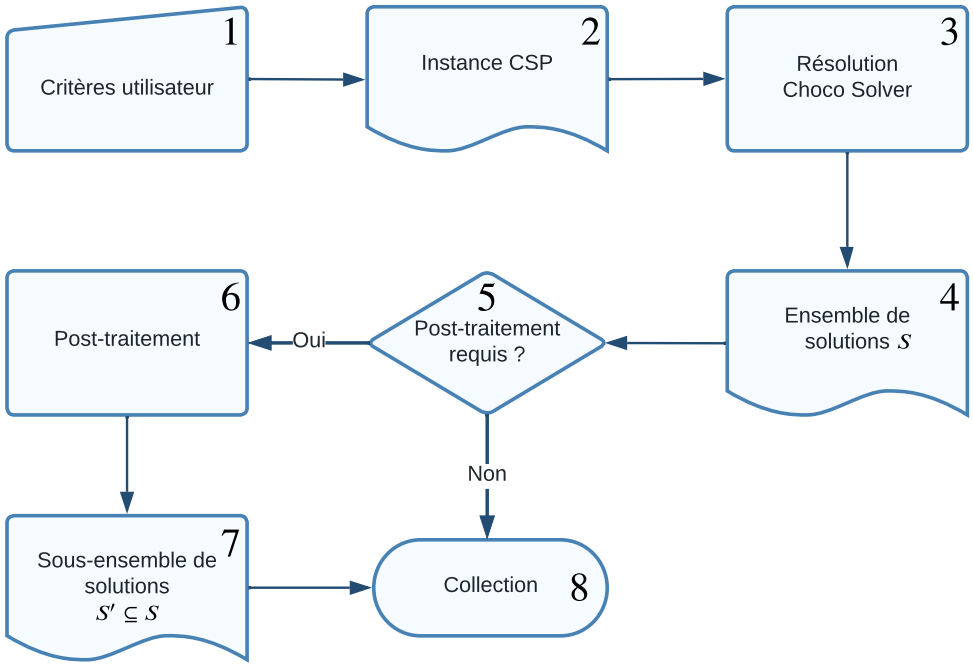


FIGURE 6.5. – Diagramme de flux du processus de génération de BenzAI

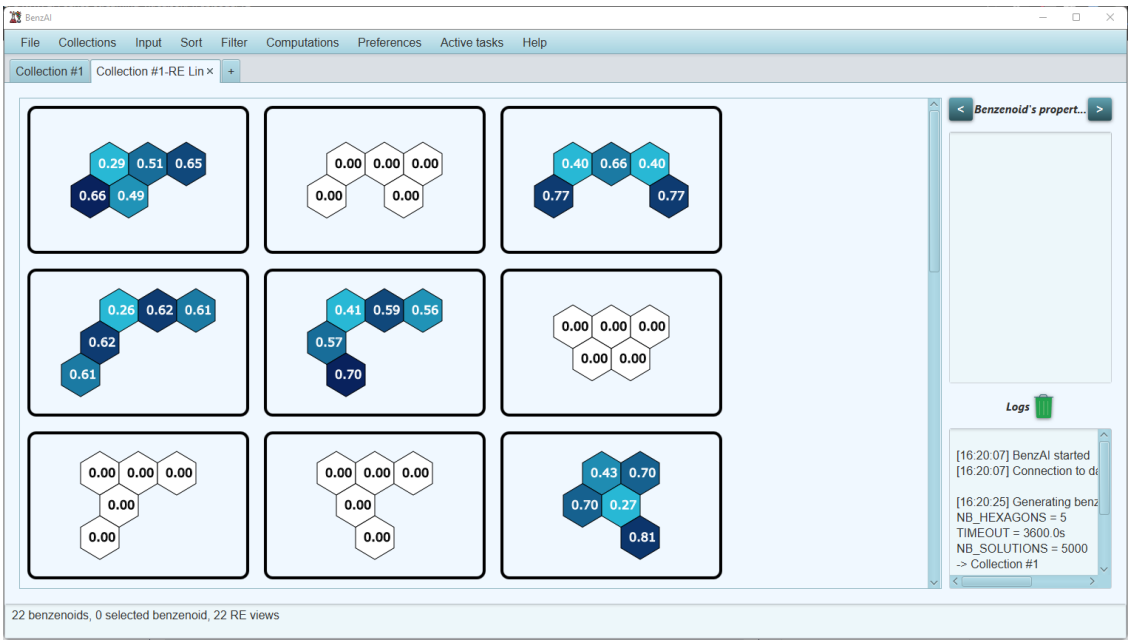


FIGURE 6.7. – Collection contenant l'énergie de résonance des benzénoïdes de 5 hexagones.

6.1.3. Fonctionnalités de BenzAI

6.1.3.1. Calculer l'énergie de résonance d'un benzénoïde

BenzAI permet à l'utilisateur de calculer l'énergie de résonance (notion définie dans le chapitre 2) d'un ou plusieurs benzénoïdes ayant été au préalable placés dans une collection via une des méthodes décrites dans la section précédente. Pour faire ces calculs, l'utilisateur a le choix entre les méthodes CERCP et LFCP qui ont été explicitées dans le chapitre 5. Une fois les calculs terminés, une nouvelle collection détaillant les résultats obtenus est créée. La figure 6.7 nous montre la collection obtenue lorsque l'on calcule l'énergie de résonance des benzénoïdes à 5 hexagones à l'aide de la méthode CERCP.

6.1.3.2. Énumérer les structures de Kekulé d'un benzénoïde

Ce logiciel est également capable d'énumérer l'intégralité ou une partie des structures de Kekulé d'un benzénoïde présent dans une collection. Pour cela, il utilise le modèle P_2 décrit dans la sous-section 5.2. Étant donné que le nombre de structures de Kekulé d'un benzénoïde est potentiellement exponentiel, nous avons choisi de limiter par défaut à 20 le nombre de structures calculées et affichées. Cette valeur est modifiable à volonté par l'utilisateur dans une interface prévue à cet effet. La figure 6.8 nous montre une collection contenant l'ensemble des structures de Kekulé d'un benzénoïde possédant 5 hexagones.

6.1.3.3. Calculer une couverture de Clar d'un benzénoïde

BenzAI permet également à l'utilisateur de calculer une couverture de Clar (notion définie dans le chapitre 2) d'un ou plusieurs benzénoïdes présents dans une collection. Le calcul de cette couverture va encore une fois être effectué à l'aide de la programmation par contraintes. En revanche, nous allons cette fois-ci considérer une instance COP (définie dans le chapitre 1). Pour rappel, une instance COP correspond à une instance CSP à laquelle nous ajoutons une fonction objectif f que nous pouvons soit minimiser, soit maximiser. Cette fonction est de la forme :

$$f : D_{x_1} \times D_{x_2} \times \ldots \times D_{x_n} \rightarrow \mathbb{Q}$$

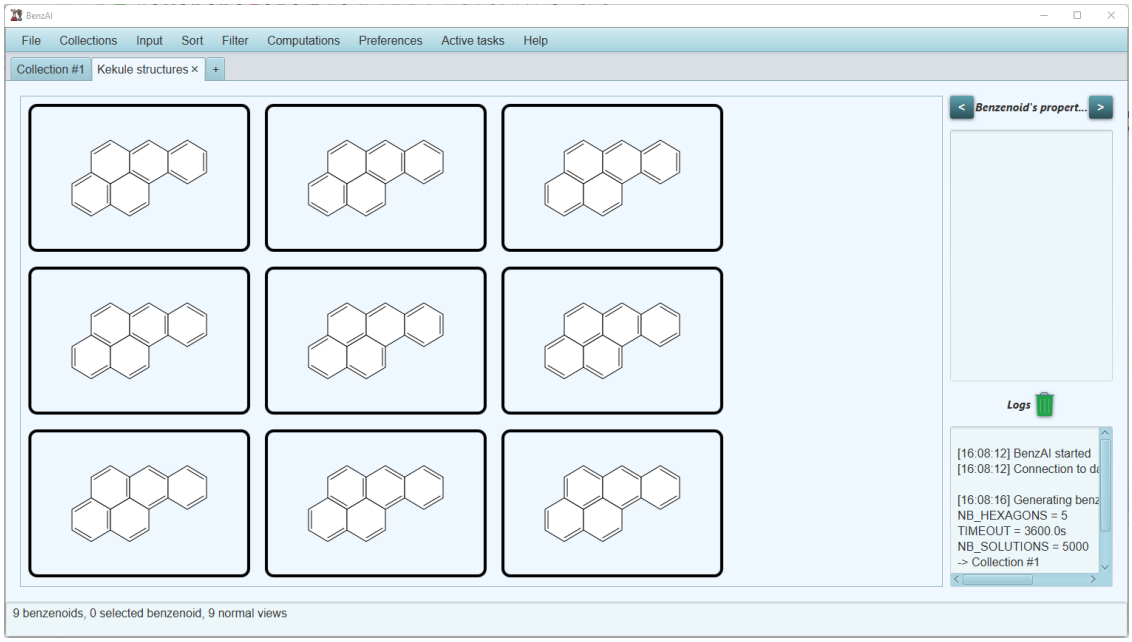


FIGURE 6.8. – Collection contenant les structures de Kekulé de l'un des benzénoïdes de 5 hexagones.

Étant donné un benzénoïde $B = (V, E)$, nous construisons l'instance COP suivante : au niveau des variables, nous considérons un ensemble de variables booléennes $X_r = \{x_{r_1}, \ldots, x_{r_n}\}$ (n étant le nombre d'hexagones de B). La variable x_{r_i} sera égale à 1 si l'hexagone i admet un rond de Clar, 0 sinon. Nous ajoutons également un second ensemble de variables booléennes $X_\ell = \{x_{\ell_{i,j}} | \{i, j\} \in E\}$. La variable $x_{\ell_{i,j}}$ vaudra 1 si la liaison entre les carbones i et j est double, et 0 si elle est simple. Nous considérons ensuite un dernier ensemble de variables booléennes $X_e = \{x_{e_1}, \ldots, x_{e_{|V|}}\}$. La variable x_{e_i} vaut 1 si le carbone i possède un électron célibataire, 0 sinon. Ensuite, nous définissons un ensemble X_{s_i} pour chaque carbone i de B qui contiendra :

- les variables de X_ℓ correspondant aux arêtes impliquant le carbone i,
- les variables de X_r correspondant aux hexagones impliquant le carbone i,
- la variable x_{e_i}.

Autrement dit, pour chaque carbone i de B, on a $X_{s_i} = \{x_{\ell_{i,j}} | \{i, j\} \in E\} \cup \{x_{r_h} | i\ figure\ dans\ l'hexagone\ h\} \cup \{x_{e_i}\}$. Pour chaque ensemble X_{s_i}, nous posons ensuite une contrainte $\texttt{Sum}(X_{s_i}, =, 1)$ spécifiant qu'une et une seule de ces variables peut prendre la valeur 1. En effet, d'un point de vue chimique, un atome de carbone ne peut pas participer à une liaison double (que ce soit au sein d'un rond de Clar ou non) et avoir un électron célibataire en même temps. De plus, une liaison double ne peut pas être comptabilisée deux fois (une fois dans un rond de Clar et une fois en dehors). Penchons-nous maintenant sur la partie optimisation de l'instance. Premièrement, nous considérons deux nouvelles variables entières : nb_ronds et $nb\_celibataires$ qui vont respectivement correspondre aux nombres de ronds et d'électrons célibataires de l'instance. Ces variables ont respectivement pour domaine $\{0, \ldots, n\}$ et $\{0, \ldots, n_c\}$ (où n_c est le nombre d'atomes de carbone de B). La correspondance entre ces deux variables et les variables précédemment introduites est établie à l'aide des contraintes $\texttt{Sum}(X_r, =, nb\_ronds)$ et $\texttt{Sum}(X_e, =, nb\_celibataires)$. Concernant l'objectif à optimiser, il s'agit d'abord de minimiser le nombre d'électrons célibataires, puis de maximiser le nombre de ronds de Clar de la couverture. Nous spécifions, pour cela, la fonction objectif f à maximiser suivante :

$$f : (nb\_ronds, nb\_celibataires) \longmapsto nb\_ronds - n \times nb\_celibataires$$

Cette fonction revient donc à maximiser le nombre de ronds de la couverture de Clar générée tout en minimisant le nombre d'électrons célibataires qu'elle admet. La multiplication du nombre d'électrons célibataires par le nombre n d'hexagones garantit que le critère sur le nombre d'électrons célibataires sera bien prioritaire sur celui du nombre de ronds. Ainsi, cela nous permet d'éviter de faire appel à une fonction objectif multicritère. La figure 6.9 nous montre une collection contenant une couverture de Clar pour chacun des benzénoïdes possédant 5 hexagones. Notons que certains benzénoïdes peuvent admettre plusieurs couvertures de Clar. Dans ce cas, on se contente d'afficher la première couverture générée minimisant le nombre d'électrons radicalaires. BenzAI est également capable de déterminer les *liaisons fixes* d'une couverture de Clar. Pour rappel, une liaison est dite fixe si elle est exclusivement simple ou double dans l'intégralité des couvertures de Clar du benzénoïde. Pour cela, il suffit de générer l'intégralité des couvertures de Clar du benzénoïde traité, et de regarder s'il existe des liaisons qui sont soit tout le temps simples, soit tout le temps doubles. La figure 6.10 nous montre les mêmes couvertures de Clar que l'exemple précédent, mais cette fois-ci en affichant les liaisons fixes. Ici, les liaisons restant simples sont représentées en bleu tandis que celles restant doubles sont représentées en rouge.

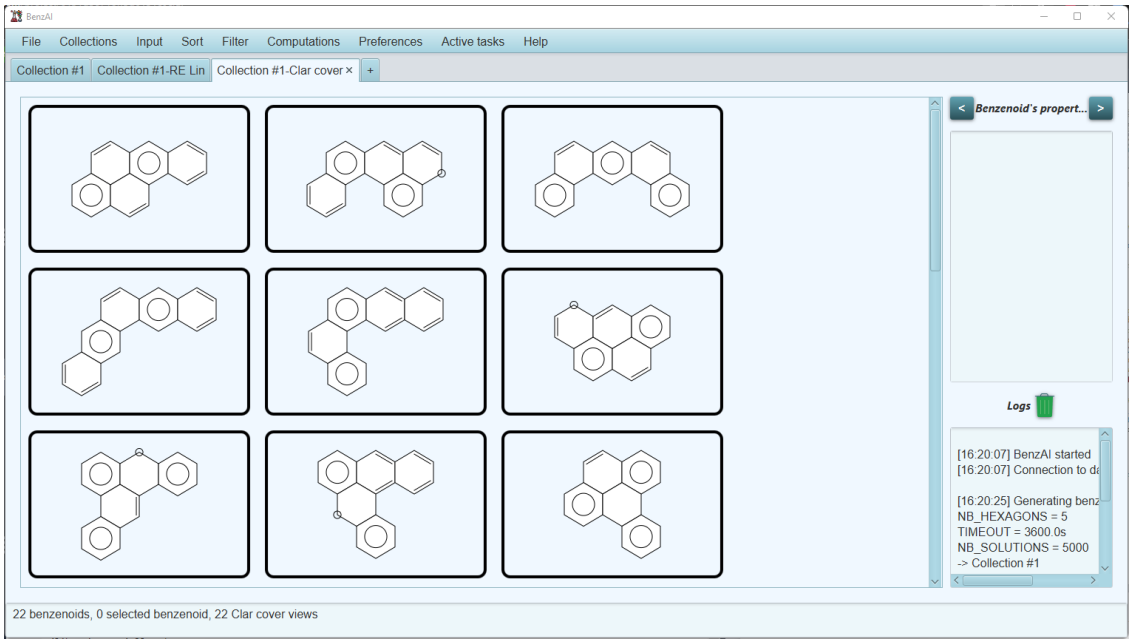


FIGURE 6.9. – Collection contenant les couvertures de Clar des benzénoïdes de 5 hexagones.

6.1.3.4. Calculer les Ring Bond Order (RBO) d'un benzénoïde

Une autre fonctionnalité de BenzAI est de calculer les Ring Bond Order (RBO) (notion définie dans le chapitre 2) d'un ou plusieurs benzénoïdes. Pour rappel, calculer les RBO d'un benzénoïde B consiste à calculer le *bond order* de chacune de ses liaisons qui est égal au ratio du nombre de structures de Kekulé de B dans lesquelles la liaison est double sur le nombre total de structures de Kekulé de B. Il faut ensuite attribuer à chaque hexagone une valeur qui sera obtenue en faisant la somme des bond orders des six liaisons participant à l'hexagone.

Pour chaque liaison du benzénoïde dont on veut calculer les RBO, BenzAI va supprimer cette liaison du graphe représentant le benzénoïde, puis appliquer l'algorithme de nettoyage de graphe décrit dans le chapitre 5 (sous-section 5.3) et va ensuite appliquer la méthode de Rispoli (explicitée dans le chapitre 2) sur le graphe obtenu afin d'obtenir le nombre de couplages parfaits qu'il admet. Il suffit ensuite de diviser la valeur obtenue par le nombre total de structures de Kekulé pour obtenir les bond order de chaque liaison, et par extension les RBO de chaque hexagone. La figure 6.11 nous montre une collection contenant les RBO de

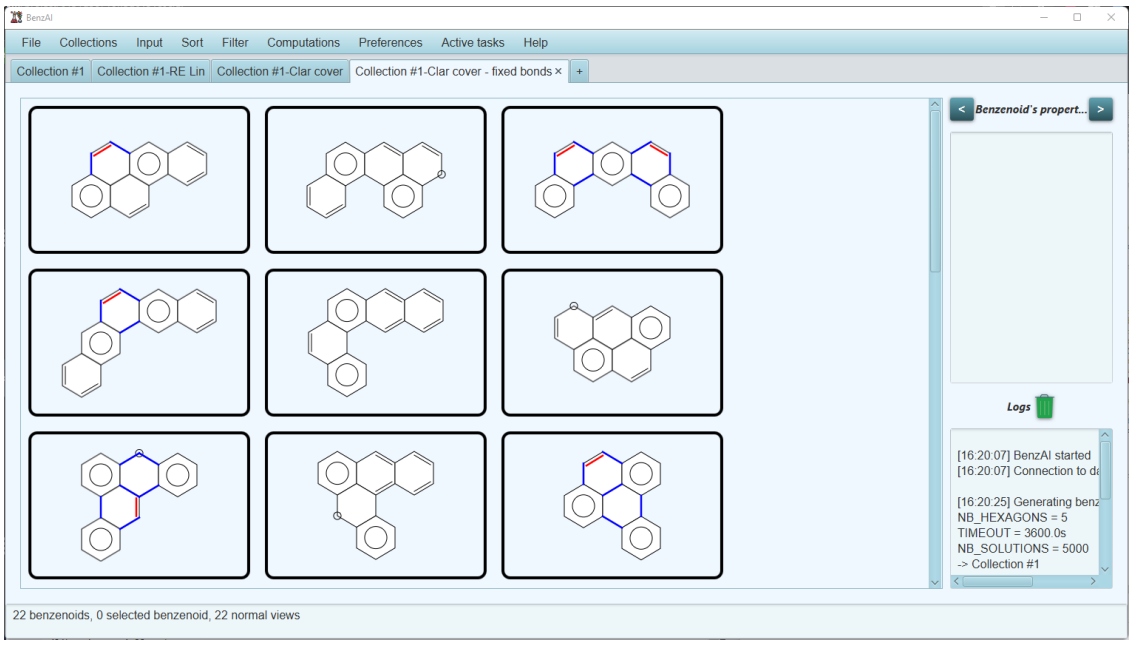


FIGURE 6.10. – Collection contenant les couvertures de Clar (avec liaisons fixes) des benzénoïdes de 5 hexagones.

tous les benzénoïdes ayant 5 hexagones.

6.1.3.5. Statistiques sur les paramètres d'irrégularité des benzénoïdes d'une collection

BenzAI est également capable d'effectuer des statistiques sur les paramètres d'irrégularités des benzénoïdes présents dans une collection. Comme mentionné dans le chapitre 2, la NASA possède une base de données listant un grand nombre de HAP ainsi que certaines de leurs propriétés chimiques. En 2019, une étude (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) effectuant des statistiques sur les paramètres d'irrégularité des benzénoïdes présents dans cette base a été publiée. Or, nous nous sommes rendus compte que cette base était incomplète. La garantie d'exhaustivité de notre générateur peut donc nous permettre d'effectuer des statistiques plus précises et complètes. C'est principalement ce point-là qui a motivé le développement de cette fonctionnalité.

Ainsi, étant donné une collection, il est possible d'afficher sous la forme de diagramme, le nombre de ses benzénoïdes ayant un paramètre d'irrégularité appartenant à un intervalle donné. Il est bien évidemment possible de modifier les valeurs de ces intervalles. La figure 6.12 nous montre les statistiques sur les paramètres d'irrégularité des benzénoïdes possédant 5 hexagones.

6.1.3.6. Afficher les spectres infrarouges des benzénoïdes d'une collection

Comme mentionné plus haut, nous avons mis en ligne une base de données contenant l'intégralité des benzénoïdes possédant 9 hexagones ou moins. Pour chacun d'entre eux, un certain nombre d'informations sont disponibles et notamment leurs spectres infrarouges. Ces derniers ont été calculés avec le logiciel Gaussian en utilisant le niveau de théorie `6-31g/b3lyp`. BenzAI est capable d'interroger cette base afin d'afficher les données de spectres infrarouges de benzénoïdes présents dans une collection (ces derniers seront classifiés selon leur nombre de carbones et d'hydrogènes). La figure 6.13 montre l'interface de notre logiciel affichant les spectres infrarouges des benzénoïdes possédant 5 hexagones. Il convient de préciser qu'une courbe de spectre infrarouge ne concerne que les molécules admettant le même nombre de carbones

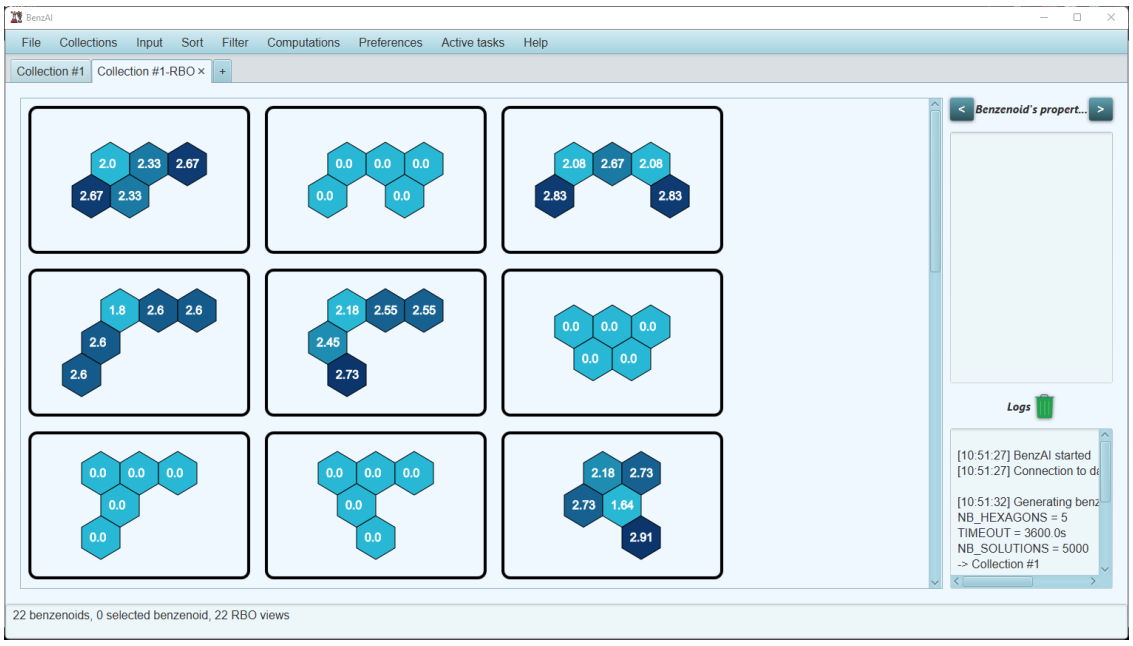


FIGURE 6.11. – Collection contenant les RBO des benzénoïdes à 5 hexagones.

et d'hydrogènes, celle décrite dans la figure 6.13 concerne les benzénoïdes possédant 20 carbones et 12 hydrogènes. Dans le cas des benzénoïdes de 5 hexagones, il existe 3 autres courbes possibles.

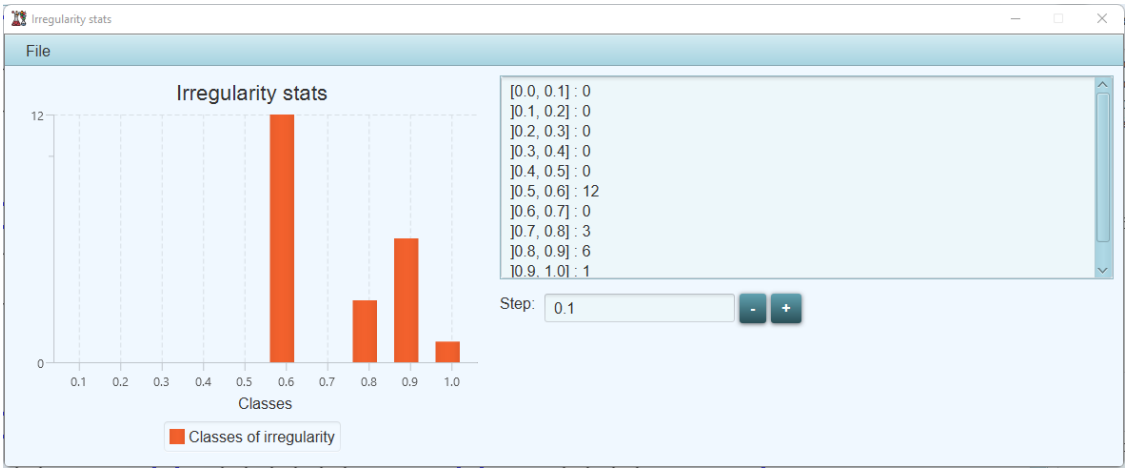


FIGURE 6.12. – Statistiques sur les paramètres d'irrégularité des benzénoïdes possédant 5 hexagones.

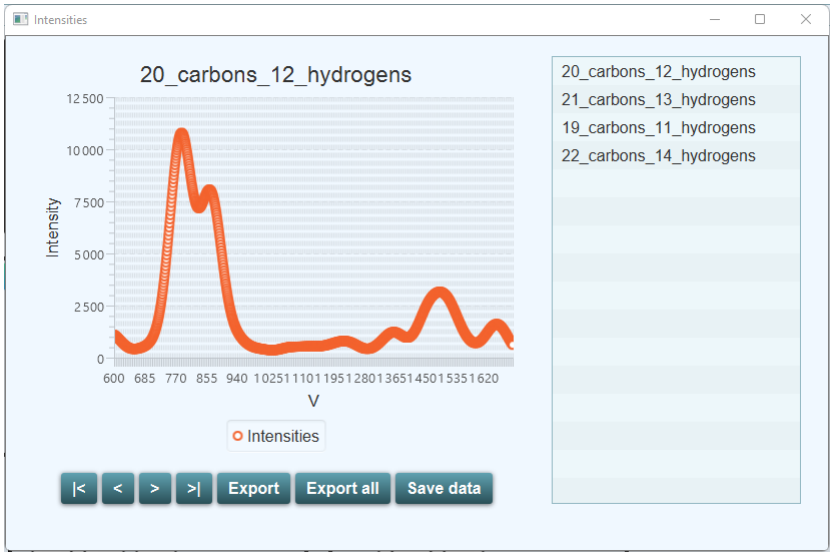


FIGURE 6.13. – Spectres infrarouges des benzénoïdes possédant 5 hexagones.

6.1.3.7. Statistiques sur les électrons radicalaires des benzénoïdes

Comme mentionné dans la définition 2.4.11, on dit qu'un électron est *radicalaire* si ce dernier ne participe ni à une liaison double, ni à un rond de Clar. Les travaux de Narita et al. en 2015 (NARITA, WANG, FENG et al. 2015) et ceux de Liu et Feng en 2020 (LIU et FENG 2020) ont permis de démontrer que la présence de ces électrons radicalaires induisent des propriétés optiques et magnétiques bien particulières. Il est donc intéressant d'être capable de déterminer quels atomes de carbones peuvent admettre ce type d'électrons, et à quelle fréquence. Notre logiciel BenzAI est capable d'effectuer ce type de statistiques, comme le montre la figure 6.14 pour les benzénoïdes possédant 5 hexagones. Ici, nous attribuons une valeur à chaque carbone d'un benzénoïde variant de 0 à 1 correspondant au rapport entre le nombre de fois ou ce carbone admet un électron radicalaire sur le nombre total de couvertures de Clar. La taille des ronds rouges présents sur les carbones dépend de cette valeur.

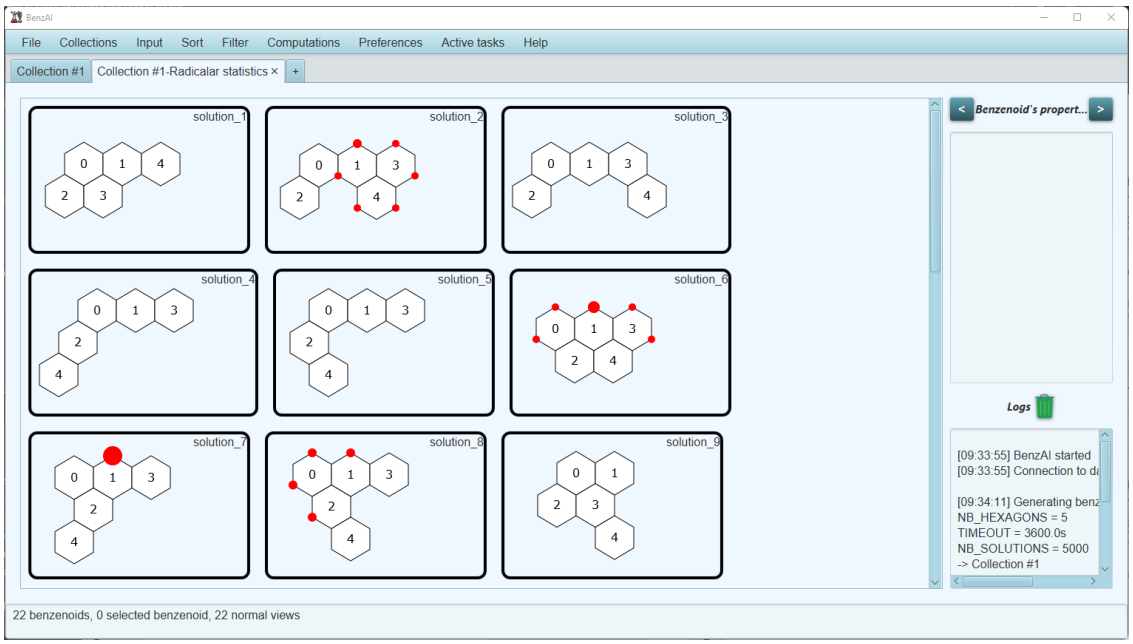


FIGURE 6.14. – Statistiques sur les électrons radicalaires des benzénoïdes de 5 hexagones.

6.1.3.8. Carte Isotropic Magnetic Shielding (IMS)

La méthode *Isotropic Magnetic Shielding (IMS)*, définie dans la sous-section 2.4.3 permet également de fournir une estimation de l'aromaticité d'un benzénoïde. Étant donné un benzénoïde, elle consiste à calculer les valeurs NICS (voir sous-section 2.4.2) sur un très grand nombre de ses points. Il en résulte une cartographie complète détaillant les zones du benzénoïde traité les plus aromatiques. Cette méthode permet d'avoir un point de vue global très précis de l'aromaticité d'un benzénoïde. En revanche, le grand nombre de calculs NICS requis la rend extrêmement coûteuse, le traitement d'un benzénoïde de taille moyenne pouvant facilement prendre plusieurs heures, voir plus d'une journée. Nous avons stocké dans la base de données BenzDB les images de ces cartes ainsi que les différentes valeurs numériques associées dans le but d'éviter aux utilisateurs d'avoir à faire ces calculs une seconde fois. Notre logiciel BenzAI est capable d'aller interroger cette base de données et de récupérer les images de ces cartes. La figure 6.15 nous montre une collection contenant la carte IMS du benzénoïde décrit dans la figure 6.16.

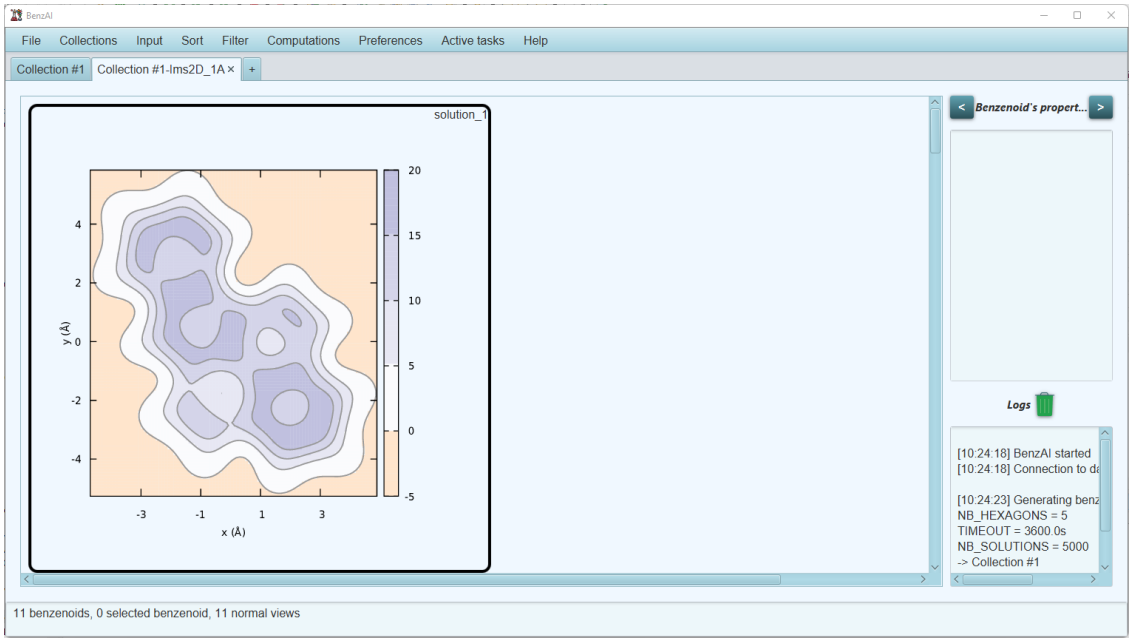


FIGURE 6.15. – Collection contenant la carte IMS d'un benzénoïde possédant 5 hexagones.

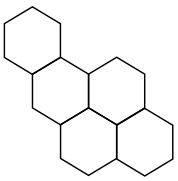


FIGURE 6.16. – Benzénoïde dont la carte IMS est affichée par BenzAI

6.2. La base de données BenzDB

6.2.1. Introduction

Dans cette section, nous présentons notre base de données baptisée *BenzDB*. L'objectif de cette dernière est de répertorier de manière exhaustive un large ensemble de benzénoïdes. Nous souhaitons mettre à

la disposition des chimistes des informations sur ces molécules issues de calculs très coûteux afin que ces derniers puissent éviter de les refaire dans le futur. Notre base porte sur l'intégralité des benzénoïdes possédant au plus 9 hexagones. Pour l'instant nous nous sommes restreint aux benzénoïdes d'au plus 9 hexagones principalement à cause du coût des calculs, mais nous n'excluons pas de l'agrandir dans le futur. Afin de s'assurer son exhaustivité, nous avons généré les benzénoïdes en utilisant le modèle général décrit dans la section 3.3. Pour chacun de ces benzénoïdes, BenzDB contient les informations suivantes :

- son nombre d'hexagones/carbones/hydrogènes,
- son paramètre d'irrégularité (voir la définition 2.2.5),
- sa représentation au format .graph (défini dans la sous-section 6.1.2),
- les coordonnées 3D de chacun de ses atomes,
- son *International Chemical Identifier (InChI)* (un identifiant unique pour chaque molécule (HELLER, MCNAUGHT, PLETNEV et al. 2015)),
- son identifiant interne (utilisé par BenzAI),
- son énergie totale et son spectre de vibration obtenus par le calcul de la théorie fonctionnelle de la densité au niveau de théorie B3LYP/6-31G,
- les valeurs NICS (voir la sous-section 2.4.2) calculées au centre de chaque hexagone avec le niveau de théorie B3LYP/6-311G**,
- sa carte *Induced Magnetic Shielding (IMS)* (définie dans la sous-section 2.4.3) calculée avec le niveau de théorie B3LYP/6-311++G(d,p).

D'un point de vue plus technique, la base de données est accessible via une API *Representational State Transfert (REST)*. Cette dernière peut être interrogée avec des requêtes GET/POST ou directement via notre logiciel BenzAI (voir section 6.1) qui propose une interface beaucoup plus intuitive.

Un des principaux avantages de notre base de données est qu'elle contient un grand nombre de résultats issus de calculs très coûteux. En effet, obtenir les valeurs nécessaires au calcul d'un spectre infrarouge, ou encore la carte IMS d'un benzénoïde nécessite des calculs très lourds qui peuvent facilement durer plusieurs heures, voir plus d'une journée. Avoir ces informations facilement accessibles permet aux chimistes de pouvoir les consulter sans avoir à refaire ces calculs. Pour finir, il convient également de mentionner que Alvarez-Ramírez et al. ont présenté en 2020 (ALVAREZ-RAMÍREZ et RUIZ-MORALES 2020) une base de données répertoriant les valeurs NICS de divers HAP, les informations contenues dans cette dernière sont donc susceptibles d'être redondantes/complémentaires avec celles contenues dans BenzDB.

6.2.2. Interroger BenzDB

Dans cette sous-section, nous détaillerons les différentes manières d'interroger notre base de données BenzDB. Nous considérons deux types de requêtes : les requêtes *globales* permettant de récupérer une information donnée pour l'ensemble des benzénoïdes présents dans la base ainsi que les requêtes dites *ciblées* qui permettent de récupérer une information donnée pour l'ensemble des benzénoïdes satisfaisant des critères spécifiques. La sous-sous-section 6.2.2.1 sera consacrée aux requêtes globales tandis que la sous-sous-section 6.2.2.2 sera consacrée aux requêtes ciblées.

6.2.2.1. Envoi de requêtes globales

Les requêtes globales peuvent être facilement accessibles via un simple navigateur web. Chacune d'entre elles a pour fonctionnalité de renvoyer une information spécifique concernant l'intégralité des benzénoïdes contenus dans la base. Une requête GET est envoyée à l'API REST qui interroge la base de données à chaque fois qu'un utilisateur accède à l'une des URL suivantes :

- https://benzenoids.lis-lab.fr/find_all_benzenoids :

  Renvoie les informations basiques (identifiant, nombre de carbones/hydrogènes/hexagones, paramètre d'irrégularité, représentation au format .graph et coordonnées des atomes de carbone de tous les benzénoïdes de la base).

- https://benzenoids.lis-lab.fr/find_all_ir/ :

  Renvoie les informations basiques et les valeurs du spectre infrarouge (fréquences, intensités, énergies et énergie au point zéro) de tous les benzénoïdes de la base.

- https://benzenoids.lis-lab.fr/find_all_nics/ :

  Renvoie les informations basiques et les valeurs NICS de tous les benzénoïdes de la base.

- https://benzenoids.lis-lab.fr/find_all_ims2d1a/ :

  Renvoie les informations basiques et les valeurs IMS de chaque benzénoïde présent dans la base.

6.2.2.2. Envoi de requêtes ciblées

Les requêtes ciblées, plus spécifiques, peuvent être envoyées à BenzDB en envoyant une requête POST à l'API REST avec une entrée au format JSON. JSON (JavaScript Object Notation) est un format de fichier utilisé pour représenter des données structurées de manière similaire aux objets Javascript. Il est très souvent utilisé par les sites web afin d'assurer la transmission de données. Un objet JSON est délimité par des accolades et est composé d'une série de lignes respectant la syntaxe suivante :

$$"clé" : valeur$$

La valeur peut être de n'importe quel type reconnu par Javascript (entier/décimal, chaîne de caractères, booléen, tableau, ...). Une requête POST devra donc être accompagnée d'un fichier JSON qui contiendra toutes les données d'entrée. Il faut ensuite savoir qu'il n'est pas possible d'envoyer une requête POST directement via un navigateur contrairement aux requêtes GET. Ce type de requête doit être envoyé via une interface ou une commande spécifique. Il est par exemple possible d'en envoyer via ReqBin (https://reqbin.com/) ou via la commande `curl`. Notons cependant que la manière la plus simple d'envoyer ce type de requête à notre API est de passer par notre logiciel BenzAI qui propose une interface intuitive et qui va automatiquement construire ces requêtes en arrière-plan et les envoyer à l'API. L'utilisation de Reqbin ou `curl` serait plutôt à privilégier si l'on souhaite récupérer les données brutes pour leur appliquer ensuite un certain traitement automatique. Les URL correspondant à ces requêtes sont similaires à celles des requêtes globales détaillées précédemment, il suffit de remplacer `find_all` par `find`.

Nous avons mentionné ci-dessus qu'une requête POST devait être accompagné d'un fichier JSON contenant les données d'entrées. Il convient donc de détailler tous les champs qui doivent apparaître dans ce fichier. Chacun de ces champs sera représenté par une ligne `"label": "opérateur valeur"` et doit obligatoirement apparaître dans le fichier d'entrée :

- `idBenzenoid` : l'identifiant unique du benzénoïde dans la base de données,

```
{
        "idBenzenoid": "",
        "label": "",
        "inchi": "",
        "nbHexagons": "= 5",
        "nbCarbons": "<= 20",
        "nbHydrogens": "<> 12",
        "irregularity": ""
}
```

FIGURE 6.17. – Entrée JSON utilisée pour obtenir des informations sur tous les benzénoïdes de 5 hexagones, avec au moins 20 atomes de carbones et un nombre d'atomes d'hydrogène différent de 12.

	Opérateurs					
Clés	<=	<	=	>	>=	<>
idBenzenoid	√	√	√	√	√	√
label			√			√
inchi			√			√
nbHexagons	√	√	√	√	√	√
nbCarbons	√	√	√	√	√	√
nbHydrogens	√	√	√	√	√	√
irregularity	√	√	√	√	√	√

TABLEAU 6.1. – Les opérateurs acceptés pour chaque champ d'une entrée JSON

- `label` : son identifiant interne (utilisé par BenzAI),
- `inchi` : son identifiant InChI,
- `nbHexagons` : son nombre d'hexagones,
- `nbCarbons` : son nombre d'atomes de carbones,
- `nbHydrogens` : son nombre d'atomes d'hydrogènes,
- `irregularity` : son paramètre d'irrégularité.

Les valeurs associées aux champs `idBenzenoid`, `nbHexagons`, `nbCarbons` et `nbHydrogens` doivent être des nombres entiers, et celle associée à `irregularity` doit être un nombre décimal. Les valeurs des autres champs doivent être des chaînes de caractères (voir la table 6.1). Les opérateurs supportés sont =, ≠, ≤, <, > et ≥ pour les nombres ainsi que = et ≠ pour les chaînes de caractères). Ces opérateurs sont notés respectivement =, <>, <=, <, > et >= dans le fichier JSON. Si un champ n'est pas impliqué dans une requête, la valeur qui lui est associée doit être la chaîne de caractères vide `""`. La figure 6.17 décrit un exemple d'entrée JSON valide. Envoyer une requête POST à l'API avec cette entrée correspond à demander des informations à la base de données sur l'intégralité des benzénoïdes ayant 5 hexagones, un nombre de carbones inférieur ou égal à 20 ainsi qu'un nombre d'hydrogènes différent de 12.

6.2.2.3. Résultat d'une requête

Lorsque l'API REST reçoit une requête, elle renvoie un message d'erreur dans le cas où cette dernière est invalide (mauvaise URL ou mauvais format d'entrée) ou si la base de données est inaccessible. Dans le cas

```
[
    {
                "idBenzenoid": 2719,
                "label": "0-1-6-7-12",
                "inchi": "1S/C19H30/c1-3-12-7-9-14-10-8-13-4-2-6-16
                        -11-15(5-1)17(12)19(14)18(13)16/h12-19H,1-11H2",
                "nbHexagons": 5,
                "nbCarbons": 19,
                "nbHydrogens": 11,
                "irregularity": 0.545
        }
]
```


FIGURE 6.18. – Sortie JSON renvoyée quand l'on demande les informations basiques des benzénoïdes de 5 hexagones, avec au plus 20 carbones et un nombre d'hydrogènes différent de 20. Dans ce cas, un seul benzénoïde est trouvé.

contraire, elle renverra les données demandées par l'utilisateur sous la forme d'une sortie JSON. Par exemple, la figure 6.18 est le résultat obtenu lorsque l'on envoie une requête POST à l'adresse https://benzenoids.lis-lab.fr/find_benzenoids/ avec l'entrée décrite dans la figure 6.17. Le résultat peut être vu comme un tableau dont chaque élément est un ensemble de champs "*clé*" : *valeur* dépendant de l'information demandée. Les champs pouvant être renvoyés par l'API sont les suivants :

- Informations basiques :
  - `nbHexagons` (de type entier) : nombre d'hexagones du benzénoïde,
  - `nbCarbons` (entier) : nombre de carbones du benzénoïde,
  - `nbHydrogens` (entier) : nombre d'hydrogènes du benzénoïde,
  - `irregularity` (décimal) : paramètre d'irrégularité du benzénoïde,
  - `inchi` (chaîne de caractères) : l'identifiant InChI du benzénoïde,
  - `label` (chaîne de caractères) : l'identifiant interne du benzénoïde (utilisé par BenzAI),
  - `geometry` (chaîne de caractères) : les coordonnées de chaque atome de carbone de la géométrie optimisée du benzénoïde sous la forme d'une chaîne de caractère au format `"C_x1_y1_z1 C_x2_y2_z2 ..."`,
  - `graphFile` (chaîne de caractères) : le benzénoïde au format .graph (défini dans la sous-section 6.1.2).
- Informations liées à NICS :
  - `nics` (chaîne de caractères) : les valeurs NICS de chaque hexagone sous la forme d'une chaîne de caractères (chaque valeur est suivie d'un espace).
- Informations liées aux spectres infrarouges :
  - `frequencies` (chaîne de caractères) : fréquences calculées par Gaussian (chaque valeur est suivie d'un espace),

- `intensities` (chaîne de caractères) : intensités calculées par Gaussian (chaque valeur est suivie d'un espace),
- `finalEnergy` (décimal) : dernière énergie calculée par Gaussian,
- `zeroPointEnergy` (décimal) : énergie au point zéro calculée par Gaussian.

- Informations liées aux cartes IMS :
  - `originPoint` (chaîne de caractères) : coordonnées du point d'origine de la carte de la forme `"x0 y0 z0"`,
  - `nbPointsX` (entier) : nombre de points sur une ligne,
  - `nbPointsY` (entier) : nombre de points sur une colonne,
  - `xVector` (chaîne de caractères) : vecteur représentant l'abscisse de la forme `"xv yv zv"`,
  - `yVector` (chaîne de caractères) : vecteur représentant l'ordonnée de la forme `"xv yv zv"`,
  - `map` (chaîne de caractères) : carte IMS du benzénoïde au format PNG (en base 64).

6.3. Conclusions

Dans ce chapitre, nous avons présenté BenzAI (section 6.1), un logiciel à destination des chimistes théoriciens que nous avons mis au point. Ce dernier regroupe les fonctionnalités ayant été présentées au cours des chapitres précédents, à savoir la génération de structures de benzénoïdes répondant à des critères divers et variés ainsi que le calcul de l'énergie de résonance selon les méthodes CERCP et LFCP. Il admet également d'autres fonctionnalités, qui ont toutes été détaillées au cours de ce chapitre. Le logiciel BenzAI constitue un projet assez novateur dans le domaine de la chimie théorique. Il n'existait en effet aucun logiciel capable d'effectuer l'intégralité de ces tâches. En plus d'être centralisateur, il dispose d'une interface simple et intuitive qui cache tous les détails techniques à l'utilisateur. Le développement de ce logiciel a donné lieu à une publication dans le journal de chimie Journal of Chemical Information and Modeling (JCIM) (Varet, Prcovic, Terrioux et al. 2022). Pour conclure, il convient de mentionner que ce logiciel continue d'évoluer et que de nouvelles fonctionnalités sont à venir.

Nous avons également présenté notre base de données BenzDB dans la section 6.2. Cette dernière contient un certain nombre d'informations sur l'ensemble exhaustif des benzénoïdes possédant au plus 9 hexagones telles que leurs informations basiques (leur nombre de carbones/hydrogènes/hexagones, leur géométrie ...), mais également des informations issues de calculs plus coûteux tel que leurs valeurs NICS, leurs valeurs de spectres infrarouges ou encore leurs cartes IMS. Cette base de données est directement interrogeable en ligne, ou via l'interface de notre logiciel BenzAI. En plus de centraliser un grand nombre d'informations sur un ensemble de benzénoïdes garanti exhaustif, cette dernière contient également des résultats issus de calculs très coûteux et les rendre accessibles à tous pourrait permettre à des chimistes d'économiser plusieurs semaines de calcul. De plus, un article détaillant les différentes fonctionnalités de cette base de données est en cours de rédaction et sera soumis au journal Journal of Chemical Information and Modeling (JCIM) dans les mois à venir.

7. Premières contributions à la chimie théorique

7.1. Introduction

Dans ce chapitre, nous montrons comment les outils que nous avons mis au point durant cette thèse peuvent être bénéfiques au domaine de la chimie. La section 7.2 détaillera les résultats des méthodes *ClarCP* et *RBO*. Ces dernières permettent d'estimer rapidement l'aromaticité d'un benzénoïde donné respectivement en énumérant ses couvertures de Clar et ses RBO. La section 7.3 présentera deux cas de figure où notre logiciel BenzAI pourrait contribuer à la synthèse de molécules entièrement aromatiques (sous-section 7.3.1) et de trimères admettant un grand nombre d'électrons radicalaires stables (sous-section 7.3.2). Pour finir, la section 7.4 détaillera comment nous pouvons utiliser notre base de données BenzDB afin d'obtenir une meilleure interprétation des spectres infrarouges concernant des benzénoïdes de 9 hexagones ou moins.

7.2. Des méthodes plus rapides pour estimer l'aromaticité d'un benzénoïde

Dans la section 2.2, nous avons vu qu'il existait, en plus de NICS et des méthodes de Lin/Lin & Fan, deux autres méthodes permettant d'estimer l'aromaticité d'un benzénoïde : en calculant ses *Ring Bond Order (RBO)* (voir la définition 2.4.15) ou en exploitant ses *couvertures de Clar* (voir la définition 2.4.13). Pour rappel, calculer les RBO d'un benzénoïde consiste à effectuer des statistiques sur l'apparition des liaisons de chaque hexagone en tant que liaison double dans l'ensemble des structures de Kekulé du benzénoïde traité. Les structures de Clar peuvent être vues, quant à elles, comme une factorisation des structures de Kekulé d'un benzénoïde qui prennent en compte les cas où l'électron π d'un carbone n'est lié à aucun autre électron π. L'aromaticité d'un benzénoïde peut donc ensuite être estimée en attribuant à chaque hexagone une valeur correspondant au rapport du nombre de ronds de Clar admis par ce dernier dans l'intégralité de ses structures de Clar sur le nombre total de structures de Clar de la molécule considérée. Ces deux méthodes seront respectivement nommées ClarCP et RBO. Ces deux méthodes sont respectivement définies dans les sous-sous-sections 6.1.3.3 et 6.1.3.4. Pour rappel, la méthode ClarCP utilise la programmation par contraintes afin d'énumérer l'intégralité des couvertures de Clar du benzénoïde traité.

Cette section détaille les résultats obtenus en estimant l'aromaticité des benzénoïdes des ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$ décrits dans la section 5.4 via ces deux méthodes. L'objectif est notamment de comparer ces deux méthodes aux méthodes CERCP (définie dans la section 5.3) et NICS (définie dans la sous-section 2.4.2) du point de vue de la qualité de l'estimation et du temps de calcul requis. Le protocole expérimental est similaire à celui détaillé dans la sous-section 5.4.1. Ainsi, nous comparons directement les différents temps d'exécution des méthodes. Pour la qualité des résultats, elle est déterminée à l'aide du coefficient de corrélation de Pearson. Les configurations matérielles sont les mêmes que celles utilisées dans la sous-section 5.4.1 : nous avons utilisé des serveurs avec des processeurs Intel Xeon Gold de 2,20 GHz et 256 GB sous CentOS Linux 8.1.1911. Le temps d'exécution de chaque algorithme a été limité à 24 heures par benzénoïde. De plus, nous avons utilisé la version 4.10.8 de Choco Solver.

On se propose de comparer les méthodes CERCP, ClarCP, RBO et NICS. La table 7.1 décrit les comparaisons des temps d'exécution pour l'ensemble de molécules $\mathscr{B}_1$. Les valeurs manquantes sont dues au fait que les instances associées n'ont pas pu être résolues dans le temps imparti fixé (24 heures). ClarCP s'avère plus rapide que CERCP sur toutes les molécules de l'ensemble $\mathscr{B}_1$, excepté la molécule 48 pour laquelle l'exécution ne s'est pas terminée dans le temps imparti. Pour rappel, il n'existe, à notre connaissance, aucune méthode permettant de compter le nombre de couvertures de Clar d'un benzénoïde à l'instar de la méthode de Rispoli (voir la section 2.2) pour les structures de Kekulé. RBO a réussi, quant à elle, à résoudre toutes les instances de $\mathscr{B}_1$ en moins d'une seconde. C'est clairement la méthode la plus performante présentée au cours de cette thèse en termes de temps de calcul. Cela est dû au fait que cette méthode utilise la méthode de Rispoli mentionnée précédemment pour compter les structures de Kekulé dans lesquelles une liaison donnée apparaît comme étant double. Cette dernière est extrêmement efficace et explique également en partie les bons résultats obtenus avec CERCP.

Intéressons-nous maintenant à la qualité de l'estimation de l'aromaticité des méthodes citées ci-dessus. La table 7.2 détaille les valeurs du coefficient de corrélation de Pearson entre ces différentes méthodes pour l'ensemble de molécules $\mathscr{B}_1$. Les valeurs manquantes sont soit dues au fait qu'une des deux instances n'a pas terminé dans le temps imparti, soit que le calcul du coefficient de corrélation donnait une valeur invalide suite à une division par 0. Cette division par zéro survient quand les valeurs de tous les hexagones des résultats sont identiques. Les champs marqués d'un « - » correspondent à des instances n'ayant pas pu être résolues dans les 24 heures imparties, tandis que les champs marqués d'un « * » correspondent à un coefficient invalide suite à une division par 0. Les valeurs retournées par ces différentes méthodes pour chaque hexagone sont disponibles dans l'annexe A. Les histogrammes décrits dans les figures 7.3 à 7.7 donnent respectivement une vision globale des différents coefficients obtenus entre en comparant les méthodes CERCP et ClarCP, CERCP et RBO, ClarCP et NICS, ClarCP et RBO ainsi que RBO et NICS. Ces derniers donnent le nombre de benzénoïdes pour différents intervalles de valeurs du coefficient pour les ensembles de molécules $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$. En regardant ces histogrammes plus en détails, nous pouvons en tirer les observations suivantes :

- **CERCP et ClarCP (figure 7.3)** : pour un total de 109 instances comparables, il apparaît que 56 instances (environ 51%) admettent un coefficient de corrélation supérieur ou égal à 0,9, que 85 (environ 78%) d'entre elles admettent un coefficient supérieur ou égal à 0,8 et que 104 (environ 95%) ont un coefficient supérieur ou égal à 0,7.

- **CERCP et RBO (figure 7.4)** : pour un total de 115 instances comparables, il apparaît que 105 d'entre elles (environ 91%) admettent un coefficient supérieur ou égal à 0,9 et que 114 d'entre elles (plus de 99%) admettent un coefficient supérieur ou égal à 0,8.

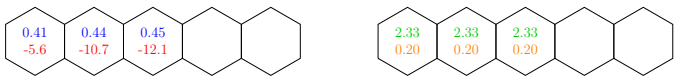
- **ClarCP et NICS (figure 7.5)** : pour un total de 110 instances comparables, 34 (environ 31%) admettent un coefficient supérieur ou égal à 0,9, que 50 d'entre elles (environ 45%) admettent un coefficient supérieur ou égal à 0,8, que 68 (environ 62 %) admettent un coefficient supérieur ou égal 0,7 et que 89 (environ 81%) admettent un coefficient supérieur ou égal à 0,6.

- **ClarCP et RBO (figure 7.6)** : pour un total de 107 instances comparables, 69 d'entre elles (environ 64%) admettent un coefficient supérieur ou égal à 0,9, 97 (environ 91%) un coefficient supérieur ou égal à 0,8 et 106 d'entre elles (plus de 99%) un coefficient supérieur ou égal à 0,7.

- **RBO et NICS (figure 7.7)** : pour un total de 108 instances comparables, 60 d'entre elles (environ 56%) admettent un coefficient supérieur ou égal à 0,9, 87 (environ 81%) admettent un coefficient supérieur ou égal à 0,8, et 96 (environ 89%) admettent un coefficient supérieur ou égal à 0,7.

Si l'on regarde ces observations plus en détails, on peut voir que les comparaisons des méthodes CERCP, ClarCP et RBO induisent de très bons coefficients de corrélation. En revanche, les comparaisons de ces méthodes avec NICS donnent des coefficients moins élevés. Cela n'est pas si surprenant, car les méthodes CERCP, ClarCP et RBO possèdent un fonctionnement similaire : deux d'entre elles (CERCP et RBO) reposent sur le comptage des structures de Kekulé du benzénoïde considéré tandis que ClarCP repose sur l'énumération de ses structures de Clar (qui peuvent être vues comme une factorisation de certaines de ses structures de Kekulé). En revanche, le fonctionnement de NICS (qui repose sur des calculs de chimie quantique) est totalement différent de celui des trois autres méthodes. Cela peut donc expliquer l'écart de résultats obtenus. Néanmoins, même si les comparaisons de NICS avec les trois autres méthodes induisent des coefficients plus faibles, ils restent acceptables et ces méthodes semblent être comparables.

Pour finir, même si les méthodes ClarCP et RBO sont, de manière générale, plus rapides que CERCP et NICS, il convient de pointer leur manque de finesse dans certains cas. Prenons, par exemple, la molécule 10 de l'ensemble $\mathscr{B}_1$ dont les valeurs obtenues avec les quatre méthodes mentionnées ci-dessus sont décrites dans la figure 7.1 (en bleu pour CERCP, en rouge pour NICS, en vert pour RBO et en orange pour ClarCP).



(a) (b)

FIGURE 7.1. – Un exemple traduisant le manque de finesse de ClarCP et RBO

Les valeurs obtenues via ClarCP et RBO sont identiques alors que celles obtenues via CERCP et NICS sont différentes. Dans le cas de ClarCP, cela s'explique par le fait que cette molécule admet cinq couvertures de Clar et que chacune admet un rond dans un hexagone différent. Dans le cas de RBO, cela s'explique par le fait que chaque hexagone admet exactement la même fréquence d'apparition de liaisons doubles dans l'intégralité des structures de Kekulé.

Pour rappel, il n'y a pas de méthode de référence pour estimer l'aromaticité d'un benzénoïde. L'utilisateur devra donc choisir le meilleur compromis entre vitesse de calcul et qualité des résultats en choisissant d'utiliser la méthode qui lui convient le mieux.

Id	n	Nombre de structures		Temps			
		de Kekulé	de Clar	ClarCP	RBO	CERCP	NICS
1	1	2	1	0,291	0,004	0,19	51,8
2	2	3	2	0,312	0,005	0,20	148,0
3	3	5	1	0,308	0,006	0,20	1 110,0
4	3	4	3	0,312	0,006	0,20	293,6
5	4	5	4	0,330	0,006	0,21	480,4
6	4	8	3	0,316	0,007	0,21	1 746,0
7	4	9	1	0,308	0,006	0,21	1 483,0
8	4	6	1	0,328	0,006	0,22	571,2
9	4	7	2	0,327	0,006	0,21	2 150,8
10	5	6	5	0,352	0,007	0,22	734,2
11	5	9	3	0,332	0,008	0,22	5 276,6
12	5	10	4	0,338	0,008	0,22	3 936,0
13	5	12	1	0,327	0,008	0,22	4 521,8
14	5	11	5	0,345	0,009	0,22	9 714,8
15	5	13	1	0,329	0,009	0,22	10 974,6
16	5	13	2	0,340	0,009	0,23	3 294,4
17	5	14	2	0,325	0,008	0,23	29 483,2
18	5	11	1	0,329	0,008	0,23	1 873,2
19	5	9	3	0,324	0,008	0,23	3 992,0
20	5	9	4	0,330	0,007	0,23	867,2
21	7	20	2	0,349	0,010	0,28	4 562,4
22	13	250	1	0,397	0,032	0,95	5 890,0
23	20	3 250	1	0,497	0,045	4,47	10 910,2
24	19	3 100	1	0,467	0,045	3,04	26 270,6
25	24	16 100	1	1,307	0,057	11,78	24 152,0
26	25	34 560	1	0,633	0,069	10,57	27 462,4
27	11	25	16	0,468	0,019	0,46	3 545,8
28	17	1 320	1	0,444	0,037	2,20	9 299,0
29	8	34	1	0,286	0,014	4,47	1 399,0
30	8	31	1	0,359	0,013	4,43	45 715,2
31	8	19	1	0,365	0,012	4,43	1 425,4
32	8	16	9	0,371	0,011	4,46	386,2
33	9	40	1	0,362	0,015	1,06	3 238,6
34	9	20	1	0,370	0,015	3,10	901,4
35	9	30	5	0,364	0,015	3,10	3 774,8
36	10	50	1	0,416	0,017	3,31	1 047,2
37	14	175	2	0,409	0,030	4,98	2 056,2
38	8	45	1	0,344	0,014	0,76	2 044,0
39	10	101	1	0,365	0,022	1,05	2 836,8
40	6	14	3	0,334	0,009	4,26	1 019,8
41	6	10	1	0,345	0,009	4,24	854,0
42	6	14	1	0,351	0,010	4,27	1 807,4
43	13	432	18	0,447	0,040	5,04	-
44	61	267 227 532	1	3 645,601	0,275	635,58	-
45	49	126 672 896	64	15,740	0,249	225,23	-
46	30	27 508	4	2,695	0,091	16,10	14 798,8
47	37	232 848	2	6,586	0,107	84,99	14 683,8
48	109	53 930 238 785 494 000	-	-	0,866	4 324,50	-

TABLEAU 7.1. – Nombre d'hexagones, de structures de Kekulé, de couvertures de Clar et temps d'exécution des méthodes ClarCP, RBO, CERCP et NICS pour les molécules de l'ensemble $\mathscr{B}_1$

Id	coef.					
	CERCP-NICS	CERCP-ClarCP	CERCP-RBO	ClarCP-NICS	ClarCP-RBO	RBO-NICS
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	1,00	1,00	1,00	1,00	1,00	1,00
4	1,00	*	*	*	*	*
5	1,00	*	*	*	*	*
6	1,00	1,00	1,00	1,00	1,00	1,00
7	1,00	1,00	1,00	1,00	1,00	1,00
8	1,00	1,00	1,00	1,00	1,00	1,00
9	0,74	1,00	0,99	0,90	0,97	0,82
10	1,00	0,00	*	0,00	*	*
11	0,51	1,00	0,97	0,41	0,94	0,66
12	0,98	1,00	1,00	0,97	1,00	0,97
13	0,90	0,97	1,00	0,98	0,98	0,92
14	0,69	0,99	0,97	0,63	0,97	0,77
15	1,00	0,91	1,00	0,89	0,91	1,00
16	0,80	0,98	0,98	0,66	0,93	0,89
17	0,95	0,96	0,99	0,82	0,93	0,97
18	0,86	0,94	0,99	0,90	0,92	0,90
19	0,89	1,00	1,00	0,85	1,00	0,85
20	1,00	1,00	0,99	1,00	1,00	1,00
21	1,00	1,00	1,00	1,00	1,00	1,00
22	0,91	0,97	1,00	0,98	0,98	0,93
23	0,90	0,96	1,00	0,98	0,97	0,91
24	0,84	0,95	1,00	0,88	0,96	0,86
25	0,90	0,96	1,00	0,98	0,97	0,92
26	0,75	0,92	0,99	0,94	0,95	0,80
27	0,97	1,00	1,00	0,98	1,00	0,98
28	0,88	0,96	1,00	0,95	0,98	0,90
29	0,81	0,76	1,00	0,64	0,75	0,85
30	0,98	0,93	1,00	0,86	0,89	1,00
31	0,95	0,83	1,00	0,87	0,85	0,91
32	1,00	1,00	1,00	1,00	1,00	1,00
33	0,92	0,85	0,99	0,81	0,81	0,93
34	0,98	0,80	0,99	0,80	0,85	0,94
35	0,98	0,91	0,99	0,92	0,87	0,95
36	0,98	0,91	0,98	0,95	0,83	0,96
37	0,99	0,95	0,96	0,96	0,89	0,94
38	0,94	0,99	1,00	0,98	0,99	0,96
39	0,94	0,99	1,00	0,98	0,99	0,96
40	0,96	0,82	1,00	0,87	0,78	0,87
41	0,98	1,00	1,00	0,96	1,00	0,96
42	0,88	0,79	0,99	0,41	0,84	0,83
43	0,96	0,98	1,00	0,93	0,97	0,96
44	*	0,71	0,83	*	0,74	*
45	-	0,83	1,00	-	0,83	-
46	0,47	0,55	0,94	0,26	0,52	0,22
47	1,00	0,94	0,90	0,97	0,86	0,92
48	-	-	1,00	-	-	-

TABLEAU 7.2. – Coefficients de corrélation de Pearson entre les méthodes CERCP, ClarCP, RBO et NICS pour l'ensemble de molécules $\mathscr{B}_1$

.

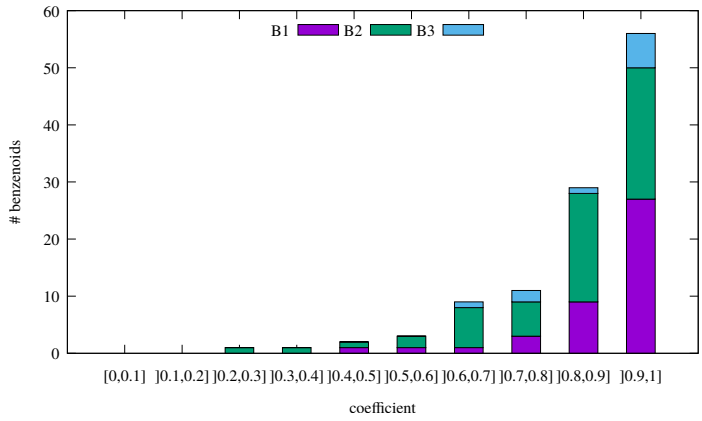


FIGURE 7.2. – Le coefficient de corrélation de Pearson entre CERCP et Nics pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

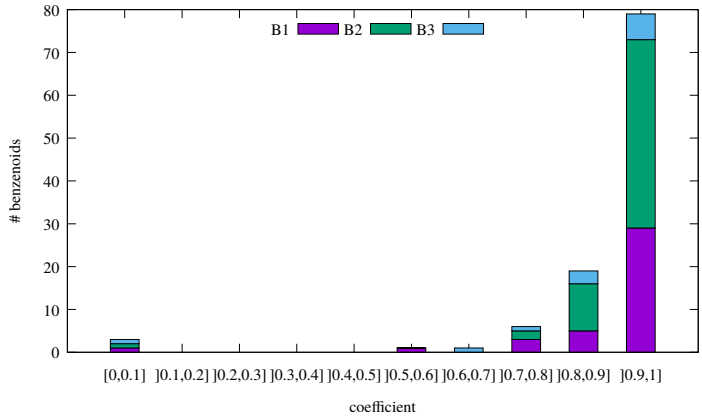


FIGURE 7.3. – Le coefficient de corrélation de Pearson entre CERCP et ClarCP pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

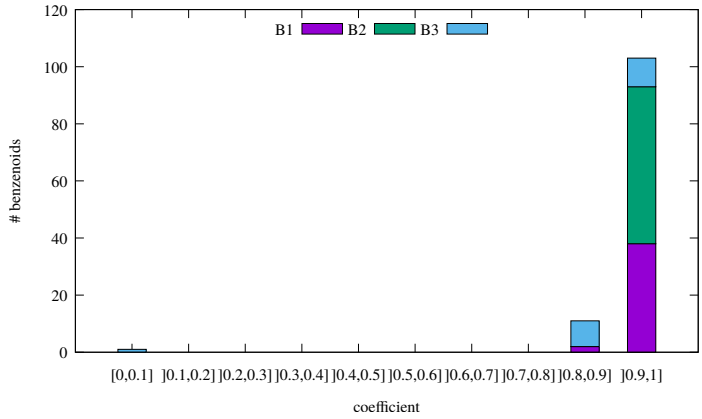


FIGURE 7.4. – Le coefficient de corrélation de Pearson entre CERCP et RBO pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

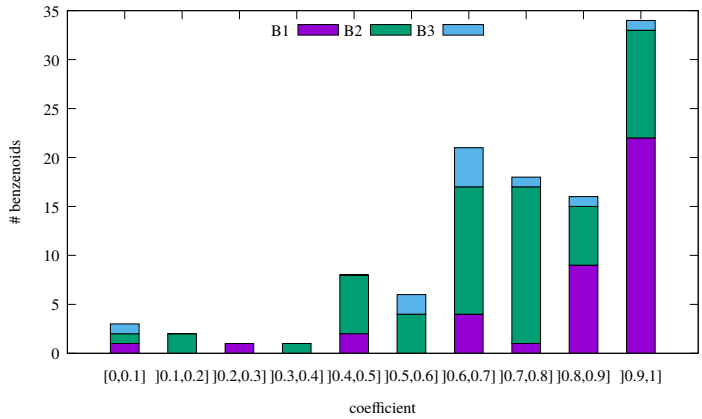


FIGURE 7.5. – Le coefficient de corrélation de Pearson entre ClarCP et NICS pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

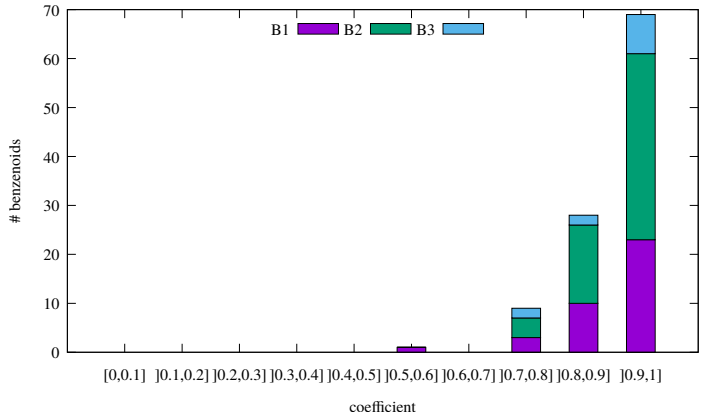


FIGURE 7.6. – Le coefficient de corrélation de Pearson entre RBO et ClarCP pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

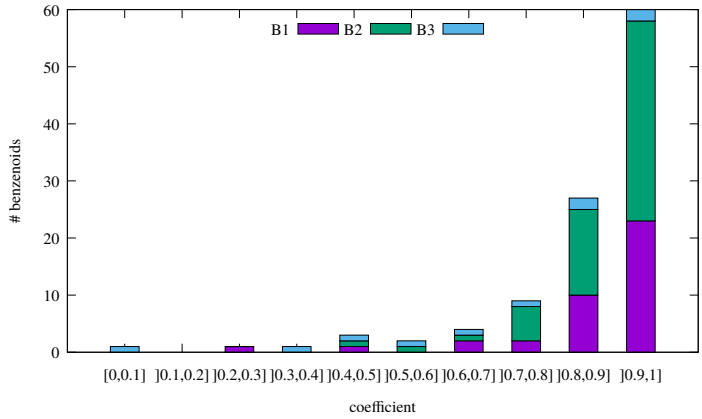


FIGURE 7.7. – Le coefficient de corrélation de Pearson entre RBO et NICS pour les ensembles $\mathscr{B}_1$, $\mathscr{B}_2$ et $\mathscr{B}_3$.

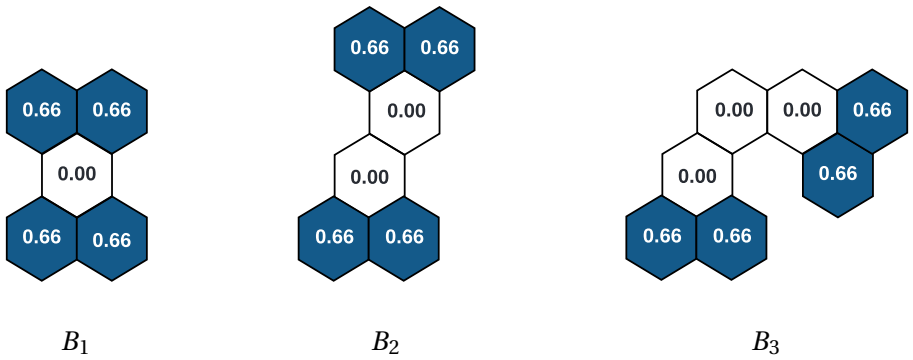


FIGURE 7.8. – Trois benzénoïdes partiellement aromatiques

7.3. Utiliser BenzAI pour contribuer à la synthèse de nouvelles molécules

Dans cette section, nous expliquerons comment notre logiciel BenzAI peut accompagner les chimistes dans la synthèse de molécules. La sous-section 7.3.1 présentera une manière d'utiliser le logiciel afin de mettre en évidence des benzénoïdes pouvant être synthétisés en ajoutant des hexagones sur un benzénoïde partiellement aromatique. La sous-section 7.3.2 détaillera quant à elle une manière d'utiliser BenzAI afin d'exhiber des benzénoïdes admettant un grand nombre d'électrons radicalaires synthétisables par trimérisation.

7.3.1. Synthèse de benzénoïdes entièrement aromatiques à partir de benzénoïdes partiellement aromatiques

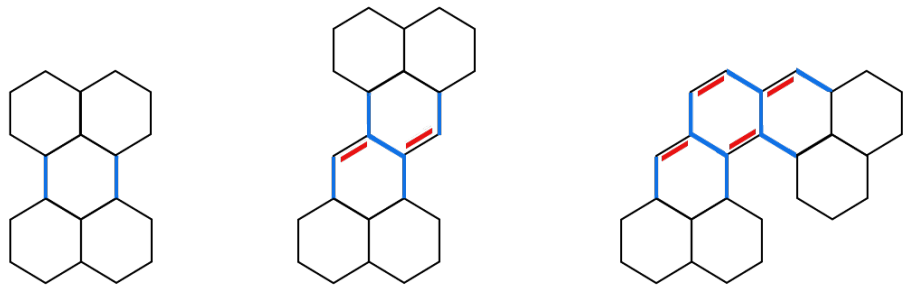
Dans cette sous-section, nous détaillons de quelle manière notre logiciel BenzAI pourrait être utile afin de synthétiser (c'est-à-dire créer réellement la molécule) des benzénoïdes entièrement aromatiques à partir de benzénoïdes partiellement aromatiques. Dans un premier temps, il convient de définir formellement ces deux notions :

Définition 7.3.1 (Benzénoïde partiellement aromatique) *Un benzénoïde est dit* partiellement aromatique *si au moins l'un de ses hexagones possède une énergie de résonance locale nulle.*

Définition 7.3.2 (Benzénoïde entièrement aromatique) *Un benzénoïde est dit* complètement aromatique *si l'intégralité de ses hexagones possèdent une énergie de résonance locale non nulle.*

Pour illustrer cela, la figure 7.8 décrit trois benzénoïdes partiellement aromatiques. Ces derniers seront dénotés respectivement B_1, B_2 et B_3 dans l'intégralité de cette section. Le phénomène derrière la notion de benzénoïde partiellement aromatique peut s'expliquer de la manière suivante : le fait qu'un hexagone possède une énergie de résonance locale nulle est dû au fait que ce dernier n'admet aucune délocalisation. Cela se traduit par la présence de liaisons fixes (des liaisons qui sont toujours simples ou doubles dans l'intégralité des structures de Clar ou de Kekulé du benzénoïde considéré) qui vont empêcher tout circuit conjugué minimal de couvrir ces hexagones. La figure 7.9 illustre ce phénomène en montrant les liaisons fixes simples (en bleu) et doubles (en rouge) des benzénoïdes B_1, B_2 et B_3.

Or, des études récentes (MISHRA, BEYER, EIMRE et al. 2019; DUMSLAFF, GU, PATERNÒ et al. 2020; MATSUOKA, ITO, SARLAH et al. 2021) ont montré que les zones des benzénoïdes admettant des liaisons fixes sont moins stables et ont une réactivité accrue, c'est-à-dire que les réactions chimiques sont davantage susceptibles d'avoir lieu dans ces régions. Cette propriété se révèle intéressante, car il serait plus facile de



B_1 B_2 B_3

FIGURE 7.9. – Liaisons fixes des benzénoïdes B_1, B_2, B_3

synthétiser des benzénoïdes entièrement aromatiques à partir de benzénoïdes qui le sont partiellement. Prenons par exemple les benzénoïdes décrits dans les figures 7.12 (b, c, d). Ces derniers peuvent être obtenus en fusionnant deux pyrènes (le benzénoïde rectangulaire de dimension 2×2) à l'aide d'un nouvel hexagone. Or, le pyrène est un benzénoïde entièrement aromatique, ce qui le rend particulièrement stable. Il est donc difficile d'obtenir ces molécules en fusionnant deux pyrènes. En revanche, il est également possible d'obtenir ces dernières en ajoutant deux hexagones à la molécule B_3. Elles seront donc plus simples à synthétiser de cette manière.

Grâce à notre logiciel BenzAI, il est possible de déterminer les benzénoïdes entièrement aromatiques qui peuvent être synthétisés à partir de benzénoïdes qui le sont partiellement. Pour cela, nous considérons tout d'abord un benzénoïde partiellement aromatique (ce dernier peut être directement dessiné ou importé s'il est connu de l'utilisateur ou alors, il est possible d'en obtenir en filtrant une collection). Ensuite, il suffit de demander au logiciel de générer l'intégralité des benzénoïdes admettant un motif correspondant à notre benzénoïde partiellement aromatique et ayant un nombre d'hexagones supérieur à ce dernier (ce nombre peut être fixé par l'utilisateur en fonction de ses besoins). Une fois la génération terminée, il suffit de filtrer la collection en ne gardant uniquement que les benzénoïdes entièrement aromatiques. Les figures 7.10, 7.11 et 7.12 illustrent respectivement ce procédé pour les molécules B_1, B_2 et B_3. Il n'existe, à notre connaissance, aucun outil permettant de faire cela actuellement. Ces travaux sont similaires à ceux de Matsuoka et al. (MATSUOKA, ITO, SARLAH et al. 2021). En effet, ces derniers présentent une méthode de synthèse nommée *growth-from-template* consistant à partir d'une molécule, puis, étant donné un motif spécifique, synthétiser des molécules de plus en plus grosses en ajoutant successivement ce dernier sur différentes parties des motifs admis par la molécule initiale.

Pour finir, intéressons-nous aux différents motifs sur lesquels les hexagones ont tendance à s'ajouter pour former des benzénoïdes entièrement aromatiques. Dans le cas du périlène (figure 7.10), on peut voir que le nouvel hexagone doit se greffer dans le creux d'un des deux *shallow armchair bay* présent sur l'un des côtés de B_1. Pour les figures 7.11 et 7.12, nous pouvons voir que les nouveaux hexagones tendent à se greffer soit dans le creux des *shallow armchair bay* ou *zigzag bay* de B_2 et B_3 (ou dans le creux d'un *ultra deep bay* dans le cas de la molécule décrite dans la figure 7.12 (a)). Une hypothèse envisageable serait que la possibilité de rendre complètement aromatique un benzénoïde partiellement aromatique dépende des motifs qu'il admet. Une perspective intéressante pourrait donc consister à exploiter BenzAI pour effectuer des statistiques sur la présence de différents motifs dans ces molécules afin d'affirmer ou infirmer cette hypothèse.

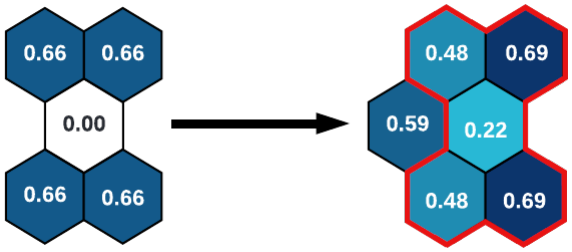


FIGURE 7.10. – L'ajout d'un hexagone sur B_1 (le périlène) le rend entièrement aromatique.

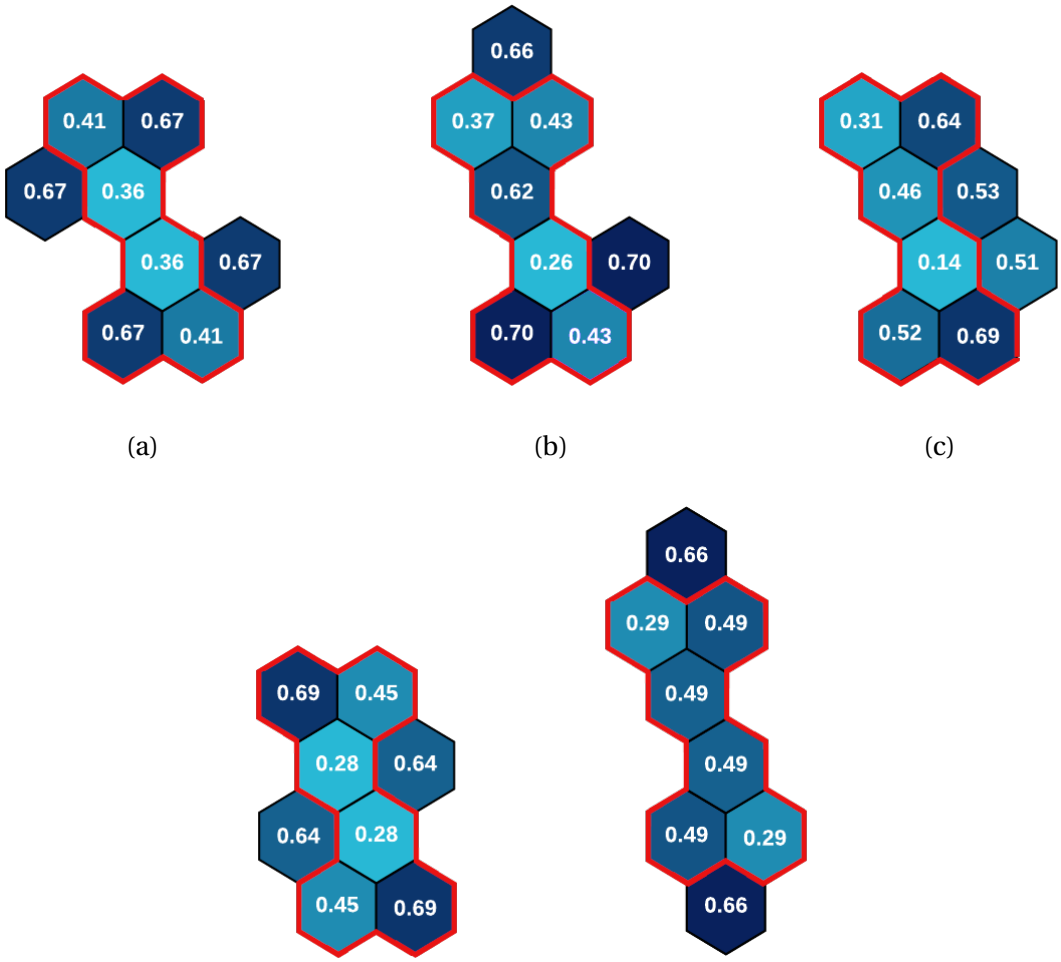


FIGURE 7.11. – Ajouter au moins deux hexagones à B_2 peut le rendre entièrement aromatique

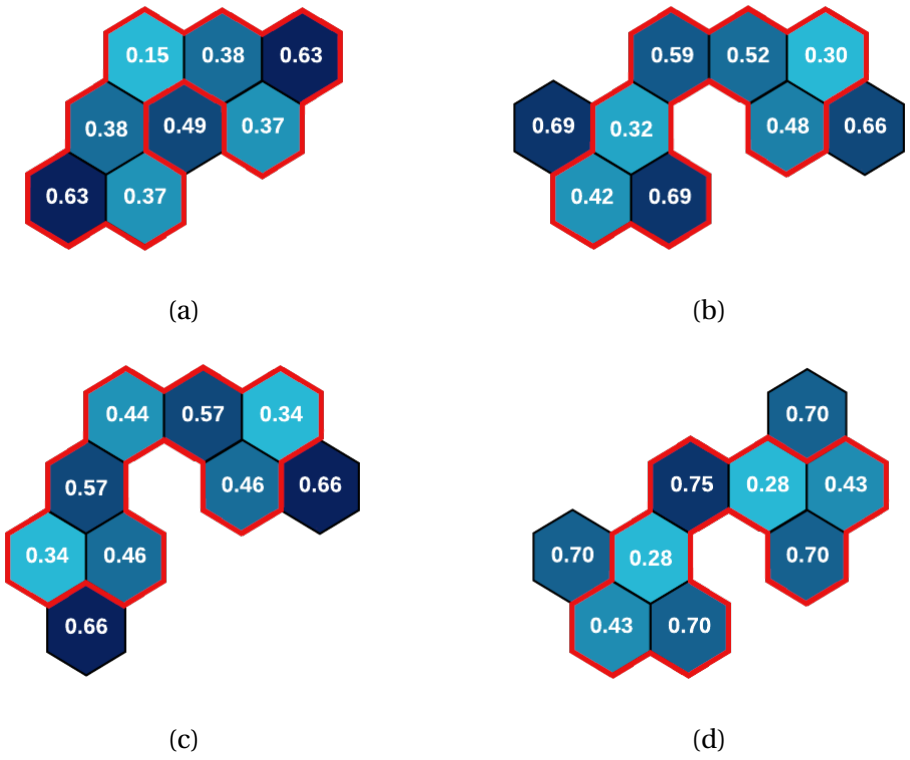


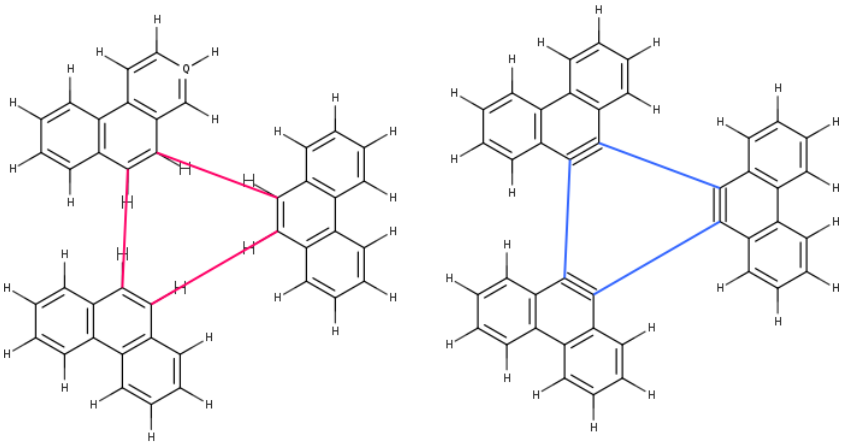
FIGURE 7.12. – Ajouter un ou deux hexagones à B_3 peut le rendre entièrement aromatique.

7.3.2. Exhiber des trimères admettant un grand nombre d'électrons radicalaires grâce à BenzAI

Les chimistes sont fortement intéressés par les benzénoïdes possédant un grand nombre d'électrons radicalaires. Cela est dû au fait que ces derniers présentent des propriétés très intéressantes. Par exemple, Allouche et al. (ALLOUCHE, LAPADULA, SIDDIQI et al. 2017) ont proposé en 2017 de stocker de l'information dans ce type de structures et de considérer ces électrons comme des bits d'informations. Ici, notre objectif était donc de mettre en évidence des benzénoïdes possédant un grand nombre d'électrons radicalaires. Un des moyens d'y arriver repose sur un processus appelé *trimérisation* qui consiste à fusionner trois molécules identiques (appelées *monomères*) autour d'un hexagone central. On appelle *trimère* une molécule obtenue de cette manière. Un point important à noter est que, dans notre cas, la trimérisation n'a d'intérêt que si elle conserve les propriétés des monomères qui ont fusionné. Ainsi, si l'on considère des monomères possédant des électrons radicalaires, la trimérisation devrait multiplier par 3 ces derniers.

Les chimistes ont récemment mené des études sur les HAP chiraux[1] basés sur le triphénylène (décrit dans la figure 7.13). Ces molécules sont obtenues par la trimérisation des hélicènes (ARTIGAS, HAGEBAUM-REIGNIER, CARISSAN et al. 2021). Ces molécules respectent la forme décrite dans la figure 7.14. Les flèches présentes sur les extrémités de cette figure correspondent aux différentes possibilités d'ajout d'hexagones. Ce processus d'ajout est appelé $\pi-$*extension*. La présence de triples liaisons dans la figure 7.14 s'explique par le fait que les carbones concernés ne doivent pas être liés à un hydrogène pour que la trimérisation puisse avoir lieu. La figure 7.15 illustre cela en décrivant un cas invalide (a) et un cas valide (b). Les flèches présentes au centre de la figure 7.14 décrivent comment ces liaisons vont se déplacer une fois la trimérisation terminée. Pour finir, le centre de cette figure représenté en pointillés correspond à l'hexagone central qui sera créé par la trimérisation.

1. Une molécule est dite *chirale* si cette dernière n'est pas superposable à son image dans un miroir, un peu comme notre main gauche n'est pas superposable à notre main droite.



(a) (b)

FIGURE 7.15. – Exemple de trimérisation invalide (a) et valide (b)

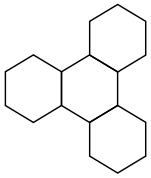


FIGURE 7.13. – Le triphénylène.

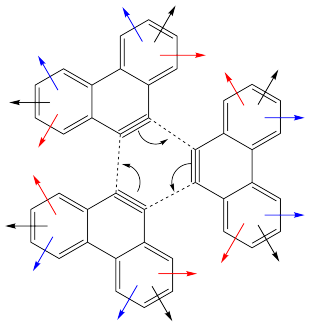


FIGURE 7.14. – Forme des molécules obtenues suite à la trimérisation d'hélicènes.

Ici, l'objectif est d'utiliser cette $\pi-$extension afin de découvrir des monomères admettant des électrons radicalaires. Par construction, la trimérisation de ces derniers devrait ainsi produire des trimères possédant trois fois plus d'électrons radicalaires que leur monomère pour un temps de calcul très inférieur à celui que nécessiterait une recherche directe de benzénoïdes possédant autant d'électrons radicalaires que le trimère.

Nous savons que le *triangulène* $C_{13}H_9$ admet un électron radicalaire sur son atome de carbone central (voir la figure 7.16). Nous avons fait l'hypothèse que nous pourrions obtenir des monomères radicalaires par assemblage de molécules contenant des triangulènes. Dans cette sous-section, nous allons voir comment nous avons pu mettre en évidence de telles molécules à l'aide de notre logiciel BenzAI.

Nous souhaitons donc utiliser notre logiciel afin de pouvoir générer des benzénoïdes dont la trimérisation produirait une nouvelle molécule en accord avec la forme décrite dans la figure 7.14. Le processus utilisé est le suivant : tout d'abord, nous avons demandé à BenzAI de générer l'intégralité des molécules admettant

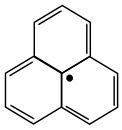
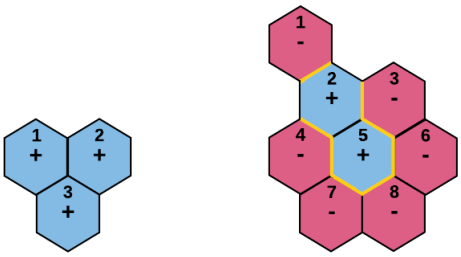


FIGURE 7.16. – Le triangulène $C_{13}H_9$

deux fois le motif décrit dans la figure 7.17 (a) et une fois celui décrit dans la figure 7.17 (b). Le premier motif est requis afin de forcer la présence du triangulène dans les molécules générées (afin de maximiser les chances de présence d'électrons radicalaires dans les molécules générées) tandis que le second est requis pour pouvoir isoler chacun des deux triangulènes (ici, l'hexagone 5 de ce motif correspondra au noyau benzénique central de la figure 7.14). Veuillez noter que nous avons spécifié que les molécules générées devaient posséder exactement 8 hexagones. Si ce nombre avait été plus élevé, alors nous aurions dû agrandir le motif de la figure 7.17 (b) afin d'éviter que les différentes extrémités de la molécule générée puissent se toucher et ainsi faire en sorte que la molécule obtenue après la trimérisation respecte la forme décrite dans la figure 7.14.

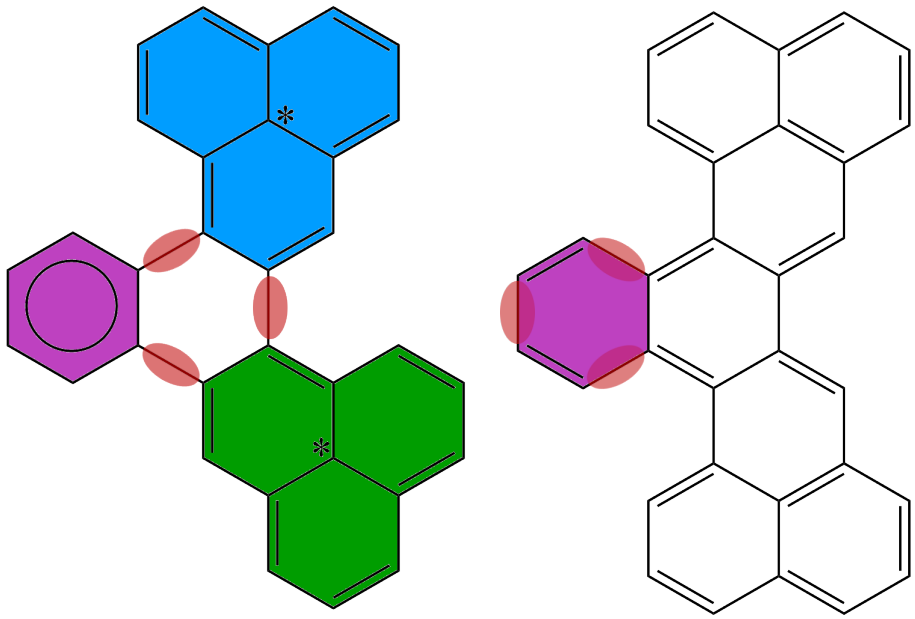


(a) (b)

FIGURE 7.17. – Les deux motifs considérés afin de générer des molécules trimérisables.

Une fois que BenzAI a terminé la génération, nous considérons les molécules générées et nous nous focalisons sur deux d'entre elles en particulier. Ces dernières sont décrites dans la 7.18. Les hexagones violets correspondent au centre de la molécule qui sera obtenue après la trimérisation. Les liaisons entourées en rouge correspondent aux liaisons simples fixes. La solution (a) admet deux électrons radicalaires, ce qui confirme notre hypothèse. Par ailleurs, l'hexagone central servant à la fusion des monomères possède un rond de Clar, ce qui permet à la fusion de conserver les propriétés radicalaires des monomères. La solution (b), quant à elle, admet n'admet aucun électron radicalaire. De plus, l'hexagone qui devrait servir de centre pour la fusion n'a pas de rond de Clar, ce qui empêche une fusion conservant les propriétés des monomères. Les molécules produites par la trimérisation des solutions (a) et (b) sont décrites dans la figure 7.19. Nous avons émis la seconde hypothèse qui est que la trimérisation de ces deux composés devrait donc produire un composé admettant six électrons radicalaires pour la solution (a) et un composé n'en admettant aucun pour la solution (b). Afin de vérifier cette dernière hypothèse, nous avons calculé les *états de spin* de chacune des molécules ainsi obtenues dans les cas où elles admettent 0, 2, 4 ou 6 électrons radicalaires. Une énergie basse correspond alors à une forte stabilité. La table 7.3 décrit les valeurs obtenues pour ces deux molécules. Ces calculs ont été effectués avec le logiciel *Turbomole* (AHLRICHS, BÄR, HÄSER et al. 1989) et avec la base $PBE0$ et la fonctionnelle $def2\text{-}SV(P)$. On observe que pour la molécule (a), l'énergie la plus haute (241) est obtenue quand elle n'admet aucun électron radicalaire, tandis que l'énergie la plus basse est obtenue quand elle en admet 6. La molécule (a) est donc stable quand elle admet 6 électrons radicalaires et instable dans le cas où elle n'en admet aucun. Nous observons une tendance inverse pour la seconde molécule. Ces résultats confirment donc notre hypothèse.

Pour conclure, nous avons montré dans cette section que nous pouvions utiliser notre logiciel BenzAI afin



(a) (b)

FIGURE 7.18. – Les deux molécules générées par BenzAI admettant deux fois le motif décrit dans la figure 7.17 (a) et une fois celui de la figure 7.17 (b) et respectant la forme décrite dans la figure 7.14.

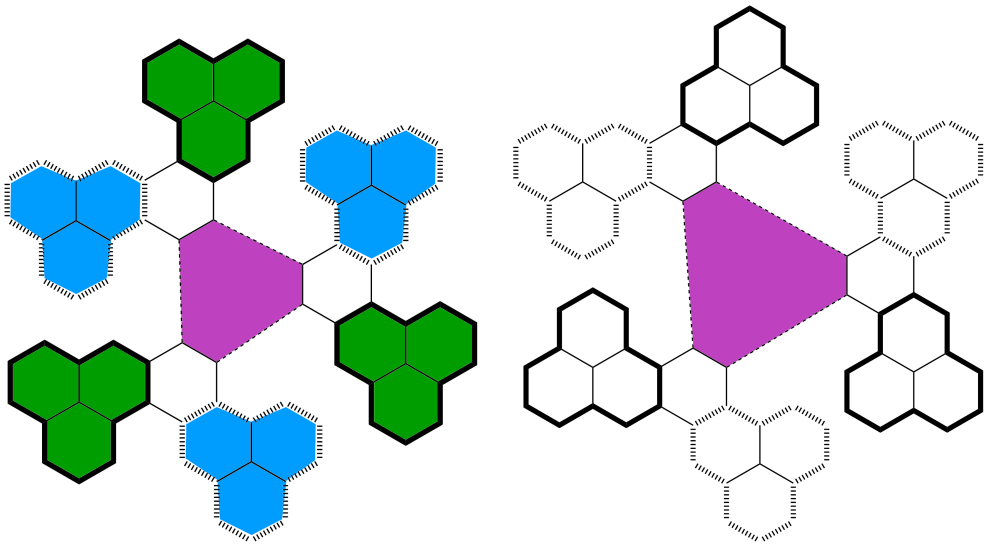
	(a)	(b)
singlet (0 électron radicalaire)	241	0
triplet (2 électrons radicalaires)	83	13
quintet (4 électrons radicalaires)	80	34
septet (6 électrons radicalaires)	0	53

TABLEAU 7.3. – Énergies des états de spin des molécules décrites dans la figure 7.19 en kJ/mol obtenues avec la base PBE0/def2-SV(P)

de générer des molécules admettant un grand nombre d'électrons radicalaires stables. Ce type de molécule est relativement rare et peuvent avoir un grand intérêt pour les chimistes si ces derniers parviennent à les synthétiser. De plus, il n'existe, à notre connaissance, aucun outil permettant de mettre en évidence de telles molécules. Nous souhaitons également mentionner que l'utilisation de BenzAI décrite dans cette section n'était pas prévue initialement, mais qu'elle a été mise en évidence suite à la prise en main de notre logiciel par les chimistes.

7.4. Une meilleure interprétation des spectres infrarouges grâce à BenzDB

En 2019, Bouwmann et al. (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) ont présenté des travaux portant sur l'étude des spectres infrarouges des HAP admettant au plus 9 cycles. Au cours de ces travaux, ils ont effectué des statistiques sur les paramètres d'irrégularité de l'ensemble de molécules considérées. Ces dernières ont été récupérées via la NASA Ames PAH IR Spectroscopic Data (https://www.astrochemistry.org/pahdb/). Or, ces derniers ont mis en évidence un biais : ils se sont rendu compte que cette base de données contenait environ 400 composés possédant plus de 34 atomes de carbone et 9 hexagones ou moins, ce qui est très peu et loin d'être exhaustif. À titre de comparaison, notre base de données *BenzDB* (voir section 6.2), qui recense tous les benzénoïdes possédant 9 hexagones ou moins de manière exhaustive (générés à l'aide de notre



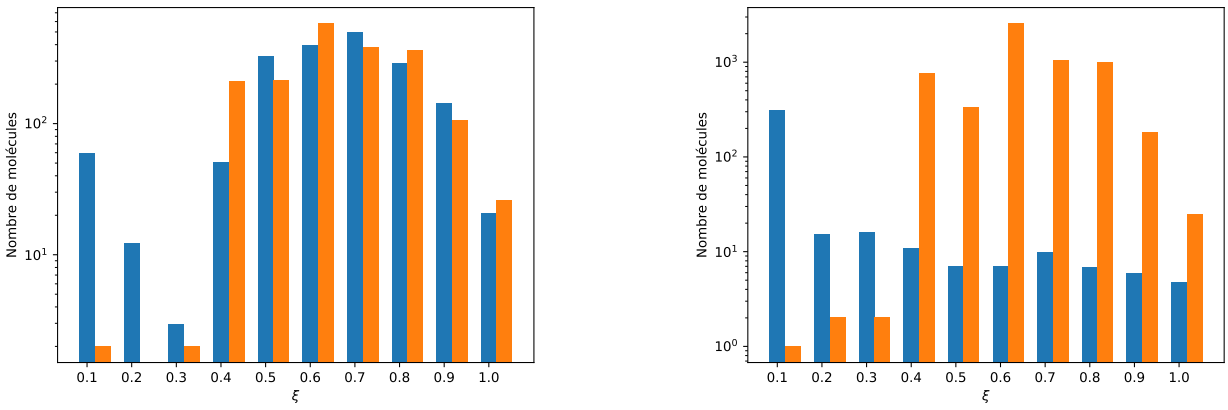
(a) (b)

FIGURE 7.19. – Molécules obtenues après la trimérisation des composés décrits dans la figure 7.18

logiciel BenzAI), admet plusieurs milliers de composés supplémentaires. Cette différence est illustrée par la figure 7.20 qui détaille la comparaison des statistiques d'irrégularité présentés dans les travaux de Bouwmann et al. (en bleu) avec les nôtres (en orange) pour l'ensemble des benzénoïde possédant 9 hexagones ou moins. Les molécules possédant 34 atomes de carbone ou moins sont répertoriés dans le graphique (a) tandis que les autres le sont dans le graphique (b). Ce choix de 34 carbones est celui fait dans les travaux de Bouwman et al..

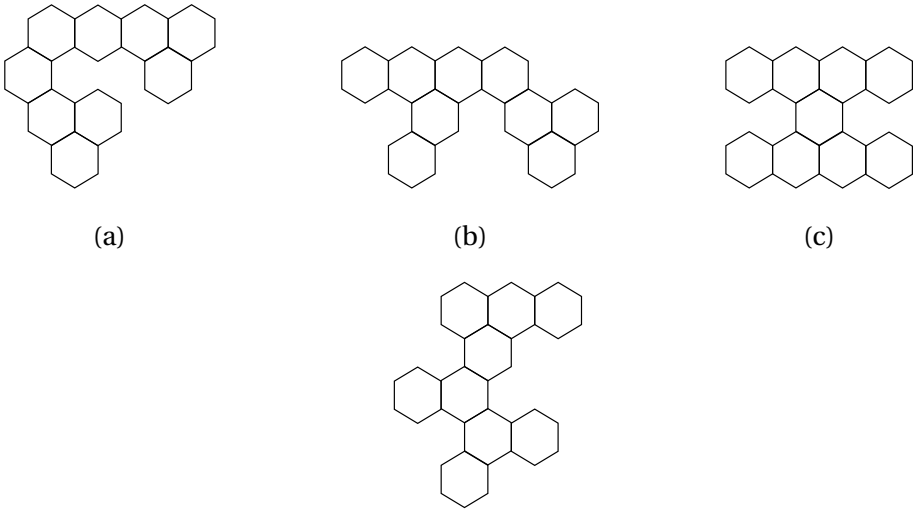
Si l'on regarde ces deux graphiques, la première chose frappante que l'on observe est le fait que notre base de données répertorie plusieurs milliers de benzénoïdes de plus que la NASA Ames PAH IR Spectroscopic Data. Les cas de figures où cette dernière admet plus de molécules que BenzDB (par exemple, pour des paramètres d'irrégularité de 0,1, 0,2, 0,3 dans les deux graphiques) sont dus au fait qu'ils ont également considéré des molécules autres que des benzénoïdes dans leurs travaux. La figure 7.21 décrit des benzénoïdes ayant un paramètre d'irrégularité de 0,6 (a), 0,7 (b), 0,8 (c) et 0,9 (d). D'après le graphique de droite, la base de données de la NASA ne recense presque aucune molécule de plus de 34 carbones admettant ces paramètres d'irrégularité, alors que BenzDB en compte plus de 4 000. Cela signifie que ces molécules ont été complètement ignorées dans les travaux de Bouwmann et al. Un autre point notable est l'absence de benzénoïdes dont le paramètre d'irrégularité est inférieur à 0,3. En effet, notre base de données n'en recense presque aucun et comme mentionné plus haut, ceux présents dans la base de la NASA ne sont pas des benzénoïdes.

L'utilisation de notre base de données couplée à BenzAI peut donc s'avérer utile pour effectuer des calculs de spectres infrarouges. Premièrement, elle contient les données nécessaires à ces calculs, ce qu'il fait que l'utilisateur n'aura pas besoin de les refaire. Pour rappel, ces calculs sont relativement coûteux et certains d'entre eux peuvent prendre plus d'une journée. Deuxièmement, la garantie d'exhaustivité permet une meilleure interprétation des spectres pour les chimistes. Pour illustrer cela, la figure 7.22 décrit les trois courbes de spectres infrarouges pour les benzénoïdes possédant quatre hexagones. Ces courbes ont été générées à l'aide de notre logiciel BenzAI : ce dernier se contente de récupérer les valeurs en interrogeant BenzDB, puis de calculer et d'afficher les courbes



(a) (b)

FIGURE 7.20. – Comparaison des distributions des paramètres d'irrégularité pour les benzénoïdes admettant au plus 9 hexagones (au plus 34 atomes de carbone (a) et plus de 34 atomes de carbone (b)) issus de la NASA Ames PAH IR Spectroscopic Data (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) (en bleu) et l'ensemble exhaustif de benzénoïdes générés à l'aide de BenzAI (en orange)



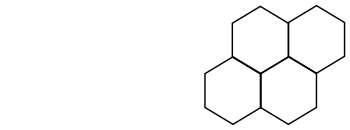
(a) (b) (c)

(d)

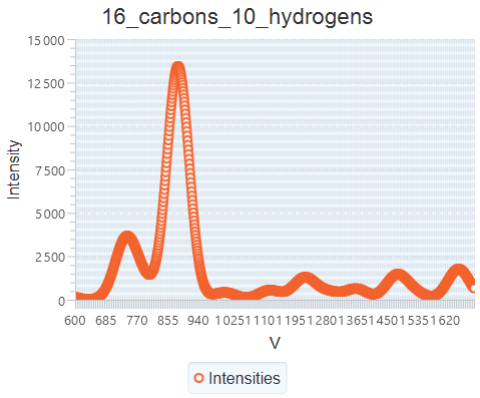
FIGURE 7.21. – Benzénoïdes de plus de 34 carbones ayant un paramètre d'irrégularité de 0.6 (a), 0.7 (b), 0.8 (c) et 0.9 (d)

7.5. Conclusions

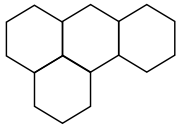
Dans ce chapitre, nous avons présenté les premières contributions à la chimie pouvant découler des travaux présentés au cours de cette thèse. Dans la section 7.2, nous avons présenté les résultats détaillés concernant deux autres méthodes, ClarCP et RBO capables d'estimer l'aromaticité d'un benzénoïde. Ces dernières reposent respectivement sur l'énumération des couvertures de Clar (notion définie dans la section 2.4.5) et sur le calcul des Ring Bond Order (RBO) (défini dans la section 2.4.6). Nous avons, par la suite,



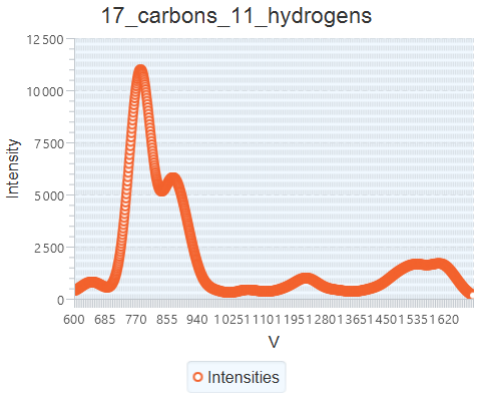
(a)



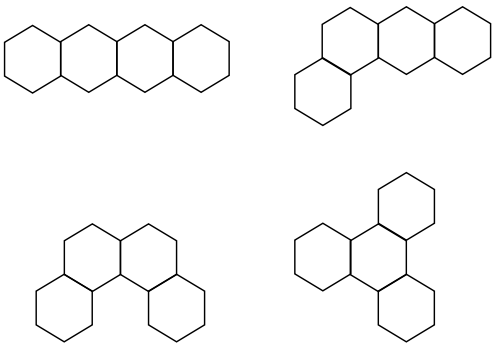
(b)



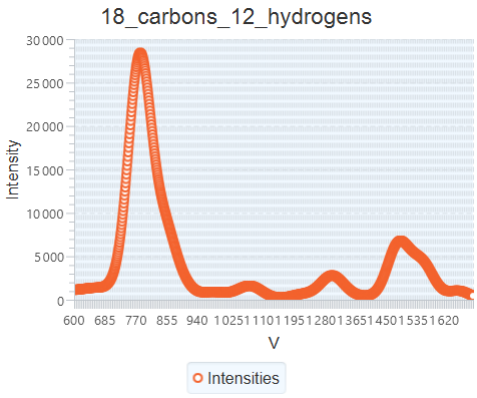
(c)



(d)



(e)



(f)

FIGURE 7.22. – Courbes de spectres infrarouges pour l'ensemble des benzénoïdes de 4 hexagones

comparé ces deux méthodes avec CERCP (présentée dans la section 5.3) et NICS (présenté dans la section 2.4.2). Pour cela, nous avons comparé les différents temps d'exécution de chaque méthode et calculé le coefficient de corrélation de Pearson pour chaque couple de méthodes et pour chaque instance résolue. Il apparaît que ces deux nouvelles méthodes sont bien plus rapides que les précédentes, et particulièrement RBO qui ne requiert aucune énumération. De plus, il apparaît que les méthodes ClarCP, RBO et CERCP ont des tendances très similaires sur la grande majorité des instances considérées. Cela peut s'expliquer par le fait que ces trois méthodes possèdent un fonctionnement similaire. Dans l'ensemble, ces trois méthodes affichent des tendances similaires à celle de NICS, malgré leurs différences de fonctionnement.

Dans la section 7.3, nous avons présenté deux manières d'utiliser notre logiciel BenzAI afin d'accompagner les chimistes dans la synthèse de molécules. La sous-section 7.3.1 décrit une manière d'utiliser le logiciel afin de mettre en évidence des benzénoïdes entièrement aromatiques obtenables en ajoutant un ou plusieurs hexagones à des benzénoïdes qui le sont partiellement. Ce procédé peut donc permettre de simplifier la synthèse de certaines molécules. La sous-section 7.3.2 présente, quant à elle, une manière d'utiliser BenzAI et sa faculté de générer des benzénoïdes admettant des motifs donnés afin d'exhiber des trimères admettant un grand nombre d'électrons radicalaires. Nous avons mis en évidence un trimère admettant six électrons radicalaires stables. Les molécules de ce type sont relativement rares et il est difficile pour les chimistes de les mettre en évidence. Il est donc intéressant pour les chimistes d'être capable d'en identifier certaines. Il convient également de mentionner que les contributions présentées dans cette section vont au-delà de ce que nous avions prévu initialement. En effet, ce sont simplement des applications des fonctionnalités du calcul d'énergie de résonance et de la génération de structures de benzénoïdes admettant des motifs présentes dans BenzAI.

Dans la section 7.4, nous avons présenté des résultats de spectres infrarouges obtenues grâce aux informations contenues dans notre base de données BenzDB. Nous avons comparé l'ensemble des benzénoïdes qu'elle contient avec ceux de la *NASA PAH Ames Database* (https://www.astrochemistry.org/pahdb/) et en analysant les travaux de Bouwmann et al. (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019) qui effectue des calculs de spectres infrarouges sur cette base. Nous avons constaté que cette dernière était incomplète et que parmi les benzénoïdes d'au plus 9 hexagones, il y avait plusieurs milliers de molécules qu'elle ne répertoriait pas. Les benzénoïdes présents dans BenzDB ont été générés avec le modèle général présenté dans la section 3.3, ce qui induit une garantie d'exhaustivité. De plus, les calculs pour récupérer les valeurs nécessaires au calcul de ces spectres sont très coûteux (traiter une molécule peut nécessiter plus d'une journée de temps de calcul) et les rendre publics peut permettre aux chimistes de les récupérer sans avoir à refaire ces calculs.

Les contributions présentées dans ce chapitre appartiennent à des domaines variés de la chimie (génération de structures de benzénoïdes, estimation de l'aromaticité, synthèse de molécules, calcul de spectres infrarouges, ...). Cela démontre la polyvalence des outils que nous avons mis au point. Pour conclure, il convient de mentionner que ces contributions restent préliminaires et que d'autres sont susceptibles d'être mises en évidence par la suite.

Quatrième partie

Conclusions

Au cours de cette thèse, nous avons abordé des problématiques relatives aux HAP et, plus particulièrement, aux benzénoïdes. Cette partie fera la synthèse des différents travaux ayant été effectués au cours de cette thèse. Tout d'abord, les chapitres 1 et 2 dressent un état de l'art de tous les aspects liés respectivement à la programmation par contraintes et à la chimie théorique qui ont été utile au cours de cette thèse. Dans les chapitres suivants, nous avons abordé deux problématiques principales : être capable de générer, de manière exhaustive, les structures de benzénoïdes satisfaisant diverses propriétés (chapitres 3 et 4) et pouvoir estimer l'aromaticité d'un benzénoïde en calculant son énergie de résonance (chapitre 5). Le chapitre 6 décrit, quant à lui, deux outils à destination des chimistes que nous avons mis au point. Pour finir, le chapitre 7 détaille les premières contributions à la chimie que nos travaux ont apporté.

Beaucoup de problématiques liées aux benzénoïdes sont considérées comme difficiles en raison de leur forte combinatoire. Nous avons émis l'hypothèse que la *Programmation Par Contraintes (PPC)* était la candidate idéale pour résoudre ces problématiques. En effet, la PPC s'est maintes fois illustrée par sa capacité à résoudre des problèmes de natures variées admettant une forte combinatoire. Ainsi, même si elle peut être, dans certains cas, moins efficace qu'une méthode ad-hoc, elle constitue un excellent compromis entre flexibilité et performance. L'objectif général de cette thèse a donc été d'utiliser la programmation par contraintes dans le but de résoudre des problématiques liées aux benzénoïdes. Comme mentionné plus haut, nous avons principalement étudié deux problématiques : la génération de structures de benzénoïdes ainsi que l'estimation de l'aromaticité d'un benzénoïde donné. Pour chacune d'entre elles, nous faisons maintenant une synthèse des travaux effectués et détaillons les premières retombées dans le monde de la chimie théorique ainsi que les perspectives envisagées.

Générer des structures de benzénoïdes satisfaisant diverses propriétés (chapitres 3 et 4)

Pour traiter cette problématique, nous avons commencé par mettre en place un modèle CSP dit *général*, capable de générer l'intégralité des structures de benzénoïdes possédant au plus un nombre donné d'hexagones. Nous considérons ensuite des ensembles de variables et de contraintes appelés *modules* que nous pouvons greffer au modèle général. Chaque module représente une propriété additionnelle que les benzénoïdes générés admettront. En procédant de la sorte, nous obtenons un outil possédant une grande flexibilité. En effet, l'utilisateur est libre de choisir les modules qu'il souhaite utiliser, de les combiner, ... De plus, si l'on souhaite modéliser une nouvelle propriété, il suffira d'implémenter le module correspondant, ce qui demandera un effort minime en termes de programmation. Le chapitre 3 détaille la majorité des propriétés globales que nous avons modélisées. Ces dernières peuvent être structurelles (être un benzénoïde catacondensé, un coronénoïde, un coronoïde, ...) ou bien chimiques (avoir un nombre de carbones et/ou d'hydrogènes donné(s) ou encore avoir une certaine valeur pour le paramètre d'irrégularité).

Le chapitre 4 se focalise, quant à lui, sur une sous-problématique spécifique : la présence de *motifs* au sein d'un benzénoïde. Pour rappel, un motif est une propriété locale qui se définit comme étant une partie d'un benzénoïde possédant une forme spécifique. En spécifiant quels motifs doivent être admis par les molécules générées, il est donc possible de déterminer leur topologie. Pour cela, nous avons proposé un ou plusieurs modules capables de modéliser chacune des propriétés suivantes :

- contenir un motif donné,
- contenir plusieurs motifs donnés,
- contenir un motif donné un certain nombre de fois,
- ne pas contenir un motif donné.

Grâce à l'efficacité et à la flexibilité de la programmation par contraintes, nous avons obtenu un outil capable de générer des ensembles de structures exhaustifs de manière efficace. Il nous est très facile de modéliser de nouvelles propriétés afin de suivre l'évolution des besoins de chimistes. De plus, contrairement

à la méthode présentée par Brinkmann et al. (BRINKMANN, CAPOROSSI et HANSEN 2002), notre approche permet de générer les coronoïdes (les benzénoïdes admettant des trous). Les expérimentations ont démontré que notre approche était capable de générer des molécules de taille suffisamment grande pour qu'elles soient intéressantes pour les chimistes et tout cela, avec des temps d'exécution très corrects. Ces travaux ont donné lieu à une publication dans les conférences CP2020 (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2020) et CP2021 (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2021) et ont également contribué à une publication dans le journal Constraints (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2022).

Estimer l'aromaticité d'un benzénoïde grâce à l'énergie de résonance (chapitre 5)

L'*aromaticité* peut être vue comme étant l'énergie induite par la délocalisation des électrons-π d'un HAP. Ce phénomène apporte aux HAP une large variété de propriétés chimiques, physiques et magnétiques intéressantes et peu communes. Il existe deux manières d'appréhender l'aromaticité d'un benzénoïde, cette dernière peut être exprimée de manière globale (en attribuant une valeur énergétique au benzénoïde traité) ou locale (en attribuant une valeur à chaque hexagone du benzénoïde). L'approche locale est la plus prisée de chimistes : elle permet de déterminer quelles parties du benzénoïde traité sont les plus stables et inversement. Elle peut donc contribuer à prévoir les différentes réactions chimiques qui peuvent avoir lieu. Il convient également de rappeler que l'aromaticité n'est pas quelque chose de calculable, ni de mesurable. Il est seulement possible de l'estimer via différentes méthodes. Parmi ces dernières, nous pouvons citer *Nucleus Independent Chemical Shift (NICS)* (DRANSFELD, SCHLEYER et VAN EIKEMA HOMMES 1996), qui repose sur de coûteux calculs de chimie quantique, ou les méthodes reposant sur l'*énergie de résonance* (RANDIĆ 1976a ; LIN et FAN 1999 ; LIN 2000).

NICS est à l'heure actuelle la méthode de référence des chimistes quand il s'agit d'estimer l'aromaticité d'un benzénoïde. Néanmoins, cette dernière est très coûteuse en temps et le traitement d'une molécule de taille conséquente peut rapidement prendre plusieurs heures (voire plus d'une journée). De plus, il n'existe, à notre connaissance, aucune implémentation disponible des algorithmes de Lin & Fan et de Lin. C'est dans ce contexte-là que nous avons présenté une implémentation de la méthode de Lin & Fan ainsi qu'une amélioration de la méthode de Lin utilisant toutes deux la programmation par contraintes et nommées respectivement LFCP et CERCP. Une fois ces deux méthodes implémentées, nous avons effectué des comparaisons expérimentales entre ces dernières et NICS afin, d'une part, de comparer les différents temps d'exécution et d'autre part la similarité des résultats obtenus (cela a été fait à l'aide du coefficient de corrélation de Pearson). La méthode CERCP s'est révélé être plus performante que NICS en termes de temps d'exécution et semble fournir des résultats d'une qualité similaire dans la très grande majorité des cas. Ces travaux ont donné lieu à une publication dans la conférence CP2020 (CARISSAN, DIM, HAGEBAUM-REIGNIER et al. 2020) et ont contribué à une publication dans le journal Constraints (CARISSAN, HAGEBAUM-REIGNIER, PRCOVIC et al. 2022).

Le logiciel BenzAI et la base de données BenzDB (chapitre 6)

Au cours de cette thèse, nous avons également développé le logiciel *BenzAI* (https://benzai-team.github.io/) à destination des chimistes qui regroupe l'intégralité des outils que nous avons mis au point. Ce logiciel s'appuie sur une base de données baptisée *BenzDB* que nous avons également déployée. Cette dernière contient un grand nombre d'informations sur l'intégralité des molécules possédant 9 hexagones ou moins (spectres infrarouges, valeurs NICS, cartes IMS, informations pour faire le lien avec BenzAI, ...). Avec ce logiciel, il est possible de générer des structures de benzénoïdes pouvant satisfaire toutes les propriétés ayant été évoquées dans les chapitres 3 et 4 et les visualiser dans des fenêtres appelées *collections* (une collection peut également être remplie en important les molécules directement depuis BenzDB). Une fois qu'une molécule est présente dans une collection, on peut effectuer les opérations suivantes :

- calcul de son énergie de résonance,
- calcul de ses RBO,

- calcul de ses couvertures de Clar,
- calcul de ses structures de Kekulé,
- statistiques sur les paramètres d'irrégularité des benzénoïdes d'une collection,
- récupérer ses données de spectre infrarouge depuis BenzDB,
- récupérer ses données de carte IMS depuis BenzDB.

Ce logiciel regroupe donc un grand nombre de fonctionnalités derrière une interface intuitive. Les utilisateurs (en grande majorité des chimistes) peuvent donc facilement s'en servir, et ce, même s'ils ignorent quels algorithmes sont utilisés en arrière-plan. En particulier, aucune connaissance en programmation par contraintes n'est requise. Un article détaillant les différentes fonctionnalités de BenzAI a été publié dans le *Journal of Chemical Information and Modeling (JCIM)* et un autre présentant la base de données BenzDB sera soumis dans les mois à venir.

Premières contributions de nos travaux à la chimie (chapitre 7)

Ce chapitre présente les différentes contributions apportées à la chimie par nos travaux. Dans la section 7.2, nous avons présenté deux méthodes (RBO et ClarCP) permettant d'estimer l'aromaticité d'un benzénoïde plus rapidement qu'en utilisant CERCP ou NICS. Néanmoins, malgré leurs grandes performances, ces méthodes peuvent manquer de finesse dans certains cas. En effet, il peut arriver que ces méthodes attribuent la même valeur à tous les hexagones. Il convient donc à l'utilisateur de choisir la méthode correspondant le plus à ses besoins.

La section 7.3 a présenté, quant à elle, deux manières d'utiliser notre logiciel BenzAI afin d'accompagner les chimistes dans la synthèse de molécules. La sous-section 7.3.1 détaille une manière d'utiliser notre logiciel afin de générer des benzénoïdes entièrement aromatiques synthétisables à partir d'autres benzénoïdes qui ne le sont que partiellement. Ensuite, nous avons présenté, dans la sous-section 7.3.2, une manière d'utiliser BenzAI afin de mettre en évidence des benzénoïdes synthétisables par trimérisation et admettant un grand nombre d'électrons radicalaires.

Dans la section 7.4, nous avons expliqué comment notre base de données BenzDB pouvait fournir aux chimistes une meilleure interprétation des spectres infrarouges concernant les benzénoïdes d'au plus 9 hexagones. Nous avons mentionné les travaux de Bouwmann et al. (BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. 2019), qui ont présenté en 2019 une étude sur les spectres infrarouges de certains benzénoïdes obtenus via la *NASA Ames PAH database*. Ces derniers ont mis en évidence que la liste des benzénoïdes présents dans cette base était loin d'être exhaustive. Or, notre base de données BenzDB contient tous les benzénoïdes possédant 9 hexagones ou moins (ici, l'exhaustivité est garantie par le fait que nous avons généré les benzénoïdes à l'aide du modèle général présenté dans la section 3.3) et contient également les données nécessaires aux calculs de spectres infrarouges de chacun d'entre eux. BenzDB peut donc permettre aux chimistes de travailler sur des ensembles de molécules complets, ce qui induit une meilleure interprétation de leurs spectres infrarouges.

Comme nous avons pu le voir, certaines de ces contributions exploitent les outils proposés au-delà de leurs fonctionnalités premières et requièrent des compétences supplémentaires en chimie. Elles dépendent donc grandement de l'appropriation de nos outils par les chimistes et l'utilisation qu'ils en feront (comme, l'accompagnement à la synthèse de différentes molécules décrit dans la section 7.3). Pour finir, il convient également de mentionner que l'étude de leurs retombées en chimie dépasse le cadre d'une thèse en informatique.

Synthèse sur l'exploitation de la programmation par contraintes

Au final, l'utilisation de la programmation par contraintes pour résoudre différentes problématiques de chimie théorique s'est avérée judicieuse. Elle nous a permis de pouvoir modéliser un grand nombre de propriétés de diverses natures (structurelles ou chimiques, locales ou globales) et de les combiner aisément. De plus, Choco Solver nous a permis de résoudre efficacement chaque propriété modélisée tout en réduisant significativement le temps consacré à la conception et à la mise œuvre de nos solutions. En théorie, la plus grande force de la programmation par contraintes est que son utilisateur n'a qu'à décrire un problème et le faire résoudre par un solveur. En pratique, cela s'avère globalement le cas même si des subtilités existent. La programmation par contraintes admet en effet certaines limites, comme on a pu le voir par exemple avec la propriété sur le nombre de structures de Kekulé pour laquelle le filtrage ne peut se faire qu'après avoir entièrement généré la molécule. À titre personnel, j'ai notamment rencontré quelques difficultés liées à son utilisation, principalement liées à Choco Solver. En effet, il est arrivé que certaines contraintes que j'ai spécifiées soient réinterprétées en arrière-plan par le solveur. De plus, il subsiste une certaine opacité au niveau de l'algorithme de parcours ainsi que des différentes heuristiques et méthodes de filtrage. Cette opacité a rendu délicate le débuggage de certains modèles. De plus, même si nous avons pu, de manière générale, modéliser rapidement les différentes propriétés, l'optimisation des modèles obtenus a nécessité beaucoup de travail et nous a demandé de nous plonger dans le code source du solveur. Pour ces raisons-là, je pense que même si des solveurs comme Choco Solver peuvent être facilement pris en main, de bonnes connaissances en programmation par contraintes sont requises pour pouvoir les utiliser à leur plein potentiel que ce soit du point de vue de la modélisation ou de celui de la résolution.

Pour conclure, nous avons pu aborder au cours de cette thèse plusieurs thématiques, allant de la résolution de problématiques de chimie théorique à la mise en place d'un logiciel. L'intégralité de ces travaux ont donné lieu à sept publications (deux dans des conférences nationales, trois dans des conférences internationales et deux dans des journaux). De plus, les problématiques que nous avons abordées sont loin d'être closes et offrent des perspectives intéressantes pour la suite.

Perspectives

Au niveau des perspectives, celles relatives à la génération de structures de benzénoïdes pourraient être de s'inspirer des travaux de Matsuoka et al. (MATSUOKA, ITO, SARLAH et al. 2021). Pour rappel, ces derniers décrivent une méthode de synthèse consistant à partir d'un benzénoïde initial et d'y ajouter successivement un même motif à différents endroits. Une première idée serait de proposer une méthode de génération reprenant ce procédé. Cette dernière pourrait être vue comme une généralisation de notre modèle général ayant la possibilité d'ajouter plusieurs hexagones en même temps. Une telle méthode pourrait permettre de générer rapidement des benzénoïdes de grande taille. L'inverse serait également possible et consisterait à mettre en place un modèle capable de générer directement les molécules synthétisables avec la méthode de Matsuoka et al. Nous pensons également travailler sur la génération de structures de benzénoïdes optimisant certains critères comme le nombre de carbones/hydrogènes/hexagones, le nombre de trous, ... Un autre point plus délicat à traiter serait de générer des benzénoïdes incluant un motif donné le plus de fois possible. Une perspective plus générale serait de ne plus se limiter à des cycles de 6 carbones en générant par exemple des *azulénoïdes* (voir la figure 3.33), s'intéresser à la génération de molécules non-planaires telles que les *fullerènes* (voir la figure 3.34) ou encore sortir du cadre des HAP en considérant des molécules possédant autre chose que du carbone et de l'hydrogène.

Pour ce qui est de l'estimation de l'aromaticité, une perspective intéressante serait de prendre en compte la notion de symétries au sein des méthodes CERCP et ClarCP afin de limiter grandement le temps de calculs. Cela a d'autant plus de sens que les chimistes s'intéressent souvent à des molécules symétriques.

Comme mentionné plus haut, les perspectives relatives à BenzAI dépendront principalement des retours de la communauté des chimistes que nous recevrons. Pour notre base de données BenzDB, nous comptons l'étoffer davantage en ajoutant de benzénoïdes admettant plus de 9 hexagones. Néanmoins, cela risque de

nécessiter un temps de calcul très conséquent. En effet, il y a plus de 30 000 benzénoïdes de 10 hexagones et les calculs effectués pour récupérer les informations de ceux ayant 9 hexagones ou moins ont demandé plusieurs mois de temps CPU. Au même titre que pour la génération, nous comptons rester à l'écoute des chimistes et intégrer à notre base les informations les plus demandées.

Nous avons vu que l'utilisation de la programmation par contraintes pouvait être adaptée à la résolution de problèmes de chimie théorique. Une dernière perspective pourrait être de s'intéresser à d'autres problèmes de chimie fortement combinatoires. De manière plus générale, nous avons prévu de rester à l'écoute de la communauté des chimistes et de tenter de proposer des modèles permettant de générer les molécules qui les intéressent.

Bibliographie

AHLRICHS, Reinhart, Michael BÄR, Marco HÄSER et al. « Electronic structure calculations on workstation computers : The program system turbomole ». In : *Chemical Physics Letters* 162.3 (1989), p. 165-169. ISSN : 0009-2614. DOI : https://doi.org/10.1016/0009-2614(89)85118-8. URL : https://www.sciencedirect.com/science/article/pii/0009261489851188 (cf. p. 181).

AJAYAKUMAR, M. R., Ji MA, Andrea LUCOTTI et al. « Persistent Peri-Heptacene : Synthesis and In Situ Characterization ». en. In : *Angew. Chem. Int. Ed.* (2021). ISSN : 1521-3773. DOI : 10.1002/anie.202102757 (cf. p. 60, 117, 135).

ALLAMANDOLA, L. J., D. M. HUDGINS et S. A. SANDFORD. « Modeling the Unidentified Infrared Emission with Combinations of Polycyclic Aromatic Hydrocarbons ». In : *The Astrophysical Journal* 511.2 (1999), p. L115-L119. DOI : 10.1086/311843 (cf. p. 60).

ALLOUCHE, Florian, Giuseppe LAPADULA, Georges SIDDIQI et al. « Magnetic Memory from Site Isolated Dy(III) on Silica Materials ». In : *ACS Central Science* 3.3 (fév. 2017), p. 244-249. DOI : 10.1021/acscentsci.7b00035. URL : https://hal.sorbonne-universite.fr/hal-01480162 (cf. p. 179).

ALVAREZ-RAMÍREZ, Fernando et Yosadara RUIZ-MORALES. « Database of Nuclear Independent Chemical Shifts (NICS) versus NICS $_{ZZ}$ of Polycyclic Aromatic Hydrocarbons (PAHs) ». In : *Journal of Chemical Information and Modeling* 60.2 (2020), p. 611-620. DOI : 10.1021/acs.jcim.9b00909 (cf. p. 163).

ANTUORI, Valentin, Emmanuel HÉBRARD, Marie-José HUGUET et al. « Leveraging Reinforcement Learning, Constraint Programming and Local Search : A Case Study in Car Manufacturing ». In : *Principles and Practice of Constraint Programming. CP 2020*. Sous la dir. de vol 12333 Lecture Notes in COMPUTER SCIENCE. Louvain La Neuve, Belgium, sept. 2020, p. 657-672. DOI : 10.1007/978-3-030-58475-7_38. URL : https://hal.laas.fr/hal-02938190 (cf. p. 27).

APT, K. *Principles of constraint programming*. Cambridge University Press, 2003 (cf. p. 31).

APT, Krzysztof R. *Principles of constraint programming*. Cambridge University Press, 2003 (cf. p. 41).

ARTIGAS, Albert, Denis HAGEBAUM-REIGNIER, Yannick CARISSAN et al. « Visualizing electron delocalization in contorted polycyclic aromatic hydrocarbons ». In : *Chem. Sci.* 12 (39 2021), p. 13092-13100. DOI : 10.1039/D1SC03368A. URL : http://dx.doi.org/10.1039/D1SC03368A (cf. p. 179).

AUDEMARD, Gilles, Christophe LECOUTRE et Emmanuel LONCA. *Proceedings of the 2022 XCSP3 Competition*. 2022. eprint : arXiv:2209.00917 (cf. p. 38).

AUMAITRE, Cyril et Jean-François MORIN. « Polycyclic Aromatic Hydrocarbons as Potential Building Blocks for Organic Solar Cells ». In : *The Chemical Record* 19.6 (2019), p. 1142-1154. DOI : 10.1002/tcr.201900016. eprint : https://onlinelibrary.wiley.com/doi/pdf/10.1002/tcr.201900016. URL : https://onlinelibrary.wiley.com/doi/abs/10.1002/tcr.201900016 (cf. p. 22).

BAUSCHLICHER, Charles W., A. RICCA, C. BOERSMA et al. « The NASA Ames PAH IR Spectroscopic Database : Computational Version 3.00 with Updated Content and the Introduction of Multiple Scaling Factors ». In : *The Astrophysical Journal Supplement Series* 234.2 (2018), p. 32. DOI : 10.3847/1538-4365/aaa019. URL : https://doi.org/10.3847\%2F1538-4365\%2Faaa019 (cf. p. 102).

BAUSCHLICHER Jr., Charles W., Els PEETERS et Louis J. ALLAMANDOLA. « The Infrared Spectra of Very Large, Compact, Highly Symmetric, Polycyclic Aromatic Hydrocarbons (PAHs) ». In : *The Astrophysical Journal* 678.1 (2008), p. 316-327. DOI : 10.1086/533424 (cf. p. 60, 95, 112).

BAYARDO, Roberto J. et Daniel P. MIRANKER. « A Complexity Analysis of Space-Bounded Learning Algorithms for the Constraints Satisfaction Problem ». In : *Proceedings of 13th National Conference on Artificial Intelligence (AAAI)*. 1996, p. 298-304 (cf. p. 47).

BELDICEANU, N et E CONTEJEAN. « Introducing global constraints in CHIP ». In : *Mathematical and Computer Modelling* 20.12 (1994), p. 97-123. ISSN : 0895-7177. DOI : https://doi.org/10.1016/0895-7177(94)90127-9. URL : https://www.sciencedirect.com/science/article/pii/0895717794901279 (cf. p. 48).

BERTELÈ, Umberto et Francesco BRIOSCHI. *Nonserial Dynamic Programming*. Academic Press, 1972 (cf. p. 39).

BESER, Uliana, Marcel KASTLER, Ali MAGHSOUMI et al. « A C216-Nanographene Molecule with Defined Cavity as Extended Coronoid ». In : *Journal of the American Chemical Society* 138.13 (2016), p. 4322-4325. DOI : 10.1021/jacs.6b01181 (cf. p. 92).

BESSIÈRE, Christian. « Arc-consistency and arc-consistency again ». In : *Artificial Intelligence* 65.1 (1994), p. 179-190 (cf. p. 42).

BESSIÈRE, Christian et Jean-Charles RÉGIN. « MAC and Combined Heuristics : Two Reasons to Forsake FC (and CBJ ?) on Hard Problems ». In : *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming (CP)*. 1996, p. 61-75 (cf. p. 44).

BESSIÈRE, Christian, Jean-Charles RÉGIN, Roland H. C. YAP et al. « An optimal coarse-grained arc consistency algorithm ». In : *Artificial Intelligence* 165.2 (2005), p. 165-185 (cf. p. 42).

BEUVIN, François, Simon de GIVRY, Thomas SCHIEX et al. « Iterated local search with partition crossover for computational protein design ». In : *Proteins : Structure, Function, and Bioinformatics* (2021) (cf. p. 38).

BOUSSEMART, Frédéric, Fred HEMERY, Christophe LECOUTRE et al. « Boosting Systematic Search by Weighting Constraints ». In : *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*. 2004, p. 146-150 (cf. p. 44, 131).

BOUWMAN, Jordy, Harold LINNARTZ et Alexander G.G.M. TIELENS. « Mid-infrared spectroscopic signatures of dibenzopyrene cations – The effect of symmetry on PAH IR spectroscopy ». In : *Journal of Molecular Spectroscopy* 378 (2021), p. 111458. ISSN : 0022-2852. DOI : https://doi.org/10.1016/j.jms.2021.111458. URL : https://www.sciencedirect.com/science/article/pii/S0022285221000424 (cf. p. 62, 95, 112, 150).

BOUWMAN, J., CASTELLANOS, P., BULAK, M. et al. « Effect of molecular structure on the infrared signatures of astronomically relevant PAHs ». In : *A&A* 621 (2019), A80. DOI : 10.1051/0004-6361/201834130. URL : https://doi.org/10.1051/0004-6361/201834130 (cf. p. 25, 58, 59, 81, 97, 159, 182, 184, 186, 190).

BRINKMANN, G., G. CAPOROSSI et P. HANSEN. « A Constructive Enumeration of Fusenes and Benzenoids ». In : *Journal of Algorithms* 45(2) (2002) (cf. p. 25, 26, 62, 81, 82, 87, 92, 115, 189).

BRUNVOLL, J., R. N. CYVIN et S. J. CYVIN. « Enumeration and Classification of Double Coronoid Hydrocarbons – Appendix : Triple Coronoids ». In : *Croatica Chemica Acta* 63.4 (1990), p. 585-601 (cf. p. 81, 92).

BURTON, Robert et Jeffrey E. STEIF. « Non-uniqueness of measures of maximal entropy for subshifts of finite type ». In : *Ergodic Theory and Dynamical Systems* 14.2 (1994), 213–235. DOI : 10.1017/S0143385700007859 (cf. p. 38).

BÉJAR, Ramón, Carmel DOMSHLAK, Cèsar FERNÁNDEZ et al. « Sensor networks and distributed CSP : communication, computation and complexity ». In : *Artificial Intelligence* 161.1 (2005). Distributed Constraint Satisfaction, p. 117-147. ISSN : 0004-3702. DOI : https://doi.org/10.1016/j.artint.2004.09.002. URL : https://www.sciencedirect.com/science/article/pii/S000437020400150X (cf. p. 38).

CABON, Bertrand, Simon DE GIVRY, Lionel LOBJOIS et al. « Radio link frequency assignment ». In : *Constraints* 4.1 (1999), p. 79-89. DOI : 10.1023/A:1009812409930. URL : https://hal.inrae.fr/hal-02695373 (cf. p. 38).

CAPOROSSI, Gilles et Pierre HANSEN. « Enumeration of Polyhex Hydrocarbons to h = 21 ». In : *Journal of Chemical Information and Computer Sciences* 38.4 (1998), p. 610-619. DOI : 10.1021/ci970116n. eprint : https://doi.org/10.1021/ci970116n. URL : https://doi.org/10.1021/ci970116n (cf. p. 26, 60, 62, 115).

CARISSAN, Y., D. HAGEBAUM-REIGNIER, N. PRCOVIC et al. « Using Constraint Programming to Generate Benzenoid Structures in Theoretical Chemistry ». In : *CP*. 2020, p. 690-706 (cf. p. 115, 189).

CARISSAN, Yannick, Chisom-Adaobi DIM, Denis HAGEBAUM-REIGNIER et al. « Computing the Local Aromaticity of Benzenoids Thanks to Constraint Programming ». In : *CP*. Springer International Publishing, 2020, p. 673-689. DOI : 10.1007/978-3-030-58475-7_39. URL : https://doi.org/10.1007/978-3-030-58475-7_39 (cf. p. 142, 143, 148, 189).

CARISSAN, Yannick, Denis HAGEBAUM-REIGNIER, Nicolas PRCOVIC et al. « Exhaustive Generation of Benzenoid Structures Sharing Common Patterns ». In : *27th International Conference on Principles and Practice of Constraint Programming*. Montpellier, France, oct. 2021. DOI : 10.4230/LIPIcs.CP.2021.19. URL : https://hal-amu.archives-ouvertes.fr/hal-03402690 (cf. p. 135, 189).

CARISSAN, Yannick, Denis HAGEBAUM-REIGNIER, Nicolas PRCOVIC et al. « How constraint programming can help chemists to generate Benzenoid structures and assess the local Aromaticity of Benzenoids ». In : *Constraints* (mai 2022). DOI : 10.1007/s10601-022-09328-x. URL : https://doi.org/10.1007/s10601-022-09328-x (cf. p. 89, 115, 147, 148, 189).

CARLSSON, Mats et Nicolas BELDICEANU. « From Constraints to Finite Automata to Filtering Algorithms ». In : *Programming Languages and Systems*. Sous la dir. de David SCHMIDT. Berlin, Heidelberg : Springer Berlin Heidelberg, 2004, p. 94-108. ISBN : 978-3-540-24725-8 (cf. p. 49).

CHEN, Xinguang. « A Theoretical Comparison of Selected CSP Solving and Modeling Techniques ». Thèse de doct. University of Alberta, 2000 (cf. p. 40).

CHEN, Ying, Chaojun LIN, Zhixing LUO et al. « Double π-Extended Undecabenzo[7]Helicene ». en. In : *Angew. Chem. Int. Ed.* 60.14 (2021), p. 7796-7801. ISSN : 1521-3773. DOI : 10.1002/anie.202014621 (cf. p. 60, 117, 135).

CHEN, Z., C.S. WANNERE, C. CORMINBOEUF et al. « Nucleus-Independent Chemical Shifts (NICS) as an Aromaticity Criterion ». In : *Chemical Reviews* 105.10 (2005), p. 3842-3888. ISSN : 0009-2665. DOI : 10.1021/cr030088+. URL : https://doi.org/10.1021/cr030088+ (cf. p. 24, 64).

CHEUNG, Kwan Yin, Kosuke WATANABE, Yasutomo SEGAWA et al. « Synthesis of a Zigzag Carbon Nanobelt ». en. In : *Nat. Chem.* 13.3 (mars 2021), p. 255-259. ISSN : 1755-4349. DOI : 10.1038/s41557-020-00627-5 (cf. p. 60, 117, 135).

CHMEISS, Assef et Philippe JÉGOU. « Efficient Path-Consistency Propagation ». In : *International Journal of Artificial Intelligence Tools* 7 (1998), p. 121-142 (cf. p. 42).

CHOU, Chien-Pin, P CHOU, Yupeng LI et al. « An Algorithm and FORTRAN Program for Automatic Computation of the Zhang–Zhang Polynomial of Benzenoids ». In : *Match (Mulheim an der Ruhr, Germany)* 68 (jan. 2012), p. 3 (cf. p. 75).

CLAR, E. *The Aromatic Sextet*. Wiley, 1972 (cf. p. 24).

COCCHI, Caterina, Deborah PREZZI, Alice RUINI et al. « Anisotropy and Size Effects on the Optical Spectra of Polycyclic Aromatic Hydrocarbons ». In : *The Journal of Physical Chemistry A* 118.33 (2014), p. 6507-6513. DOI : 10.1021/jp503054j (cf. p. 95, 112).

COLLET, Mathieu, Arnaud GOTLIEB, Nadjib LAZAAR et al. « RobTest : A CP Approach to Generate Maximal Test Trajectories for Industrial Robots ». In : *CP 2020 - 26th International Conference on Principles and Practice of Constraint Programming*. T. 12333. Lecture Notes in Computer Science. Louvain-la-Neuve, Belgium : Springer International Publishing, sept. 2020, p. 707-723. DOI : 10.1007/978-3-030-58475-7_41. URL : https://hal-lirmm.ccsd.cnrs.fr/lirmm-03687386 (cf. p. 27).

COOK, Stephen A. « The Complexity of Theorem-Proving Procedures ». In : *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Shaker Heights, Ohio, USA : Association for Computing Machinery, 1971, 151–158. ISBN : 9781450374644. DOI : 10.1145/800157.805047. URL : https://doi.org/10.1145/800157.805047 (cf. p. 37).

COOPER, Martin et Thomas SCHIEX. « Arc consistency for soft constraints ». In : *Artificial Intelligence* 154.1 (2004), p. 199-227. ISSN : 0004-3702. DOI : https://doi.org/10.1016/j.artint.2003.09.002. URL : https://www.sciencedirect.com/science/article/pii/S0004370203001875 (cf. p. 31, 37, 38).

CRAMER, Christopher J. *Essentials of Computational Chemistry : Theories and Models*. 2nd Edition. Chichester, West Sussex, England ; Hoboken : Wiley–Blackwell, sept. 2004. ISBN : 978-0-470-09182-1 (cf. p. 66).

CYVIN, Sven J., Jon BRUNVOLL et Bjørg N. CYVIN. « The hunt for concealed non-Kekuléan polyhexes ». In : *Journal of Mathematical Chemistry* 4 (1990), p. 47-54 (cf. p. 57, 58).

DEBRUYNE, Romuald et Christian BESSIÈRE. « From Restricted Path Consistency to Max-Restricted Path Consistency ». In : *Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming (CP)*. 1997, p. 312-326 (cf. p. 42).

– « Some Practicable Filtering Techniques for the Constraint Satisfaction Problems ». In : *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*. 1997, p. 412-417 (cf. p. 42).

DECHTER, Rina. « Learning While Searching in Constraint-Satisfaction-Problems ». In : *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*. 1986, 178–183 (cf. p. 46, 47).

– « Enhancement Schemes for Constraint Processing : Backjumping, Learning, and Cutset Decomposition ». In : *Artificial Intelligence* 41 (1990), p. 273-312 (cf. p. 40, 47).

– *Constraint processing*. Morgan Kaufmann Publishers, 2003 (cf. p. 31, 41).

DECHTER, Rina. « Tractable Structures for Constraint Satisfaction Problems ». In : Handbook of Constraint Programming. Elsevier, 2006. Chap. 7, p. 209-244 (cf. p. 31, 39).

DECHTER, Rina et Itay MEIRI. « Experimental Evaluation of Preprocessing Techniques in Constraint Satisfaction Problems. » In : t. 1. Jan. 1989, p. 271-277 (cf. p. 43).

– « Experimental Evaluation of Preprocessing Algorithms for Constraint Satisfaction Problems ». In : *Artificial Intelligence* 68.2 (1994), p. 211-241 (cf. p. 43, 44).

DEWAR, M. J. S. « A Molecular Orbital Theory of Organic Chemistry. I. General Principles ». In : *Journal of the American Chemical Society* 74.13 (1952), p. 3341-3345. ISSN : 0002-7863. DOI : 10.1021/ja01133a038. URL : https://doi.org/10.1021/ja01133a038 (cf. p. 23).

DI GIOVANNANTONIO, Marco, Xuelin YAO, Kristjan EIMRE et al. « Large-Cavity Coronoids with Different Inner and Outer Edge Structures ». In : *Journal of the American Chemical Society* 142.28 (2020), p. 12046-12050. DOI : 10.1021/jacs.0c05268 (cf. p. 92).

DIAS, Jerry Ray. « Structure and Electronic Characteristics of Coronoid Polycyclic Aromatic Hydrocarbons as Potential Models of Graphite Layers with Hole Defects ». In : *The Journal of Physical Chemistry A* 112.47 (2008), p. 12281-12292. DOI : 10.1021/jp806987f (cf. p. 92).

DOMSHLAK, Carmel et Jörg HOFFMANN. « Fast Probabilistic Planning Through Weighted Model Counting ». In : *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006), AAAI, 243-252 (2006)* (jan. 2006) (cf. p. 38).

DRAINE, B. T. « Astronomical Models of PAHs and Dust ». In : *EAS Publications Series* 46 (2011), p. 29-42. DOI : 10.1051/eas/1146003 (cf. p. 22).

DRANSFELD, Alk, P.v.R. SCHLEYER et Nicolaas J. R. VAN EIKEMA HOMMES. « Nucleus-independent chemical shifts : a simple and efficient aromaticity probe. » English. In : *Journal of the American Chemical Society* 118 (1996). NICS, p. 6317-6318. ISSN : 0002-7863. DOI : 10.1021/ja960582d (cf. p. 24, 26, 65, 189).

DUMSLAFF, Tim, Yanwei GU, Giuseppe M. PATERNÒ et al. « Hexa-Peri-Benzocoronene with Two Extra K-Regions in an Ortho-Configuration ». en. In : *Chem. Sci.* 11.47 (déc. 2020), p. 12816-12821. ISSN : 2041-6539. DOI : 10.1039/D0SC04649C (cf. p. 24, 60, 61, 117, 118, 148, 176).

EATON, Philip E. et Thomas W. COLE. « Cubane ». In : *Journal of the American Chemical Society* 86.15 (1964), p. 3157-3158. DOI : 10.1021/ja01069a041 (cf. p. 95).

EHRENHAUSER, Franz S. « PAH and IUPAC Nomenclature ». In : *Polycyclic Aromatic Compounds* 35.2-4 (2015), p. 161-176. DOI : 10.1080/10406638.2014.918551. eprint : https://doi.org/10.1080/10406638.2014.918551. URL : https://doi.org/10.1080/10406638.2014.918551 (cf. p. 24).

FAGES, J.-G., X. LORCA et C. PRUD'HOMME. *Choco Solver User Guide Documentation*. https://choco-solver.readthedocs.io/en/latest/ (cf. p. 27, 47).

FARADAY, M. « On new compounds of carbon and hydrogen, and on certain other products obtained during the decomposition of oil by heat ». In : *Philos. Trans. R. Soc.* 1 (1825), p. 440-466 (cf. p. 22).

FREUDER, Eugene C. « A Sufficient Condition for Backtrack-Free Search ». In : *Journal of the ACM* 29 (1) (1982), p. 24-32 (cf. p. 43).

FREUDER, Eugene C. et Charles D. ELFE. « Neighborhood inverse consistency preprocessing ». In : *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI)*. 1996, p. 202-208 (cf. p. 31, 42).

FREUDER, Eugene C. et Richard J. WALLACE. « Partial constraint satisfaction ». In : *Artificial Intelligence* 58.1 (1992), p. 21-70. ISSN : 0004-3702. DOI : https://doi.org/10.1016/0004-3702(92)90004-H. URL : https://www.sciencedirect.com/science/article/pii/000437029290004H (cf. p. 31, 37, 38).

FRISCH, M. J., G. W. TRUCKS, H. B. SCHLEGEL et al. *Gaussian~16 Revision C.01*. Gaussian Inc. Wallingford CT. 2016 (cf. p. 66).

FROST, Daniel et Rina DECHTER. « Dead-End Driven Learning ». In : *Proceedings of he 12th National Conference on Artificial Intelligence (AAAI)*. 1994, p. 294-300 (cf. p. 47).

– « Look-Ahead Value Ordering for Constraint Satisfaction Problems ». In : *IJCAI-95* (jan. 1997) (cf. p. 47).

FUJISE, Kei, Eiji TSURUMAKI, Kan WAKAMATSU et al. « Construction of Helical Structures with Multiple Fused Anthracenes : Structures and Properties of Long Expanded Helicenes ». In : *Chemistry – A European Journal* 27.14 (mars 2021), p. 4548-4552. ISSN : 0947-6539. DOI : 10.1002/chem.202004720 (cf. p. 60, 117, 135).

GAREY, Michael R. et David S. JOHNSON. *Computer and Intractability*. Freeman, 1979 (cf. p. 37).

GASCHNIG, John. *Performance Measurement and Analysis of Certain Search Algorithms*. Rapp. tech. CMU-CS-79-124. Carnegie-Mellon University, 1979 (cf. p. 40).

GEELEN, Pieter A. « Dual Viewpoint Heuristics for Binary Constraint Satisfaction Problems ». In : *Proceedings of the European Conference on Artificial Intelligence (ECAI)*. 1992, p. 31-35 (cf. p. 44).

GENT, Ian P., Ian MIGUEL et Peter NIGHTINGALE. « Generalised arc consistency for the alldifferent constraint : An empirical survey ». In : *ARTIFICIAL INTELLIGENCE* (2008) (cf. p. 48).

GINSBERG, Matthew L. « Dynamic Backtracking ». In : *Journal of Artificial Intelligence Research* 1 (1993), p. 25-46 (cf. p. 40, 47).

GOLOMB, Solomon W. et Leonard D. BAUMERT. « Backtrack programming ». In : *Journal of the ACM* 12 (1965), 516–524 (cf. p. 44).

GOMES, Carla P., Bart SELMAN, Nuno CRATO et al. « Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems ». In : *Journal of Automated Reasoning* 24.1/2 (2000), p. 67-100 (cf. p. 46).

HABET, Djamal et Cyril TERRIOUX. « Conflict History based Search for Constraint Satisfaction Problem ». In : *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*. 2019, p. 1117-1122. DOI : 10.1145/3297280.3297389 (cf. p. 131).

– « Conflict history based heuristic for constraint satisfaction problem solving ». In : *Journal of Heuristics* 27.6 (juin 2021), p. 951-990. DOI : 10.1007/s10732-021-09475-z. URL : https://hal-amu.archives-ouvertes.fr/hal-03276535 (cf. p. 44).

HARALICK, Robert M. et Gordon L. ELLIOT. « Increasing tree search efficiency for constraint satisfaction problems ». In : *Artificial Intelligence* 14 (1980), p. 263-313 (cf. p. 43).

HARVEY, William D. « Nonsystematic backtracking search ». Thèse de doct. Stanford University, 1995 (cf. p. 46).

HELLER, S.R., A. MCNAUGHT, I. PLETNEV et al. « InChI, the IUPAC International Chemical Identifier ». In : *Journal of Cheminformatics* 7 (2015). DOI : 10.1186/s13321-015-0068-4 (cf. p. 163).

HENTENRYCK, Pascal Van et Jean-Philippe CARILLON. « Generality versus Specificity : An Experience with AI and OR Techniques ». In : *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*. AAAI'88. Saint Paul, Minnesota : AAAI Press, 1988, 660–664 (cf. p. 49).

HILLIER, Frederick S. et Gerald J. LIEBERMAN. *Introduction to Operations Research*. Seventh. New York, NY, USA : McGraw-Hill, 2001 (cf. p. 26).

HOEVE, Willem Jan van. « The alldifferent Constraint : A Survey ». In : *CoRR* cs.PL/0105015 (2001). URL : https://arxiv.org/abs/cs/0105015 (cf. p. 48).

HOOS, Holger H. et Thomas STÜTZLE. *Stochastic Local Search : Foundations and Applications*. Elsevier / Morgan Kaufmann, 2004 (cf. p. 39).

HOOS, Holger H. et Edward P. K. TSANG. « Local Search Methods ». In : Handbook of Constraint Programming. Elsevier, 2006. Chap. 5, p. 135-167 (cf. p. 39).

HWANG, Joey et David G. MITCHELL. « 2-Way vs. d-Way Branching for CSP ». In : *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP)*. 2005, p. 343-357 (cf. p. 45).

HÜCKEL, Erich. « Zusammenfassende Darstellung. Grundzüge der Theorie ungesättigter und aromatischer Verbindungen ». In : *Zeitschrift für Elektrochemie und angewandte physikalische Chemie* 43.10 (1937), p. 827-849. DOI : https://doi.org/10.1002/bbpc.19370431016 (cf. p. 22).

JÉGOU, Philippe et Cyril TERRIOUX. « Hybrid backtracking bounded by tree-decomposition of constraint networks ». In : *Artificial Intelligence* 146 (2003), p. 43-75 (cf. p. 47).

JUSSIEN, Narendra, Romuald DEBRUYNE et Patrice BOIZUMAULT. « Maintaining Arc-Consistency within Dynamic Backtracking ». In : *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP)*. 2000, p. 249-261 (cf. p. 40, 47).

KANCHERLA, Sindhu et Kåre B. JØRGENSEN. « Synthesis of Phenacene–Helicene Hybrids by Directed Remote Metalation ». In : *J. Org. Chem.* 85.17 (sept. 2020), p. 11140-11153. ISSN : 0022-3263. DOI : 10.1021/acs.joc.0c01097 (cf. p. 60, 117, 135).

KASK, Kalev, Rina DECHTER et Vibhav GOGATE. « Counting-Based Look-Ahead Schemes for Constraint Satisfaction ». In : *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*. Sous la dir. de Mark WALLACE. T. 3258. Lecture Notes in Computer Science. Springer, 2004, p. 317-331. DOI : 10.1007/978-3-540-30201-8_25. URL : https://doi.org/10.1007/978-3-540-30201-8_25 (cf. p. 38).

KASTELEYN, P. W. « Graph theory and crystal physics ». In : Academic Press, 1967, 43–110 (cf. p. 56, 75).

KASTLER, Marcel, Jochen SCHMIDT, Wojciech PISULA et al. « From Armchair to Zigzag Peripheries in Nanographenes ». In : *Journal of the American Chemical Society* 128.29 (2006), p. 9526-9534. DOI : 10.1021/ja062026h (cf. p. 95).

KATSIRELOS, George et Fahiem BACCHUS. « Unrestricted Nogood Recording in CSP Search ». In : *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP)*. 2003, p. 873-877 (cf. p. 47).

– « Generalized NoGoods in CSPs ». In : *Proceedings of 20th National Conference on Artificial Intelligence (AAAI)*. 2005, p. 390-396 (cf. p. 47).

KEERTHI, Ashok, Carlos SÁNCHEZ-SÁNCHEZ, Okan DENIZ et al. « On-Surface Synthesis of a Chiral Graphene Nanoribbon with Mixed Edge Structure ». en. In : *Chem. – Asian J.* 15.22 (2020), p. 3807-3811. ISSN : 1861-471X. DOI : 10.1002/asia.202001008 (cf. p. 60).

KEKULÉ, A. « Sur la constitution des substances aromatiques ». In : *Bull. Soc. Chim. Paris* 3 (1865), p. 98-110 (cf. p. 22, 56).

KONISHI, Akihito, Koki HORII, Daisuke SHIOMI et al. « Open-Shell and Antiaromatic Character Induced by the Highly Symmetric Geometry of the Planar Heptalene Structure : Synthesis and Characterization of a Nonalternant Isomer of Bisanthene ». In : *Journal of the American Chemical Society* (2019). DOI : 10.1021/jacs.9b04080 (cf. p. 95).

KUMAR, T. K. Satish. « An Information-Theoretic Characterization of Abstraction in Diagnosis and Hypothesis Selection ». In : *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*. Berlin, Heidelberg : Springer-Verlag, 2002, 126–139. ISBN : 3540439412 (cf. p. 38).

LAMPKIN, Bryan J., Peter B. KARADAKOV et Brett VANVELLER. « Detailed Visualization of Aromaticity Using Isotropic Magnetic Shielding ». In : *Angewandte Chemie International Edition* 59.43 (2020). _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.202008362, p. 19275-19281. ISSN : 1521-3773. DOI : https://doi.org/10.1002/anie.202008362. URL : http://onlinelibrary.wiley.com/doi/abs/10.1002/anie.202008362 (cf. p. 66).

LECOUTRE, C. *Constraint Networks : Techniques and Algorithms*. Wiley, 2009 (cf. p. 31, 32, 41, 48).

LECOUTRE, Christophe, Chavalit LIKITVIVATANAVONG, Scott G. SHANNON et al. « Maintaining Arc Consistency with Multiple Residues ». In : *Constraint Programming Letters (CPL)* 2 (2008) (cf. p. 42).

LECOUTRE, Christophe et Olivier ROUSSEL, éd. *Proceedings of the 2018 XCSP3 Competition*. 2018, p. 1-104. URL : http://arxiv.org/abs/1901.01830 (cf. p. 125).

LECOUTRE, Christophe, Lakhdar SAÏS, Sébastien TABARY et al. « Recording and Minimizing Nogoods from Restarts ». In : *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 1.3-4 (2007), p. 147-167 (cf. p. 45, 47, 49).

LIBERATORE, Paolo. « On the complexity of choosing the branching literal in DPLL ». In : *Artificial Intelligence* 116.1-2 (2000) (cf. p. 43).

LIN, C. « Efficient Method for Calculating the Resonance Energy Expression of Benzenoid Hydrocarbons Based on the Enumeration of Conjugated Circuits ». In : *J. Chem. Inf. Comput. Sci.* 40 (2000), p. 778-783 (cf. p. 24, 28, 64, 74, 138, 142, 143, 148, 189).

LIN, C. et G. FAN. « Algorithms for the Count of Linearly Independent and Minimal Conjugated Circuits in Benzenoid Hydrocarbons ». In : *J. Chem. Inf. Comput. Sci* 39 (1999), p. 782-787 (cf. p. 23, 28, 64, 71, 73, 137, 148, 189).

LIU, Junzhi et Xinliang FENG. « Synthetic Tailoring of Graphene Nanostructures with Zigzag-Edged Topologies : Progress and Perspectives ». In : *Angewandte Chemie International Edition* 59 (juill. 2020). DOI : 10.1002/anie.202008838 (cf. p. 24, 60, 117, 135, 161).

LONGUET-HIGGINS, H.C. « The Symmetry Groups of Non-Rigid Molecules ». In : *Molecular Physics* 6.5 (1963), p. 445-460. DOI : 10.1080/00268976300100501 (cf. p. 95).

LUBY, Michael, Alistair SINCLAIR et David ZUCKERMAN. « Optimal speedup of Las Vegas algorithms ». In : *Information Processing Letters* 47(4) (1993), p. 173-180 (cf. p. 46).

LUCH, A. *The Carcinogenic Effects of Polycyclic Aromatic Hydrocarbons*. London : Imperial College Press, 2005. URL : https://www.worldscientific.com/worldscibooks/10.1142/p306 (cf. p. 22).

MACKWORTH, Alan K. « Consistency in networks of relations. » In : *Artificial Intelligence* 8 (1977), 99–118 (cf. p. 41, 42).

MANN, M. et B. THIEL. « Kekulé Structures Enumeration Yields Unique SMILES ». In : *Proceedings of Workshop on Constraint Based Methods for Bioinformatics*. 2013 (cf. p. 27, 137, 142).

MANN, Martin, Guido TACK et Sebastian WILL. « Decomposition During Search for Propagation-Based Constraint Solvers ». In : *CoRR* abs/0712.2389 (2007). arXiv : 0712.2389. URL : http://arxiv.org/abs/0712.2389 (cf. p. 38).

MARQUES SILVA, João P., Inês LYNCE et Sharad MALIK. « Conflict-Driven Clause Learning SAT Solvers ». In : Handbook of Satisfiability. 2009, p. 131-153 (cf. p. 31, 37).

MARTIN, Max M., Frank HAMPEL et Norbert JUX. « A Hexabenzocoronene-Based Helical Nanographene ». en. In : *Chem. – Eur. J.* 26.45 (2020), p. 10210-10212. ISSN : 1521-3765. DOI : 10.1002/chem.202001471 (cf. p. 60, 117, 118).

MATSUOKA, Wataru, Hideto ITO, David SARLAH et al. « Diversity-oriented synthesis of nanographenes enabled by dearomative annulative π-extension ». In : *Nature Communications* 12 (2021) (cf. p. 115, 148, 176, 177, 191).

MISHRA, Shantanu, Doreen BEYER, Kristjan EIMRE et al. « Synthesis and Characterization of π-Extended Triangulene ». In : *Journal of the American Chemical Society* 141.27 (2019), p. 10621-10625. DOI : 10.1021/jacs.9b05319 (cf. p. 148, 176).

MISHRA, Shantanu, Xuelin YAO, Qiang CHEN et al. « Large Magnetic Exchange Coupling in Rhombus-Shaped Nanographenes with Zigzag Periphery ». en. In : *Nat. Chem.* (mai 2021), p. 1-6. ISSN : 1755-4349. DOI : 10.1038/s41557-021-00678-2 (cf. p. 60, 117, 118, 135).

MITCHELL, David G. « Resolution and Constraint Satisfaction ». In : *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP)*. 2003, p. 555-569 (cf. p. 45).

MODI, Pragnesh Jay et Manuela VELOSO. « Multiagent Meeting Scheduling with Rescheduling ». In : *In Distributed Constraint Reasoning, (DCR*. 2004 (cf. p. 38).

MOHR, Roger et Thomas C. HENDERSON. « Arc and Path Consistency Revisited ». In : *Artificial Intelligence* 28.2 (1986) (cf. p. 42).

MONTANARI, Ugo. « Networks of constraints : Fundamental properties and applications to picture processing ». In : *Information Sciences* 7 (1974), p. 95-132. ISSN : 0020-0255. DOI : https://doi.org/10.1016/0020-0255(74)90008-5. URL : https://www.sciencedirect.com/science/article/pii/0020025574900085 (cf. p. 32).

MORI, Tadashi. « Chiroptical Properties of Symmetric Double, Triple, and Multiple Helicenes ». en. In : *Chem. Rev.* 121.4 (fév. 2021), p. 2373-2412. ISSN : 0009-2665, 1520-6890. DOI : 10.1021/acs.chemrev.0c01017 (cf. p. 60, 117, 118).

MOSKEWICZ, Matthew W., Conor F. MADIGAN, Ying ZHAO et al. « Chaff : Engineering an Efficient SAT Solver ». In : *Proceedings of the 38th Design Automation Conference (DAC)*. 2001, p. 530-535 (cf. p. 49).

NARITA, Akimitsu, Xiao-Ye WANG, Xinliang FENG et al. « New Advances in Nanographene Chemistry ». In : *Chemical Society Reviews* 44.18 (2015), p. 6616-6643. DOI : 10.1039/C5CS00183H (cf. p. 22, 161).

NATHUSIUS, Marvin, Barbara EJLLI, Frank ROMINGER et al. « Chrysene-Based Blue Emitters ». In : *Chemistry – A European Journal* 26.66 (nov. 2020), p. 15089-15093. ISSN : 0947-6539. DOI : 10.1002/chem.202001808 (cf. p. 60, 117, 118).

NAVARRO, J. « A Story of the Electron ». In : *Cambridge University Press* (2012) (cf. p. 22).

NIU, Wenhui, Ji MA, Paniz SOLTANI et al. « A Curved Graphene Nanoribbon with Multi-Edge Structure and High Intrinsic Charge Carrier Mobility ». In : *J. Am. Chem. Soc.* 142.43 (oct. 2020), p. 18293-18298. ISSN : 0002-7863. DOI : 10.1021/jacs.0c07013 (cf. p. 60, 61, 121).

PALACIOS, Héctor, Blai BONET, Adnan DARWICHE et al. « Pruning Conformant Plans by Counting Models on Compiled D-DNNF Representations ». In : *Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling*. ICAPS'05. Monterey, California, USA : AAAI Press, 2005, 141–150. ISBN : 1577352203 (cf. p. 38).

PAULING, Linus, L. O. BROCKWAY et J. Y. BEACH. « The Dependence of Interatomic Distance on Single Bond-Double Bond Resonance1 ». In : *Journal of the American Chemical Society* 57.12 (1935), p. 2705-2709. DOI : 10.1021/ja01315a105 (cf. p. 24, 76).

PEARL, Judea. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1988. ISBN : 1558604790 (cf. p. 38).

PESANT, G. « Counting Solutions of CSPs : A Structural Approach ». In : *IJCAI*. 2005 (cf. p. 38).

PESANT, Gilles. « A Regular Language Membership Constraint for Finite Sequences of Variables ». In : *Principles and Practice of Constraint Programming – CP 2004*. Sous la dir. de Mark WALLACE. Berlin, Heidelberg : Springer Berlin Heidelberg, 2004, p. 482-495. ISBN : 978-3-540-30201-8 (cf. p. 49).

PESANT, Gilles et Alessandro ZANARINI. « Counting-Based Search : Branching Heuristics for Constraint Satisfaction Problems ». In : *Journal of Artificial Intelligence Research* 43 (jan. 2012), p. 173-210. DOI : 10.1613/jair.3463 (cf. p. 38).

PETCU, Adrian et Boi FALTINGS. « An Efficient Constraint Optimization Method for Large Multiagent Systems ». In : (jan. 2005) (cf. p. 38).

PIZZOCHERO, Michele et Efthimios KAXIRAS. « Imprinting Tunable π-Magnetism in Graphene Nanoribbons via Edge Extensions ». In : *J. Phys. Chem. Lett.* 12.4 (fév. 2021), p. 1214-1219. DOI : 10.1021/acs.jpclett.0c03677 (cf. p. 60).

PROSSER, Patrick. « Hybrid Algorithms for the constraint satisfaction problem ». In : *Computational Intelligence* 9 (1993), p. 268-299 (cf. p. 40).

– *MAC-CBJ : maintening arc consistency with conflict-directed backjumping*. 1995 (cf. p. 40).

QIU, Zijie, Akimitsu NARITA et Klaus MÜLLEN. « Carbon nanostructures by macromolecular design from branched polyphenylenes to nanographenes and graphene nanoribbons ». In : *Faraday Discussions* (2020). Publisher : The Royal Society of Chemistry. DOI : 10.1039/D0FD00023J (cf. p. 60, 117, 135).

QIU, Zijie, Akimitsu NARITA et Klaus MÜLLEN. « Spiers Memorial Lecture Carbon nanostructures by macromolecular design – from branched polyphenylenes to nanographenes and graphene nanoribbons ». In : *Faraday Discuss.* 227 (0 2021), p. 8-45. DOI : 10.1039/D0FD00023J. URL : http://dx.doi.org/10.1039/D0FD00023J (cf. p. 60, 117, 118).

RANDIĆ, M. « A graph theoretical approach to conjugation and resonance energies of hydrocarbons ». In : *Tetrahedron* 33 (15 1976), p. 1905-1920. DOI : 10.1016/0009-2614(76)80257-6 (cf. p. 23, 189).

– « Conjugated circuits and resonance energies of benzenoid hydrocarbons ». In : *Chemical Physics Letters* 38 (1 1976), p. 68-70. DOI : 10.1016/0009-2614(76)80257-6 (cf. p. 67).

– « Benzenoid Rings Resonance Energies and Local Aromaticity of Benzenoid Hydrocarbons ». In : *Journal of Computational Chemistry* 40(5) (2019), p. 753-762 (cf. p. 64, 67, 68, 143, 144).

RANDIĆ, M. et A. T. BALABAN. « Local Aromaticity and Aromatic Sextet Theory beyond Clar ». In : *Int. J. Quantum Chem.* 108(17) (2018) (cf. p. 77).

RANDIĆ, M. et X. GUO. « Recursive Method for Enumeration of Linearly Independent and Minimal Conjugated Circuits of Benzenoid Hydrocarbons ». In : *Journal of Chemical Information and Modeling* 34(2) (1994) (cf. p. 68, 69).

RANDIĆ, M., X. GUO et D. J. KLEIN. « Analytical Expressions for the Count of LM-Conjugated Circuits of Benzenoid Hydrocarbons ». In : *Int. J. Quantum Chem.* 60 (1996), p. 943-958 (cf. p. 74).

REFALO, Philippe. « Understanding and Improving the MAC Algorithm ». In : *Proceedings of the 10th International Conference on Principles and Practices of Constraint Programming (CP)*. 2004, p. 557-571 (cf. p. 44).

RICCA, Alessandra, Charles W. BAUSCHLICHER, Christiaan BOERSMA et al. « The Infrared spectroscopy of compact polycyclic aromatic hydrocarbons containing up to 384 carbons ». In : *The Astrophysical Journal* 754.1 (2012), p. 75. DOI : 10.1088/0004-637X/754/1/75 (cf. p. 60).

RICCA, Alessandra, Joseph E. ROSER, Els PEETERS et al. « Polycyclic Aromatic Hydrocarbons with Armchair Edges : Potential Emitters in Class B Sources ». In : *The Astrophysical Journal* 882.1 (2019), p. 56. DOI : 10.3847/1538-4357/ab3124 (cf. p. 60, 81).

RICHARDS, E. Thomas et Barry RICHARDS. « Nogood Learning for Constraint Satisfaction ». In : *Proceedings CP-96 Workshop on Constraint Programming Applications*. CP, 1996 (cf. p. 47).

RIEGER, R. et K. MÜLLEN. « Forever Young : Polycyclic Aromatic Hydrocarbons as Model Cases for Structural and Optical Studies ». In : *Journal of Physical Organic Chemistry* 23.4 (2010), p. 315-325. DOI : 10.1002/poc.1644 (cf. p. 22, 62).

RISPOLI, Fred J. « Counting Perfect Matchings in Hexagonal Systems Associated with Benzenoids ». In : *Mathematics Magazine* 14 (2001), p. 194-200 (cf. p. 56, 57, 75, 141, 142).

ROSSI, F., P. van BEEK et T. WALSH. *Handbook of Constraint Programming*. Elsevier, 2006 (cf. p. 26).

ROSSI, Francesca, Peter van BEEK et Toby WALSH, éd. *Handbook of Constraint Programming*. T. 2. Foundations of Artificial Intelligence. Elsevier, 2006. ISBN : 978-0-444-52726-4. URL : https://www.sciencedirect.com/science/bookseries/15746526/2 (cf. p. 31, 38, 41).

ROY, Myriam, Veronika BEREZHNAIA, Marco VILLA et al. « Stereoselective Syntheses, Structures, and Properties of Extremely Distorted Chiral Nanographenes Embedding Hextuple Helicenes ». In : *Angewandte Chemie International Edition* 59.8 (2020), p. 3264-3271 (cf. p. 95, 112).

RUIZ-MORALES, Yosadara. « The Agreement between Clar Structures and Nucleus-Independent Chemical Shift Values in Pericondensed Benzenoid Polycyclic Aromatic Hydrocarbons : An Application of the Y-Rule ». In : *Journal of Physical Chemistry A - J PHYS CHEM A* 108 (nov. 2004). DOI : 10.1021/jp040179q (cf. p. 143, 144).

RÉGIN, J.-C. « A filtering algorithm for constraints of difference in CSPs ». In : *Proceedings of AAAI*. 1994, p. 362-367 (cf. p. 48, 137).

SABIN, Daniel et Eugene C. FREUDER. « Contradicting Conventional Wisdom in Constraint Satisfaction ». In : *Proceedings of 11th European Conference on Artificial Intelligence (ECAI)*. 1994, p. 125-129 (cf. p. 45).

SANDHOLM, Tuomas. « Algorithm for optimal winner determination in combinatorial auctions ». In : *Artificial Intelligence* 135.1 (2002), p. 1-54. ISSN : 0004-3702. DOI : https://doi.org/10.1016/S0004-3702(01)00159-X. URL : https://www.sciencedirect.com/science/article/pii/S000437020100159X (cf. p. 38).

SCHIEX, Thomas et G. VERFAILLIE. « Stubbornness : a possible enhancement for backjumping and nogood recording ». In : *11. European Conference on Artificial Intelligence*. Proceedings. Toulouse, France, 1994. URL : https://hal.inrae.fr/hal-02774709 (cf. p. 47).

SCHIEX, Thomas et Gérard VERFAILLIE. « Nogood Recording for Static and Dynamic Constraint Satisfaction Problems ». In : *International Journal of Artificial Intelligence Tools* 3(2) (1994), p. 187-207 (cf. p. 47).

SIMONCINI, David, David ALLOUCHE, Simon DE GIVRY et al. « Guaranteed Discrete Energy Optimization on Large Protein Design Problems ». In : *Journal of Chemical Theory and Computation* 11.12 (2015), p. 5980-5989. DOI : 10.1021/acs.jctc.5b00594 (cf. p. 27).

STALLMAN, Richard M. et Gerald J. SUSSMAN. « Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis ». In : *Artifical Intelligence* 9(2) (1977), p. 135-196 (cf. p. 40, 47).

STERGIOU, Kostas et Toby WALSH. « Encodings of Non-Binary Constraint Satisfaction Problems ». In : *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI)*. 1999, p. 163-168 (cf. p. 46).

SYCARA, K., S.F. ROTH, N. SADEH et al. « Distributed constrained heuristic search ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 21.6 (1991), p. 1446-1461. DOI : 10.1109/21.135688 (cf. p. 38).

TAYLOR, Peter R. « Molecular Symmetry and Quantum Chemistry ». In : *Lecture Notes in Quantum Chemistry : European Summer School in Quantum Chemistry*. Sous la dir. de Björn O. ROOS. Lecture Notes in Chemistry. 1992 (cf. p. 95).

TERNANSKY, Robert J., Douglas W. BALOGH et Leo A. PAQUETTE. « Dodecahedrane ». In : *Journal of the American Chemical Society* 104.16 (1982), p. 4503-4504. DOI : 10.1021/ja00380a040 (cf. p. 95).

THOMSON, J.J. « Cathode Rays ». In : *Edinburgh Dublin Philos. Mag. J. Sci.* (1897) (cf. p. 22).

THOMSON, J.J. *The Corpuscular Theory of Matter, Charles Scribner's Sons*. 1907 (cf. p. 22).

ULLMANN, J. R. « An Algorithm for Subgraph Isomorphism ». In : *J. ACM* 23.1 (1976), 31–42. ISSN : 0004-5411. DOI : 10.1145/321921.321925. URL : https://doi.org/10.1145/321921.321925 (cf. p. 43).

URYU, Mizuho, Taito HIRAGA, Yoshito KOGA et al. « Synthesis of Polybenzoacenes : Annulative Dimerization of Phenylene Triflate by Twofold C-H Activation ». en. In : *Angew. Chem.* 132.16 (2020), p. 6613-6616. ISSN : 1521-3757. DOI : 10.1002/ange.202001211 (cf. p. 60, 117, 135).

VALIANT, L.G. « The complexity of enumeration and reliability problems ». In : *SIAM J. Comput.* 8 (1979), p. 410-421 (cf. p. 38).

VANDEL, J et S de GIVRY. « A New Local Move Operator for Reconstructing Gene Regulatory networks ». In : *CP-11 workshop on Constraint Based Methods for Bioinformatics*. Perugia, Italy, 2011, p. 67-72 (cf. p. 38).

VANDEL, J, B MANGIN, M VIGNES et al. « Extended Bayesian scores for reconstructing gene regulatory networks ». In : *ECCS-10 workshop on graphical models for reasoning on biological systems : computational challenges*. Lisbon, Portugal, 2010 (cf. p. 38).

VARET, Adrien, Nicolas PRCOVIC, Cyril TERRIOUX et al. « BenzAI : A Program to Design Benzenoids with Defined Properties Using Constraint Programming ». In : *Journal of Chemical Information and Modeling* (2022). To appear. DOI : 10.1021/acs.jcim.2c00353 (cf. p. 150, 167).

VERHAEGHE, Hélène, Christophe LECOUTRE et Pierre SCHAUS. « Extending Compact-Table to Negative and Short Tables. » In : *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2017, p. 3951-3957. URL : https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14359/14122 (cf. p. 123).

VIRICEL, Clément, Simon de GIVRY, Thomas SCHIEX et al. « Cost Function Network-based Design of Protein-Protein Interactions : predicting changes in binding affinity ». In : *Bioinformatics* 34.15 (2018), p. 2581-2589 (cf. p. 38).

WALTZ, David. « Generating Semantic Descriptions From Drawings of Scenes With Shadows ». Thèse de doct. Oct. 1972 (cf. p. 42).

WATTEZ, Hugues, Christophe LECOUTRE, Anastasia PAPARRIZOU et al. « Refining Constraint Weighting ». In : *Proceedings of the 31st IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. 2019, p. 71-77. DOI : 10.1109/ICTAI.2019.00019 (cf. p. 44, 131).

WITEK, Henryk A. et Johanna LANGNER. « Clar Covers of Overlapping Benzenoids : Case of Two Identically-Oriented Parallelograms ». In : *Symmetry* 12.10 (2020). ISSN : 2073-8994. DOI : 10.3390/sym12101599. URL : https://www.mdpi.com/2073-8994/12/10/1599 (cf. p. 75).

WOHNER, Natalie, Pui K. LAM et Klaus SATTLER. « Systematic energetics study of graphene nanoflakes : From armchair and zigzag to rough edges with pronounced protrusions and overcrowded bays ». In : *Carbon* 82 (2015), p. 523-537. DOI : 10.1016/j.carbon.2014.11.004 (cf. p. 119, 131).

WU, Christine Wei. « Modelling Chemical Reactions Using Constraint Programming and Molecular Graphs ». In : *Principles and Practice of Constraint Programming*. 2004, p. 808-808 (cf. p. 27).

WU, Jishan, Wojciech PISULA et Klaus MÜLLEN. « Graphenes as Potential Material for Electronics ». In : *Chemical Reviews* 107.3 (2007), p. 718-747. DOI : 10.1021/cr068010r (cf. p. 22).

XIA, Zeming, Sai Ho PUN, Han CHEN et al. « Synthesis of Zigzag Carbon Nanobelts through Scholl Reactions ». en. In : *Angew. Chem. Int. Ed.* 60.18 (2021), p. 10311-10318. ISSN : 1521-3773. DOI : 10.1002/anie.202100343 (cf. p. 60, 117, 135).

YANG, Xuan, Frank ROMINGER et Michael MASTALERZ. « Benzo-Fused Perylene Oligomers with up to 13 Linearly Annulated Rings ». en. In : *Angew. Chem. Int. Ed.* 60.14 (2021), p. 7941-7946. ISSN : 1521-3773. DOI : 10.1002/anie.202017062 (cf. p. 60, 117, 118).

YANO, Yuuta, Nobuhiko MITOMA, Hideto ITO et al. « A Quest for Structurally Uniform Graphene Nanoribbons : Synthesis, Properties, and Applications ». In : *The Journal of Organic Chemistry* 85.1 (2019), p. 4-33. ISSN : 0022-3263. DOI : https://doi.org/10.1021/acs.joc.9b02814. URL : https://www.sciencedirect.com/science/article/pii/S0022326321043620 (cf. p. 24).

YAO, Xuelin, Wenhao ZHENG, Silvio OSELLA et al. « Synthesis of Nonplanar Graphene Nanoribbon with Fjord Edges ». In : *J. Am. Chem. Soc.* 143.15 (avr. 2021), p. 5654-5658. ISSN : 0002-7863. DOI : 10.1021/jacs.1c01882 (cf. p. 60).

ZENG, Cheng, Bohan WANG, Huanhuan ZHANG et al. « Electrochemical Synthesis, Deposition, and Doping of Polycyclic Aromatic Hydrocarbon Films ». In : *J. Am. Chem. Soc.* 143.7 (fév. 2021), p. 2682-2687. ISSN : 0002-7863. DOI : 10.1021/jacs.0c13298 (cf. p. 60, 117, 118, 135).

ANNEXES

A. Annexe 1 - Résultats détaillés pour l'ensemble $\mathscr{B}_1$

Les figures A.1-A.8 décrivent les structures de benzénoïdes considérées dans l'ensemble $\mathscr{B}_1$. Ces dernières ont des tailles et des formes différentes et peuvent admettre ou non des symétries. De plus, nous spécifions pour chacune de ces molécules les résultats obtenus par CERCP et NICS respectivement en bleu et en rouge ou de RBOCP et de ClarCP en vert et orange. Pour des soucis de lisibilité, nous ne donnons les valeurs que d'un hexagone de chaque classe de symétrie. En effet, tous les hexagones d'une même classe de symétrie admettent la même valeur peu importe la méthode utilisée.

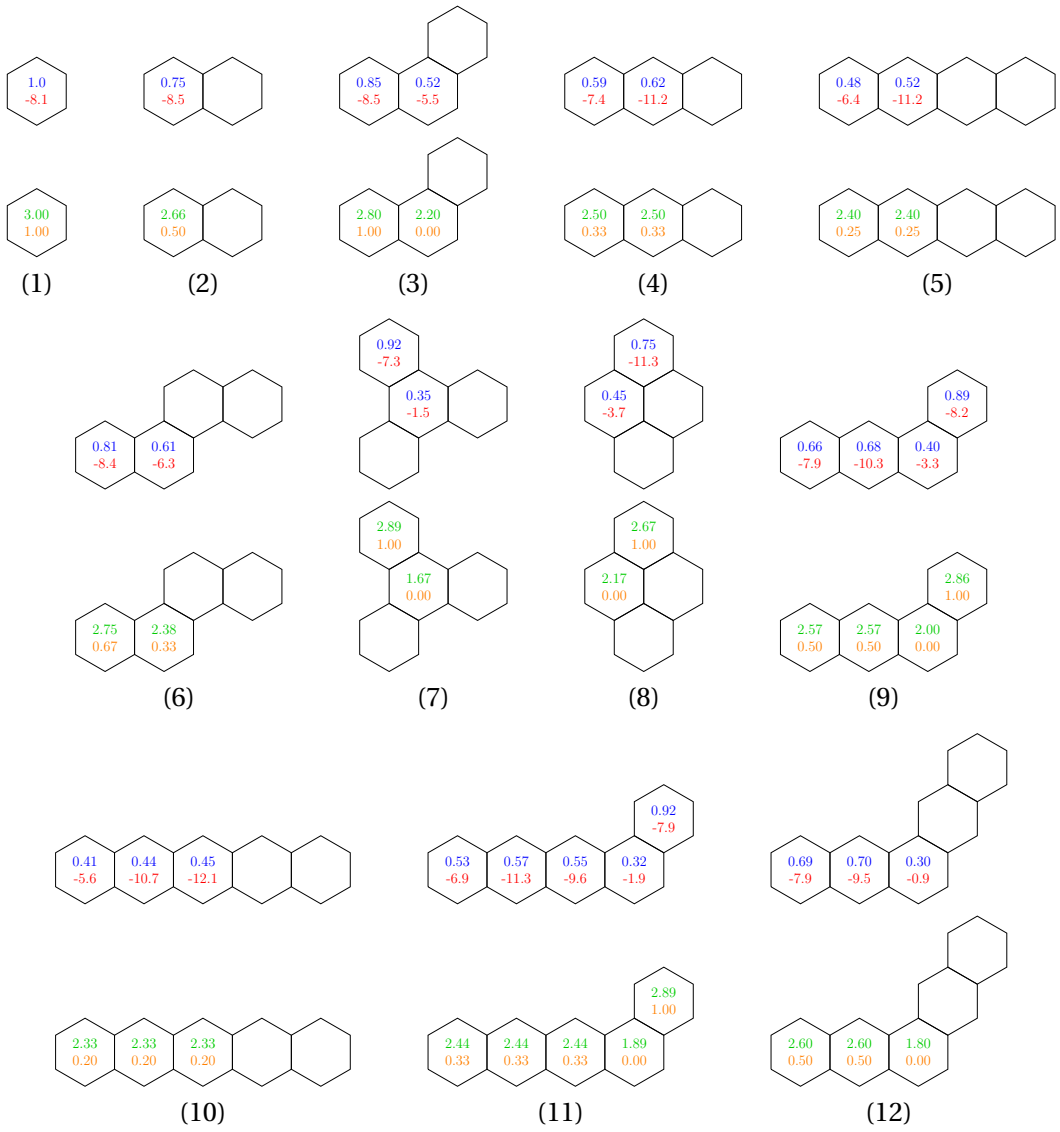


FIGURE A.1. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$.

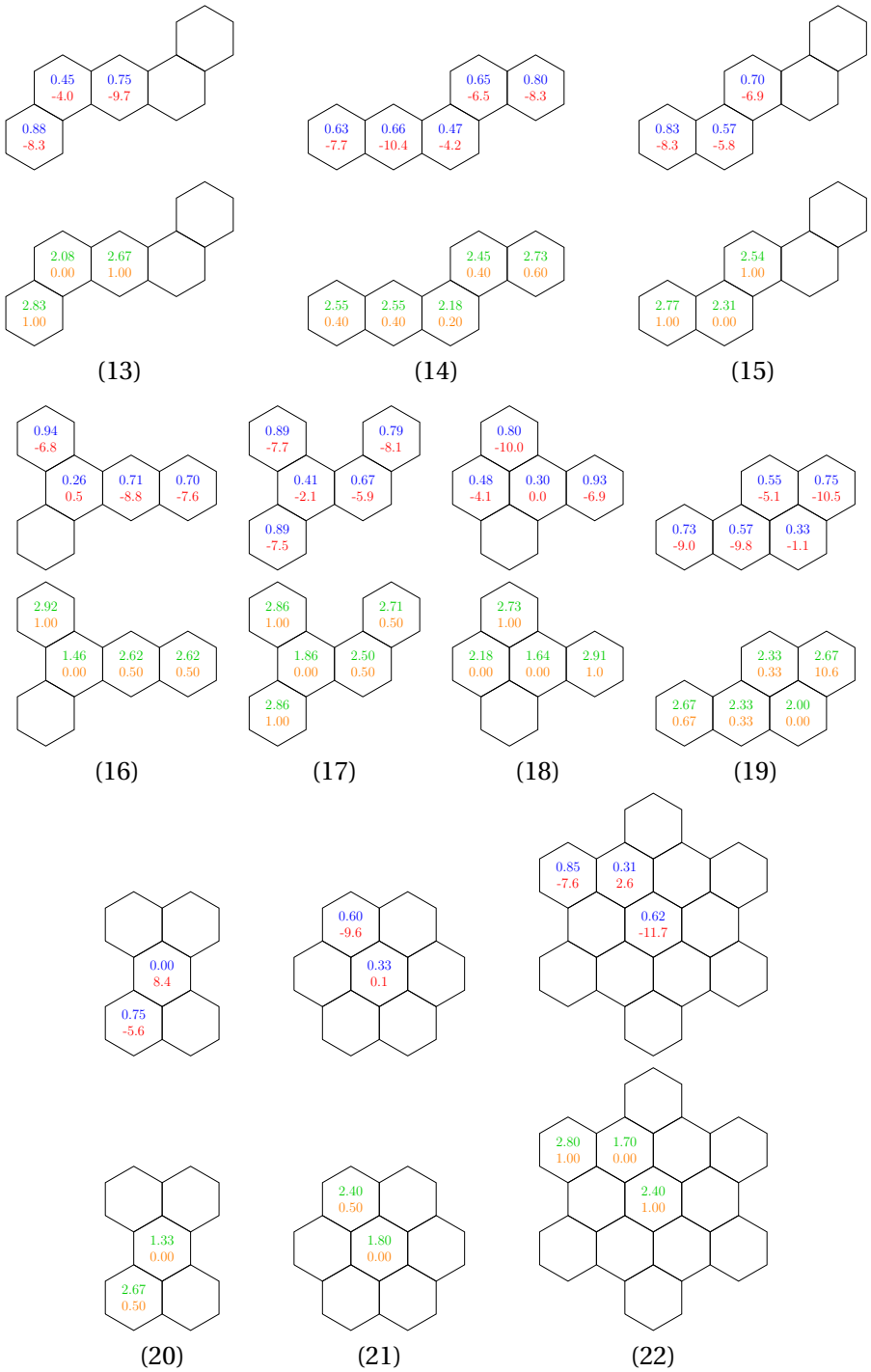


FIGURE A.2. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.1).

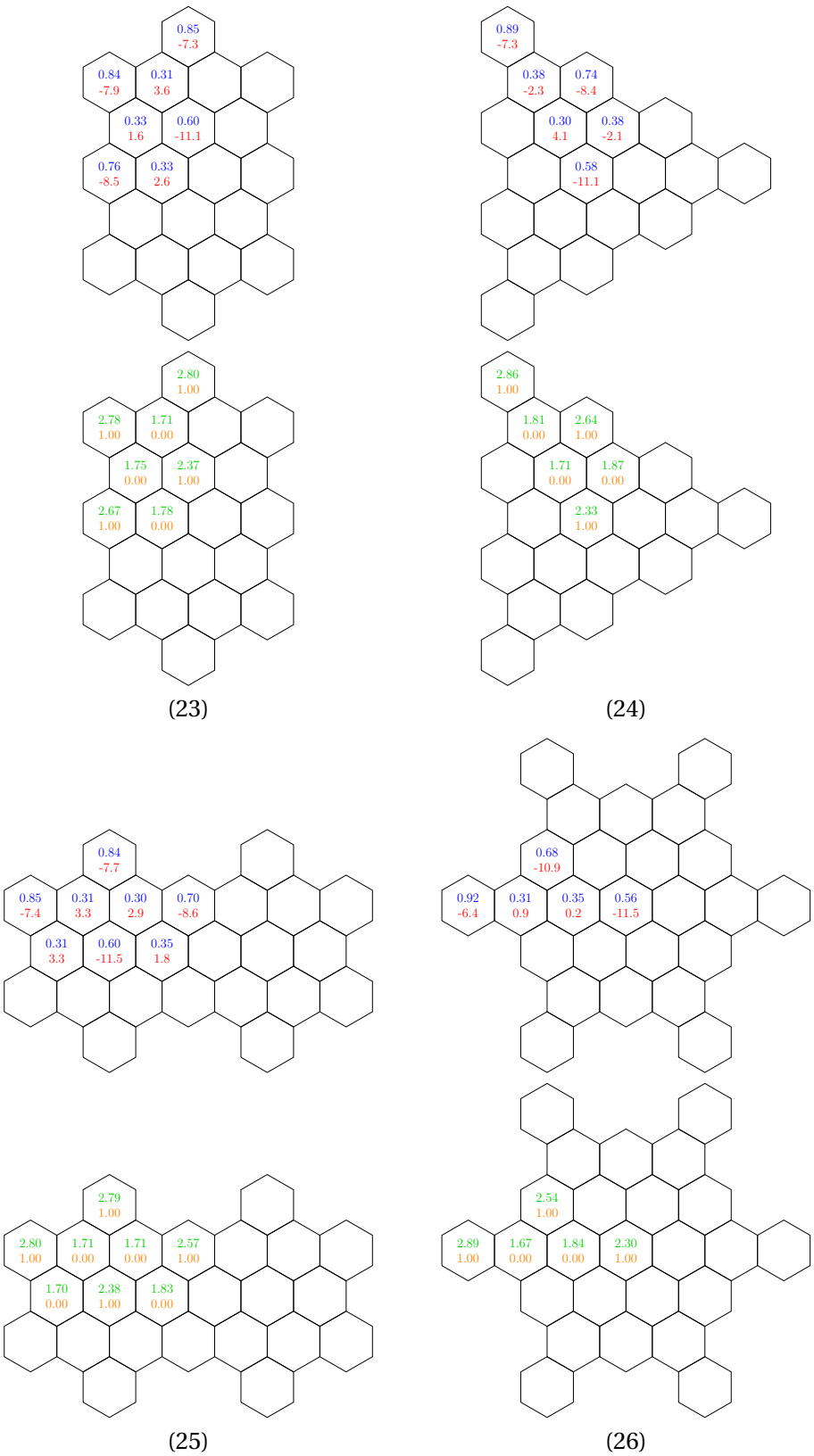


FIGURE A.3. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.2).

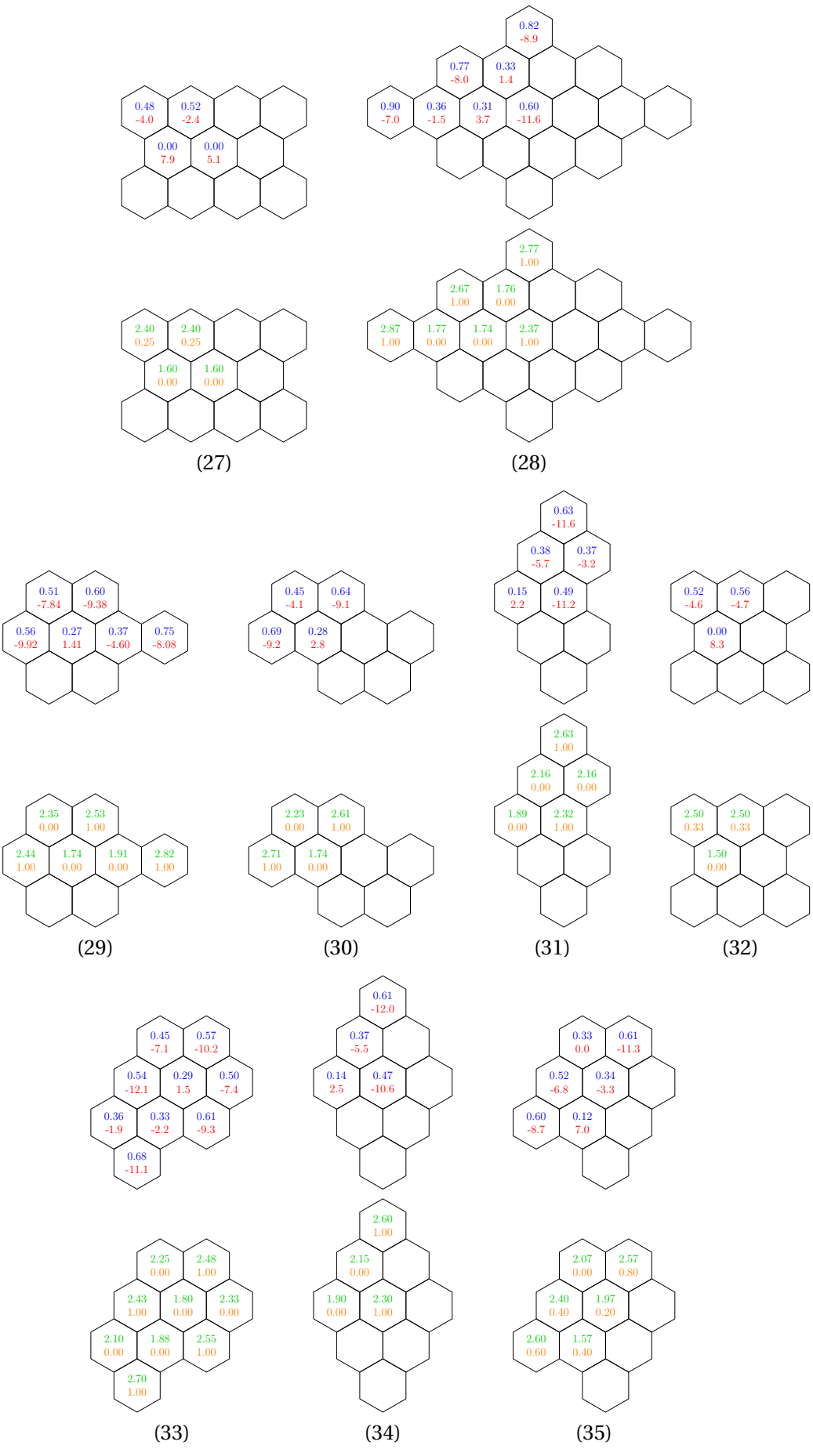


FIGURE A.4. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.3).

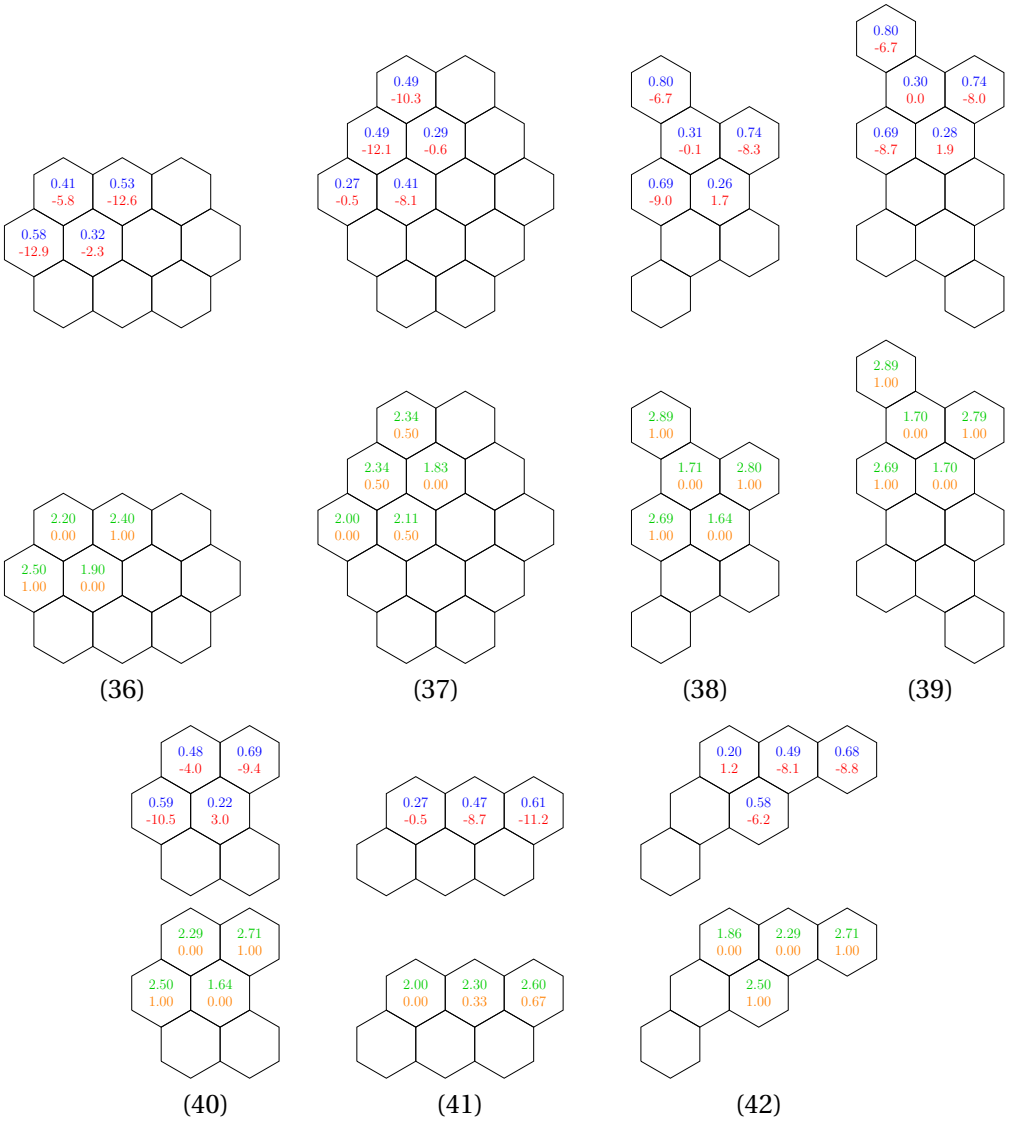
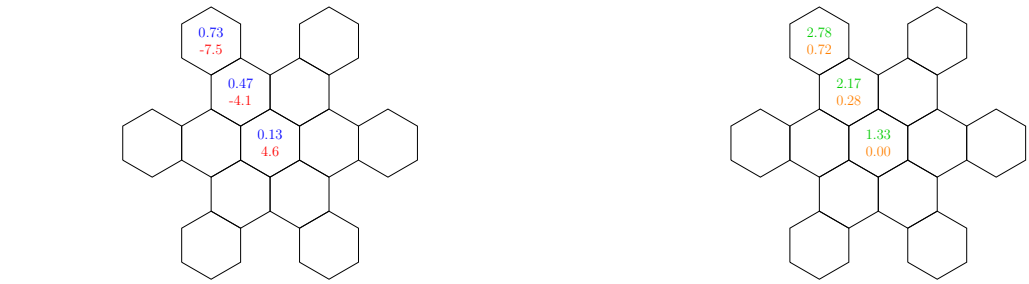
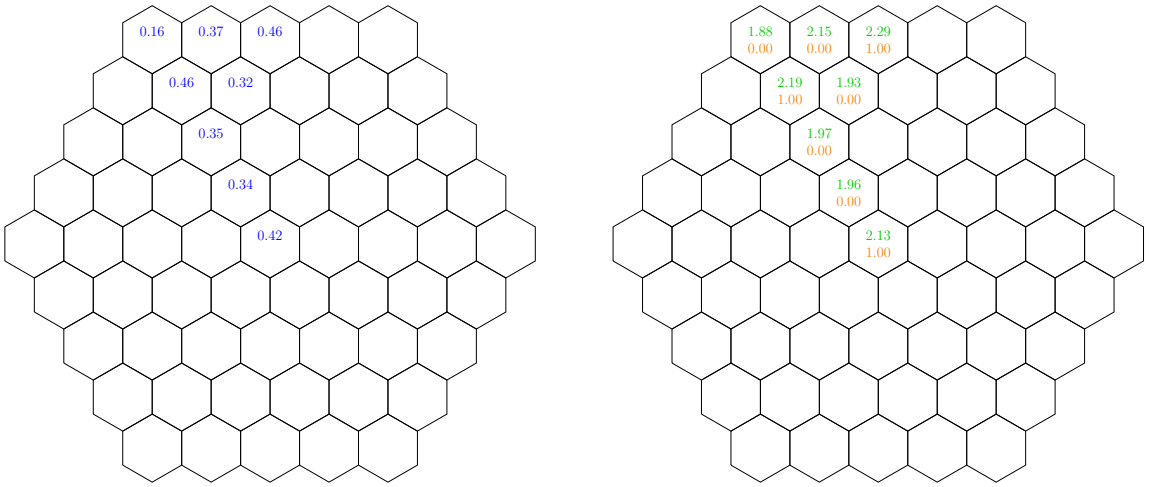


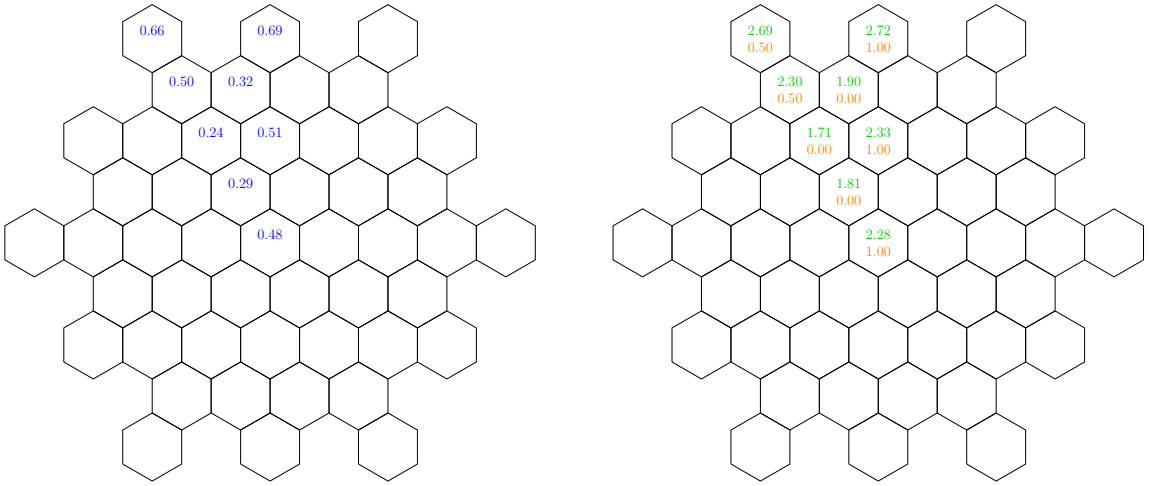
FIGURE A.5. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.4).



(43)

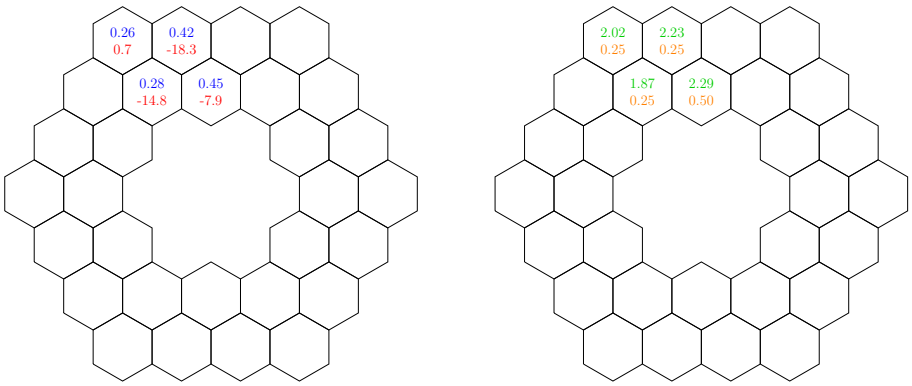


(44)

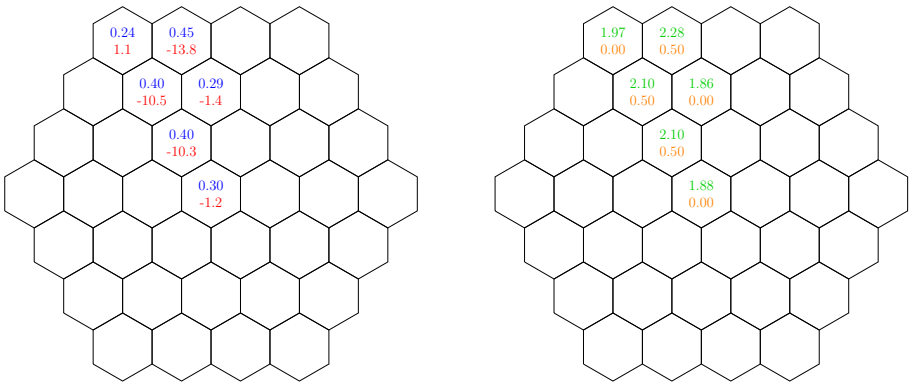


(45)

FIGURE A.6. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.5).

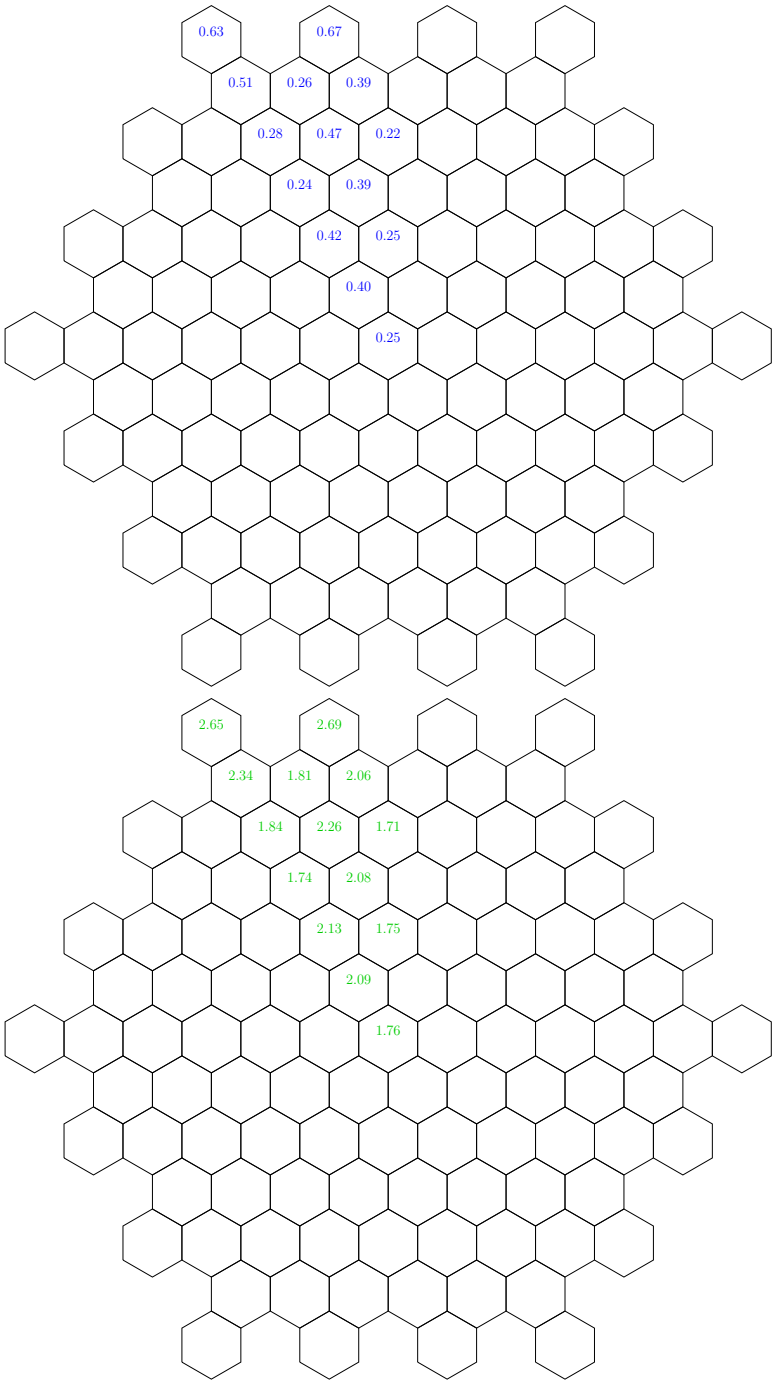


(46)



(47)

FIGURE A.7. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.6).



(48)

FIGURE A.8. – Résultats sur les 48 molécules du sous-ensemble $\mathscr{B}_1$ (suite de la figure A.7).

B. Annexe 2 - Résultats détaillés pour l'ensemble $\mathscr{B}_3$

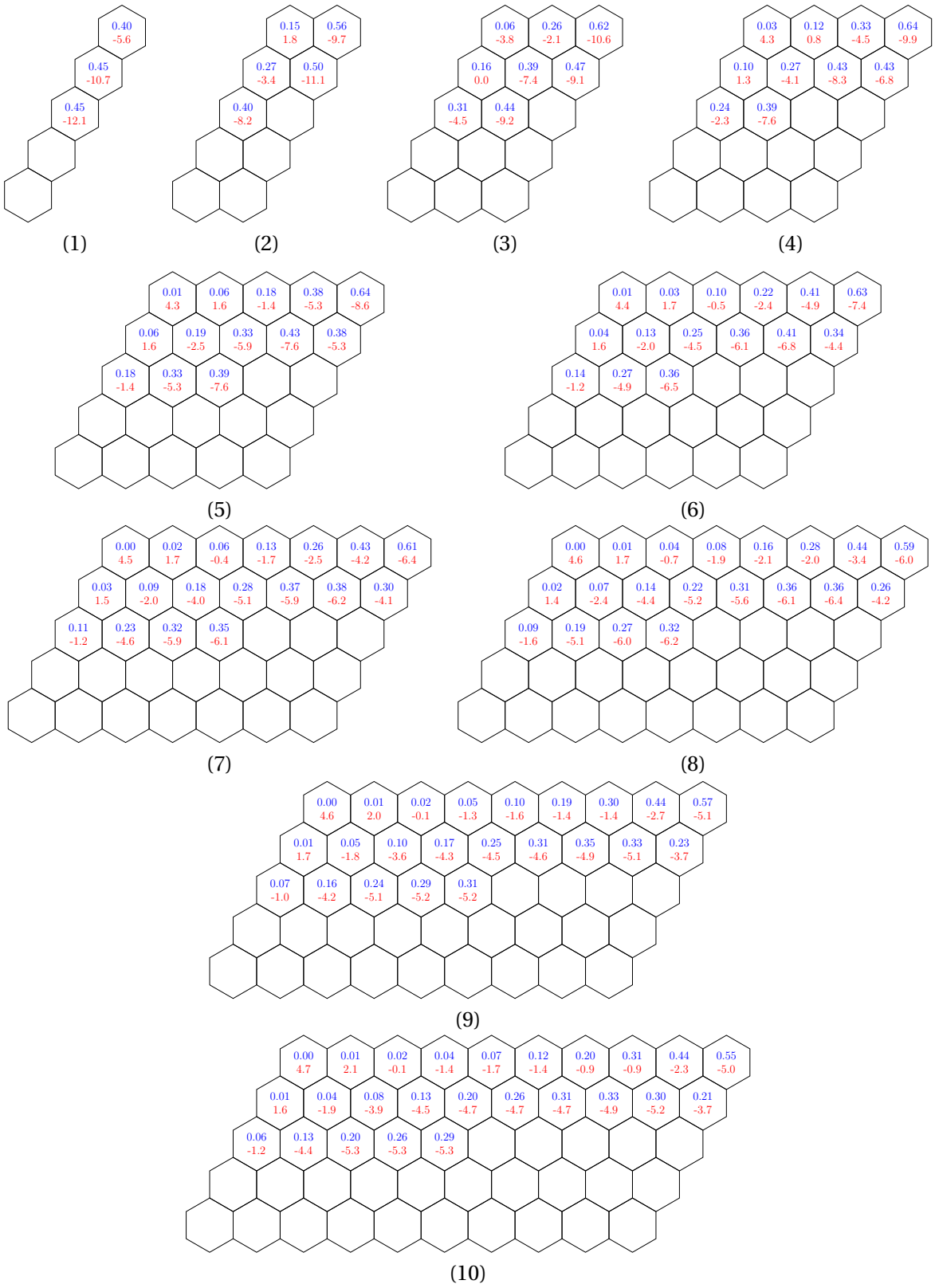


FIGURE B.1. – Énergie de résonance (en bleu) et valeurs NICS (en rouge) des molécules de l'ensemble $\mathscr{B}_3$