

# THÈSE DE DOCTORAT DE

L'UNIVERSITÉ BRETAGNE SUD  
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601  
*Mathématiques et Sciences et Technologies  
de l'Information et de la Communication*  
Spécialité : Informatique

Par

**Nicolas BLOYET**

## Caractérisation et plongement de sous-graphes colorés

Application à la construction de modèles structures à activité (QSAR)

Thèse présentée et soutenue à VANNES, le « date »  
Unité de recherche : IRISA (UMR 6074), LMBA (UMR 6205)

### Rapporteurs avant soutenance :

Sophie TISON Professeur, Université de Lille  
Dominique BARTH Professeur, Université de Versailles Saint Quentin

### Composition du Jury :

*Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du Jury ne comprend que les membres présents*

Président :	Prénom Nom	Fonction et établissement d'exercice (à préciser après la soutenance)
Examinateurs :	Luc BRUN Rumen ANDONOV Manuel ALLAIN	Professeur, ENSICAEN Professeur, Université de Rennes 1 Docteur, Société Avril
Dir. de thèse :	Pierre-François MARTEAU	Professeur, Université de Bretagne Sud
Co-dir. de thèse :	Emmanuel FRÉNOD	Professeur, Université de Bretagne Sud



# **Remerciements**

---

Je tiens à remercier ...



# Sommaire

---

<b>Introduction</b>	<b>9</b>
La chimie organique . . . . .	10
Caractérisation d'une substance chimique . . . . .	15
Modèles Structure-Activité ([Q]SAR) . . . . .	17
Contexte et Problématique . . . . .	21
Contributions . . . . .	22
<b>1 Caractérisation de sous-graphe à un isomorphisme près</b>	<b>27</b>
1.1 Problème . . . . .	27
1.1.1 Isomorphisme de graphes . . . . .	27
1.1.2 Forme canonique de graphe . . . . .	28
1.1.3 Algorithmes . . . . .	29
1.1.4 Proposition . . . . .	29
1.2 Écriture canonique d'un arbre . . . . .	30
1.2.1 Définitions . . . . .	30
1.2.2 Fonction de trace $\sigma$ . . . . .	32
1.3 Écriture d'un graphe quelconque en arbre : SCOTT . . . . .	36
1.3.1 Étapes principales . . . . .	36
1.3.2 Ordonnancement selon une racine (étape 1) . . . . .	37
1.3.3 Transformation en arbre (étape 2) . . . . .	37
1.3.4 Désignation d'une racine et encodage (étape 3) . . . . .	49
1.3.5 Grammaire . . . . .	49
1.4 Analyse de complexité temporelle . . . . .	50
1.4.1 Identification de la racine $\rho$ , $\varphi_\rho$ . . . . .	50
1.4.2 Encodage d'un l'arbre, $\varphi_t$ . . . . .	50
1.4.3 Morphisme de $\mathbb{G}$ vers $\mathbb{T}$ par $f$ , $\varphi_f$ . . . . .	50
1.4.4 Complexité globale . . . . .	52
1.4.5 Parallélisation . . . . .	53
1.5 Application . . . . .	53
1.5.1 Shrunken multipedes graphs . . . . .	54
1.5.2 Graphes moléculaires . . . . .	55
1.6 Conclusion . . . . .	56
1.6.1 Performances . . . . .	56
1.6.2 Cas d'usage type . . . . .	56
1.6.3 Perspectives . . . . .	56
<b>Glossaire</b>	<b>61</b>
<b>2 Plongements sémantiques de fragments de graphe</b>	<b>65</b>
2.1 Fragments de graphes . . . . .	65
2.1.1 Définition . . . . .	65
2.1.2 Fragmentation d'un graphe . . . . .	65
2.1.3 Vocabulaire généré . . . . .	66

## SOMMAIRE

---

2.1.4	Dimensionnalité d'un espace de fragments . . . . .	68
2.2	Analogie avec les langages naturels . . . . .	69
2.2.1	Distributions de fragments . . . . .	69
2.2.2	Rapprochement empirique avec le TALN . . . . .	70
2.2.3	Formulation de l'analogie . . . . .	72
2.3	Construction de plongements lexicaux . . . . .	72
2.3.1	Les plongements lexicaux en TALN . . . . .	72
2.3.2	Les plongements adaptés aux graphes . . . . .	74
2.4	Adaptation aux fragments moléculaires . . . . .	77
2.4.1	Plongements de fragments moléculaires . . . . .	77
2.4.2	Protocole d'apprentissage . . . . .	78
2.4.3	Requêtes par rapport à un fragment de référence . . . . .	79
2.4.4	Densification et projection des plongements . . . . .	80
2.5	Conclusion . . . . .	81
<b>3</b>	<b>Exploitation des plongements de fragments dans les Qsar</b> . . . . .	<b>85</b>
3.1	Caractérisation de graphe . . . . .	85
3.1.1	Graphes et mesures de similarité définies sur l'ensemble des graphes . . . . .	85
3.1.2	Graphe comme ensemble de fragments . . . . .	86
3.1.3	Graphe comme un réseau de noeuds . . . . .	87
3.1.4	Récapitulatif . . . . .	90
3.2	Construction d'un modèle prédictif . . . . .	90
3.2.1	Graph Attention Network (GAT) . . . . .	91
3.2.2	Protocole d'évaluation . . . . .	92
3.3	Expérimentations et résultats . . . . .	93
3.3.1	Tâche de régression . . . . .	94
3.3.2	Tâches de classification . . . . .	96
3.3.3	Conclusion . . . . .	100
<b>Conclusion</b>		<b>103</b>
Problématique		103
Contexte		103
Objectifs et enjeux		103
Réponse apportée		104
Mise en perspective		104
État de l'art		104
Apports au sujet		104
Limites des travaux		105
Avenir du sujet		106
Perspectives		106
<b>Bibliographie</b>		<b>107</b>
<b>Annexes</b>		<b>118</b>
<b>A CEMRACS</b>		<b>118</b>

# Table des Figures

---

1	Tableau périodique des éléments . . . . .	11
2	Représentation de la molécule caractéristique de la substance caféine . . . . .	12
3	Quelques ordres de liaisons covalentes rencontrées en chimie . . . . .	12
4	Visualisation des différentes échelles de la matière impliquées en chimie . . . . .	14
5	Calcul d'un identifiant SMILES à partir d'un graphe moléculaire (source : Wikimedia Commons) . . . . .	16
6	Projection d'une molécule sur un espace de fragments . . . . .	19
7	Illustration du principe de MMPA . . . . .	21
1.1	Encodage d'un arbre enraciné $t_r$ par une séquence de symboles, d'après la notation de <i>Neveu</i> (étendue aux arêtes), et par la fonction de trace proposée $\sigma^T$ (1.2.14) . . . . .	31
1.2	Proposition $\mathcal{P}_N : (K_N, \leq)$ . . . . .	34
1.3	SCOTT appliqué à un exemple simple . . . . .	36
1.4	Ordonnancement des noeuds en niveau d'après une racine . . . . .	37
1.5	Les différentes configurations aboutissant à des cycles dans un graphe . . . . .	39
1.6	Ordonnancement de productions sur un exemple simple . . . . .	43
1.7	Meilleur (gauche) et pire (droite) des cas pour l'encodage d'un arbre . . . . .	51
1.8	Graphe maximisant la taille de l'arbre généré . . . . .	52
1.9	Deux graphes "shrunken multipedes" isomorphes (72 noeuds) . . . . .	53
1.10	Canonisation de graphes "shrunken multipedes" (y-log) . . . . .	54
1.11	Temps d'exécution médian par nombre de noeuds sur des graphes moléculaires . . . . .	55
1.12	Proposition d'adaptation pour arêtes orientées . . . . .	58
2.1	Fragments de différentes tailles issus d'un noeud (en gras) . . . . .	66
2.2	Projection d'un graphe sur un espace de fragments de taille 1 . . . . .	67
2.3	Fréquence des fragments en fonction du rang, en repère log-log . . . . .	70
2.4	Occurrence par rapport au rang pour des distributions Zipf et Mandelbrot . . . . .	71
2.5	Relation entre rang d'utilisation d'un fragment/mot et sa fréquence . . . . .	71
2.6	Les deux types d'architectures décrites dans <i>Word2Vec</i> (source : [80]) . . . . .	74
2.7	Une itération du processus de coloration <i>Weisfeiler-Lehman</i> . . . . .	76
2.8	Construction d'un contexte de fragment dans le cadre du <i>Radial Skip-Gram</i> . . . . .	76
2.9	Principe de formation de <i>n-grams</i> à partir des fragments issus de SCOTT . . . . .	78
2.10	Cas de substitution d'halogènes observés sur des fragments proches d'un fragment de référence . . . . .	79
2.11	Projection dans l'espace des plongements de fragments réduits par <i>t-SNE</i> . . . . .	80
3.1	Convolution en 2D - Convolution sur Graphe . . . . .	88
3.2	Obtention d'une prédiction (classification) à partir d'un graphe. Les neurones sont sur la figure représentés par des carrés pour les différencier de noeuds de graphes . . . . .	89
3.3	Récapitulatif des méthodes de traitements sur graphes . . . . .	91
3.4	Action d'une couche <i>DropOut</i> dans un réseau de neurones profond . . . . .	91
3.5	Modèle construit et évalué à base de GAT . . . . .	92
3.6	Récapitulatif des jeux de données présents dans <i>MoleculeNet</i> . . . . .	93

## TABLE DES FIGURES

---

3.7	Influence du nombre de dimensions des plongements . . . . .	95
3.8	Scatter-plot des valeurs réelles en fonction des valeurs prédites, <i>Lipophilicity</i> . . . . .	95
3.9	Entropie catégorielle obtenue lorsque l'observation $i$ appartient à la classe 1 . . . . .	96
3.10	Influence du nombre d'epochs sur la performance du modèle . . . . .	97
3.11	Influence du nombre de dimensions des plongements sur la <i>ROC-AUC, BACE</i> . . . . .	98
3.12	Résultats comparés des architectures <i>Skip-Gram</i> et <i>CBOW, BACE</i> . . . . .	99

# Introduction

---

Ces dernières années ont vu se démocratiser tout un ensemble de solutions d'acquisition, de gestion et d'analyse des données. À l'échelle des infrastructures, l'arrivée de technologies permettant une gestion dynamique des ressources, et même une servicialisation de celles-ci, a contribué à un contrôle très largement accru des coûts liés à ces installations, et même à une réduction globale de ceux-ci. Dans le domaine logiciel, l'émergence de technologies (langages, *frameworks* et bibliothèques) sous licence libre favorise leur large adoption, et rend ainsi disponible à tous des outils modernes et efficaces.

Cet accès facilité aux infrastructures et aux traitements, conjugués à une urbanisation croissante des systèmes d'information, rend désormais possible une réelle mutualisation de l'information. Au travers d'un *pipe-line* de données, celle-ci est ainsi produite, transférée, agrégée et mise à disposition automatiquement, pouvant par la suite alimenter des traitements. Ces traitements sont de forme et de finalités très diverses, et vont de leur simple représentation à l'élaboration de modèles explicatifs ou prédictifs. Il est à noter que lorsque la volumétrie le permet, ces modèles n'utilisent plus seulement des méthodes statistiques pour leur optimisation, mais de plus en plus des algorithmes dits d'apprentissage automatique (Machine Learning) tels que les réseaux de neurones ou les forêts aléatoires auxquels il était auparavant plus difficile de faire appel dans les cas d'usage courants. De part leur souplesse d'utilisation et la diversité de cas d'usage adressés par ces méthodes, ces méthodes viennent considérablement enrichir le domaine de l'analyse de données.

Dans le secteur industriel, cet essor technologique ouvre de nouvelles possibilités quant à la façon dont sont optimisés les procédés. Individuellement, les données liées aux processus industriels sont bien souvent considérées comme des productions à part entière, et permettent souvent de fournir des métriques qualitatives, une historisation de l'activité, voire même dans les meilleurs cas de constituer une base pour des études statistiques. Mais lorsque ces données sont croisées, que ce soit avec des données issues du domaine public, ou avec d'autres données issues de processus liés à des métiers différents, celles-ci peuvent alors se révéler porteuses d'une information "enrichie" à haute valeur ajoutée. Leur collecte et leur traitement devient alors un aspect stratégique dans l'optimisation de ces procédés.

Parmi les applications possibles de ces modèles, on distingue les modèles explicatifs, ayant pour objectif de renforcer la connaissance métier d'un procédé, des modèles prédictifs qui ont davantage pour objectif d'accélérer ou de conforter une prise de décision en proposant une réponse estimée à un problème donné, sur la base d'un ensemble d'observations. Selon si la variable cible est quantitative ou qualitative, on parle respectivement de tâches de *régression* ou de *classification*.

Toutefois, la nature des dites observations statistiques est naturellement très variable suivant le domaine adressé ...

## La chimie organique

Parmi ces domaines industriels, la chimie organique est une discipline bien singulière du point de vue du traitement automatique de l'information. En effet, si elle a beaucoup évolué par le biais des disciplines conjuguées de l'informatique et des statistiques, elle reste très spécifique à traiter compte tenu de la nature même des données, I.e. les entités chimiques avec lesquelles il faut composer, plus particulièrement les *molécules*.

La chimie organique étant le domaine applicatif cible des travaux effectués au cours de cette thèse, nous introduisons très succinctement dans cette section quelques éléments nécessaires à la compréhension des enjeux abordés.

## Notions élémentaires

Il convient tout d'abord d'insister sur le caractère synthétique et général de la présentation proposée ci-après. Si la chimie est bien le domaine applicatif principal visé par les travaux de cette thèse, elle constitue pour nous un "domaine métier" que nous abordons avec le regard et les connaissances d'un non-expert. Les apports scientifiques des travaux décrits dans ce manuscrit relèvent essentiellement du domaine informatique. Il existe toutefois d'excellents ouvrages de référence tels que [1] ou les plus récents [2, 3] (deux volumes) qui permettront de dissiper les imprécisions qui pourraient apparaître dans les paragraphes qui suivent.

### Entités chimiques

**Atomes et ions** Nous raisonnons en chimie sur des entités élémentaires de la matière. L'unité de base est l'*atome*, défini comme appartenant à un *élément chimique* lui conférant ses propriétés. Cet élément est directement lié à la configuration électronique de l'atome, elle-même fonction du nombre de protons que comporte son noyau, que l'on désigne par le terme *nombre atomique*. Le noyau d'un atome est chargé positivement, tandis qu'un ensemble d'électrons chargés négativement orbitant<sup>1</sup> autour de ce noyau rendent l'ensemble électriquement neutre.

Il est à noter que si les atomes sont neutres, il existe des entités chargées positivement ou négativement dérivées de ces atomes, que l'on appelle les ions. Ces déséquilibres électroniques provenant d'une modification du nombre d'électrons et non de protons, le nombre atomique d'un ion reste inchangé par rapport à son atome de référence, ils appartiennent donc toujours au même élément chimique. On dénombre à ce jour 118 éléments chimiques, communément classifiés selon la table périodique des éléments (Fig. 1), faisant état de la périodicité de leurs propriétés physico-chimiques. On retrouve ainsi dans une même colonne des éléments ayant des configurations électroniques similaires, à l'instar de leurs comportements chimiques.

**Molécules** Ces atomes/ions peuvent être assemblés entre eux par la formation de liaisons covalentes, mettant en commun un ou plusieurs électrons. Ces assemblages chimiques forment des *molécules*, les entités chimiques constituant les *substances*. N'ayant pas de taille limite, les combinaisons d'atomes formant des substances sont virtuellement infinies, et plusieurs dizaines de millions de molécules sont à ce jour répertoriées dans des encyclopédies et bases de données telles que *PubChem* ou *ChemSpider*. Nous illustrons en Fig. 2 la molécule caractéristique de la substance connue sous le nom de "caféine", bien connue à travers le monde pour ses propriétés stimulantes.

---

1. inexact eu égard au caractère à la fois ondulatoire et corpusculaire de l'électron, mais il s'agit ici d'une interprétation pratique communément utilisée

Tableau périodique des éléments chimiques																			
Période	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
I	Hydrogène 1 H 1,007975	Béryllium 4 Be 9,0121831	Magnésium 12 Mg 24,3055	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Chalcogénies	Hélium 2 He 4,002602		
II	Lithium 3 Li 6,9395	Scandium 21 Sc 44,955906 (d)	Titanium 22 Ti 47,867 (t)	Vanadium 23 V 50,9415 (t)	Chromium 24 Cr 51,9691 (d)	Manganèse 25 Mn 54,938044	Fer 26 Fe 55,845 (d)	Cobalt 27 Co 58,933194	Nickel 28 Ni 58,9394 (d)	Cuivre 29 Cu 63,546 (s)	Zinc 30 Zn 65,38 (d)	Gallium 31 Ga 69,723 (t)	Aluminium 13 Al 26,9815385	Silicium 14 Si 28,085 (t)	Phosphore 15 P 30,97376200	Soufre 16 S 32,0675	Oxygène 8 O 15,99940	Fluor 9 F 18,999402316	Néon 10 Ne 20,1797 (d)
III A	Calcium 20 Ca 40,078 (d)	Yttrium 39 Y 88,90584	Zirconium 40 Zr 91,224 (t)	Niobium 41 Nb 92,90637	Molibdène 42 Mo 95,95 (n)	Rhenium 43 Tc [98]	Ruthénium 44 Ru 101,07 (t)	Rhodium 45 Rh 102,90550	Palladium 46 Pd 106,42 (t)	Argent 47 Ag 107,6682 (d)	Cadmium 48 Cd 112,414 (t)	Indium 49 In 114,818 (t)	Antimoine 51 Sb 118,710 (t)	Tellure 52 Te 121,760 (t)	Iode 53 I 122,60 (t)	Krypton 36 Kr 83,798 (d)			
IV A	Rubidium 37 Rb 85,4678 (d)	Strontium 38 Sr 87,62 (t)	Baryum 56 Ba 132,90545	Lanthanides 57–71 [137,32] (t)	Hafnium 72 Hf 178,49 (z)	Tantale 73 Ta 180,94788	Tungstène 74 W 183,84 (t)	Rhénum 75 Re 186,207 (t)	Osmium 76 Os 192,217 (s)	Iridium 77 Ir 196,96659	Platine 78 Pt 195,084 (g)	Dor 79 Au 196,96659	Mercure 80 Hg 204,3835	Thallium 81 Tl 207,2 (t)	Plomb 82 Pb 208,98040	Bismuth 83 Bi [209]	Polonium 84 Po [210]	Astatine 85 At [211]	Radon 86 Rn [222]
V A	Francium 87 Fr [223]	Radium 88 Ra [226]	Actinides 89–103 [227]	Actinium 89 Ac [227]	Lanthane 57 La 138,90547	Céritium 58 Ce 140,116 (t)	Praséodyme 59 Pr 140,90766	Néodyme 60 Nd 144,242 (t)	Prométhium 61 Pm [145]	Samarium 62 Sm 150,36 (t)	Europium 63 Eu 151,964 (t)	Gadolinium 64 Gd 157,25 (t)	Terbium 65 Tb 162,500 (t)	Dysprosium 66 Dy 164,93033	Holmium 67 Ho 167,259 (t)	Erbiium 68 Er 168,93422	Thulium 69 Tm 173,045	Ytterbium 70 Yb 174,9668	Lutécium 71 Lu [231]
VI A	Uranium 92 U [232,03588]	Thorium 90 Th [232,03577]	Protactinium 91 Pa [231,03588]	Neptunium 93 Np [238,02891]	Plutonium 94 Pu [244]	Americium 95 Am [243]	Curium 96 Cm [247]	Berkélium 97 Bk [247]	Californium 98 Cf [251]	Einsteinium 99 Es [252]	Fermium 100 Fm [257]	Mendelevium 101 Md [258]	Nobélium 102 No [259]	Lawrencium 103 Lr [266]					
VII A																			
	Alcalins	Alcalino-terreux	Lanthanides	Actinides	Métaux de transition	Métaux pauvres	Métalloïdes	Autres non-métaux	Halogénes	Gaz nobles	Non classés	primordial	scission d'autres éléments	synthétique					

FIGURE 1 – Tableau périodique des éléments

(source : Wikimedia Commons)

Cette molécule est constituée d'atomes de différents éléments (Carbone *C*, Azote *N*, Oxygène *O* et Hydrogène *H*) reliés par différentes liaisons covalentes (détaillées dans la section suivante).

**Composés** Enfin, ces molécules forment des larges populations, et peuvent cohabiter avec d'autres substances sans interagir chimiquement avec elles. On appelle *mélange* les populations de molécules hétérogènes, tandis qu'une substance "pure", bien que non atteignable en pratique est désignée par le terme de *composé chimique*. Les état de la matière (gazeux, solide, liquide, plasma) tels que nous percevons macroscopiquement sont une manifestation de la densité et de l'agitation qui existe au sein de ces populations. Bien qu'il existe des cas particuliers comme les cristaux ou les polymères, ou qu'il soit possible de considérer différentes échelles d'agencement (étude des protéines), l'échelle moléculaire est ainsi celle avec laquelle nous raisonnons en chimie pour étudier un mélange, car les propriétés physico-chimiques que nous mesurons sur un composé (point d'ébullition, solubilité, acidité, etc.), et que nous associons par conséquent à une substance, sont directement liées à cette échelle. Il faut également noter que c'est à cette échelle que se produisent les réactions chimiques, que nous ne détaillerons pas ici car c'est un sujet extrêmement vaste et non-essentiel pour les travaux de cette thèse.

## Liaisons covalentes

Les liaisons covalentes impliquées dans les assemblages menant aux molécules sont de natures électriques, et ont pour origine une recherche de stabilité. Les électrons sont organisés en couches nommées *orbitales atomiques*, correspondant à des fonctions d'onde, qui décrivent de manière probabiliste leur position par rapport au noyau de l'atome [4, 5]. Ces orbitales sont connues et décrites, et peuvent être regroupées en *couches* et *sous-couches* électroniques. Ces couches admettent un

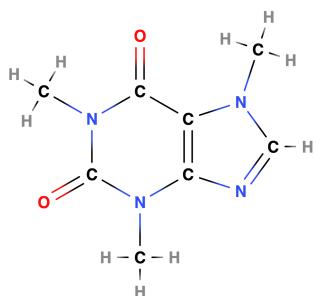


FIGURE 2 – Représentation de la molécule caractéristique de la substance caféine

nombre maximum d'électrons, correspondant au nombre d'orbitales qu'elles contiennent, et atteint sa stabilité maximale lorsque ce nombre maximal est atteint. Les atomes ayant naturellement des couches électroniques saturées sont donc les plus stables, et sont considérés chimiquement inertes : il s'agit des gaz nobles, dernière colonne du tableau périodique. Les atomes des autres éléments en revanche, vont chercher à atteindre cette stabilité en cédant ou en acquérant des électrons sur leur couche la plus éloignée du noyau, désignée sous le terme de couche de valence. C'est sur cette couche que se trouvent les électrons de valence, qui vont pouvoir être partagés avec d'autres atomes afin d'accroître leurs stabilités respectives. Cette mise en commun d'électrons se réalise par une superposition spatiale des orbitales atomiques. Il existe différentes catégories (nommées *ordres*) de liaisons covalentes, illustrées en Fig. 3

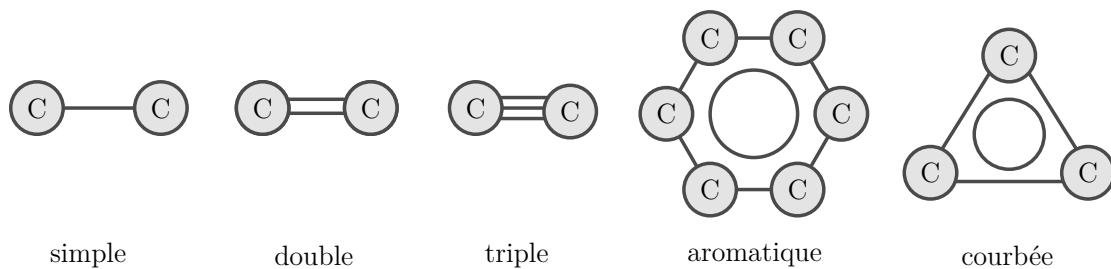


FIGURE 3 – Quelques ordres de liaisons covalentes rencontrées en chimie

Le nombre d'électrons présents sur cette couche externe étant parfaitement défini par le nombre atomique, un élément chimique ne peut donc former qu'un nombre limité de liaisons covalentes. Ainsi, un atome de Carbone est dit tétravalent, car apte à former quatre liaisons, l'Azote est trivalent, l'Oxygène bivalent, la famille des halogènes sont tous monovalents, etc. La valence des atomes suit donc logiquement les colonnes du tableau périodique.

### Exemple d'une tasse de café

Nous illustrons en Fig. 4 les différentes échelles de matière évoquées avec un exemple très simple. Prenons une tasse de café, il s'agit d'une solution dans laquelle évoluent plusieurs composés chimiques, tels que l'eau, des arômes, sucres, et également la caféine auparavant citée. Ces composés pris individuellement sont des substances pures, et sont caractérisées par une molécule unique, comme la molécule de caféine par exemple. Ces substances pures ont des propriétés physico-

chimiques propres, que l'on associe à la molécule elle-même. Enfin, nous représentons à l'échelle atomique quelques liaisons covalentes.

### **Conformation spatiale**

Bien que l'on représente communément les molécules fixes et dans le plan par leur représentation topologique illustrée en Fig. 2, elles sont en réalité agencées dans l'espace, et en mouvement permanent. Il existe en effet plusieurs catégories de liaisons covalentes, dont la nature et le nombre impliqué vont fortement influer sur la géométrie, et donc la *conformation spatiale* de la molécule. Nous listons en Fig. 3 les liaisons les plus couramment rencontrées. Les liaisons simples sont les plus longues, et permettent un axe de rotation, tandis que les liaisons double et triple sont plus courtes, mais plus rigides et ne permettent pas de rotation. Elles affectent en revanche fortement l'angle des autres liaisons. Les liaisons aromatiques impliquent des cycles de cinq à six atomes dans lesquels des électrons sont partagés par plus de deux atomes. Enfin, il existe des liaisons plus exotiques et beaucoup moins stables telles que la liaison courbée, dite "banane".

Au delà des modalités de ces différentes modalités de liaison, il demeure des degrés de liberté lorsqu'il s'agit d'organiser ces molécules dans l'espace : pour une formule topologique (planaire) donnée, il existe plusieurs conformations stables offertes par les angles de liaison. Ces différentes configurations sont désignées par le terme d'*isomères*, et sont identifiés en qualifiant les liaisons à l'origine de cette ambiguïté. Les liaisons covalentes peuvent donc être doublement qualifiées.

### **Activité physico-chimique**

L'influence de la conformation spatiale d'une molécule sur ses propriétés physico-chimiques est primordiale. Ces propriétés, qui peuvent être quantitatives (point d'ébullition, de fusion, densité, solubilité, constante de dissociation  $pK_a$ ) ou qualitatives (couleur, réactivité avec un substrat donné, odeur, toxicité à seuil donné) et que l'on désigne de manière générale par le terme *activité*, témoignent du comportement chimique des molécules. Ces propriétés sont directement fonctions de caractéristiques physiques telles que la polarité, la surface de contact, l'affinité électronique, qui elles-même sont héritées de la géométrie et de la cinétique des molécules.

L'activité physico-chimique d'une substance est donc fonction de sa structure.

## **La chimie dans l'industrie**

Ces quelques rappels de chimie étant posés, intéressons nous à présent à la place de cette discipline dans l'industrie. Étant historiquement très liée aux processus de production, elle est présente dans de nombreux domaines tiers, mais l'industrie chimique en elle-même comporte également de nombreuses branches, selon la destination et l'origine des composés manipulés. Citons par exemple la chimie pharmaceutique (composés biologiquement actifs), la phytochimie (végétaux), l'agrochimie (biocides) ou encore l'oléochimie (huiles et acides gras).

Toutes ces branches ont en commun la recherche et la synthèse de composés chimiques les plus performants possibles pour une tâche donnée. En cosmétique par exemple, on peut attendre d'un composé incorporé à une crème de jour qu'il ait une odeur "agréable", ou d'un savon qu'il démontre d'une action nettoyante efficace. Une part importante de l'activité de Recherche et Développement menée dans ces domaines consiste à la fois en l'élaboration de nouveaux composés ayant des performances accrues sur les propriétés attendues, mais également de capitaliser sur ces recherches en cherchant à identifier quels éléments structurels sont les plus pertinents dans l'obtention de ces performances applicatives. En effet, malgré une connaissance métier forte, il arrive bien souvent que ces liens ne soient pas parfaitement résolus du point de vue théorique, et il reste donc des

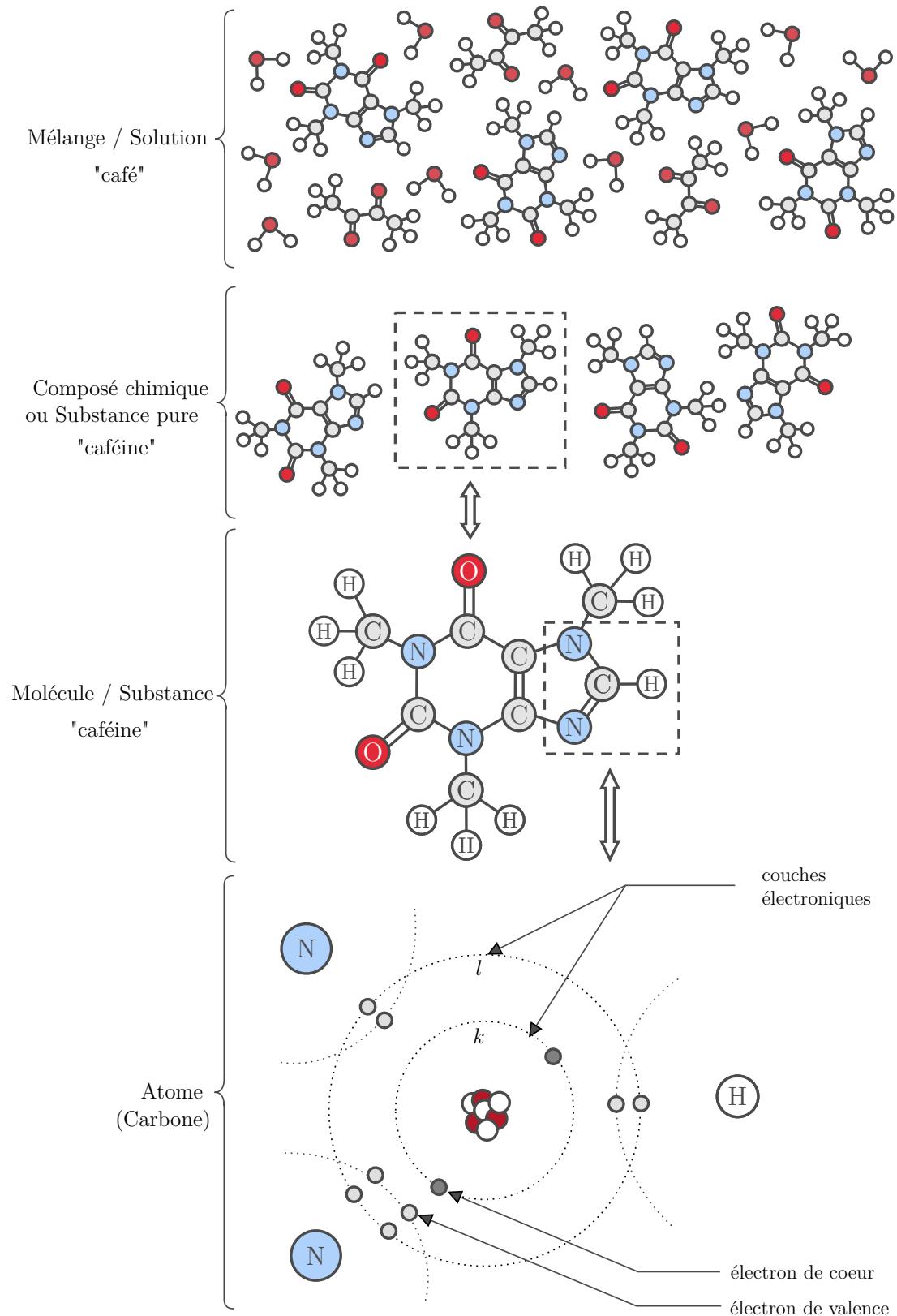


FIGURE 4 – Visualisation des différentes échelles de la matière impliquées en chimie

connaissances à acquérir. Ceci est d'autant plus vrai lorsque les propriétés cibles sont de plus en plus "abstraites" ou qualitatives et non reliées spécifiquement à une grandeur physique, comme par exemple l'odeur, le toucher, ou autres propriétés sensorielles.

Faute de savoir exactement quels sont les moyens d'action les plus efficaces dans l'obtention d'une propriété cible, la recherche d'un composé chimique cible est réalisé souvent par criblage (*screening*), occasionnant un grand nombre d'essais expérimentaux avec ajustement paramètre par paramètre. Cette manière de procéder engendre des coûts très importants liés à ces expériences, car ils nécessitent des moyens humains et matériels conséquents. Ce secteur bénéficie donc beaucoup des modèles physiques ou statistiques pouvant fournir une estimation d'activité pour une substance donnée, afin de réduire l'ampleur de ce criblage. Il faut par ailleurs insister sur le degré de précision exigé dans les mesures, les réglementations de pureté des composés étant très strictes, car liés à des risques de première importance, notamment sanitaires.

## Caractérisation d'une substance chimique

Nous venons de rappeler brièvement la nature des différentes entités chimiques, et les principaux mécanismes régissant leurs interactions. Nous allons à présent nous rapprocher du domaine du traitement automatique de l'information qui nous intéresse en nous demandant comment caractériser une substance chimique, et quelles sont les structures de données les plus adaptées pour les représenter.

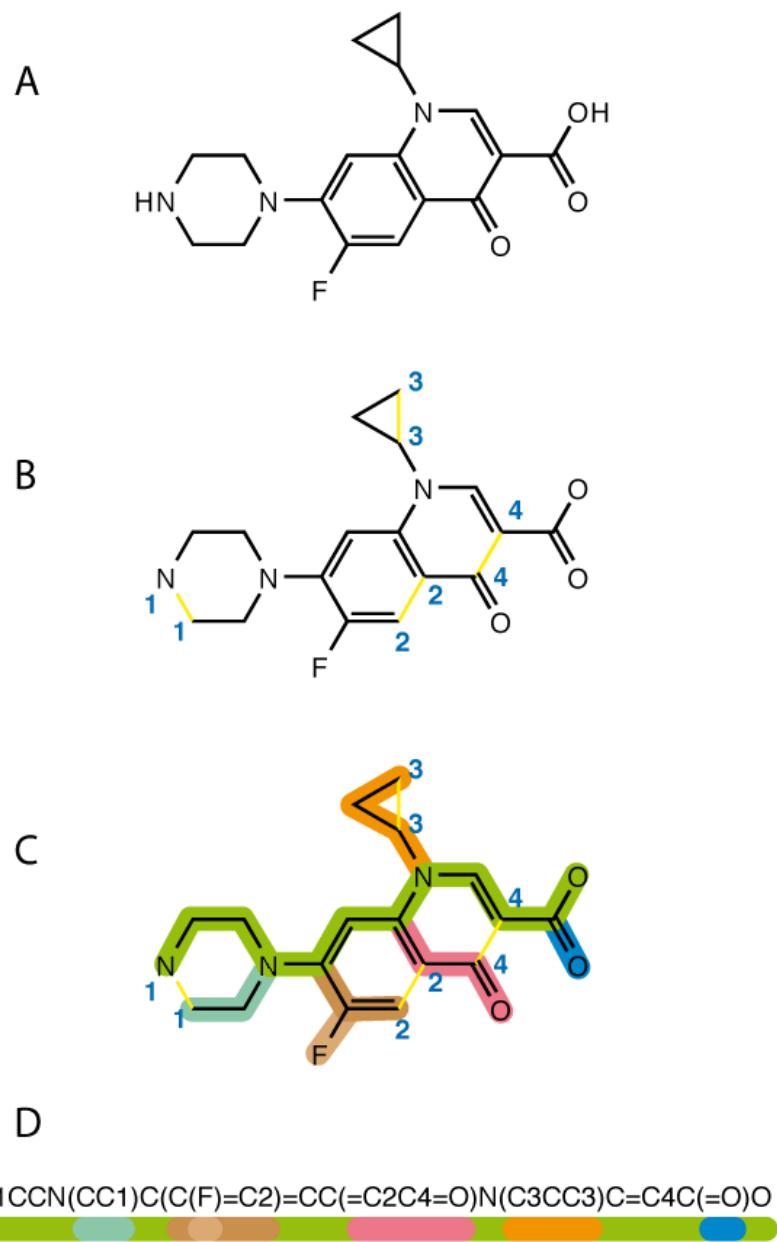
## Graphes moléculaires

La manière la plus naturelle de représenter une molécule est d'utiliser la notation topologique que nous avons découvert en Fig. 2. Bien que l'on plonge dans le plan une molécule spatiale, il n'existe aucune ambiguïté sur la manière dont sont connectés les atomes entre eux, et il est même possible de préciser la géométrie spatiale des liaisons (*stéréochimie*) en annotant de manière plus précise les liaisons. Cette notation est donc suffisante pour caractériser de manière formelle et complète une substance chimique.

Cette formule topologique peut être associée à un *graphe*, objet mathématique composé d'un ensemble de *nœuds* (que l'on rencontre également sous le terme de *vertices*), dont certaines paires sont connectées par des *arêtes* (ou *arcs*). Les éléments de ces ensembles respectifs peuvent porter une coloration (ou label), c'est-à-dire une annotation porteuse d'information. Dans le cas présent, les atomes sont associés logiquement aux noeuds, colorés selon leur élément chimique, et les liaisons covalentes correspondent aux arêtes, colorées selon leur modalité de liaison, c'est-à-dire leur ordre, voire leur géométrie.

Les graphes ont l'avantage d'être suffisants pour représenter une molécule, mais ont l'inconvénient de ne pas être eux-mêmes facilement encodables, en tout cas pas de manière unique. Ils sont donc difficilement utilisables en tant qu'identifiants. Cependant, nous avons pu constater que la valence des atomes, et donc le nombre de liaisons dans lequel ceux-ci sont impliqués, est parfaitement défini par leur emplacement dans le tableau périodique. Sur un graphe moléculaire, le degré d'un noeud est donc contraint par son label.

Cette contrainte restreint fortement les degrés de liberté lors de la construction du graphe. Lorsque cette contrainte est associée à un ensemble d'invariants chimiques permettant d'établir un ordonnancement à la fois des atomes et des arêtes, il est alors possible de décrire ces graphes de manière unique au moyen d'une chaîne de caractère. C'est l'objectif des algorithmes de calcul d'identifiant SMILES [6], illustré en Fig. 5, ou le plus récent "International Chemical Identifier" (*InChI*) [7].



---

FIGURE 5 – Calcul d'un identifiant SMILES à partir d'un graphe moléculaire  
(source : Wikimedia Commons)

Bien que l'utilisation de ces algorithmes de linéarisation de graphes soit restreinte au domaine unique de la chimie du fait des invariants utilisés, les identifiants produits sont très pratiques, et constituent actuellement l'identifiant formel standard d'une substance, recommandé par l'Union internationale de chimie pure et appliquée (IUPAC) consortium ayant autorité en la matière. On retrouve ainsi cet identifiant dans toutes les bases de données de molécules existantes.

## Représentations vectorielles

Les structures de données les plus répandues dans le traitement automatique de l'information et les statistiques ne sont toutefois pas les graphes, mais plutôt les vecteurs, embarquant sur un nombre fini de dimensions un ensemble de valeurs scalaires, qui peuvent aussi bien être qualitatives que quantitatives. Il est possible de dériver un graphe moléculaire en un ensemble de scalaires (nombre d'atomes, nombre de carbones, nombre de doubles/triples liaisons, etc.), mais d'une part cela rend l'identification de la substance beaucoup plus complexe, et d'autre part celle-ci ne saurait être exacte : s'il est bien possible de projeter sur un vecteur de dimension suffisante arbitraire un élément individuel (noeud ou arête), il est en revanche impossible d'y projeter un ensemble (de cardinalité indéterminée) sans perte d'information. Une représentation vectorielle s'accompagne donc nécessairement d'une perte d'information par rapport à une représentation sous forme de graphe.

Malgré cette perte d'information, les vecteurs restent le format le plus courant pour stocker et traiter de l'information. Les méthodes statistiques ou d'apprentissage automatique les plus courantes, qui sont par ailleurs souvent les plus efficaces, traitent des données encodées de façon vectorielle. C'est en cherchant à dépasser cette restriction de composer uniquement avec des vecteurs pour réaliser des analyses que nous allons à présent nous intéresser aux modèles dits structure-activité.

## Modèles Structure-Activité ([Q]SAR)

Dans le cas où l'on aborde ces tâches de prédiction/estimation de propriétés physico-chimiques d'un point de vue statistique, le succès réside bien souvent dans la manière dont on décrit les observations, en l'occurrence des espèces chimiques. Comme nous venons de le voir, celles-ci s'apparentent bien plus à un réseau d'atomes qu'à une information représentée sous forme vectorielle composée d'un nombre fini de descripteurs. Cette information de type structurelle renferme ainsi une grande partie de l'information caractéristique de cette molécule : elle définit la manière dont celle-ci va se conformer dans l'espace, ce qui conditionnera considérablement ses propriétés macroscopiques où sa réactivité potentielle en présence d'une autre espèce.

La relation qui lie cette information structurelle à une propriété macroscopique (physico-chimique) est spécifiquement rencontré sous le terme de *relation structure à activité* ([Quantitative] Structure-Activity Relationship, ou plus simplement QSAR dans la littérature). Nous utiliserons par la suite l'abréviation SAR, car les modèles construits ne sont pas nécessairement quantitatifs (régression), mais peuvent tout aussi bien qualitatifs (classification).

## Objectifs

On distingue deux cas d'usage des modèles SAR : le premier est purement applicatif, c'est-à-dire que l'on cherche à obtenir un modèle précis permettant d'estimer le comportement macroscopique d'une molécule vis-à-vis d'une propriété donnée. Ce cas d'usage constitue donc une forme d'al-

ternative aux méthodes de simulation physique<sup>2</sup>, typiquement l'objet des méthodes *ab initio* ou de *Monte Carlo*, pour lesquelles il faut composer avec une complexité algorithmique bien souvent prohibitive car très onéreuse. Par exemple, une implémentation générale de la méthode *Hartree-Fock* [8] présente une complexité temporelle en  $O(n^3)$  dans le meilleur des cas,  $O(n^7)$  dans le cas le plus complexe. Par ailleurs, s'il existe des méthodes d'approximation beaucoup plus efficaces pour certaines propriétés physico-chimiques telles que le *LogP* [9] ou les paramètres de solubilité [10], pour certaines propriétés il n'existe pas de méthode de simulation connue.

Le second cas d'usage est la restitution d'information. Il arrive que certaines propriétés soient assez mal comprises, et que l'on ne sache pas quels sont les éléments structurels les plus déterminants dans l'obtention d'une propriété cible. Les connaissances nécessaires ne peuvent être obtenues autrement que par une grande batterie d'essais, durant lesquels on fait varier chaque élément suspecté d'avoir une influence. Pouvoir interpréter un modèle SAR permet justement d'identifier quelle serait la relation liant les éléments structurels aux propriétés macroscopiques de la molécule. Dans ce cas, il est nécessaire de s'appuyer sur des prédicteurs ayant un certain sens chimique.

Il est à noter qu'historiquement, le terme [Q]SAR est lié au domaine de la chemio-informatique, mais on pourrait tout à fait imaginer d'autres domaines d'application où les observations statistiques s'apparenteraient à des structures de type réseaux (réseaux sociaux, circuits électroniques, réseaux informatiques, etc.). Les domaines applicatifs de ces modèles sont donc amenés à se multiplier.

## Jeux de données

Dans un contexte où les mesures expérimentales sont coûteuses et longues à obtenir, les données labellisées (autrement dit les observations) sont en conséquence rares. C'est bien le cas de la chimie qui fait face à une situation paradoxale : en effet, il existe une quantité extraordinaire de molécules référencées dans des bases de données publiques, pour autant il est compliqué de constituer un jeu de données à la fois large et portant sur une propriété pertinente d'un point de vue industriel. La base de données *PubChem* [11], initiative de la bibliothèque nationale de médecine des USA (U.S. National Library of Medicine), revendique ainsi 236 millions de substances référencées, tandis que *ChemSpider* [12], son homologue britannique, agrège 270 sources de données pour proposer des informations fiables et traçables sur 76 millions de structures. Ces deux bases de données seules constituent une formidable source d'information, malheureusement elles ne peuvent pas facilement être mises à profit pour l'élaboration de modèles. En effet, si les substances sont bien référencées, il est extrêmement rare que les propriétés physico-chimiques réclamant des mesures complexes soient indiquées, et on ne trouve ainsi dans le domaine public que des propriétés issues de simulations.

Par ailleurs, au delà même des réticences que pourraient avoir les industriels en mettant à disposition du domaine public des données qui ont une valeur industrielle, il serait impossible d'obtenir un jeu de données très volumineux concernant une seule propriété. Les propriétés qu'il est intéressant d'estimer sont bien souvent liées à une tâche bien précise, et le nombre de substances pour lesquelles cette mesure est pertinente est finalement très réduit (de l'ordre de la centaine ou du millier). Il serait impensable de réaliser des mesures pour se constituer un patrimoine d'observations sur des substances qui ne seraient jamais utilisées pour cette tâche. Si l'on s'éloigne des propriétés les plus pertinentes à prédire pour des tâches précises, il existe toutefois des jeux de données portant sur des propriétés qui ne sont pas triviales, mais assez générales pour avoir fait l'objet de mesures sur quelques milliers de substances. On retrouve ces jeux de données dans la littérature en tant que points de comparaison pour les modèles proposés. [13, 14, 15, 16]

---

2. par exemple, par un ensemble d'équations aux dérivées partielles (probabilistes ou déterministes)

Nous pouvons résumer la situation comme suit : toutes (ou presque toutes) les substances chimiquement valides et ayant été manipulées en laboratoire sont référencées, mais leurs propriétés non triviales sont beaucoup plus rarement disponibles publiquement. Il existe malgré tout des jeux de données constitués pour les *benchmarks* de modèles SAR, mais ces propriétés restent assez peu pertinentes dans le milieu industriel. De manière générale, plus on se rapproche d'un objectif industriel, plus les jeux de données associés à cette tâche sont restreints en nombre d'observations. Il s'agit donc d'un domaine où nous aurions tout intérêt à pouvoir transférer une information générale acquise sur des corpus larges, pouvant aider à la représentation de structures, vers des jeux de données beaucoup plus restreints et compliqués à traiter seuls.

## État de l'art

Nous exposons dans cette section une revue des méthodes actuelles pour l'élaboration de tels modèles. À l'instar des différentes façons de représenter une molécule, on distingue deux orientations principales : les approches basées sur une vectorisation des substances chimiques alimentant des modèles plus classiques, ou les approches traitant directement l'information sous forme de graphe. La plus grande partie de ces méthodes est évaluée sur un jeu de données commun dans [13]. Il est important de noter que s'il existe bien une bibliographie à ce sujet, ces modèles sont encore assez peu répandus dans l'industrie. En cause, une émergence des données encore récente dans un domaine qui est étroitement lié à la physique, et donc culturellement plus favorable aux simulations qu'aux études statistiques.

## Approches orientées vecteurs

Parmi les méthodes basées sur une vectorisation des graphes, nous excluons du périmètre celles consistant en une énumération de caractéristiques de la structure de base, car nous considérons que nous ne sommes plus en présence de modèles structure-activité dans ce cas, mais de modèles statistiques classiques, d'autant que ces modèles incorporent également beaucoup d'invariants chimiques extérieurs à la connaissance apportée par le seul graphe.

Il existe cependant des SAR ayant pour base une vectorisation des graphes. C'est le cas des modèles dits *Bag of Fragments (BoF)*. Le principe est le suivant : premièrement, sur un ensemble de graphes (corpus), une identification de tous les sous-graphes (fragments) rencontrés est réalisée. Ces sous-graphes forment un dictionnaire de fragments dont les entrées correspondent aux dimensions d'un espace vectoriel. Par la suite, individuellement sur chaque graphe, on dénombre chacun des fragments observés, rendant ainsi compte de sa composition. Un ordre étant fixé dans le dictionnaire, chaque graphe peut être projeté sur cet espace de fragments, on obtient ainsi un vecteur de dimension fixe (Fig. 6). [17, 18]

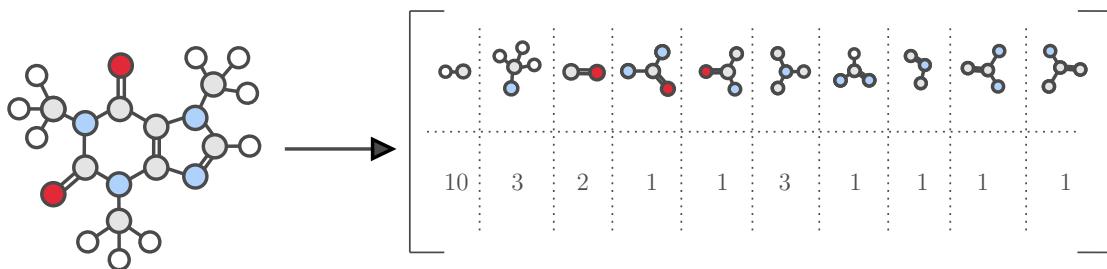


FIGURE 6 – Projection d'une molécule sur un espace de fragments

Les vecteurs produits sont ensuite exploités dans des méthodes d'analyse classiques, telle que des régressions linéaires ou de moindre carrés partiels, ou encore des forêts aléatoires, machines à vecteurs de support (SVM) et réseaux de neurones. [19, 21, 20]

La première limite de cette approche est liée à une explosion combinatoire qui engendre un accroissement exponentiel de dimensionnalité pour cet espace de fragments, à mesure que l'on considère une taille de fragment plus élevée. En conséquence, les projections s'apparentent à des vecteurs creux dès lors que l'on considère des fragments allant au delà d'un atome central et de son voisinage direct. Les fragments utilisables sont donc naturellement imprécis, d'autant qu'une grande proportion de ces fragments sont très rares et ne sont donc observés que quelques fois sur le corpus, ce qui les rend inexploitables dans des modèles statistiques si ce n'est au risque d'un phénomène de sur-apprentissage.

Cette difficulté peut être atténuée en reconSIDérant les modalités des labels des noeuds : il est possible en effet de rapprocher artificiellement sous une même modalité des atomes aux propriétés similaires (propriétés pharmacophoriques) pour diminuer le nombre de dimensions obtenues. [22] Une autre méthode (*ECP*) consiste à borner le nombre de dimensions à une valeur arbitraire, et par utilisation d'une fonction de hachage, à projeter les sacs de fragment dans ce nouvel espace métrique, au prix de collisions non contrôlées, dont on fait l'hypothèse qu'elles perturberont moins le modèle que l'utilisation de vecteurs creux [23]. Il faut également souligner que l'identification des fragments n'est pas parfaite : une notation canonique telle que les SMILES serait nécessaire, mais celle-ci n'est canonique que sur des molécules complètes et non sur des fragments. La notation utilisée est suffisamment approchante d'une forme réellement canonique<sup>3</sup> pour être utilisée, mais celle-ci induit là encore une légère perte d'information. Par ailleurs, cette notation canonique n'est applicable qu'en chimie, ne rendant pas cette approche transposable à d'autres domaines.

La seconde limite est la perte d'information liée au regroupement dans un sac non ordonné de structures locales dont l'agencement est porteur d'information. En plus des approximations citées précédemment, la quantité d'information perdue devient dommageable. Ces modèles peuvent malgré tout se révéler efficaces pour la prédiction de propriétés faciles à modéliser (densité, affinité avec l'eau, etc.), mais montrent vite leurs limites pour les propriétés de plus haut niveau telles que les points d'ébullition ou les aspects sensoriels, et se révèlent alors moins bons que leurs homologues classiques [15].

Ces modèles restent toutefois très simples à implémenter, et sont tout aussi simples à interpréter, les fragments gardant une signification chimique très forte. Ils seront donc toujours pertinents à étudier, ne serait-ce que pour disposer d'un modèle explicatif ou d'une "*baseline*".

## Approches orientées graphes

La littérature fait également état de plusieurs modèles SAR exploitant directement les structures de graphes. Au sein de cette famille, il existe plusieurs sous-branches, interprétant ces structures de différentes façons pour réaliser leur apprentissage.

La première se concentre sur les *distances d'édition* : l'idée est d'observer, pour deux structures ayant une même ossature commune, la différence d'activité induite par un changement structurel localisé, et d'expliquer la variance de la variable observée par cette édition locale. C'est le concept dit de "fossé d'activité" (*activity cliff*) [24, 25], débouchant sur la méthode dite de l'analyse de paire de molécules appairées (*Matched Molecular Pair Analysis*, MMPA) [26], illustrée en Fig. 7. Cette méthode marche d'autant mieux que les fossés d'activité sont nets, elle est de fait très pertinente en chimie. Par ailleurs les substances sont typiquement des entités pouvant être organisées par couples (acide ou base conjuguées), il est donc aisément d'appairer des substances. Enfin, ces mo-

---

3. le taux d'erreur constaté est de l'ordre de 1%

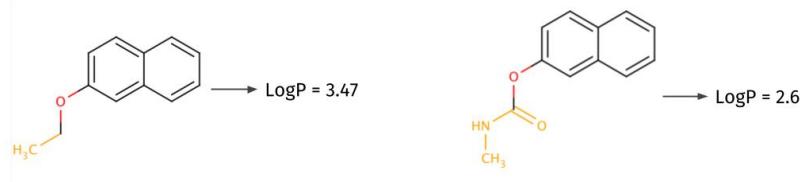


FIGURE 7 – Illustration du principe de MMPA

dèles sont excellents pour restituer l'information, car au plus proche de la logique métier associée à la discipline.

En revanche, il demeure que pour des substances sans appariement, il sera très compliqué d'estimer les comportements physico-chimiques. D'autre part, si l'apprentissage sur des différentiels de structure peuvent donner de bons résultats "comparatifs", i.e qu'il sera assez facile de déterminer quelle molécule de la paire étudiée sera par exemple la plus soluble dans l'eau, l'estimation absolue de cette propriété reste à estimer par rapport à l'ossature, et réclame donc une quantité de données importante pour être réellement pertinent en prédition. Enfin, si ce modèle reste très pertinent en chimie, il est difficilement généralisable à d'autres domaines.

La seconde sous-branche est plus générale, dans le sens où les méthodes qui en relèvent pourraient s'appliquer à n'importe quel domaine, pourvu que les observations soient représentées par des graphes. Ces méthodes consistent à apprendre de manière non-supervisée une représentation optimale des éléments constitutifs du graphe, éventuellement affinés pour une tâche donnée. [27, 28] Ces représentations sont ensuite placées en entrée d'un réseau à apprentissage profond jusqu'à une couche de régression ou de classification. L'utilisation de réseaux de convolution (issus du traitement des images) permet de traiter des graphes n'ayant pas un nombre d'éléments fixé. Cette catégorie de modèles n'est pas plus détaillée ici, car elle fera l'objet du chapitre 3. Elle est en tout cas bien adaptée aux problématiques de la chimie, car si les représentations apprises sont projectives (associées à une perte d'information), il est en revanche possible de les entraîner de manière non-supervisée sur un corpus large, sans forcément connaître l'activité associée à ces substances. Ces méthodes se prêtent donc très bien à un apprentissage par transfert, avantage notable quand la quantité de données labellisées est limitée.

## **Contexte et Problématique**

C'est dans ce contexte à la fois scientifique et industriel que cette thèse CIFRE s'intègre : *Oleon*, filiale du groupe *Avril* et acteur majeur de l'oléochimie dans le monde, développe et synthétise des substances chimiques aux applications diverses (cosmétique, agronomie, lubrifiants mécaniques, agroalimentaires, etc.). Une part importante de son activité est consacrée à l'innovation, et à l'élaboration de nouvelles substances ayant de meilleures propriétés physico-chimiques cibles, et fait face aux difficultés liées à ces tâches de caractérisation de propriétés que nous avons évoqué précédemment. La finalité opérationnelle de cette thèse est d'aider *Oleon* à se perfectionner sur la construction de modèles prédictifs structure-activité, avec les contraintes suivantes :

- ne pas compter uniquement sur les données dont Oleon dispose,
  - tirer parti des ressources disponibles dans le domaine public.

Ce contexte général nous amène au sujet de recherche suivant :

## Caractérisation et plongement de sous-graphes colorés Application à la construction de modèles structures à activité

Ce sujet de thèse cible trois objectifs principaux suivants :

1. Améliorer/Développer un outillage algorithmique pour la canonisation de graphes et de sous-graphes (fragments) colorés, en veillant à ce que cet outillage soit le plus générique possible (i.e. exploitable en dehors du domaine de la chimie),
2. Revisiter les méthodes de vectorisation de graphe construites sur la base d'une fragmentation, et développer une approche basée sur des plongements de fragments (à l'instar du modèle *Word2Vec* développé pour le traitement automatique des langues naturelles),
3. Exploiter cette vectorisation sur des tâches de régression ou de classification pour estimer des propriétés physico-chimiques propres aux molécules.

Si ces trois objectifs sont concourants, le deuxième objectif est central : il nécessite en effet une représentation canonique pour les fragments et s'appuie donc sur le premier objectif. Le troisième objectif exploite directement les vectorisations produites par le deuxième objectif.

Comme précédemment suggéré l'hypothèse suivante justifie notre second objectif : si les substances chimiques publiques seules ne peuvent nous aider à prédire de manière précise une propriété cible pour une molécule donnée, elles peuvent a priori nous permettre d'apprendre une bonne représentation vectorielle pour les fragments des substances chimiques, voire pour les molécules. Derrière cette notion de bonne représentation nous entendons une représentation susceptible d'encoder du sens, ici une réalité physico-chimique, à l'échelle du fragment puis par composition à l'échelle de la molécule.

L'exploitation de ces données publiques passe donc par un apprentissage des meilleures vectorisations possibles, notamment par une mise en contexte globale de fragments, inspiré des *plongements lexicaux* (premier objectif). Pour étudier précisément ces contextes lexicaux, et construire les meilleurs plongements possibles, il est nécessaire de savoir caractériser formellement un sous-graphe, et reconnaître sa classe d'isomorphisme (premier objectif). Enfin, nous ne perdons pas de vue la finalité industrielle de cette thèse, en cherchant à exploiter ces travaux pour construire des SAR (troisième objectif).

## Contributions

Pour conclure cette introduction, nous listons dans cette section un résumé des travaux principaux entrepris dans le cadre cette thèse. Les trois objectifs principaux listés précédemment font l'objet des trois chapitres "contributions" de ce manuscrit. Compte tenu des domaines distincts abordés au regard des objectifs fixés, nous avons réparti notre état de l'art dans ces trois chapitres pour faciliter la lecture du manuscrit. Enfin, au cours de la thèse, des travaux de recherche de périmètre plus restreints et extérieurs au sujet principal ont également été traités. Ceux-ci sont regroupés en annexe.

En réponse aux trois objectifs principaux fixés initialement, notre contribution principale fait donc l'objet des trois chapitres suivants :

## Caractérisation de sous-graphe à un isomorphisme près

Sur ce volet plus théorique, nous proposons un algorithme apportant une solution au problème de la production d'une forme canonique pour les graphes. Les algorithmes existants ne remplissent pas entièrement nos attentes : les algorithmes généraux ne traitent pas nativement les graphes aux arêtes colorées, tandis que les algorithmes spécialisés en chimie sont soit restreints aux molécules entières, soit inexacts. Enfin, tous ces algorithmes ont une implémentation uniquement séquentielle. Nous proposons un algorithme plus général, traitant nativement les arêtes colorées, et pouvant être exécuté de façon distribuée. Si notre implémentation n'est pas plus rapide que ceux de l'état de l'art, elle n'est pas non plus drastiquement plus lente passé une certaine taille de graphe, et demeure de complexité tout à fait raisonnable (linéaire) sur les graphes moléculaires.

## Plongements sémantiques de fragments de graphes

Sachant caractériser de façon formelle un fragment moléculaire, il devient possible d'étudier leur distribution sur un corpus large. Nous observons empiriquement des similitudes étonnantes avec les corpus issus du traitement automatique des langages naturels. Ce domaine disposant de méthodes permettant de traiter des problèmes de forte dimensionnalité, nous formulons une analogie entre fragments moléculaires et unité lexicale (mot). Nous définissons une manière d'adapter les plongements lexicaux à ces fragments, pour obtenir des vectorisations apprises sur un corpus large semblant capter un sens chimique.

## Utilisation des plongements de fragments pour l'élaboration de Qsar

Enfin, nous exploitons ces représentations nouvellement apprises pour l'objectif final, i.e. l'élaboration de modèles structures-activité. Disposant de vectorisations de fragments faisant état de leur sémantique, nous les utilisons ensemble pour classifier ou calculer une régression sur les graphes entiers. Après avoir testé leur utilisation dans différentes situations et pour des tâches variées, ceux-ci se révèlent bien adaptés en tant que valeurs initiales de "nodes features" pour les méthodes orientées graphe, et permettent d'améliorer sensiblement leur efficacité, et ce même sur des jeux de données assez restreints, ce qui vient répondre à la problématique initiale. Dans certains cas, l'association de ces plongements avec les méthodes de convolution sur graphes aboutissent même aux meilleurs résultats connus à ce jour sur ces jeux de données.

## Références

- [1] Jerry MARCH, *Advanced organic chemistry : reactions, mechanisms, and structure*, John Wiley & Sons, 1992.
- [2] Francis A CAREY et Richard J SUNDBERG, *Advanced organic chemistry : part A : structure and mechanisms*, Springer Science & Business Media, 2007.
- [3] Francis A CAREY et Richard J SUNDBERG, *Advanced Organic Chemistry : Part B : Reaction and Synthesis*, Springer Science & Business Media, 2007.
- [4] Ira N. LEVINE, *Quantum Chemistry, 7th Edition*, Pearson, 2014.
- [5] David J GRIFFITHS et Darrell F SCHROETER, *Introduction to quantum mechanics*, Cambridge University Press, 2018.
- [6] David WEININGER, Arthur WEININGER et Joseph L WEININGER, « SMILES. 2. Algorithm for generation of unique SMILES notation », in : *Journal of Chemical Information and Computer Sciences* 29.2 (1989), p. 97-101.
- [7] Stephen HELLER et al., « InChI-the worldwide chemical structure identifier standard », in : *Journal of cheminformatics* 5.1 (2013), p. 7.
- [8] Charlotte Froese FISCHER, « General hartree-fock program », in : *Computer physics communications* 43.3 (1987), p. 355-365.
- [9] Antoine DAINA, Olivier MICHELIN et Vincent ZOETE, « iLOGP : a simple, robust, and efficient description of n-octanol/water partition coefficient for drug design using the GB/SA approach », in : *Journal of chemical information and modeling* 54.12 (2014), p. 3284-3301.
- [10] Steven ABBOTT et Charles M HANSEN, *Hansen solubility parameters in practice*, Hansen-Solubility, 2008.
- [11] Sunghwan KIM et al., « PubChem substance and compound databases », in : *Nucleic acids research* 44.D1 (2015), p. D1202-D1213.
- [12] Harry E PENCE et Antony WILLIAMS, *ChemSpider : an online chemical information resource*, 2010.
- [13] Zhenqin WU et al., « MoleculeNet : a benchmark for molecular machine learning », in : *Chemical science* 9.2 (2018), p. 513-530.
- [14] JC BRADLEY, A WILLIAMS et A LANG, *Jean-Claude Bradley Open Melting Point Dataset, 2014*, URL : [https://figshare.com/articles/Jean\\_Claude\\_Bradley\\_Open\\_Melting\\_Point\\_Datset/1031637](https://figshare.com/articles/Jean_Claude_Bradley_Open_Melting_Point_Datset/1031637).
- [15] Muthukumarasamy KARTHIKEYAN, Robert C GLEN et Andreas BENDER, « General melting point prediction based on a diverse compound data set and artificial neural networks », in : *Journal of chemical information and modeling* 45.3 (2005), p. 581-590.

- [16] Sophie MARTEL et al., « Large, chemically diverse dataset of logP measurements for benchmarking studies », in : *European Journal of Pharmaceutical Sciences* 48.1 (2013), p. 21-29.
- [17] Alexandre VARNEK et al., « Substructural fragments : an universal language to encode reactions, molecular and supramolecular structures », in : *Journal of computer-aided molecular design* 19.9-10 (2005), p. 693-703.
- [18] Igor BASKIN et Alexandre VARNEK, « Building a chemical space based on fragment descriptors », in : *Combinatorial chemistry & high throughput screening* 11.8 (2008), p. 661-668.
- [19] Fiorella RUGGIU et al., « ISIDA Property-Labelled Fragment Descriptors », in : *Molecular informatics* 29.12 (2010), p. 855-868.
- [20] Fiorella RUGGIU, « Property-enriched fragment descriptors for adaptive QSAR », thèse de doct., 2014.
- [21] Fiorella RUGGIU et al., « Individual Hydrogen-Bond Strength QSPR Modelling with ISIDA Local Descriptors : a Step Towards Polyfunctional Molecules », in : *Molecular informatics* 33.6-7 (2014), p. 477-487.
- [22] Alexandre VARNEK et al., « ISIDA-Platform for virtual screening based on fragment and pharmacophoric descriptors », in : *Current Computer-Aided Drug Design* 4.3 (2008), p. 191.
- [23] David ROGERS et Mathew HAHN, « Extended-connectivity fingerprints », in : *Journal of chemical information and modeling* 50.5 (2010), p. 742-754.
- [24] Ye HU, Dagmar STUMPFE et Jürgen BAJORATH, « Advancing the activity cliff concept. », in : *F1000Research* 2 (2013), p. 199, ISSN : 2046-1402, DOI : 10.12688/f1000research.2-199.v1.
- [25] D STUMPFE et al., « Follow up : Advancing the activity cliff concept, part II ANRT biblio Chemistry ML NeuralNets QSAR », in : *F1000Research* (2014).
- [26] Noel M. O'BOYLE et al., « Using matched molecular series as a predictive tool to optimize biological activity », in : *Journal of Medicinal Chemistry* 57.6 (2014), p. 2704-2713, ISSN : 15204804, DOI : 10.1021/jm500022q.
- [27] Annamalai NARAYANAN et al., « subgraph2vec : Learning distributed representations of rooted sub-graphs from large graphs », in : *arXiv preprint arXiv :1606.08928* (2016).
- [28] Aditya GROVER et Jure LESKOVEC, « node2vec : Scalable feature learning for networks », in : *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, p. 855-864.

- [29] David K DUVENAUD et al., « Convolutional networks on graphs for learning molecular fingerprints », in : *Advances in neural information processing systems*, 2015, p. 2224-2232.
- [30] Joseph GOMES et al., « Atomic convolutional networks for predicting protein-ligand binding affinity », in : *arXiv preprint arXiv :1703.10603* (2017).
- [31] Steven KEARNES et al., « Molecular graph convolutions : moving beyond fingerprints », in : *Journal of computer-aided molecular design* 30.8 (2016), p. 595-608.
- [32] Mikael HENAFF, Joan BRUNA et Yann LECUN, « Deep convolutional networks on graph-structured data », in : *arXiv preprint arXiv :1506.05163* (2015).
- [33] Petar VELIČKOVIĆ et al., « Graph attention networks », in : *arXiv preprint arXiv :1710.10903* (2017).
- [34] Thomas N KIPF et Max WELLING, « Semi-supervised classification with graph convolutional networks », in : *arXiv preprint arXiv :1609.02907* (2016).
- [35] Jie ZHOU et al., « Graph neural networks : A review of methods and applications », in : *arXiv preprint arXiv :1812.08434* (2018).
- [36] Annamalai NARAYANAN et al., « graph2vec : Learning Distributed Representations of Graphs », in : *arXiv preprint arXiv :1707.05005* (2017).
- [37] Palash GOYAL et Emilio FERRARA, « Graph Embedding Techniques, Applications, and Performance : A Survey », in : *arXiv preprint arXiv :1705.02801* (2017).

# Caractérisation de sous-graphes

---

Pour étudier des graphes, et à fortiori des sous-graphes dans un ensemble d'observations, il s'avère nécessaire de pouvoir les identifier formellement, et ainsi désigner de façon uniforme des structures qui seraient identiques.

En particulier, il est important de pouvoir rendre indépendante la notation que l'on fera d'un (sous-)graphe vis-à-vis d'un quelconque ordre dans une énumération de noeuds ou d'arêtes. On cherche ainsi à définir une notation de graphe qui soit canonique, de sorte que deux (sous-)graphes isomorphes aient la même forme canonique. Cette notation pourrait dès lors être utilisée pour identifier des modalités de motifs.

Ce document présente un algorithme, que nous avons dénommer Scott, qui propose une solution à ce problème. Cette solution se veut générale et indépendante du domaine d'application, et ne nécessite donc que la connaissance encodée sur les éléments des graphes traités.

## 1.1 Problème

Un graphe  $G = (V_G, E_G) \in \mathbb{G}$  est une structure de données composée d'un ensemble de noeuds  $V_G$ , reliés entre eux par des arêtes rassemblées dans un ensemble  $E_G$ .

Ce sont des objets mathématiques particulièrement adaptés pour représenter une population d'entités interconnectées via des liens (uniformes ou non), telles que des réseaux de télécommunication, des molécules, des relations sociales, etc.

Un graphe peut être décrit par une énumération des éléments de  $V_G$  et  $E_G$ . Cependant, une telle énumération n'est pas unique, car on ne peut pré-supposer l'existence d'une relation d'ordre ordonnant de tels ensembles. Sur la base de ce type de description non-unique, on constate que deux graphes identiques peuvent avoir des encodages différents. La nécessité de pouvoir confirmer ou infirmer une égalité structurelle entre deux graphes amène naturellement à la définition du problème d'isomorphisme de graphes.

### 1.1.1 Isomorphisme de graphes

Soient deux graphes  $G = (V_G, E_G)$  et  $H = (V_H, E_H)$ , un isomorphisme  $f : V_G \rightarrow V_H$  de  $G$  vers  $H$  est une bijection de l'ensemble des sommets de  $G$  dans l'ensemble des sommets de  $H$ , conservant les arêtes.

$$G \simeq H \iff \exists f, \forall u, v \in V_G, (u, v) \in E_G \Leftrightarrow (f(u), f(v)) \in E_H$$

Si  $f$  existe, alors  $G$  et  $H$  admettent une relation d'isomorphisme ( $\simeq$ ) et sont donc isomorphes, on note alors  $G \simeq H$ . Déterminer l'existence (ou la non existence) de  $f$  est la façon la plus élémentaire

de répondre à ce problème. Déterminer une expression de  $f$  est une tâche au moins aussi complexe, qui répond également au problème.

Il est important de noter que malgré de nombreux travaux portant sur ce problème d'isomorphisme, ([38] [39]), sa classe de complexité reste à ce jour mal connue ([40], [41]) : il n'est pas prouvé que ce problème soit **NP-complet**, mais on ne connaît pas non plus d'algorithme polynomial pour le cas général des graphes quelconques. Ce problème dispose donc de sa propre classe de complexité notée **GI** (Graph Isomorphism) ([42], [43]). Des travaux récents ([44]) semblent toutefois indiquer que ce problème soit quasi-polynomial, bien que cela reste à confirmer.

### 1.1.2 Forme canonique de graphe

Une autre façon de résoudre ce problème est de calculer, pour tout graphe  $G$ , un représentant dit canonique. Ce représentant (généralement un graphe isomorphe à  $G$ ) sera unique pour toute la classe d'isomorphisme de  $G$ ,  $[G]^1$ . Deux graphes isomorphes deviennent donc littéralement égaux lorsqu'ils sont canonisés. La fonction de canonisation  $Canon$  est ainsi un morphisme de l'ensemble des graphes  $\mathbb{G}$  vers l'ensemble quotient  $\mathbb{G}/\simeq$ .

$$\begin{aligned} Canon : \mathbb{G} &\rightarrow \mathbb{G}/\simeq \\ G \simeq H &\iff Canon(G) = Canon(H) \end{aligned}$$

C'est un problème qui est au moins aussi complexe que le test d'isomorphisme classique, car sa résolution fournit une solution à ce dernier dans tous les cas. Par ailleurs dans le cas général, il est plus coûteux de calculer une forme canonique que de tester un isomorphisme. En effet, il est souvent possible de statuer sur les test d'isomorphisme avant l'exécution complète de l'algorithme.

Ceci étant, malgré un surcoût algorithmique, ce problème connexe présente de nombreux avantages. La forme canonique d'un graphe  $G$  étant unique, celle-ci ne nécessite d'être calculée qu'une seule fois, après quoi on pourra directement comparer les formes canoniques de graphes candidats à un isomorphisme, en un temps beaucoup plus court. Cette complexité supplémentaire est donc très vite amortie dès lors que l'on doit effectuer beaucoup de tests d'appariements sur une population de (sous-)graphes.

Lors de l'élaboration de modèles structures à activité notamment, on cherche justement à rendre les fragments (sous-graphes) étudiés indépendants de la manière dont ils sont encodés, afin de pouvoir regrouper les éléments qui seraient effectivement identiques et porteurs de la même information. On pense également au cas où l'on souhaite indexer ce type de fragments, pour la construction d'un index dans une base de données, où il serait impensable de réaliser à chaque recherche un test d'isomorphisme.

Dans certains domaines bien spécifiques, des travaux ont pu aboutir à des heuristiques pour standardiser des écritures, grâce à la recherche de nombreux invariants de graphes et du domaine. On peut citer en ce sens la notation *SMILES* [45] ou *InChI* [46] en chimie organique, qui se veulent être des écritures uniques pour une molécule. La principale limitation de ces écritures réside dans l'exploitation d'invariants chimiques pour lever les ambiguïtés, forme de connaissance entièrement relative au domaine et donc extérieure au contexte du graphe considéré. Elles ne sont donc pas applicables dans le cas général.

---

1. classe d'équivalence de la relation d'équivalence "est isomorphe à", notée  $\simeq$

### 1.1.3 Algorithmes

On note donc un écart important entre le traitement théorique et pratique de ce problème : si des algorithmes de complexité polynomiale existent pour un grand nombre de classes restreintes de graphes (degré borné [47], graphes planaires [48], etc.), et si le cas général semble également traitable en temps quasi-polynomial ([44]), les meilleurs algorithmes implémentés à ce jour demeurent de complexité exponentielle dans le cas général.

Un état de l'art complet de ces implémentations est réalisé dans [49], ainsi qu'un benchmark sur des cas d'isomorphismes complexes construits à l'aide de [50]. Les algorithmes *saucy* ([51], [52]), *conauto* ([53]), *bliss* ([54], [55]) et *nauty/traces* ([56], [57], [58]) sont ainsi évalués dans un contexte proche du pire des cas possibles.

Nous nous intéressons ici à la résolution du problème de génération d'une forme canonique permettant de représenter les graphes dans le cas le plus général possible, i.e. dans le cas où les noeuds, tout comme les arêtes sont labellisés. Parmi les algorithmes existants, seuls *bliss*, *nauty* et *traces* permettent d'obtenir une labellisation canonique. Leur principe de fonctionnement est basé sur la recherche d'une coloration dite *équitable* (cf. papiers) des noeuds, qui permet ensuite d'induire une relation d'ordre sur les noeuds à l'aide d'arbres de recherche (backtracking), dont les élagages progressifs permettent une diminution drastique de l'espace des solutions possibles. Dès lors que l'on dispose d'une telle relation d'ordre, il est possible d'énumérer canoniquement tous les éléments constitutifs d'un graphe.

### 1.1.4 Proposition

Malheureusement, il n'est pas prouvé que ces approches basées sur une coloration des noeuds puissent aboutir dans un contexte d'arêtes non-homogènes, autrement dit labellisées (colorées), autrement que par une ré-écriture profonde du graphe. Typiquement, en dupliquant le graphe en autant de modalités d'arêtes, puis en les masquant respectivement. Cette difficulté a motivé l'élaboration de l'approche algorithmique présentée ici. Par ailleurs, la majeure partie des algorithmes formant l'état de l'art sont basés sur la méthode de backtracking, évoquée précédemment. Aucun n'utilise à notre connaissance des réécritures de graphe, bien que ce procédé puisse présenter des avantages, comme celui de traiter nativement des arêtes colorées.

L'algorithme développé, qui constitue notre première contribution, propose un moyen de projeter un graphe quelconque vers un arbre, classe de graphe pour lequel on peut facilement définir un encodage symbolique canonique. La relation d'ordre nécessaire à la canonicité du processus découle de l'arbre obtenu. Cet encodage va au-delà d'une simple signature, et se veut facile à ré-interpréter pour éventuellement remonter au graphe d'origine. En cela, il s'agit également d'un encodage efficace de graphe.

En outre cet algorithme peut aisément bénéficier d'une implémentation distribuée, qui assure de fait une scalabilité horizontale, là où les algorithmes de l'état de l'art, bien que démontrant un temps d'exécution impressionnant du fait de leur implémentation très optimisée depuis parfois plusieurs décennies, demeurent séquentiels.

## 1.2 Écriture canonique d'un arbre

### 1.2.1 Définitions

Soit  $\Sigma$  un alphabet, autrement dit un ensemble de symboles. Une séquence de  $n \in \mathbb{N}$  symboles (potentiellement nulle) appartient à l'ensemble des séquences de  $n$  éléments  $\Sigma^n$ , et permet de définir l'ensemble des chaînes de symboles de longueur indéterminée  $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$ .

Un ensemble  $\mathcal{E}$  est totalement ordonné par une relation binaire  $\leq$  si celle-ci est réflexive, antisymétrique et transitive, et si deux éléments de  $\mathcal{E}$  sont toujours comparables  $\forall x, y \in \mathcal{E}, x \leq y$  ou  $y \leq x$ . On note que  $(\mathcal{E}, \leq)$ . En particulier  $(\Sigma^*, \leq)$ , car une suite de symboles est ordonnée par l'ordre lexicographique.

Pour tout graphe  $G = (V, E) \in \mathbb{G}$ , on assimile à une fonction  $e_G(\cdot, \cdot) : V \times V \rightarrow \mathbb{N}$ , la valeur (potentiellement nulle) associée à une liaison entre deux noeuds  $i$  et  $j$  sur ce graphe, qui n'est autre que la valeur  $A_{ij}^G$ , avec  $A^G$  la matrice d'adjacence de  $G$ .

Notons l'ensemble des arbres  $\mathbb{T} \subset \mathbb{G}$ , sous-ensemble des graphes tel que  $t = (V, E) \in \mathbb{T}$  vérifie les propriétés suivantes :

- $t$  est connecté et acyclique
- $t$  a exactement  $n - 1$  arêtes, où  $n = |V|$
- il existe un unique chemin de poids minimal reliant deux noeuds

Plus particulièrement, nous nous intéressons aux arbres planaires enracinés  $t_\rho$  dont un des noeuds  $\rho \in V$  est identifié comme une racine. À chaque noeud  $\nu \in V$  de cet arbre  $t_\rho$ , nous pouvons attribuer un niveau  $N$ , qui est la longueur du chemin vers  $\rho$ , notée  $\|\mu(\nu, \rho)\|$ , avec  $\mu(\cdot, \cdot)$  la fonction associant à deux noeuds le chemin minimal les reliant. Pour faciliter la suite des notations, nous considérons que le niveau minimal 0 ne désigne pas  $\rho$  mais au contraire sa feuille la plus éloignée. On ne prend donc pas la distance minimale brute, mais la différence la distance maximale constatée sur l'arbre enraciné  $t_\rho$  et celle-ci. Le niveau est donc à valeur dans l'intervalle  $[0, N_G]$ ,  $N_G$  désignant le niveau de la racine  $\rho$ , et donc le niveau maximum observé sur  $G$ .

$$\begin{aligned} lvl : V &\rightarrow \mathbb{N}, \\ lvl(\nu) &= \max_{\nu' \in V_G} \|\mu(\nu', \rho)\| - \|\mu(\nu, \rho)\| = N_G - \|\mu(\nu, \rho)\| \end{aligned}$$

On note  $V_N$  l'ensemble des noeuds de niveau  $N$ , tel que  $\nu \in V_N \Leftrightarrow lvl(\nu) = N$ . Propriété remarquable des arbres, chaque noeud  $\nu \in V_N$  est l'unique parent d'un ensemble de noeuds de niveau  $N - 1$ , que l'on nomme descendants. Si l'on exclut tous les noeuds de niveau supérieur à  $N$ , chaque noeud  $\nu \in V_N$  d'un arbre est donc lui-même la racine d'un arbre enraciné sur  $\nu$ , noté  $t_\nu$ . On note  $T_N$  l'ensemble de ces arbres enracinés associé à un niveau  $N$ . Soit  $\Lambda_\nu \subseteq V$  l'ensemble des descendants d'un noeud  $\nu \in V$ ,  $\lambda \in \Lambda_\nu \Leftrightarrow \begin{cases} lvl(\lambda) = lvl(\nu) - 1 \\ e_G(\nu, \lambda) \neq 0 \end{cases}$ .

**Theorem 1** (Neveu). *Un arbre enraciné doté d'une relation d'ordre portant sur tout arbre est encodable canoniquement.*

Il est prouvé qu'un plongement planaire d'un arbre enraciné admet une notation (dite de Neveu) non-ambiguë (canonique) sous forme de séquence de mots, de sorte que deux arbres ayant la même notation soient égaux. [59]

Cette notation, dont nous donnons un exemple en Fig. 1.1, n'est toutefois canonique que si le plongement planaire (représentation en deux dimensions) est lui-même canonique, et qu'il existe

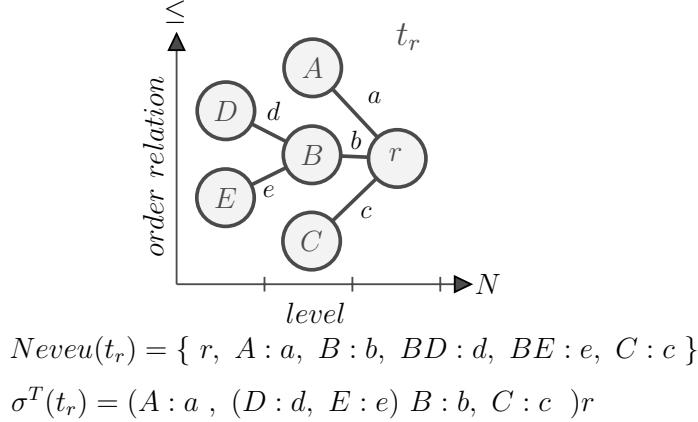


FIGURE 1.1 – Encodage d'un arbre enraciné  $t_r$  par une séquence de symboles, d'après la notation de Neveu (étendue aux arêtes), et par la fonction de trace proposée  $\sigma^T$  (1.2.14 ).

de fait une relation d'ordre définie sur les noeuds d'un même niveau  $N$  (noeuds frères). Nous détaillons par la suite une méthode permettant de définir de façon systématique cette relation d'ordre dans le cas des arbres ayant des noeuds et arêtes colorés (hétérogènes). Nous proposons également une alternative à la notation de Neveu, grâce à une fonction de trace  $\sigma^T$  définie pour tout arbre, et limitant la redondance.

**Lemma 1.** *Tout noeud est encodable sous forme de chaîne de symboles*

*Démonstration.* Les noeuds d'un graphe  $G$  peuvent porter une coloration, c'est-à-dire une suite de symboles dont les différentes modalités sont désignées comme des *labels*. On caractérise cette coloration comme une association entre chaque élément de  $V_G$  et une modalité à valeur dans cet ensemble de couleurs/labels,  $L_G^{V_G} L_G^{V_G} : V_G \rightarrow \Sigma^*$ . Dans ce cas, on renforce la définition d'isomorphisme introduite précédemment, car en plus de conserver les arêtes, il faut que ceux-ci conservent les labels. Cette définition s'applique également à tout graphe non coloré, celui-ci n'étant finalement qu'un cas particulier où un seul label est représenté.

$$\begin{aligned} \forall G = (V_G, E_G), H = (V_H, E_H) \in \mathbb{G} \\ G \simeq H \Leftrightarrow \exists f : V_G \rightarrow V_H , \begin{cases} \forall \nu_1, \nu_2 \in V_G , e_G(\nu_1, \nu_2) = e_H(f(\nu_1), f(\nu_2)) \\ \forall \nu \in V_G , L_G^{V_G}(\nu) = L_H^{V_H}(f(\nu)) \end{cases} \end{aligned}$$

Or, la première condition est toujours vérifiée dès lors que  $f$  est une permutation impliquant deux noeuds (cas particulier d'automorphisme).

$$f : V_G \rightarrow V_H , \forall \nu_1, \nu_2 \in V_G , \begin{cases} f(\nu_1) = \nu_2 \\ f(\nu_2) = \nu_1 \end{cases} \Rightarrow e_G(\nu_1, \nu_2) = e_H(f(\nu_1), f(\nu_2))$$

Autrement dit, pour permutez deux noeuds  $\nu_1$  et  $\nu_2$  deux-à-deux sans changer de classe d'isomorphisme, il faut et il suffit que  $L_G^{V_G}(\nu_1) = L_G^{V_G}(\nu_2)$ . Deux noeuds remplissant cette condition peuvent donc faire l'objet d'une ambiguïté sans impacter la classe d'isomorphisme de leur graphe.

Dans le cadre d'une notation canonique à un isomorphisme près, un noeud de tout graphe  $G$  est donc suffisamment défini par son label (éventuellement nul), que l'on peut choisir de repréSEN-

ter de manière générale comme une suite de symboles  $\Sigma^*$ , ensemble doté d'une relation d'ordre lexicographique.

Cet encodage est donné par une fonction de trace  $\sigma^{V_G} : V_G \rightarrow \Sigma^*$ , et permet par transitivité de doter  $V_G$  d'une relation d'ordre, car  $(\Sigma^*, \leq)$  est ordonné lexicographiquement.  $\square$

**Corollary 1.** *Toute arête est encodable sous forme de chaîne de symboles*

*Démonstration.* De la même manière, toute arête dans un graphe  $G$  peut faire l'objet d'un label, introduit par la relation  $L_G^{E_G} : E_G \rightarrow \Sigma^*$ . Un isomorphisme de ce type de graphe implique à son tour une condition sur les permutations d'arêtes, dans le sens où d'une part leurs noeuds de départ et d'arrivée doivent être invariants, et d'autre part le label relatif à l'arête conservé. Là encore, si on permute des arêtes deux-à-deux, il suffit d'avoir deux noeuds du même label pour qu'ils puissent faire l'objet d'une permutation n'affectant pas la classe d'isomorphisme du graphe.

Dans le cadre d'une notation canonique à un isomorphisme près, une arête est donc suffisamment définie par son noeud source et son noeud cible, eux mêmes représentés par leurs chaînes de symboles respectives, ainsi que son label (éventuellement nul), que l'on peut choisir de représenter avec le même alphabet  $\Sigma^*$ . Cet encodage est donné par une fonction de trace  $\sigma^{E_G} : E_G \rightarrow \Sigma^*$ , et permet de doter  $E_G$  d'une relation d'ordre.  $\square$

### 1.2.2 Fonction de trace $\sigma$

De la même façon que nous avons défini des fonctions de trace permettant d'encoder canoniquement les noeuds et les arêtes d'un graphe, nous cherchons ici à définir une fonction de trace définie pour tout arbre, sans présupposer d'une relation d'ordre sur ceux-ci.

**Proposition 1.** *Tout arbre est encodable à un isomorphisme près sous la forme d'une séquence de symboles produite par une fonction  $\sigma^T : \mathbb{T} \rightarrow \Sigma^*$ .*

$$\exists \sigma^T : \mathbb{T} \rightarrow \Sigma^*, \forall t_1, t_2 \in \mathbb{T}, \sigma^T(t_1) = \sigma^T(t_2) \Leftrightarrow t_1 \simeq t_2 \quad (1.2.1)$$

Nous définissons également l'inverse à droite<sup>2</sup>  $\tilde{\sigma}^T$  de cette fonction de trace, associant à la trace produite par  $\sigma^T$  d'un arbre  $t$  le représentant unique de sa classe d'équivalence  $[t]$ . L'espace de sortie de cette application est donc  $\mathbb{T}$  quotienté par la relation d'isomorphisme  $\simeq$ , d'où l'impossibilité de la décrire comme une véritable fonction inverse.

$$\exists \tilde{\sigma}^T : \Sigma^* \rightarrow \mathbb{T}/\simeq, \tilde{\sigma}^T \circ \sigma^T : \mathbb{T} \rightarrow \mathbb{T}/\simeq, \sigma^T(t) \mapsto [t] \quad (1.2.2)$$

**Proposition 2** (Corollaire). *Un arbre  $t \in \mathbb{T}$ , pouvant être un sous-graphe d'un graphe  $g \in \mathbb{G}$ , peut être assimilé de façon réversible à un noeud unique de label  $\sigma^T(t)$ .*

Conséquence immédiate de la Prop. 1, si une fonction de trace peut encoder un arbre sous forme de suite de symboles sans perte d'information, et comme ces mêmes suites peuvent être utilisées comme labels sur des noeuds, il est possible de compresser un (sous-) arbre en un unique noeud portant ce label, et ce même si cet arbre est lui-même un sous-ensemble d'un graphe quelconque.

2. soit une *section* au sens des catégories

Nous pouvons formaliser cette compression par une fonction  $\kappa$  projetant tout arbre vers l'ensemble  $V$  des structures homogènes à des vertices de  $V_G$ .

$$(1.2.1), (1.2.2) \Rightarrow \exists \kappa : \mathbb{T} \rightarrow V, \exists \tilde{\kappa} : V \rightarrow \mathbb{T}/\simeq, \tilde{\kappa} \circ \kappa \simeq id_{\mathbb{G}} \quad (1.2.3)$$

Nous cherchons à présent à prouver les Prop. 1 et 2, et de déterminer une expression de cette fonction  $\sigma^T$ .

**Lemma 2.** *Les sous-arbres  $t_\nu$  d'un niveau  $N$  sont ordonnés lexicographiquement.*

*Démonstration.* Nous formulons pour un niveau quelconque  $N \in [0, N_G]$  la proposition  $\mathcal{P}_N$  suivante, dans laquelle nous supposons vraies les Prop. 1 et Prop. 2.

$\mathcal{P}_N$  Soit  $T_N$  l'ensemble des arbres enracinés sur les noeuds d'un niveau  $N$ ,

$$T_N = \bigcup_{\nu \in V_G \mid lvl(\nu) = N} t_\nu$$

Nous restreignons les domaines de définition respectifs de  $\sigma^T$  et de  $\kappa$  au seul niveau  $N$  de cette proposition,

$$\exists \sigma_N^T : T_N \rightarrow \Sigma^*, \tilde{\sigma}_N^T \circ \sigma_N^T \simeq id_{\mathbb{G}} \quad (1.2.4)$$

$$\exists \kappa_N : T_N \rightarrow V, \exists \tilde{\kappa}_N : V \rightarrow T_N/\simeq, \tilde{\kappa}_N \circ \kappa_N \simeq id_{\mathbb{G}} \quad (1.2.5)$$

Cette fonction de trace proposée dote les arbres frères d'un même niveau  $N$  d'une relation d'ordre lexicographique, de sorte que l'ensemble noté  $K_N$  des noeuds portant comme labels leurs traces respectives est lui-même ordonné.

$$K_N = \bigcup_{t \in T_N} \kappa_N(t), (K_N, \leq) \quad (1.2.6)$$

Nous supposons donc, pour un niveau  $N$ , être en mesure d'ordonner tous les arbres frères de ce niveau, par une fonction de trace  $\sigma_N^T$  fournissant pour chacun d'eux un encodage canonique. Cette proposition est illustrée en Fig.1.2.

$\mathcal{P}_{N+1}$  Hypothèse d'induction (*HI*) : supposons que  $\mathcal{P}_N$  soit vraie, alors  $(K_{N+1}, \leq)$  est-il ordonné ? Peut-on par ailleurs en déduire une expression de  $\sigma^T$  ?

Tout élément de  $K_N$  n'est ainsi relié qu'à un unique élément de  $V_{N+1}$  exclusivement par une arête  $e \in E_N$ . On décrit donc formellement et sans ambiguïté tous les arbres  $T_{N+1}$  par une énumération de tous les triplets de  $K_N \times E_N \times V_{N+1}$  existants.

$$\mathcal{P}_N \Rightarrow (K_N, \leq) \left. \begin{array}{c} (E_N, \leq) \\ (V_{N+1}, \leq) \end{array} \right\} \Rightarrow (K_N \times E_N \times V_{N+1}, \leq) \quad (1.2.7)$$

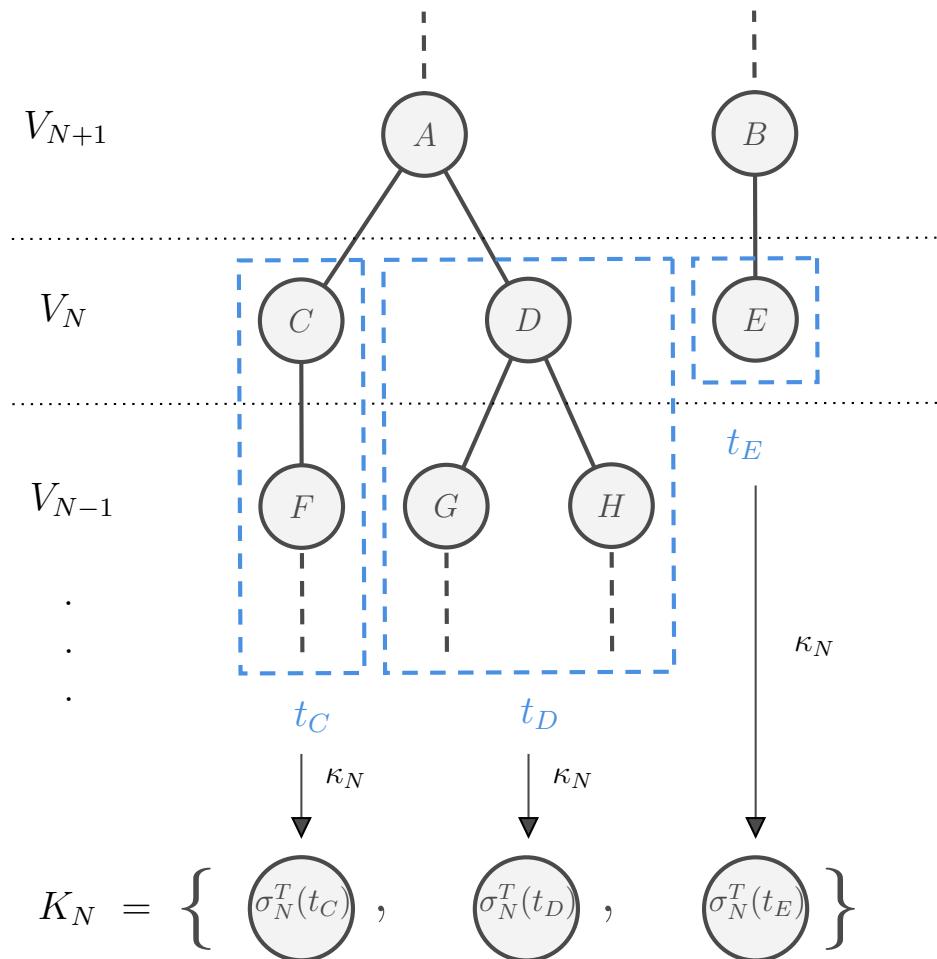


FIGURE 1.2 – Proposition  $\mathcal{P}_N : (K_N, \leq)$

$$\forall t_\nu \in T_{N+1} , \left( \bigcup_{\lambda \in \Lambda_\nu} \sigma^V(\lambda) \times \sigma^E(e(\nu, \lambda)) \times \sigma^V(\nu), \leq \right) \quad (1.2.8)$$

$$\mathcal{P}_{N+1} \text{ est vraie} \quad (1.2.9)$$

Ainsi, si nous sommes en mesure de donner un encodage canonique d'un (sous-) arbre pour un certain niveau, alors nous pouvons en faire de même pour le niveau suivant.

**$\mathcal{P}_0$**  Initialisation : Cette proposition est triviale au niveau des feuilles, car des arbres sans descendants sont directement assimilables à des noeuds (noeud  $E$  en Fig. 1.2), et peuvent donc être encodés comme tels par  $\sigma^V$ .

$$\Lambda_\nu = \emptyset \Leftrightarrow \sigma^T(t_\nu) \equiv \sigma^V(\nu) \quad (1.2.10)$$

$$\mathcal{P}_0 \text{ est vraie} \quad (1.2.11)$$

$$(1.2.9), (1.2.11) \Rightarrow \mathcal{P}_N \text{ est vraie} \quad (1.2.12)$$

□

Nous venons de prouver que si les ensembles  $V_G$  et  $E_G$  sont ordonnés par une relation d'ordre, alors il est possible d'ordonner tout ensemble d'arbres, ce qui conjointement au théorème de Neveu, suffit à valider la Prop. 1. Au delà de l'existence de cette relation d'ordre, nous cherchons finalement à caractériser cette fonction de trace, d'après les ensembles impliqués dans la récurrence, à savoir  $K_N$ ,  $E_N$  et  $V_{N+1}$ . Cette fonction de trace  $\sigma^T$  serait donc récursive, et basée sur les fonctions d'encodage de noeuds et d'arêtes.

**Lemma 3.** *La fonction de trace  $\sigma^T$  fournit une représentation canonique de tout arbre  $t \in \mathbb{T}$ .*

*Démonstration.* Soit  $\Gamma$  l'opérateur de concaténation prenant deux arguments : un sous-ensemble  $\epsilon$  d'un ensemble  $\mathcal{E}$  ordonné par une relation d'ordre totale  $\leq$ , et un symbole de concaténation arbitraire  $\cdot \in \Sigma$ , tel que défini en Eq. 1.2.13.

$$\begin{aligned} \Gamma : \mathcal{E} \times \Sigma &\rightarrow \Sigma^* \\ \forall \epsilon \subseteq \mathcal{E}, \forall \cdot \in \Sigma, \Gamma(\epsilon, \cdot) &= \bigoplus_{e_i \in \epsilon \mid e_{i-1} \leq e_i} e_i = e_1 \cdot e_2 \cdot \dots \cdot e_{|\epsilon|} \end{aligned} \quad (1.2.13)$$

$$\begin{aligned} \sigma^T : \mathbb{T} &\rightarrow \Sigma^* \\ \sigma^T(t_\nu) &= \Gamma \left( \bigcup_{\lambda \in \Lambda_\nu} \{\sigma^T(\lambda) : \sigma^E(\lambda, \nu)\}, \cdot \right) \cdot \sigma^V(\nu) \end{aligned} \quad (1.2.14)$$

Cette fonction de trace nous permet d'encoder canoniquement l'ensemble des arbres  $\mathbb{T}$ . En effet, deux de ces traces sont uniques si et seulement si tous leurs éléments constitutifs sont rigoureusement égaux, ce qui signifie qu'une permutation de ces ensembles d'éléments n'aurait pas d'impact sur la classe d'isomorphisme de l'arbre considéré, ce qui valide  $\sigma^T$  comme étant solution des Prop. 1 et 2. Ainsi,  $(\mathbb{T}, \leq)$ .

□

Nous illustrons en Fig. 1.1 l'encodage associé au graphe exemple par la fonction de trace  $\sigma^T$ . Par commodité et pour rester proche de la notation de Neveu, nous utilisons la virgule "," comme symbole de concaténation des arbres, au lieu du point utilisé dans la définition ci-dessus en Eq. 1.2.14.

## 1.3 Écriture d'un graphe quelconque en arbre : Scott

Dans cette section, nous cherchons à associer à un graphe quelconque  $G \in \mathbb{G}$  un représentant de sa classe d'isomorphisme sous forme d'arbre  $t \in \mathbb{T}$ , afin de pouvoir appliquer pour tout graphe  $G$  la notation que nous avons introduite à la section précédente.

Formellement, nous cherchons une application  $f : \mathbb{G} \rightarrow \mathbb{T}$  tel que  $f$  soit injective. Nous proposons une solution à ce problème, que nous nommons SCOTT (Structure Canonisation using Ordered Tree Translation).

### 1.3.1 Étapes principales

SCOTT est composé de trois étapes principales :

Étape. 1 Ordonnancement des noeuds en niveaux, étant donnée une racine préalablement déterminée.

Étape. 2 Ré-écriture, de manière réversible, des arêtes induisant des cycles.

Étape. 3 Encodage canonique de l'arbre obtenu.

La figure 1.3 illustre sur un exemple simple les étapes citées ci-dessus. Les sections suivantes détaillent et formalisent chacune de ces étapes, pour aboutir à la forme canonique finale. La grammaire utilisée est décrite en §1.3.5 (Grammaire 1.1).

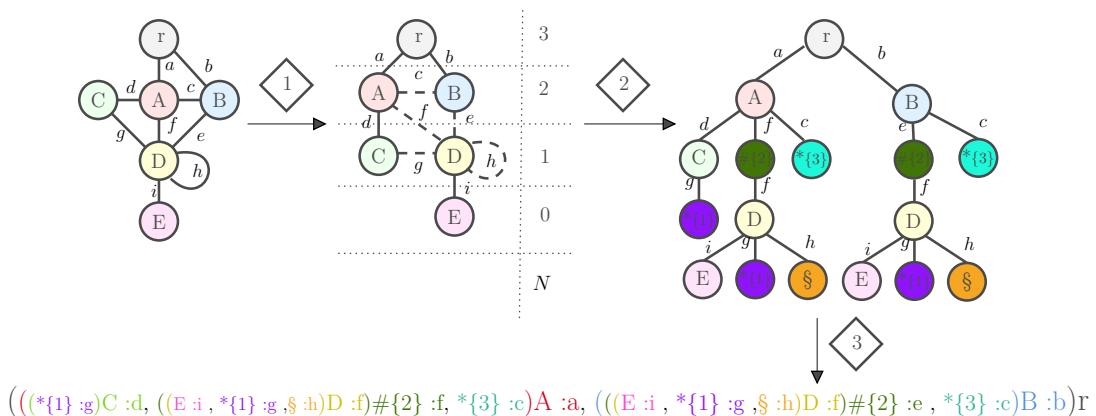


FIGURE 1.3 – SCOTT appliqué à un exemple simple

### 1.3.2 Ordonnancement selon une racine (étape 1)

La première étape consiste à ordonner tous les noeuds de  $g$  selon leur distance minimale (appelé niveau) avec un noeud  $\rho$  désigné comme racine. On suppose pour l'instant que cette racine est connue sans ambiguïté.

L'objectif est de pouvoir définir un ordre dans lequel les cycles seront réécrits, et ainsi rendre le processus déterministe. En effet, en ordonnant les noeuds par niveau, le dernier "étage" du graphe sera forcément relié aux noeuds de l'étage inférieur exclusivement, faute de quoi par définition, ce chemin ne serait pas minimal.

La fonction  $lvl : V \rightarrow \mathbb{N}$  introduite à la section précédente, peut être ré-utilisée ici telle quelle puisque même si le chemin minimal  $\mu(\nu, \rho)$  n'est pas unique, sa longueur  $\|\mu(\nu, \rho)\|$  l'est. On rappelle :

$$\forall \nu \in V, lvl(\nu) = \max_{\nu' \in V} \|\mu(\nu', \rho)\| - \|\mu(\nu, \rho)\|$$

Nous illustrons l'application de cette fonction d'ordonnancement sur un graphe exemple en Fig. 1.4.

### 1.3.3 Transformation en arbre (étape 2)

Considérant l'ordonnancement précédent, nous nous plaçons dans le contexte d'un niveau  $N$ . Nous faisons l'hypothèse que le niveau inférieur  $N + 1$  soit traité, c'est-à-dire que chacun de ces noeuds  $\nu \in V_{N+1}$  est la racine d'un arbre, et peut donc sans perte d'information être résumé à un simple noeud, de label  $\sigma_\nu$ , qui sera ainsi une feuille dans ce contexte. (Prop. 2)

Tous les descendants d'un noeud  $\nu$  peuvent donc être résumés par un unique noeud de label  $\sigma^T(\nu)$ . L'objectif est de traiter chaque étage de façon récursive, dans l'idée d'éditer les sous-graphes correspondants dans chaque contexte (niveau), et ainsi faire disparaître sur l'entièreté du graphe toute forme de cycle et tout chemin multiple entre deux noeuds.

#### Définition d'une production

Ces éditions de graphes peuvent être formalisées sous le terme de productions [60]. Parmi les différents formalismes proposés, celui que nous utiliserons ici repose sur le double pushout (DPO), dans lequel nous définissons une production  $p : L \xleftarrow{l} K \xrightarrow{r} R$  comme une paire de morphismes de

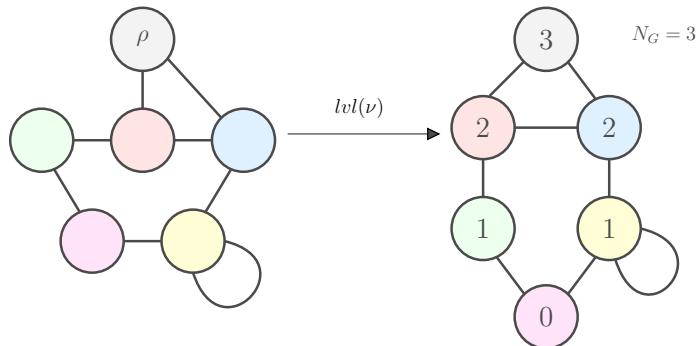


FIGURE 1.4 – Ordénancement des noeuds en niveau d'après une racine

graphes  $l : K \rightarrow L$  et  $r : K \rightarrow R$ , tous deux injectifs. Ces trois graphes  $L$ ,  $R$  et  $K$  représentent respectivement la partie gauche (LHS), la partie droite (RHS) et l'interface de cette production. Cette interface  $K$  représente les éléments non affectés par cette production, tandis que les éléments (noeuds et arêtes)  $L - K$  seront supprimés et les éléments  $R - K$  ajoutés. La mise en application d'une de ces productions sur un graphe  $G$  se fait par le biais d'une fonction bijective dite de matching  $m : V_G \rightarrow V_{LHS}$  associée à un système d'assertions et de prédictats logiques  $S_m$  permettant d'associer à tout noeud  $\nu \in V_G$  le "rôle" (son noeud correspondant dans  $V_{LHS}$ ) qu'il tiendra dans l'application de cette production.

## Écriture matricielle

Nous pouvons reprendre cette définition sous la forme de logique booléenne, plus précisément en représentant chacun de ces graphes par une matrice d'adjacence

$$G_{i,j}^E = \begin{cases} 1 & \text{si } (i, j) \text{ sont reliés} \\ 0 & \text{sinon} \end{cases} \quad \text{et une matrice de noeuds}^3 G_i^V = \begin{cases} 1 & \text{si } i \text{ existe} \\ 0 & \text{sinon} \end{cases}.$$

Ces deux matrices pourraient être regroupées dans un même tenseur, mais nous préférions pour gagner en clarté les garder séparés, et définir toutes les matrices suivantes à la fois pour les arêtes et pour les noeuds. On décrit à l'aide de l'algèbre logique les matrices  $l$  et  $r$ , respectivement de deletion et d'addition<sup>4</sup> :

$$l = L \wedge \bar{R} = \bar{R} \wedge L \tag{1.3.1}$$

$$r = R \wedge \bar{L} = \bar{L} \wedge R \tag{1.3.2}$$

Ces matrices nous permettent de décrire dynamiquement une production, et de pouvoir l'exprimer comme une opération matricielle déterministe pour passer d'un membre à l'autre, et surtout décrit en l'état une opération réversible :

$$R = r \vee (\bar{l} \wedge L) \tag{1.3.3}$$

$$L = l \vee (\bar{r} \wedge R) \tag{1.3.4}$$

$$K = L \wedge R \tag{1.3.5}$$

## Extension aux graphes labellisés

Dans la définition précédente, il n'est pas question d'arêtes labellisées, elles ne transportent donc pas plus d'information que leur propre existence, d'où le fait qu'elles puissent être représentées par des valeurs booléennes. Nous souhaiterions ici pouvoir traiter le cas général d'arêtes non-homogènes (i.e. colorées), portant donc des labels.

À partir de  $\mathcal{L} = \{a, b, \dots\}$  l'ensemble des labels applicables à tout noeud ou arête d'un graphe, et  $\bar{\mathcal{L}} = \{\bar{a}, \bar{b}, \dots\}$  l'ensemble de ces labels conjugués, définissons le système d'axiomes suivants, dérivé de la logique du premier ordre, où ? désigne une forme indéterminée :

3. Cette dernière peut sembler superflue, mais il ne faut pas oublier que des noeuds peuvent être supprimés.

4. chacune de ces matrices est définie à la fois pour les arêtes et pour les noeuds

$$\wedge : (\mathcal{L} \cup \overline{\mathcal{L}} \cup \{0, 1\}) \times (\mathcal{L} \cup \overline{\mathcal{L}} \cup \{0, 1\}) \rightarrow (\mathcal{L} \cup \overline{\mathcal{L}} \cup \{0, 1\}), \forall a, b \in \mathcal{L},$$

$$\begin{array}{ll} a \wedge 0 = 0 & 0 \wedge 0 = 0 \\ a \wedge 1 = a & 0 \wedge 1 = 0 \\ a \wedge \bar{a} = 0 & 1 \wedge 0 = 0 \\ a \wedge b = ? & 1 \wedge 1 = 1 \end{array}$$

$$\vee : (\mathcal{L} \cup \overline{\mathcal{L}} \cup \{0, 1\}) \times (\mathcal{L} \cup \overline{\mathcal{L}} \cup \{0, 1\}) \rightarrow (\mathcal{L} \cup \overline{\mathcal{L}} \cup \{0, 1\}), \forall a, b \in \mathcal{L},$$

$$\begin{array}{ll} a \vee 0 = a & 0 \vee 0 = 0 \\ a \vee 1 = ? & 0 \vee 1 = 1 \\ a \vee \bar{a} = ? & 1 \vee 0 = 1 \\ a \vee b = ? & 1 \vee 1 = 1 \end{array}$$

### Triplet de production solution

Sur un graphe dont les noeuds ont été ordonnancés en niveaux, seules trois configurations d'arêtes (schématisées en Fig. 1.5 par leurs exemples minimaux) aboutissent à la présence de cycles, qui peuvent être résolues respectivement par autant de productions associées :

- $p_c$  co-bound, une arête d'un noeud vers un noeud distinct appartenant au même niveau  $N$  (e.g les arêtes  $c$  et  $g$  en Fig. 1.3)
- $p_i$  in-bound, deux noeuds d'un même niveau  $N$  reliés à un descendant commun au niveau  $N - 1$  (e.g les arêtes  $f$  et  $e$  en Fig. 1.3)
- $p_s$  self-bound, une arête d'un noeud vers lui-même (e.g l'arête  $h$  en Fig. 1.3)

Nous détaillons ci-après ces trois productions  $p_c$ ,  $p_i$  et  $p_s$ . Nous donnons pour chacune d'elles leurs matrices représentatives des  $LHS$  ( $L^N, L^E$ ) et  $RHS$  ( $R^N, R^E$ ), et à titre indicatif, nous calculons dans le premier cas les matrices de deletion  $l^N$  et  $l^E$ , ainsi que les matrices d'addition  $r^N$  et  $r^E$ .

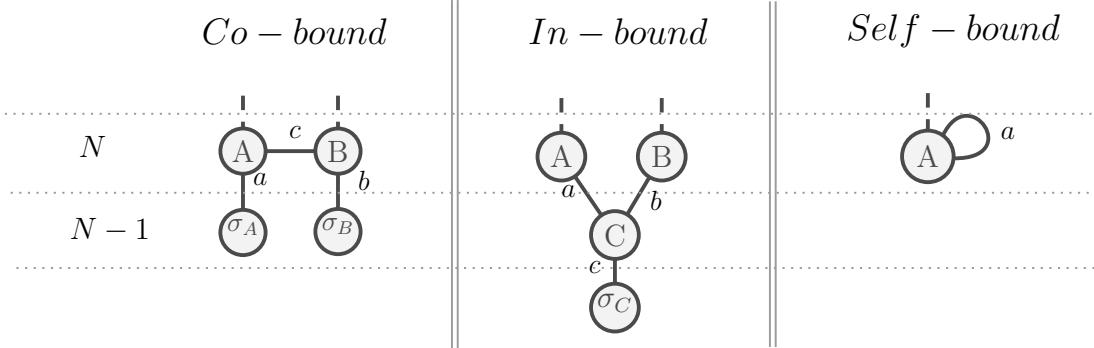
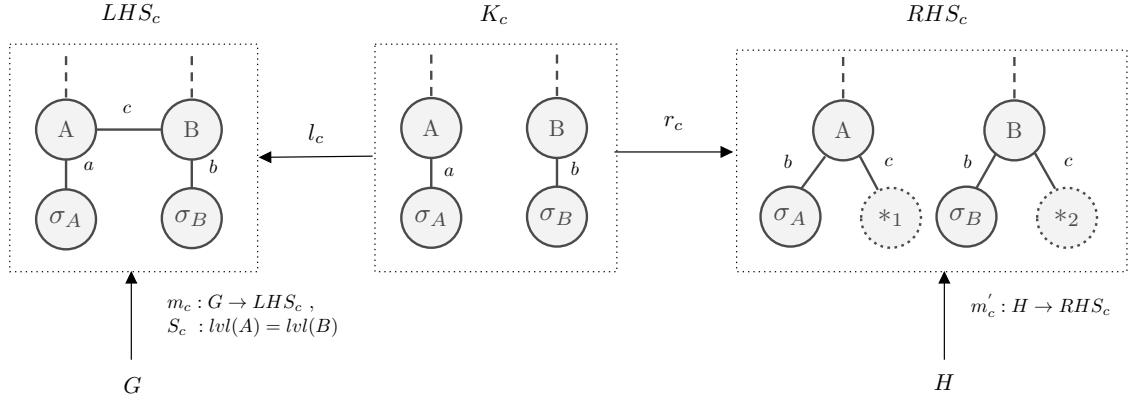


FIGURE 1.5 – Les différentes configurations aboutissant à des cycles dans un graphe

**Co-bound :  $p_c$** 

Dans le contexte cible d'un arbre enraciné, les noeuds  $A$  et  $B$  ont un ancêtre en commun, qui peut être  $\rho$  lui-même. Supposons que ces chemins  $\mu(A, \rho)$  et  $\mu(B, \rho)$  soient directs (on néglige les noeuds intermédiaires), autrement dit on considère que le niveau supérieur est uniquement le noeud racine  $\rho$ .



$$L^N = \begin{bmatrix} A & r \\ B & r \\ \sigma_A & r \\ \sigma_B & r \\ *_1 & 0 \\ *_2 & 0 \end{bmatrix}, \quad L^E = \begin{bmatrix} 0 & \mathbf{c} & \mathbf{a} & 0 & 0 & 0 \\ \mathbf{c} & 0 & 0 & \mathbf{b} & 0 & 0 \\ \mathbf{a} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{b} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad R^N = \begin{bmatrix} r \\ r \\ r \\ v \\ v \end{bmatrix}, \quad R^E = \begin{bmatrix} 0 & 0 & \mathbf{a} & 0 & \mathbf{c} & 0 \\ 0 & 0 & 0 & \mathbf{b} & 0 & \mathbf{c} \\ \mathbf{a} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{b} & 0 & 0 & 0 & 0 \\ \mathbf{c} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{c} & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\overline{R^N} \wedge L^N = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \overline{R^E} \wedge L^E = \begin{bmatrix} 0 & \mathbf{c} & 0 & 0 & 0 & 0 \\ \mathbf{c} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \overline{L^N} \wedge R^E = \begin{bmatrix} 0 \\ 0 \\ 0 \\ v \\ v \end{bmatrix}, \quad \overline{L^E} \wedge R^E = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{c} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{c} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{c} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{c} & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dans  $LHS_c$ , il existe donc une chaîne (série de noeuds connectés)  $\langle \rho, \dots, A, B, \dots, \rho \rangle$ , formant un cycle.  $LHS_c$  n'est donc pas un arbre, tout comme tout graphe  $g \in \mathbb{G}$ , satisfaisant au moins une fois la relation de matching  $m_c$ .

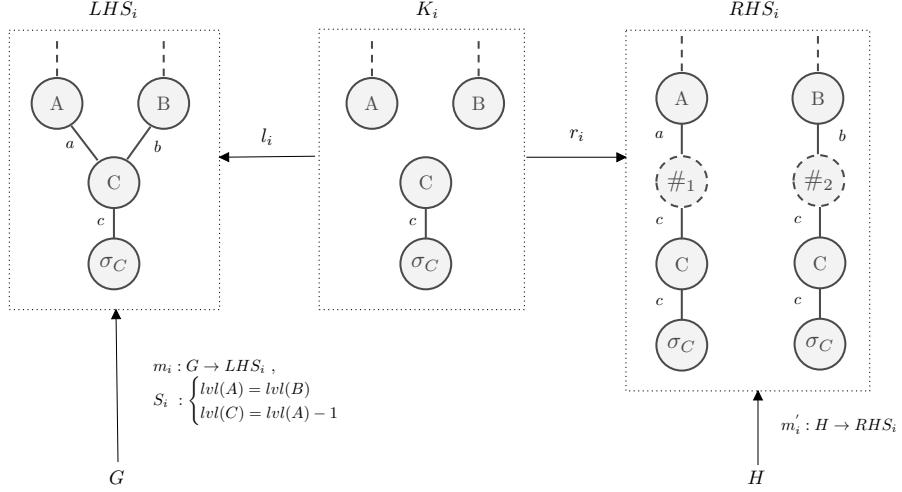
En revanche, sous cette hypothèse,  $RHS_c$  est un arbre. Plus particulièrement, l'application de cette production  $p_c$  supprime les cycles élémentaires présents au niveau  $lvl(A)$ .

**In-bound :  $p_i$** 

Là encore, nous sommes dans la même configuration de graphe cyclique que dans l'exemple précédent, car il existe une chaîne  $\langle \rho, \dots, A, C, B, \dots, r \rangle$ . De plus, si  $a = b$ , il n'existe pas un unique chemin minimal  $\mu(C, \rho)$ .

$LHS_i$  n'est donc pas un arbre, ainsi que tout graphe  $g \in \mathbb{G}$  satisfaisant au moins une fois la relation de matching  $m_i$ .

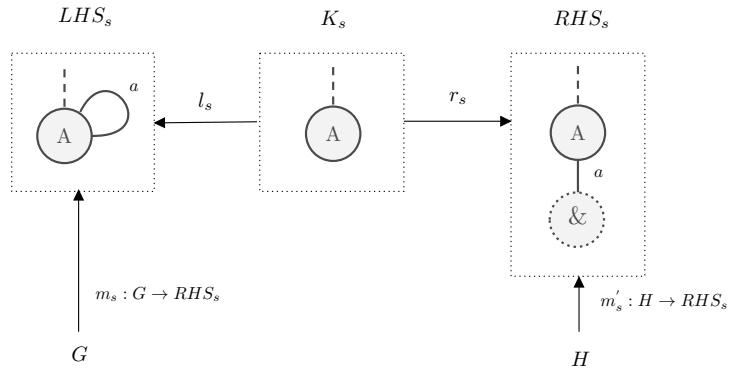
### 1.3. Écriture d'un graphe quelconque en arbre : SCOTT



$$L^N = \begin{bmatrix} A & r \\ B & r \\ C & r \\ \sigma_C & r \\ \#_1 & 0 \\ \#_2 & 0 \\ C' & 0 \\ \sigma'_C & 0 \end{bmatrix}, L^E = \begin{bmatrix} 0 & 0 & \mathbf{a} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{b} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{a} & \mathbf{b} & 0 & \mathbf{c} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{c} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, R^N = \begin{bmatrix} r \\ r \\ r \\ r \\ m \\ m \\ r \\ r \end{bmatrix}, R^E = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{a} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{b} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{c} & \mathbf{a} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{c} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{a} & 0 & \mathbf{a} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{b} & 0 & 0 & 0 & 0 & \mathbf{b} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{b} & 0 & \mathbf{c} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c} & 0 \end{bmatrix}$$

En revanche, sous cette hypothèse, RHS est un arbre. Plus particulièrement, l'application de cette production  $p_i$  supprime les cycles non-élémentaires présents au niveau  $lvl(A)$ , et lève la non-unicité des chemins menant à  $\rho$  (et donc tout autre noeud).

**Self-bound :**  $p_s$



$$L^N = \begin{bmatrix} A & r \\ \& & 0 \end{bmatrix}, L^E = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}, R^N = \begin{bmatrix} r \\ s \end{bmatrix}, R^E = \begin{bmatrix} 0 & a \\ a & 0 \end{bmatrix}$$

Cette configuration est triviale, et permet de réécrire tout graphe comme un graphe simple.

Les productions décrites vérifient toutes (1.3.3) (1.3.4) et (1.3.5), elles peuvent donc être décrites dynamiquement grâce aux matrices d'addition  $r$  et de deletion  $l$ , et appliquées dans le contexte d'un graphe, dès lors que l'on dispose d'un matching  $m : G \rightarrow LHS_m$ .

## Ordonnancement des productions

Les productions que nous venons de décrire permettent de ré-écrire toute forme de cycle dans un graphe. Leur application est toutefois effectuée suivant un ordonnancement précis, par niveau et par catégorie, formant un morphisme général  $f_G$  tel que :

$$f_G = f_{N_G} \circ \dots \circ f_0 \quad (1.3.6)$$

$$\forall N \in [0, N_G], f_N = p_i^N \circ p_c^N \circ p_s^N \quad (1.3.7)$$

$$\forall x \in \{i, c, s\}, k = \Omega(\{p_k^N\}), p_x^N = p_{x_k}^N \circ \dots \circ p_{x_1}^N \quad (1.3.8)$$

avec  $\Omega(\cdot)$  le cardinal d'un ensemble, et  $\{p_k^N\}$  l'ensemble des productions individuelles composant le morphisme  $p_x^n$  associé au niveau  $N$  et une catégorie de production  $x$ .

Ces ensembles  $\{p_x^N\}$  sont construits en déterminant tous les antécédents possibles aux fonctions de matching associés aux productions de catégorie  $k$ , pour un niveau  $N$  donné.

Soit  $G_\rho = (V_G, E_G)$ , un graphe  $G$  muni d'un noeud  $\rho \in V$  identifié comme racine.

On rappelle  $V_N \subseteq V_G$ ,  $\nu \in V_N \Leftrightarrow lvl(\nu) = N$ , l'ensemble des noeuds de  $G_\rho$  de niveau  $N$ .

Soit  $W_N \supseteq V_N$ ,  $W_N = \bigcup_{k=0}^N V_k$ ,  $\nu \in W_N \Leftrightarrow lvl(\nu) \leq N$ , l'ensemble des noeuds de  $G$  d'un niveau inférieur ou égal à  $N$ .

Soit  $G_{\rho, N} = (W_N, E_N \in (W_N \times W_N)) \subseteq G_\rho$  le sous-graphe de  $G_\rho$  ne comportant que les noeuds  $W_N$  d'un niveau inférieur ou égal à  $N$  et les arêtes  $E_N$  associant exclusivement deux éléments de  $W_N$ .

Enfin, nous notons  $G_\rho^N = f_N(G_\rho)$  le graphe  $G_\rho$  traité jusqu'au niveau  $N$  par  $f_N \circ \dots \circ f_0$ . On allège la syntaxe d'un sous-graphe entièrement traité  $G_{\rho, N}^N = G_\rho^*$ .

Nous construisons l'ensemble des productions  $p_x^N$  en appliquant autant de fois que possible la fonction de matching  $m_x$  associée à cette catégorie de production  $x$  dans le contexte de  $G_{\rho, N}$ .

$$\{p_k^N\} = arg(G_{\rho, N} \rightarrow m_k : LHS_k) \quad (1.3.9)$$

Ces productions peuvent être caractérisées par la trace des arbres associés aux noeuds auxquels elles s'appliquent ( $\{\sigma_A\}$  pour les self-bounds,  $\{\sigma_A, \sigma_B\}$  pour les co-bounds et les in-bounds) ainsi que les modalités d'arêtes impliquées, induisant un ordre sur ces productions.

$$(\mathbb{T}, \leq) \Rightarrow (\{p_k^N\}, \leq) \quad (1.3.10)$$

Nous illustrons en Fig.1.6 un exemple d'ordonnancement de productions sur un exemple simple. L'application de ces productions suivant cet ordonnancement permet d'assurer à l'algorithme d'aboutir à un arbre unique à toute classe d'isomorphisme, ce que nous prouvons ci-après.

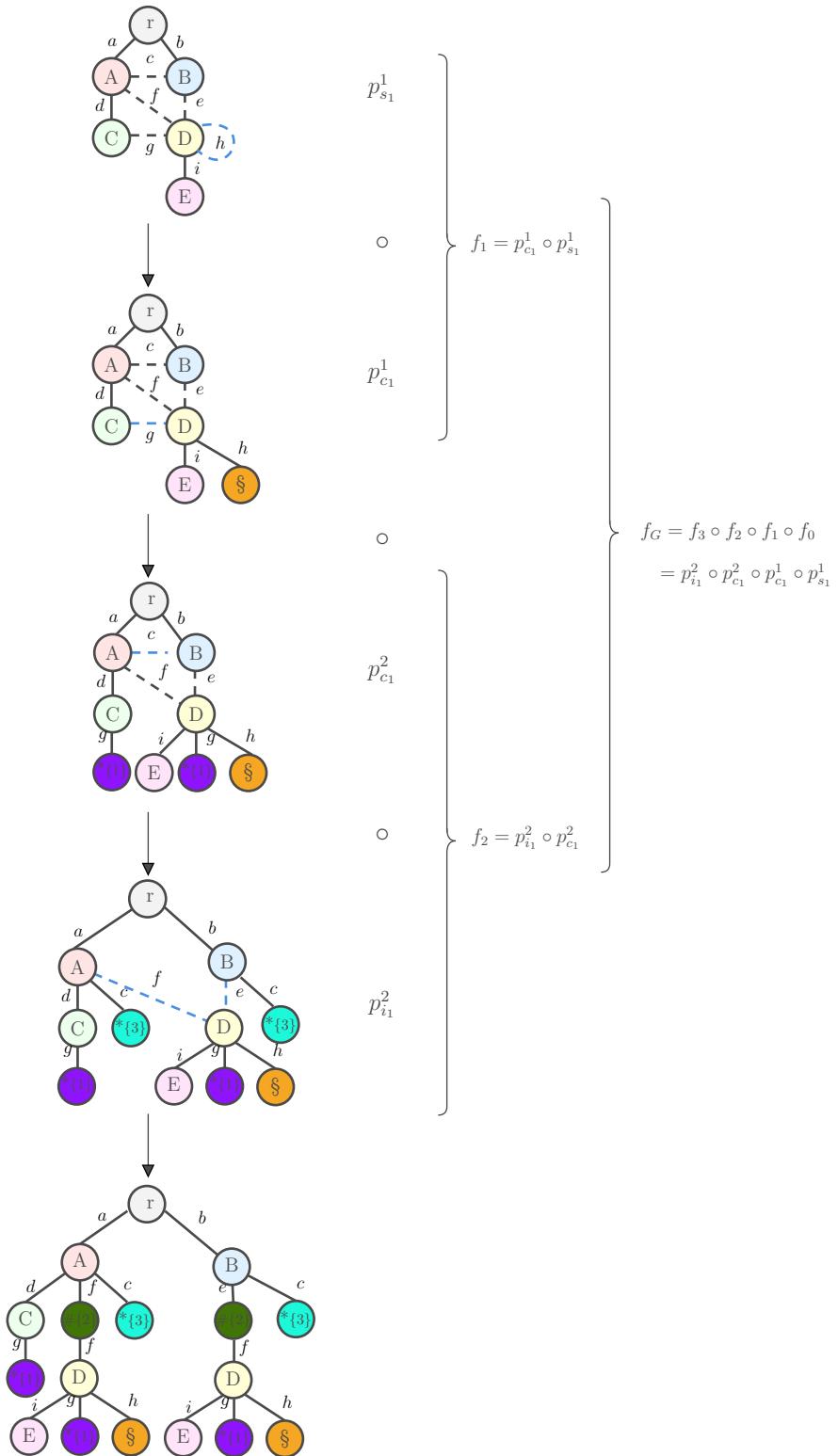
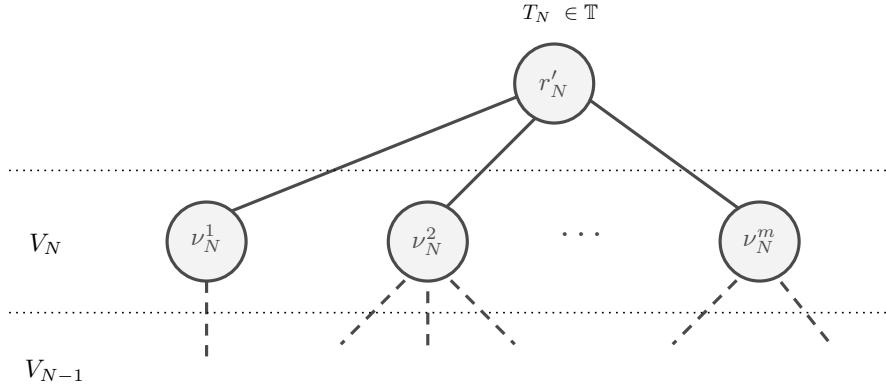


FIGURE 1.6 – Ordonnancement de productions sur un exemple simple

### Espace d'arrivée de $f_G$

**Lemma 4.** Le morphisme général  $f$  aboutit à un arbre.  $\forall g \in \mathbb{G}, f(g) \in \mathbb{T}$

*Démonstration.* Nous formulons la proposition suivante :



$\mathcal{P}_N$

Dans le contexte de  $G_{\rho, N}^*$ , sous-graphe jusqu'au niveau  $N$ , dans lequel il n'est plus possible d'appliquer une quelconque production  $p_s^N$ ,  $p_i^N$  ou  $p_c^N$ , si un noeud  $\rho'_N$  connecte tous les noeuds  $\nu_N \in V_N$ , alors le graphe obtenu  $T_N$  est un arbre de racine  $\rho'_N$ .

$$T_N = \left( W_N \cup \rho'_N, E_N \cup \bigcup_{\nu_i \in V_N} (\rho'_N, \nu_i) \right), T_N \in \mathbb{T}$$

En d'autres termes, on affirme que pour un rang  $N \in [0, N_G]$ , l'application de toutes les productions possibles sur  $G_{\rho, N}$  aboutit au fait que tous les noeuds  $\nu_N \in V_N$  soient des arbres.

$\mathcal{P}_0$

Les noeuds appartenant à  $V_0$  sont des feuilles.

Si un noeud  $\rho'_0$  connecte ces feuilles, le graphe résultant est par définition un arbre de racine  $\rho'_0$ .

$$\mathcal{P}_0 \text{ est vraie} \quad (1.3.11)$$

$\mathcal{P}_{N+1}$

Supposons que  $\mathcal{P}_N$  soit vraie. (Hypothèse d'induction)

$P_N \Leftrightarrow \forall \nu_N \in V_N, \nu_N$  est racine d'un arbre (potentiellement nul) composé de tous ses descendants des niveaux inférieurs. Tout  $\nu_N$  peut donc être remplacé sans perte d'information par  $\sigma^T(t_{\nu_N})$  conformément à la section précédente, et deviennent donc des feuilles.

Cette transformation faite, nous pouvons remonter au contexte des noeuds  $V_{N+1}$  et appliquer chacune des productions autant de fois que l'on observe de matchings, pour obtenir  $G_{\rho, N+1}^*$ .

$$\begin{aligned} \nexists m_c : G_{\rho, N+1}^* \rightarrow LHS_c &\Leftrightarrow \nexists < \rho'_{N+1}, A, B, \rho'_{N+1} > \\ \nexists m_i : G_{\rho, N+1}^* \rightarrow LHS_i &\Leftrightarrow \nexists < \rho'_{N+1}, A, C, B, \rho'_{N+1} > \\ \nexists m_s : G_{\rho, N+1}^* \rightarrow LHS_s &\Leftrightarrow \nexists < \rho'_{N+1}, A, A, \rho'_{N+1} > \end{aligned}$$

### 1.3. Écriture d'un graphe quelconque en arbre : SCOTT

---

Il n'existe donc pas de cycle dans  $G_{\rho, N+1}^*$ , ce que l'on peut constater par construction, car dans toutes les productions décrites, si un noeud  $\rho'$  est le parent direct de A et B (ou simplement A dans le cas du self-bound), alors ce noeud  $\rho'$  est racine d'un arbre.

$$\begin{aligned} \mathcal{P}_{N+1} &\text{ est vraie} \\ (1.3.11), (1.3.3) \Rightarrow \mathcal{P}_N &\text{ est vraie} \end{aligned}$$

Enfin pour  $N = lvl(\rho)$ ,  $\rho'_N = \rho$ , et donc  $T_N = G_\rho$ .

□

#### Injectivité de $f_G$

**Lemma 5.** *L'arbre produit par f est unique pour toute classe d'isomorphisme.*

*Notations et rappels.*

$$\begin{aligned} f_G &= f_{N_G} \circ \dots \circ f_0 \\ \forall N \in [0, N_G], f_N &= p_i^N \circ p_c^N \circ p_s^N \\ \forall x \in \{i, c, s\}, k &= \Omega(\{p_k^N\}), p_x^N = p_{x_k}^N \circ \dots \circ p_{x_1}^N \end{aligned}$$

Avec  $\forall p_x^i \in p_x^N$ ,  $p_x^i : L \xleftarrow{l} K \xrightarrow{r} R$  désigne une production (bijective) à appliquer sur le graphe  $G$ , muni d'une fonction de matching associée  $m_x^i : V_G \rightarrow LHS_m$  respectant un système de conditions  $S_x$ . Les ensembles respectifs de tous ces éléments sur un graphe  $G$  sont notés  $\{p_G\}$  et  $\{m_G\}$ .

On rappelle que  $G \simeq H$  si et seulement si il existe au moins une bijection de  $V_G$  dans  $V_H$  conservant la structure et les labels :

$$G \simeq H \Leftrightarrow \exists \pi : V_G \rightarrow V_H, \forall \nu_1, \nu_2 \in V_G, \begin{array}{l} L_G^E(\nu_1, \nu_2) = L_H^E(\pi(\nu_1), \pi(\nu_2)) \\ \wedge \quad L_G^V(\nu_1) = L_H^V(\pi(\nu_1)) \\ \wedge \quad L_G^V(\nu_2) = L_H^V(\pi(\nu_2)) \end{array}$$

On note une telle bijection  $\pi : V_G \leftrightarrow V_H$ . On s'autorise le raccourci syntaxique  $\pi(V_G) = V_H$ .

$$G \simeq H \Leftrightarrow \exists \pi : V_G \leftrightarrow V_H$$

Nous en déduisons la contraposée  $G \not\simeq H$  :

$$\begin{aligned} G \not\simeq H &= \overline{G \simeq H} \\ G \not\simeq H &\Leftrightarrow \forall \pi : V_G \rightarrow V_H, \exists \nu_1, \nu_2 \in V_G, \begin{array}{l} L_G^E(\nu_1, \nu_2) \neq L_H^E(\pi(\nu_1), \pi(\nu_2)) \\ \vee \quad L_G^V(\nu_1) \neq L_H^V(\pi(\nu_1)) \\ \vee \quad L_G^V(\nu_2) \neq L_H^V(\pi(\nu_2)) \end{array} \end{aligned}$$

Nous notons de même  $\pi : V_G \leftrightarrow V_H$ ,  $\pi(V_G) \neq V_H$ .

$$G \not\simeq H \Leftrightarrow \forall \pi : V_G \leftrightarrow V_H$$

Démonstration.

$$\begin{cases} G \simeq H \Leftrightarrow f_G(G) \simeq f_H(H) \\ G \not\simeq H \Leftrightarrow f_G(G) \not\simeq f_H(H) \end{cases} \Leftrightarrow \begin{cases} G \simeq H \Rightarrow f_G(G) \simeq f_H(H) & (\mathcal{P}_A) \\ G \not\simeq H \Rightarrow f_G(G) \not\simeq f_H(H) & (\mathcal{P}_B) \end{cases}$$

On reformule ainsi le problème sous forme de deux propositions ( $\mathcal{P}_A$ ) et ( $\mathcal{P}_B$ ).

$$(1) : G \simeq H \Leftrightarrow \forall g = (V_g, E_g) \subseteq G, \exists h = (V_h, E_h) \subseteq H, \exists \pi : V_g \leftrightarrow V_h$$

$$\overline{(1)} : G \not\simeq H \Leftrightarrow \exists g = (V_g, E_g) \subseteq G, \forall h = (V_h, E_h) \subseteq H, \forall \pi : V_g \leftrightarrow V_h$$

On rappelle que  $\{m_G\}$  désigne l'ensemble des fonctions de matchings associées à des productions qu'il est possible d'appliquer par  $G$ , autrement dit générées par  $G$ , tel que :

$$\begin{aligned} m_c : V_G &\rightarrow V_{LHS_c}, S_c \\ \{m_G\} = \bigcup m_i : V_G &\rightarrow V_{LHS_i}, S_i \\ m_s : V_G &\rightarrow V_{LHS_s}, S_s \end{aligned}$$

Pour chacune de ces fonctions de matching  $m^j \in \{m_G\}$ , un unique sous-graphe  $g^j \subseteq G$  est sélectionné. Formellement,  $m^j(V_{g^j}) = V_{LHS^j}$ ,  $p^j(LHS^j) = RHS^j$ .

$$(2) : \exists \pi : V_G \leftrightarrow V_H \underset{g \subseteq G, h \subseteq H}{\xrightarrow{\text{(1)}}} \forall m^j \in \{m_G\}, \exists m^{j'} \in \{m_H\}, h^{j'} = \pi(g^j)$$

$$\overline{(2)} : \exists m^j \in \{m_G\}, \forall m^{j'} \in \{m_H\}, h^{j'} \neq \pi(g^j) \underset{g \subseteq G, h \subseteq H}{\xrightarrow{\text{(1)}}} \forall \pi : V_G \leftrightarrow V_H$$

$$(3) \{m_G\} = \{m_H\} \Leftrightarrow \forall m (m \in \{m_G\} \Leftrightarrow m \in \{m_H\})$$

$$\overline{(3)} \{m_G\} \neq \{m_H\} \Leftrightarrow \exists m \in \{m_G\}, m \notin \{m_H\} \vee \exists m \in \{m_H\}, m \notin \{m_G\}$$

$$(4) : G \simeq H \xrightarrow{\text{(2), (3)}} \{m_G\} = \{m_H\}$$

$$\overline{(4)} : \{m_G\} \neq \{m_H\} \xrightarrow{\text{(2), (3)}} G \not\simeq H$$

Les ensembles des fonctions de matching sont donc égales si  $G \simeq H$ . Il n'est cependant pas possible de conclure que ces ensembles sont égaux seulement si  $G \simeq H$ . On ne peut également pas affirmer que si  $G \not\simeq H$ , alors  $\{m_G\} \neq \{m_H\}$ , bien que la réciproque soit vraie.

Cela ne suffit malheureusement pas non plus à prouver ( $\mathcal{P}_A$ ), car avec  $f_1$  et  $f_2$  deux applications, en général  $f_1 \circ f_2 \neq f_2 \circ f_1$ . Au delà de l'égalité de ces ensembles, nous allons à présent prouver que l'on peut les appliquer suivant un ordonnancement aboutissant à un résultat invariant à la classe

d'isomorphisme du graphe.

Les productions formant une fonction  $f$  sont ordonnées selon un ordre non-strict :

$$(5) : \begin{cases} ([0, N_G], <) \Rightarrow (\{f_G\}, <) \\ (\{s, c, i\}, <) \Rightarrow \forall f_N \in f_G (\{f_N\}, <) \\ \forall p^j \in p_x^N, \left( \bigcup_{\nu \in g^j} \sigma^T(t_\nu), \leq \right) \Rightarrow (\{p_x^N\}, \leq) \end{cases} \Rightarrow \left( \bigcup_{m^j \in \{m_G\}} m^j, \leq \right)$$

Un ensemble  $\{m_G\}$  est donc générateur d'une séquence ordonnée de  $K$  fonctions de matching,

$$\{m_G\} \equiv < m_G^1, \dots, m_G^K >, \forall k \in [1, K-1], m_G^k \leq m_G^{k+1}$$

Il ne s'agit toutefois que d'un ordre partiel, aussi il peut y avoir des égalités, notées  $m^a \equiv m^b$  :

$$(6) : \begin{aligned} m^a, m^b \in \{m_G\}, m^a \equiv m^b &\Leftrightarrow m^a \leq m^b \wedge m^b \leq m^a \\ m^a \equiv m^b &\Leftrightarrow \bigcup_{\nu \in g^a} \sigma^T(t_\nu) = \bigcup_{\nu \in g^b} \sigma^T(t_\nu) \Leftrightarrow g^a \simeq g^b \end{aligned}$$

Cet ensemble de productions est stable, car aucune application de production ne peut rajouter ou supprimer une autre production que celle qui est traitée (car il est impossible d'appliquer une quelconque fonction de matching dans une RHS).

$$(7) : \begin{aligned} G \simeq H &\stackrel{(4), (5)}{\Rightarrow} \stackrel{(6)}{\Rightarrow} \forall k \in [1, K], m_G^k \equiv m_H^k \\ G \simeq H &\Rightarrow \forall k \in [1, K], g^k \simeq h^k \end{aligned}$$

On note  $\overset{k}{G}$  le graphe  $G$  auquel on a appliqué les  $k$  premières productions.

Étudions les conditions d'isomorphisme entre  $\overset{k}{G}$  et  $\overset{k}{H}$  :

$$\forall k \in [1, K], \overset{k}{G} \simeq \overset{k}{H} \Leftrightarrow p_G^{k-1} \left( \overset{k-1}{G} \right) \simeq p_H^{k-1} \left( \overset{k-1}{H} \right)$$

On rappelle que les productions sont déterministes, et sont donc des bijections. On souligne ici leurs propriétés injectives :

$$\begin{aligned} \forall p_G \in \{p_G\}, \forall G_1, G_2 \in \mathbb{G}, p_G(G_1) \simeq p_G(G_2) &\Rightarrow G_1 \simeq G_2 \\ \forall p_H \in \{p_H\}, \forall H_1, H_2 \in \mathbb{G}, p_H(H_1) \simeq p_H(H_2) &\Rightarrow H_1 \simeq H_2 \end{aligned}$$

Du fait de ces propriétés, si deux applications de productions aboutissent au même résultat, alors les opérandes (les graphes intermédiaires) et les opérateurs (les productions) sont respectivement

isomorphes et identiques :

$$p_G^{k-1} \binom{k-1}{G} \simeq p_H^{k-1} \binom{k-1}{H} \Leftrightarrow (p_G^{k-1} = p_H^{k-1}) \wedge \binom{k-1}{G} \simeq \binom{k-1}{H}$$

Nous réécrivons à l'aide de la proposition  $P_k \Leftrightarrow \overset{k}{G} \simeq \overset{k}{H}$  :

$$(8) : P_k \Leftrightarrow (m_G^{k-1} \equiv m_H^{k-1}) \wedge P_{k-1}$$

On identifie en particulier,

$$P_K \Leftrightarrow f_G(G) \simeq f_H(H)$$

Supposons que  $G \simeq H$ , on peut prouver  $(\mathcal{P}_A)$  par récurrence.

Il est immédiat que  $G \simeq H \implies \overset{0}{G} \simeq \overset{0}{H}$  car  $G \simeq H \equiv \overset{0}{G} \simeq \overset{0}{H}$ .

$$(9) : \begin{cases} G \simeq H \xrightarrow{(7)} \forall k \in [1, K], m_G^k \equiv m_H^k \implies P_1 \text{ est vraie} \\ \overset{0}{G} \simeq \overset{0}{H} \\ P_1 \end{cases}$$

$$G \simeq H \xrightarrow{(9)} P_K \text{ est vraie}$$

$$(\mathcal{P}_A) : G \simeq H \implies f_G(G) \simeq f_H(H)$$

Supposons que  $G \not\simeq H$ , on peut prouver  $(\mathcal{P}_B)$  par invalidation directe de l'initialisation  $P_1$ . Par hypothèse,  $G \not\simeq H \equiv \overset{0}{G} \not\simeq \overset{0}{H}$ .

$$G \not\simeq H \equiv \overset{0}{G} \not\simeq \overset{0}{H} \implies P_1 \text{ est fausse}$$

$$(10) : \bar{P}_1 \xrightarrow{(8)} \forall k \in [2, K], P_k \text{ est fausse}$$

$$G \not\simeq H \xrightarrow{(10)} P_K \text{ est fausse}$$

$$(\mathcal{P}_B) : G \not\simeq H \implies f_G(G) \not\simeq f_H(H)$$

□

Par ailleurs  $\overset{K}{G} \in \mathbb{T}, \overset{K}{G} \in \mathbb{T}$  d'après le Lemme 4, et la trace de ces deux arbres par  $\sigma^T$  sera bien identique d'après le Lemme 3.

$$\sigma^T \binom{K}{G} = \sigma^T \binom{K}{\overset{K}{G}}$$

### 1.3.4 Désignation d'une racine et encodage (étape 3)

Nous résumons cette section par la définition d'une fonction de trace applicable à tout graphe doté d'une racine.

$$\begin{aligned}\sigma^{G_\rho} &: G_\rho \rightarrow \Sigma^*, \\ \sigma^{G_\rho} &: G_\rho \mapsto \sigma^T(f(G_\rho))\end{aligned}$$

On supposait jusqu'ici que l'identité de la racine  $\rho$  était connue. Pour désigner quelle est cette racine parmi tous les noeuds, nous pouvons simplement prendre celui générant une trace minimum, car tout ensemble muni d'un ordre total admet un élément minimum. Tout graphe est donc encodé par son encodage minimum.

$$\rho_G = \arg \min_{\nu \in V_G} (\sigma^{G_\rho}(G_\nu))$$

L'identité de la racine  $\rho_G$  pour tout graphe  $G$  étant à présent un invariant, le morphisme  $f$  est bijectif, qui jusqu'ici reposait sur l'hypothèse que  $\rho$  soit identifiée. Nous en déduisons une fonction de trace injective et applicable à tout graphe, utilisable comme une fonction de hash.

$$\begin{aligned}\sigma^G &: G \rightarrow \Sigma^*, \\ \sigma^G &: G \mapsto \sigma^T(f(G_{\rho_G}))\end{aligned}$$

### 1.3.5 Grammaire

```

⟨tree⟩ ::= [ ⟨descendant_list⟩ ] ⟨node⟩ : ⟨edge_modality⟩
⟨descendant_list⟩ ::= ( ⟨tree⟩ { , ⟨tree⟩ } )
⟨node⟩ ::= ⟨node_label⟩ | ⟨mirror_node_label⟩ | ⟨virtual_node_label⟩ | ⟨self_node_label⟩
⟨node_label⟩ ::= ⟨label⟩
⟨mirror_node_label⟩ ::= # ⟨magnet⟩
⟨virtual_node_label⟩ ::= * ⟨magnet⟩
⟨self_node_label⟩ ::= §
⟨magnet⟩ ::= {⟨label⟩}
⟨edge_modality⟩ ::= ⟨label⟩
⟨label⟩ ::= Σ*

```

Grammar 1.1 – Grammaire SCOTT supportant  $\sigma^T$ , dérivée de [61]

## 1.4 Analyse de complexité temporelle

Dans cette section, nous évaluons la complexité temporelle de l'algorithme d'encodage canonique décrit, identifié à la fonction  $\sigma^G$ , et son comportement asymptotique suivant  $n = ||V_G||$ .

Nous décrivons cette complexité avec une borne supérieure  $O(\sigma^G)$  correspondant au pire des cas possibles, et une borne inférieure  $\Omega(\sigma^G)$  correspondant au meilleur des cas possibles.

Celle-ci consiste en la composition des complexités des étapes suivantes :

- identification de la racine  $\rho$ , notée  $\varphi_\rho$
- morphisme de  $\mathbb{G}$  vers  $\mathbb{T}$  par  $f$  avec  $\rho$  identifié, notée  $\varphi_f$
- encodage de l'arbre, notée  $\varphi_t$

$$\sigma^G \in \mathcal{O}(\varphi_\rho) \cdot [\mathcal{O}(\varphi_f) + \mathcal{O}(\varphi_t)]$$

avec  $\mathcal{O}(f(n))$  une classe de complexité parmi les deux citées.

### 1.4.1 Identification de la racine $\rho$ , $\varphi_\rho$

De manière naïve, on itère sur chaque noeud  $\nu$  de  $G$ , qui peut être la racine  $\rho$ .

Cependant, il est possible de restreindre l'ensemble des noeuds pouvant candidater à cette élection. Grâce à une fonction bien choisie, on associe un score calculable en  $O(1)$  à chaque noeud, restreignant ainsi le domaine de recherche à un nombre  $k_\rho$  de noeuds, avec  $1 \leq k_\rho \leq n$ .

$$\varphi_\rho \in \Omega(1) \quad \varphi_\rho \in O(n)$$

### 1.4.2 Encodage d'un l'arbre, $\varphi_t$

On peut considérer qu'à ce point nous avons conservé l'information de l'étage de chaque noeud de la précédente étape. L'encodage d'un noeud et d'une arête se fait en  $O(1)$ . Soit  $t$  le nombre de noeuds d'un arbre, son encodage de façon non-canonique se fait donc en  $O(t)$ , auquel il faut ajouter la complexité du tri opéré pour chaque ensemble de descendants d'un noeud. Le meilleur et le pire cas pour cet encodage (illustrés en Fig. 1.7) est donc directement lié aux cas correspondants pour le tri.

Un tri est typiquement de complexité  $O(t \cdot \log(t))$ .

$$\varphi_t \in \Omega(t) \quad \varphi_t \in O(t + t \cdot \log(t))$$

### 1.4.3 Morphisme de $\mathbb{G}$ vers $\mathbb{T}$ par $f$ , $\varphi_f$

L'application des morphismes aboutissant à un arbre représentant une classe d'isomorphisme intervient sur la complexité globale de l'algorithme de deux façons. Premièrement, l'étape en elle-même génère des actions qui ont un coût propre, mais également la taille de l'arbre généré aura une influence sur la suite de l'algorithme, car ce sont ces arbres qui seront encodés dans la dernière étape.

De façon invariable, il est nécessaire d'ordonner tous les noeuds en étages vis-à-vis de  $\rho$ . Cette opération  $\varphi_0$  s'assimile à un parcours de  $n$  étapes exactement.

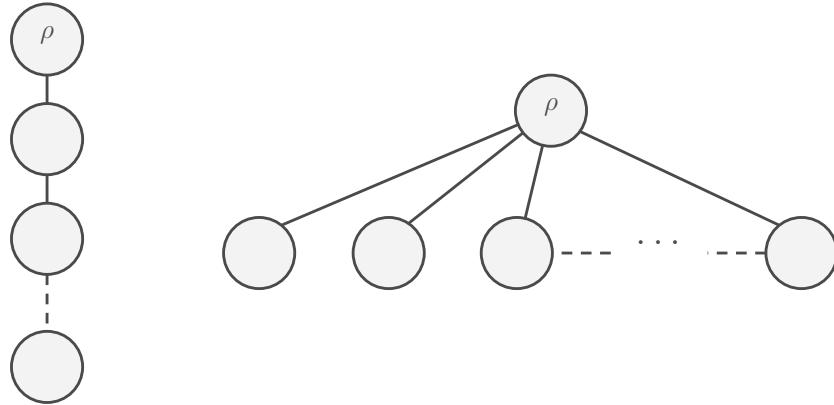


FIGURE 1.7 – Meilleur (gauche) et pire (droite) des cas pour l'encodage d'un arbre

Le nombre d'applications de ces productions n'est pas constant, et dépend du nombre d'arêtes  $e = \|E_G\|$  par rapport à  $n$ , avec  $(n - 1) \leq e \leq \frac{n(n-1)}{2}$  pour un graphe simple, sans self-bound que nous omettons pour l'instant.

### Coût des morphismes

La complexité de chacune des productions nécessaire au morphisme est relative à leur position par rapport à  $\rho$ . En effet, le calcul de "magnet" servant à apparier deux noeuds virtuels est réalisé à partir de la trace d'un sous-arbre. Dans tous les cas ce magnet est issu de la trace de deux noeuds, pris dans le contexte de leurs descendants exclusifs. Bien que l'on puisse conserver le résultat une fois celui-ci réalisé pour un noeud, on peut considérer qu'une telle trace soit calculée pour tous les noeuds dans le pire des cas. Chaque noeud  $v$  génère donc au préalable une opération en  $O(w_v + w_v \cdot \log(w_v))$  avec  $w_v$  le nombre de descendants de ce noeud. Les calculs suivants seront effectués en  $O(1)$ .

$$\varphi_{f_1} \in \Omega(1) \quad \varphi_{f_1} \in O\left(n * \frac{d_{max}}{2} + \frac{d_{max}}{2} \cdot \log\left(\frac{d_{max}}{2}\right)\right)$$

Nous pouvons exprimer le nombre d'arêtes séparant un graphe complet simple (sans self-bound) d'un arbre, correspondant au nombre maximum de productions à appliquer, par  $\frac{n(n-1)}{2} - (n - 1) = \frac{n^2 - 3n + 2}{2}$ , auquel on ajoute une self-bound à résoudre par noeud dans le pire des cas, opération en  $O(n)$ . Dans le meilleur des cas, si  $G$  est déjà un arbre, il n'y a aucun morphisme à appliquer.

$$\varphi_{f_2} \in \Omega(1) \quad \varphi_{f_2} \in O\left(n + \frac{n^2 - 3n + 2}{2}\right) = O\left(\frac{n^2 + n + 2}{2}\right)$$

On obtient, pour la séquence  $\varphi_f = \varphi_{f_2} \circ \varphi_{f_1} \circ \varphi_{f_0}$  :

$$\varphi_f \in \Omega(n) \quad \varphi_f \in O\left(n + n + \frac{n^2 - 3n + 2}{2}\right) = O\left(\frac{n^2 + n + 2}{2}\right)$$

### Taille de l'arbre généré

Les productions cobound et inbound ne génèrent pas le même nombre de noeuds et d'arêtes supplémentaires. En effet, si une production résolvant une cobound générera invariablement deux noeuds

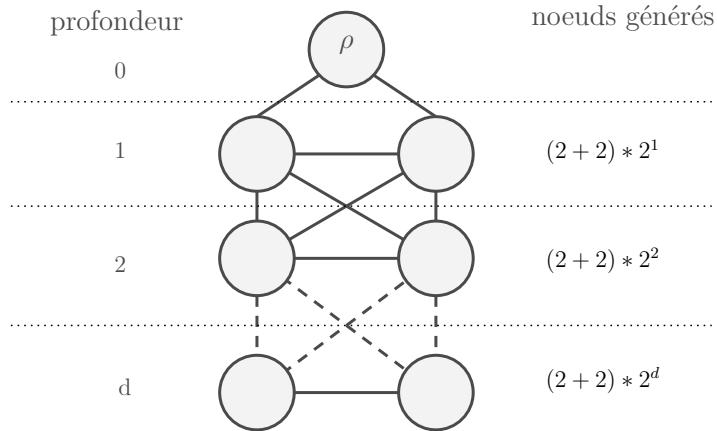


FIGURE 1.8 – Graphe maximisant la taille de l’arbre généré

virtuels, une production inbound doublera le nombre de noeuds successeurs du noeud incriminé. Là encore, le pire cas possible en terme de taille d’arbre généré n’est pas comme on pourrait le penser le graphe présentant le plus d’inbounds, qui serait un graphe à deux couches complètement connectées, mais le graphe présentant le plus de niveaux de deux noeuds possibles.

Là où chaque étage générera invariablement les deux noeud de chaque co-bound, les noeuds doublés dus à l’inbound le seront également dans les étages précédents, d’où l’application d’une puissance supplémentaire pour chaque étage.

Le nombre de niveau  $d$  que l’on peut obtenir sur ce type de graphe est de  $(n-1)/2$  (division entière). Nous pouvons donc estimer le nombre de noeuds générés dans le pire des cas à  $\sum_{d=1}^{(n-1)/2} 8^d = \frac{2}{7} \left( 2^{\frac{3n}{2} + \frac{1}{2}} - 4 \right) \sim k^{2 \cdot n}$ , avec  $k \simeq \frac{4}{7}$ . La taille des arbres générés  $n'$  lors de cette étape est donc encadrée de la manière suivante :

$$n \leq n' \leq n + k^{2 \cdot n}, \quad k \simeq 0.57$$

#### 1.4.4 Complexité globale

On rappelle la complexité globale de l’algorithme  $\sigma^G$  comme étant la composition suivante de  $\sigma^G \in \mathcal{O}(\varphi_\rho) \cdot [\mathcal{O}(\varphi_f) \circ \mathcal{O}(\varphi_t)]$ .

Si certaines complexités sont difficiles à déterminer, d’où des bornes majorantes larges, on constate toutefois un écart de complexité très important entre le cas le plus simple d’un graphe directement sous forme d’arbre, et le cas le plus complexe d’un graphe trop régulier pour diminuer le nombre de racines candidates, et présentant un grand nombre d’inbounds.

$$\sigma^G \in \Omega(n)$$

$$\sigma^G \in O \left( n \cdot \left( (n + k^{2n}) + (n + k^{2n}) \cdot \log((n + k^{2n})) + \frac{n^2 + n + 2}{2} \right) \right) \sim O(n \cdot k^{2n}), \quad k \simeq 0.57$$

## 1.4.5 Parallélisation

Malgré une complexité temporelle importante, et même exponentielle dans le pire des cas, notre algorithme dispose d'axes de parallélisation permettant de gagner jusqu'à un ordre de grandeur sur les architectures multicoeurs, voire deux sur des architectures multi-serveurs de type cluster.

Le premier axe se situe au niveau de l'identification du noeud racine. Une fois l'ensemble des candidats identifiés (et si possible minimisé), l'exécution des traces qui leur seront associés est indépendante, et peuvent être exécutés sur plusieurs stations. En effet, la sortie de l'algorithme est entièrement fonction de l'identité de ce candidat, et se prête naturellement à une distribution du calcul par programmation fonctionnelle.

Le second axe de parallélisation consiste à distribuer l'ordonnancement des productions réalisées pour une catégorie et un niveau donné. En effet, le calcul de ces invariants impliquent de nombreuses évaluations de sous-graphes en arbres, dont l'impact n'est pas négligeable sur la durée d'exécution de l'algorithme. Cependant, là encore, le score associé à une production est uniquement fonction des éléments impliqués directement dans cette production, autrement dit le niveau courant, qui n'est pas encore en cours de traitement, et les niveaux inférieurs déjà traités. Chaque évaluation de production peut donc travailler sur une copie locale et figée de ces éléments sans aucun risque d'effet de bord puisque le traitement n'a pas commencé. Ici également, une implémentation selon le paradigme fonctionnel permet de mettre en oeuvre un parallélisme à ce niveau de l'algorithme. L'application des productions altérant cependant à chaque étape la structure du graphe, cette séquence de transformations est cependant séquentielle.

Une mise en oeuvre de cette parallélisation peut par exemple s'effectuer sur un framework *Map Reduce*. Des travaux en ce sens ont été entamés sur la technologie *Spark*, permettant de diviser le temps total de calcul par le nombre de coeurs disponible.

## 1.5 Application

Nous détaillons ci-après quelques mesures de performance sur des cas concrets, soit théoriques soit issus du domaine applicatif de la chimie, et donc de complexité variable. Les calculs sont réalisés, sauf indication contraire, sur un poste doté de 8 coeurs (Intel Atom C2750 @ 2.40GHz) dédiés et 16 Go de mémoire.

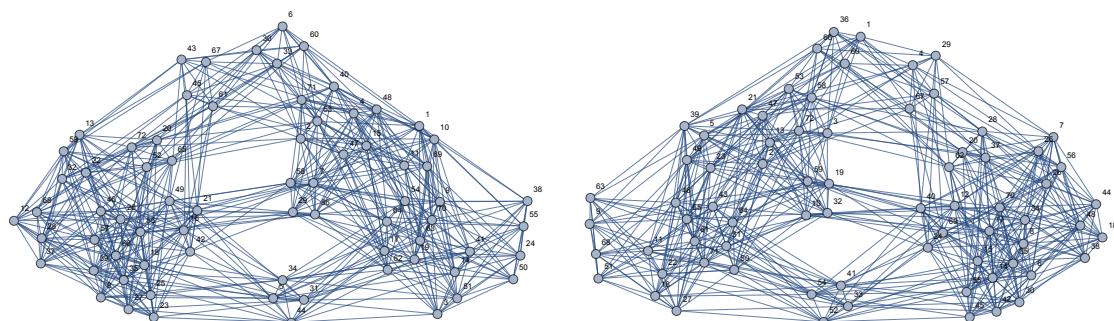


FIGURE 1.9 – Deux graphes "shrunken multipedes" isomorphes (72 noeuds)

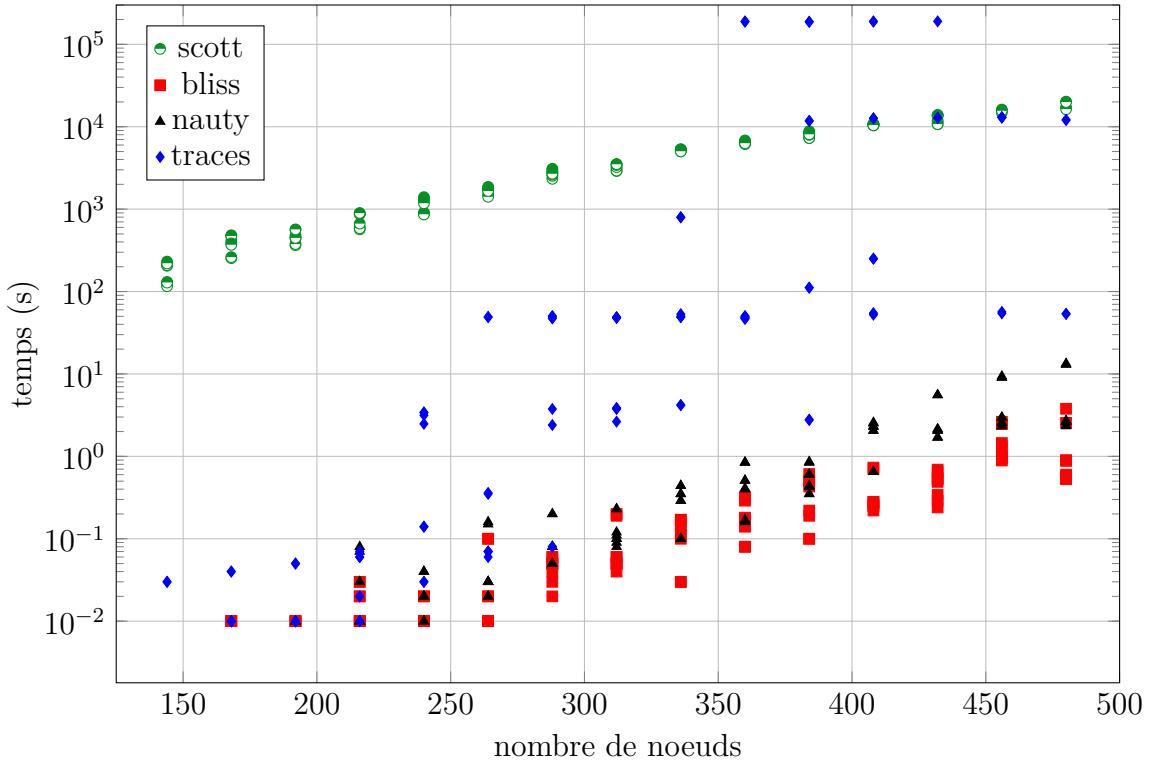


FIGURE 1.10 – Canonisation de graphes "shrunken multipedes" (y-log)

### 1.5.1 Shrunken multipedes graphs

En premier lieu, nous évaluons SCOTT sur des graphes combinatoires issus du benchmark [49]. Parmi les familles de graphes utilisées et mises à disposition sur cet article, nous nous intéressons ici aux "shrunken multipedes", une classe elle-même construite à partir des graphes "*Cai, Fürer and Immerman*" dont un couple d'exemples est illustré en figure 1.9. Nous pouvons ainsi comparer la performance de notre algorithme par rapport aux autres testés.

Ceci étant, il faut rappeler que cette famille de graphe est spécialement conçue pour présenter des cas d'isomorphismes (ou de non-isomorphismes) non triviaux à résoudre, et par conséquent aux caractéristiques assez éloignés des cas idéaux dont il est fait mention précédemment. Notamment, le nombre d'arêtes est important par rapport au nombre de noeuds (d'un facteur 7.8 en moyenne). Ces graphes sont donc en situation initiale très éloignés d'arbres. Nous calculons la forme canonique de 157 graphes ayant jusqu'à 500 noeuds, regroupés correctement en 79 classes d'isomorphisme. Nous comparons la performance de notre algorithme par rapport aux algorithmes de l'état de l'art qui proposent eux-aussi une forme canonique, à savoir TRACES, NAUTY et BLISS. Nous présentons en Fig. 1.10 le temps nécessaire pour calculer la trace canonique d'un graphe en fonction de son nombre de noeuds. Plusieurs ordre de grandeurs étant représentés sur le premier axe, une échelle log est utilisée.

Nous constatons que si les meilleurs algorithmes cet exercice démontrent une avance bien significative, SCOTT demeure tout à fait comparable avec certains d'entre eux au-delà de 380 noeuds. Il est donc raisonnablement utilisable en pratique, même pour des graphes issus de procédés combinatoires, très éloignés des conditions optimales. Nous notons également que pour un nombre de

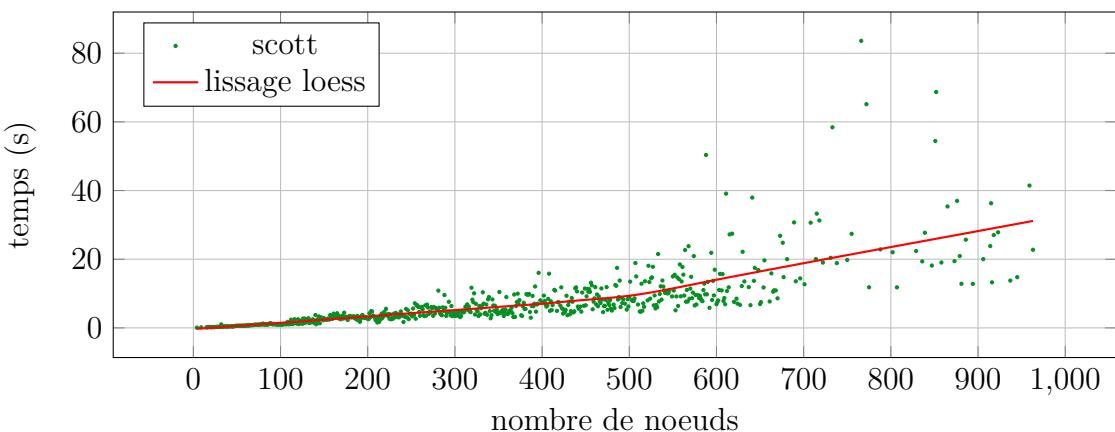


FIGURE 1.11 – Temps d’exécution médian par nombre de noeuds sur des graphes moléculaires

noeuds donné, la variabilité du temps de traitement est bien moindre sur SCOTT que sur les autres algorithmes, où l’on peut observer un écart de plusieurs ordre de grandeur pour plusieurs graphes ayant le même nombre de noeuds.

### 1.5.2 Graphes moléculaires

Nous testons finalement SCOTT sur des graphes moléculaires, le domaine pour lequel nous nous adressons de prime abord. Les molécules ne sont pas des graphes complexes, car le nombre de noeuds n’excède que rarement 500, et surtout sont naturellement assez proches des arbres. Bien entendu, il existe des cycles et des embranchements à traiter, mais le nombre d’arêtes est généralement voisin du nombre de noeuds, et il est en tout cas du même ordre. C’est donc une catégorie de graphes facilement adressée par notre algorithme.

Nous extrayons de *PubChem* un ensemble de molécules allant de 4 à 1000 atomes (noeuds). Nous échantillonons aléatoirement jusqu’à 500 noeuds, pour ne pas avoir à traiter des dizaines de millions de molécules. Par la suite, les molécules deviennent rares, et il n’est plus nécessaire de recourir à un échantillonage. Nous retenons au total 4736 molécules.

Nous ne distinguons pas dans la description des molécules les liaisons orientées *cis* et *trans*, nous nous attendons à observer des collisions de molécules stéréo-isomères, tout comme la notation SMILES classique. Nous obtenons sur le jeu de données 4490 traces uniques par SCOTT, soit le même nombre qu’avec la notation SMILES, attestant d’une équivalence dans ce contexte, et de l’absence de collision inattendue. Pour détecter les stéréo-isomères et ainsi reproduire le comportement des SMILES isomériques, il faudrait re-qualifier les liaisons en question sous des modalités plus fines.

Nous traçons en figure 1.11 le temps médian obtenu en fonction du nombre de noeuds considéré. Un lissage *loess* (*Locally Weighted Least Squares Regression*) est réalisé. Nous constatons que ce lissage indique un caractère linéaire par morceaux sur ce type de graphes, soit une complexité bien moindre que celle observée sur des graphes issus de procédés combinatoires.

## 1.6 Conclusion

### 1.6.1 Performances

Un travail considérable reste à réaliser au niveau de l'implémentation et de l'optimisation de cet algorithme, ne lui permettant pas en l'état d'être concurrentiel par rapport aux meilleurs algorithmes de l'état de l'art. Ceci étant, la solution proposée ici reste la seule à traiter nativement des arêtes colorées, et dispose de bonnes possibilités de parallélisation. En l'état, notre algorithme parvient toutefois, sur certains cas de graphes, au niveau d'outils tels que TRACES au delà d'un certain nombre de noeuds.

Si la complexité temporelle reste problématique pour des graphes théoriques issus de méthodes combinatoires, avec un temps d'exécution exponentiel par rapport au nombre de noeuds (tout comme les algorithmes de référence), elle est beaucoup plus raisonnable sur des graphes plus simples ayant un nombre d'arêtes voisin au nombre de noeuds. Pour cette catégorie de graphes, on observe plutôt un caractère linéaire, tout à fait utilisable en pratique.

### 1.6.2 Cas d'usage type

Le cas d'usage optimal de cet algorithme correspond exactement au contexte de son élaboration : pouvoir identifier formellement par une chaîne de caractères et à un isomorphisme près, de manière ré-utilisable, des (sous-)graphes de taille raisonnable, rendant triviale toute comparaison ultérieure avec un autre sous-graphe (fragment). Au delà de l'approche basée sur des morphismes de graphes plutôt que sur du *backtracking*, le principal point de démarcation par rapport à la bibliographie consiste à pouvoir traiter de manière native des arêtes colorées, ce qui peut être essentiel dans certains domaines (chimie organique par exemple).

### 1.6.3 Perspectives

#### Heuristiques et implémentation

Les algorithmes à base de *backtracking*, qui peuplent la très grande majorité de l'état de l'art sur ce problème, ont au fil des années pu développer des heuristiques permettant d'élaguer progressivement l'espace des solutions possibles, réduisant grandement la complexité globale de l'approche. Si notre implémentation reste à ce jour naïve, et ne développe aucune heuristique similaire, il est en revanche très probable qu'il en existe. Notamment, sur le nombre et la diversité d'invariants rapides à calculer permettant de limiter les racines potentielles, ou même pour éliminer certaines de ces racines avant même le calcul complet de leur arbre associé. De la même manière, il est probable qu'une implémentation de l'algorithme avec un langage performant (compilé), comme ceux utilisés pour les algorithmes de la bibliographie (C/ C++) puisse apporter un gain de performances significatif par rapport au Python, utilisé dans notre implémentation courante.

Le format de la trace produite pourrait lui aussi être sujet à des considérations nouvelles. Nous sommes restés focalisés ici à une notation "1-ligne" compacte, mais il serait possible d'avoir une notation tout aussi canonique en utilisant un format plus hiérarchique, et donc plus lisible, si l'usage s'y prête davantage. La seule condition est de respecter la relation d'ordre induite par la conversion en arbre.

## Extension aux graphes orientés

Une généralisation de SCOTT aux graphes orientés, typiquement utilisés dans le cadre de la représentation de relations sociales ou de dépendances, pourrait tout à fait être envisagée en adaptant le jeu de productions (voir la proposition en Fig. 1.12) permettant de transformer un graphe en arbre équivalent. Un arbre enraciné aux arêtes non-orientées étant équivalent au même arbre enraciné aux arêtes orientées, la fonction de trace une fois l'arbre obtenu serait utilisable sans modifications. La validité de cette extension resterait toutefois à formaliser.

## Généralisation à la notion de Noyau (*Kernel*)

Nous avons démontré que SCOTT était injectif vis-à-vis des classes d'équivalence, c'est-à-dire qu'il produit un résultat caractéristique de chaque classe d'équivalence, de sorte que l'on puisse se prononcer exactement sur le fait que deux graphes soient isomorphes ou non une fois leur trace obtenue.

Au-delà de ce résultat binaire (oui/non), on pourrait s'intéresser à préciser ce résultat, et voir dans quelle mesure des graphes très proches, quasi-isomorphes, produiront ou non des traces également proches, et inversement avec des graphes très différents, et ainsi nous rapprocher d'une "métrique d'isomorphisme".

Cette notion de similarité et de sa conservation après application d'un morphisme est couverte en algèbre par la notion de noyau (*kernel*). Un kernel  $k$  donne une mesure de similarité dans un espace qui peut être transposée à un produit scalaire dans un autre espace via une projection  $\phi$ . Étant donné deux objets  $x$  et  $x'$  et  $\langle \cdot, \cdot \rangle$  le produit scalaire,

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

Un exemple portant sur le problème d'isomorphisme de graphes serait le suivant : soient deux graphes  $G$  et  $G'$  isomorphes, il existe un espace dans lequel le produit scalaire des projections respectives de ces graphes par  $\phi$  vaut 0. Si tel est le cas, un kernel peut nous permettre de mesurer cette similarité sans forcément disposer de  $\phi$  ni même de cet espace. Un kernel peut donc être vu comme une mesure de similarité, qui n'est pas forcément parfait (injectif). Un kernel parfaitement injectif est d'ailleurs qualifié de kernel complet. Il existe par ailleurs un grand nombre de kernels définis sur les graphes [62, 63], sans qu'ils soient tous complets (injectifs) ou même qu'ils soient définis sur les graphes labellisés.

L'algorithme SCOTT permet de produire une représentation canonique préservant la proximité des graphes dans le sens où deux graphes similaires produiront des représentations canoniques similaires (et réciproquement). Ainsi, des distances applicables aux chaînes de caractères de type distance d'édition (en complexité temporelle  $O(N^2)$ ) ou par recouvrement de sous-séquences (en  $O(N \cdot \log(N))$ ) pourraient permettre, sur la base des représentations canoniques produites par SCOTT de disposer de mesures de similarité efficaces pour les graphes, et venir compléter l'ensemble des kernel pour graphes disponibles. Nous pourrions également directement traiter de distances d'édition et de recouvrement des arbres produits par SCOTT, sans forcément passer par les traces sous forme de chaîne de caractères.

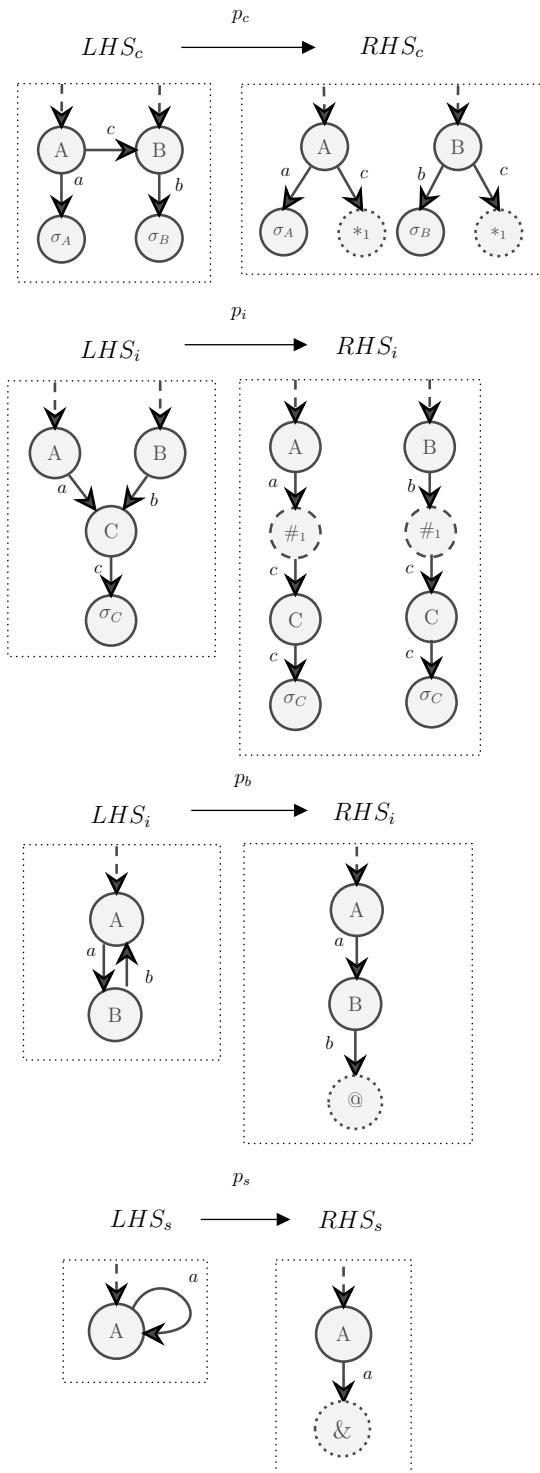


FIGURE 1.12 – Proposition d'adaptation pour arêtes orientées

# Glossaire

---

## graphé

Un graphe  $G = (V_G, E_G) \in \mathbb{G}$  est un objet mathématique composé d'un ensemble de sommets  $V_G$ , pouvant être associés deux à deux pour former des arêtes, formant l'ensemble  $E_G$ . Il peut être représenté par une énumération des éléments de ces deux ensembles, ou par une matrice d'adjacence  $A^G$ . Les sommets comme les arêtes peuvent être colorés par des fonctions d'étiquetage, respectivement  $L_G^{V_G} : V \rightarrow \Sigma^*$  et  $L_G^{E_G} : E \rightarrow \Sigma^*$ .. 27

$\mathbb{G}$

Ensemble des graphes. 27

## arbre

Un arbre  $t = (V_t, E_t) \in \mathbb{T}$  est un arbre appartenant au sous-ensemble de graphes énoncés  $\mathbb{T}$ , présentant les particularités d'être acycliques, autrement dit le chemin minimal entre deux noeuds  $\nu_1, \nu_2 \in V_t$   $\mu(\nu_1, \nu_2)$  est unique.. 29

$\mathbb{T}$

Ensemble des arbres, lui-même sous-ensemble des graphes, tel que  $\mathbb{T} \subset \mathbb{G}$ .. 30

$L_G^{V_G}$

Fction de labellisation des noeuds du graphe  $G$ , associant à chacun de ces noeuds un label.. 31

$L_G^{E_G}$

Fction de labellisation des arêtes du graphe  $G$ , associant à chacune de ces arêtes un label.. 32

## isomorphisme

Dans le cas de deux graphes  $G = (V_G, E_G)$  et  $H = (V_H, E_H)$ , un isomorphisme  $f : V_G \rightarrow V_H$  de  $G$  vers  $H$  est une bijection de l'ensemble des sommets de  $G$  dans l'ensemble des sommets de  $H$ , conservant les arêtes.. 27

## relation d'isomorphisme ( $\simeq$ )

relation binaire entre deux éléments (ici, graphes) satisfaisant au moins un isomorphisme. 27

$[G]$

Classe d'équivalence du graphe  $G$ , ensemble comprenant tous les graphes partageant un isomorphisme avec  $G$ , et ayant donc la même forme canonique. 28

$\leq$

relation binaire réflexive, antisymétrique et transitive, pouvant ordonner un ensemble  $\mathcal{E}$ . On note alors  $(\mathcal{E}, \leq)$ . 30

$\Sigma^*$

l'ensemble des chaînes de symboles de longueur indéterminée  $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$ . 30

$\leftrightarrow$

Si  $E \leftrightarrow F$ , alors il existe une bijection entre les ensembles  $E$  et  $F$ . 45

$\nleftrightarrow$

Si  $E \nleftrightarrow F$ , alors il n'existe une bijection entre les ensembles  $E$  et  $F$ . 45

$\rho$

dans le cadre d'un arbre  $t \in \mathbb{T}$ , il s'agit d'un noeud tenant le rôle de racine, c'est-à-dire qu'il est l'ancêtre de tous les autres noeuds. Dans le cadre d'un graphe quelconque  $g \in \mathbb{G}$ , il s'agit simplement d'un noeud identifié comme tel, n'ayant pas de propriété particulière.. 30

$e_G(\cdot, \cdot)$

fonction associant à deux noeuds la valeur de l'arête les reliant, éventuellement nulle, soit la valeur  $A_{ij}^G$ . 30

$\mu(\cdot, \cdot)$

fonction associant à deux noeuds  $\nu_1$  et  $\nu_2$  le chemin (séquence d'arêtes) minimal  $\mu(\nu_1, \nu_2)$  les reliant. La norme  $\|\mu(\nu_1, \nu_2)\|$  désigne la somme pondérée par les poids respectifs des arêtes de cette séquence. cf. *chaîne*. 30

**chaîne**

Séquence de noeuds notée  $\langle \nu_1, \dots, \nu_m \rangle$  que l'on visite en suivant un chemin. cf.  $\mu$ . 40

**production**

Une production  $p : L \xleftarrow{l} K \xrightarrow{r} R$  est une paire de morphismes de graphes  $l : K \rightarrow L$  et  $r : K \rightarrow R$ , tous deux injectifs, permettant d'appliquer une transformation déterministe sur une portion d'un graphe  $G$ .. 37

$m : V_G \rightarrow V_{LHS}$

Fonction dite de match mettant en bijection un sous-ensemble de noeuds de  $G$  avec les noeuds de  $LHS_m$ . Cette fonction attribue les rôles dans la production  $p$  associée à  $m$ .. 38

$\{p_x^N\}$

Ensemble des productions de catégorie  $x$  parmi self-bound, co-bound et in-bound à appliquer au niveau  $N$  d'un graphe  $G$ . Interprétation ensembliste de  $p_x^N$ .. 42

$p_x^N$

Morphisme défini par la composition de toutes les productions d'une catégorie  $x$  parmi self-bounds, co-bounds et in-bounds effectuées au niveau  $N$ , tel que  $p_x^N = p_{x_k}^N \circ \dots \circ p_{x_1}^N$ , avec  $k = \Omega(\{p_k^N\})$ . 42

$f_N$

Morphisme défini comme la composition de toutes les productions effectuées sur un niveau  $N$ , effectuées dans l'ordre tel que  $f_N = p_i^N \circ p_c^N \circ p_s^N$ . 42

$f_G$

Morphisme général appliqué à un graphe  $G$  pour le transformer en arbre. Composition des morphismes par niveau  $f_{N_G} \circ \dots \circ f_0$  avec  $N \in [0, N_G]$ . 42

$N$

Niveau d'un noeud  $\nu \in V_G$  à valeur dans  $[0, N_G]$ , indiquant la position d'un noeud par rapport à la racine  $\rho$ , 0 indiquant la feuille la plus éloignée. On note  $V_N$  l'ensemble des noeuds de niveau  $N$ . 30

$N_G$

Nombre de niveaux observé sur le graphe enraciné  $G_\rho$  entre sa racine  $\rho$  et ses feuilles les plus éloignées. 30

$V_N$

Ensemble des noeuds situés au niveau  $N$  d'un graphe.. 30

$\Lambda_\nu$

Dans le cadre d'un arbre enraciné, pour un noeud  $\nu$  de niveau  $N$ , désigne l'ensemble des noeuds  $\lambda$  de niveau  $N - 1$  ayant une arête les reliant à  $\nu$ . Autrement dit, les descendants directs de  $\nu$ .. 30

$t_\nu$	Sous arbre associé un noeud $\nu$ , composé uniquement des éléments descendants de $\nu$ , qui tient donc le rôle de racine.. 30
$T_N$	Ensemble des arbres associés respectivement à l'ensemble des noeuds $V_N$ . Chacun de ces noeuds $\nu \in t_\nu$ est la racine d'un arbre $t_\nu$ .. 33
$\kappa$	Fonction de "compression" d'un sous arbre $t_\nu$ vers un unique noeud de label $\sigma^T(t_\nu)$ .. 33
$K_N$	Ensemble des noeuds produits par l'application de la fonction de compression <i>kappa</i> à $T_N$ .. 33
$G_\rho$	Un graphe quelconque $G \in \mathbb{G}$ , muni d'un noeud $\rho \in V_G$ identifié comme racine, et défini comme enraciné.. 42
$G_{\rho, N}$	Un graphe enraciné $G_\rho$ tronqué au niveau $N$ . $G_{\rho, N} = (W_N, E_N \in (W_N \times W_N)) \subseteq G_\rho$ .. 42, 61
$\overset{k}{G}$	Graphe intermédiaire dans le traitement de $G$ par $f_G$ , $\overset{k}{G}$ est le résultat de l'application de la $k$ -ième production.. 47
$g^j$	Sous-graphe de $G$ "acteur" de la production $p^j$ , appairé par la fonction de matching $m^j$ . 46
$W_N$	Ensemble des noeuds d'un niveau inférieur ou égal à $N$ .. 42
$G_{\rho, N}^*$	Graphe enraciné sur $\rho$ , tronqué au niveau $N$ , pour lequel on a appliqué toutes les productions $p_s$ , $p_c$ et $p_i$ possibles. On démontre que c'est un arbre.. 42
$\Gamma$	Opérateur de concaténation d'un ensemble muni d'une relation d'ordre. Exemple : $\Gamma(\{A, B, C\}, \cdot) = A \cdot B \cdot C \dots$ 35
$\sigma$	fonction de trace associant à un noeud ( $\sigma^V$ ), une arête ( $\sigma^E$ ), un arbre ( $\sigma^T$ ) ou encore un graphe ( $\sigma^G$ ) une représentation de sa classe d'équivalence, sous forme de séquence de symboles. Deux éléments satisfaisant une relation d'isomorphisme produiront ainsi une trace identique. On adjoint à cette fonction une section (inverse à droite) $\tilde{\sigma}$ . Dans les faits, la trace produite en premier lieu est suffisante, car elle permet sans ambiguïté d'identifier une classe d'équivalence ainsi que ses membres, elle est donc canonique.. 5, 32
$\tilde{\sigma}$	Section (inverse à droite) d'une fonction de trace $\sigma$ , associant à la trace produite le-dit représentant de la classe d'équivalence. $\tilde{\sigma} \circ \sigma$ est donc une canonisation. L'espace de sortie de cette application est donc quotienté par la relation d'isomorphisme $\simeq$ . 32

## Références

- [38] László BABAI et Eugene M LUKS, « Canonical labeling of graphs », in : *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, ACM, 1983, p. 171-183.
- [39] Viktor N ZEMLYACHENKO, Nickolay M KORNEENKO et Regina I TYSHKEVICH, « Graph isomorphism problem », in : *Journal of Soviet Mathematics* 29.4 (1985), p. 1426-1481.
- [40] Uwe SCHÖNING, « Graph isomorphism is in the low hierarchy », in : *Journal of Computer and System Sciences* 37.3 (1988), p. 312-323.
- [41] Vikraman ARVIND et Piyush P KURUR, « Graph isomorphism is in SPP », in : *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*. IEEE, 2002, p. 743-750.
- [42] Kellogg S BOOTH et Charles J COLBOURN, *Problems polynomially equivalent to graph isomorphism*, Computer Science Department, Univ., 1979.
- [43] Johannes KOBLER, Uwe SCHÖNING et Jacobo TORÁN, *The graph isomorphism problem : its structural complexity*, Springer Science & Business Media, 2012.
- [44] László BABAI, « Graph isomorphism in quasipolynomial time », in : *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, ACM, 2016, p. 684-697.
- [45] David WEININGER, « SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules », in : *Journal of chemical information and computer sciences* 28.1 (1988), p. 31-36.
- [46] Stephen R HELLER et Alan D McNAUGHT, « The IUPAC international chemical identifier (InChI) », in : *Chemistry International* 31.1 (2009), p. 7.
- [47] Eugene M LUKS, « Isomorphism of graphs of bounded valence can be tested in polynomial time », in : *Journal of computer and system sciences* 25.1 (1982), p. 42-65.
- [48] John E HOPCROFT et Jin-Kue WONG, « Linear time algorithm for isomorphism of planar graphs (preliminary report) », in : *Proceedings of the sixth annual ACM symposium on Theory of computing*, ACM, 1974, p. 172-184.
- [49] Daniel NEUEN et Pascal SCHWEITZER, « Benchmark graphs for practical graph isomorphism », in : *arXiv preprint arXiv :1705.03686* (2017).
- [50] Yuri GUREVICH et Saharon SHELAH, « On finite rigid structures », in : *The Journal of Symbolic Logic* 61.2 (1996), p. 549-562.

- 
- [51] Paul T DARGA, Karem A SAKALLAH et Igor L MARKOV, « Faster symmetry discovery using sparsity of symmetries », in : *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, IEEE, 2008, p. 149-154.
  - [52] Paolo CODENOTTI et al., « Conflict analysis and branching heuristics in the search for graph automorphisms », in : *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, IEEE, 2013, p. 907-914.
  - [53] José Luis LÓPEZ-PRESA, Antonio Fernández ANTA et Luis Núñez CHIROQUE, « Conauto-2.0 : Fast isomorphism testing and automorphism group computation », in : *arXiv preprint arXiv :1108.1060* (2011).
  - [54] Tommi JUNTTILA et Petteri KASKI, « Engineering an efficient canonical labeling tool for large and sparse graphs », in : *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*, SIAM, 2007, p. 135-149.
  - [55] Tommi JUNTTILA et Petteri KASKI, « Conflict propagation and component recursion for canonical labeling », in : *International Conference on Theory and Practice of Algorithms in (Computer) Systems*, Springer, 2011, p. 151-162.
  - [56] Brendan D MCKAY et al., « Practical graph isomorphism », in : (1981).
  - [57] Adolfo PIPERNO, « Search space contraction in canonical labeling of graphs », in : *arXiv preprint arXiv :0804.4881* (2008).
  - [58] Brendan D MCKAY et Adolfo PIPERNO, « Practical graph isomorphism, II », in : *Journal of Symbolic Computation* 60 (2014), p. 94-112.
  - [59] Jacques NEVEU, « Arbres et processus de Galton-Watson », in : *Annales de l'IHP Probabilités et statistiques*, t. 22, 2, 1986, p. 199-207.
  - [60] Pedro Pablo Perez VELASCO, *Matrix Graph Grammars*, 2008, p. 321, arXiv : 0801.1245, URL : <http://arxiv.org/abs/0801.1245>.
  - [61] G OLSEN, « Gary Olsen's interpretation of the “Newick's 8 : 45” tree format standard », in : *URL http ://evolution. genetics. washington. edu/phylip/newick doc. html* (1990).
  - [62] S Vichy N VISHWANATHAN et al., « Graph kernels », in : *Journal of Machine Learning Research* 11.Apr (2010), p. 1201-1242.
  - [63] Benoit GAÜZERE, Luc BRUN et Didier VILLEMIN, « Two new graphs kernels in chemoinformatics », in : *Pattern Recognition Letters* 33.15 (2012), p. 2038-2047.
  - [64] Reinhard DIESTEL, « Graph theory. 2005 », in : *Grad. Texts in Math* 101 (2005).
  - [65] László BABAI, William M KANTOR et Eugene M LUKS, « Computational complexity and the classification of finite simple groups », in : *24th Annual Symposium on Foundations of Computer Science (Sfcs 1983)*, IEEE, 1983, p. 162-171.

- [66] Akihiro INOKUCHI, Takashi WASHIO et Hiroshi MOTODA, « An apriori-based algorithm for mining frequent substructures from graph data », in : *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2000, p. 13-23.
- [67] Xifeng YAN et Jiawei HAN, « gspan : Graph-based substructure pattern mining », in : *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, IEEE, 2002, p. 721-724.
- [68] Arthur CAYLEY, « A theorem on trees », in : *Quart. J. Math.* 23 (1889), p. 376-378.
- [69] Frank HARARY et Edgar M PALMER, *Graphical enumeration*, Elsevier, 2014.
- [70] Pedro Pablo Perez VELASCO et Juan de LARA, « Matrix Approach to Graph Transformation : Matching and Sequences », in : *Lecture Notes in Computer Science* 4178 (2006), p. 122, ISSN : 16113349, DOI : [10.1007/11841883\\_10](https://doi.org/10.1007/11841883_10).
- [71] H. EHRIG, K. HOFFMANN et J. PADBERG, « Transformations of Petri nets », in : *Electronic Notes in Theoretical Computer Science* 148.1 SPEC. ISS. (2006), p. 151-172, ISSN : 15710661, DOI : [10.1016/j.entcs.2005.12.016](https://doi.org/10.1016/j.entcs.2005.12.016).

# Plongements sémantiques de fragments de graphe

---

En nous appuyant sur l'algorithme Scott, nous pouvons à présent caractériser formellement tout graphe (ou sous-graphe) aux noeuds et aux arêtes labellisés. Nous allons à présent développer la notion de fragments de graphes, un concept très proche de la notion de sous-graphes et qui est à la base de nombreuses approches exploitées dans la construction de modèles structure-activité, malgré les faiblesses et limitations inhérentes mentionnées en introduction. En explorant une piste basée sur des observations empiriques, nous verrons comment il est possible d'adapter certaines méthodes utilisées en traitement automatique des langages naturels pour apprendre des vectorisations sémantiques construite à partir d'une étude "lexicographique" portant sur ces fragments.

## 2.1 Fragments de graphes

### 2.1.1 Définition

Explicitons dans un premier temps la notions de fragments. Il s'agit simplement d'un sous-graphe pour lequel la distance maximale entre deux noeuds est bornée supérieurement, la borne considérée étant désignée par le terme de *taille de fragment*. Par exemple, pour un fragment de taille 2, la plus longue séquence de noeuds qu'il sera possible de parcourir sur ce fragment n'excédera pas 3, tandis que des fragments de taille 0 s'apparentent eux à de simples noeuds. Nous illustrons en Fig. 2.1 les fragments de taille  $[0, 2]$  issus d'un noeud (indiqué en gras), sur la molécule de caféine.

### 2.1.2 Fragmentation d'un graphe

Lorsque l'on fragmente entièrement un graphe, c'est-à-dire que l'on cherche à énumérer tous les fragments d'une taille donnée présents sur ce graphe, chaque noeud va ainsi générer un fragment dont il sera l'origine. Ces fragments étant des sous-graphes, et donc des graphes, il est possible de les identifier formellement grâce par exemple à l'algorithme SCOTT, qui présente l'avantage d'obtenir pour chaque classe d'équivalence de ces fragments une transcription sous forme de chaîne de caractères unique.<sup>1</sup>

Ces fragments de graphes peuvent être utilisés en tant que descripteurs topologiques du graphe dont ils sont issus. Chacun des  $N$  fragments extraits d'un corpus de graphes devient ainsi le représentant d'une variable  $\chi_i$ , avec  $i \in [0, N]$ , et forment un espace vectoriel sur lequel il devient possible de projeter une collection de fragments, et par extension un graphe. Cette projection peut

---

1. Note : le noeud générant le fragment ne sera pas forcément le noeud racine de l'arbre calculé par SCOTT

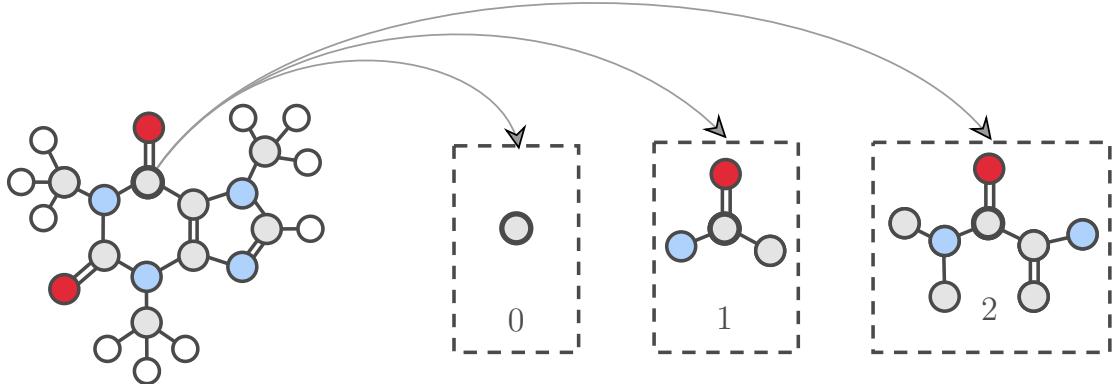


FIGURE 2.1 – Fragments de différentes tailles issus d'un noeud (en gras)

être une simple somme d'occurrences sur le-dit graphe, dans sa version la plus naïve, ou à base de pondérations plus élaborées, telles que la pondération *tf-idf* [72] pour appliquer un coefficient aux valeurs associées respectivement à chaque fragment selon leur niveau de rareté dans le corpus et leur fréquence dans le graphe. En Fig. 2.2 nous illustrons la projection d'un graphe moléculaire sur un espace de fragments de taille 1, dans sa version la plus simple, c'est-à-dire une somme des occurrences de chaque fragment. On indique sur chaque noeud le fragment  $\chi_i$  qu'il génère. Cette vectorisation de graphe peut être utilisée pour alimenter des modèles structure-activité les plus basiques, en les soumettant par exemple à un classifieur de type bayésien, *Random Forest* ou de type réseau de neurones artificiels. Nous observons d'ores et déjà sur l'exemple de la Fig. 2.2 qu'il semble y avoir un écart important entre le fragment le plus observé (10 occurrences) et les fragments les plus rares (seulement 1 occurrence, pour 6 fragments sur 10). Cette observation fera l'objet d'une discussion par la suite.

### 2.1.3 Vocabulaire généré

Il est important de noter qu'avant l'implémentation de l'algorithme SCOTT proposé, il n'existant pas de méthode prouvée formellement permettant d'identifier de manière unique des fragments aux noeuds et arêtes colorés dans le cas général (pas seulement des molécules entières). Par exemple, l'identification de fragments utilisée dans [19] repose sur des invariants chimiques qui ne sont pas toujours vérifiés sur certaines portions de molécule : celle-ci n'est donc pas par conséquent réellement canonique. Bien que le taux de collision soit faible, il n'est pas pour autant entièrement négligeable comme le montre l'étude ci-dessous.

Pour estimer le taux de collision, nous considérons un jeu de données issu de [9], comportant 707 observations de *LogP*, une propriété proche de la solubilité. Nous réalisons une fragmentation (de taille 2) des graphes constituant les observations selon les deux approches évoquées (SCOTT et ISIDA), et réalisons une régression de type *Bayesian Ridge*. Les fragmentations de tous les graphes aboutissent à un ensemble de fragments observables formant un *vocabulaire*. Nous remarquons que les vocabulaires générés varient selon la méthode de fragmentation utilisée : nous distinguons 151 fragments avec une fragmentation SCOTT, tandis que seulement 147 sont répertoriés via une fragmentation ISIDA, soit un taux de collision de 2.65% pour la méthode ISIDA. Nous donnons en Tab. 2.1 les métriques moyennes obtenues sur une validation croisée en *Leave-One-Out*<sup>2</sup>, pour

2. chaque observation est donc évaluée dans le cas où toutes les autres composent le jeu d'apprentissage

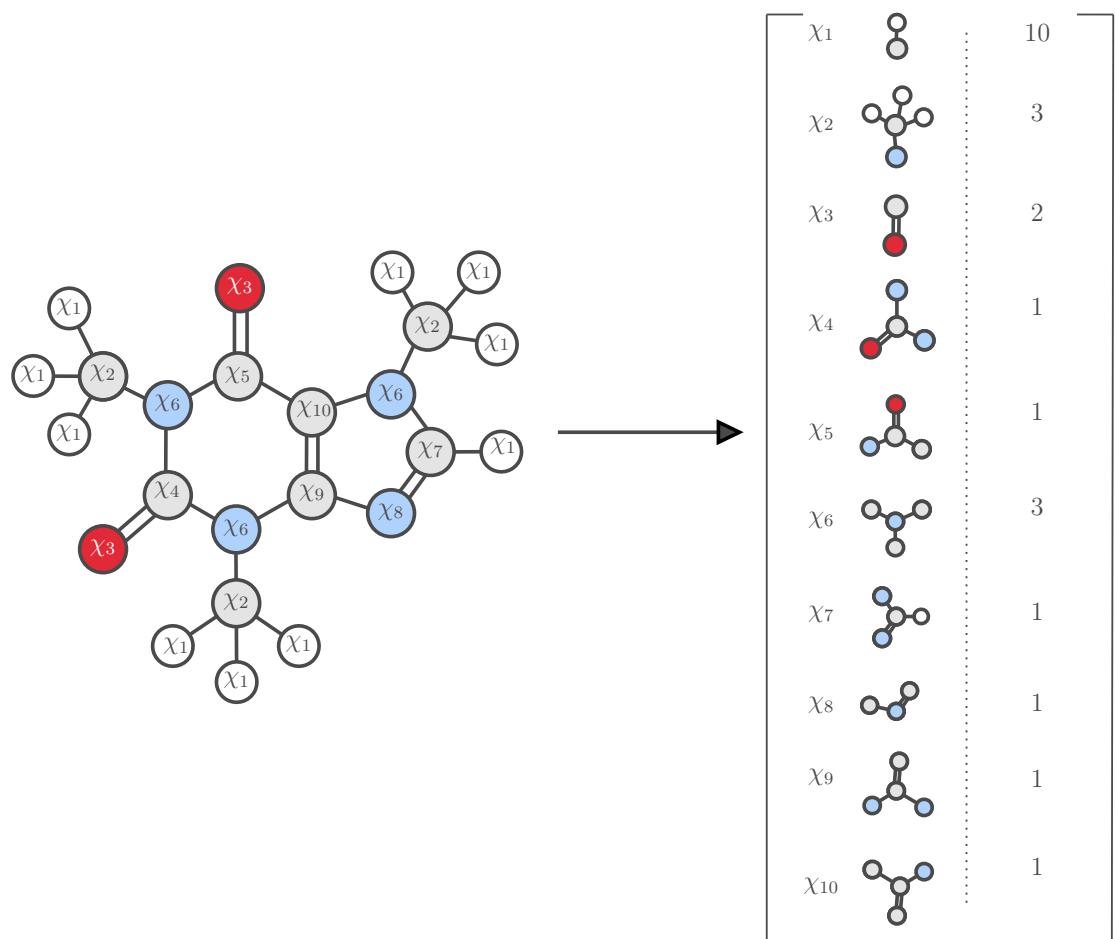


FIGURE 2.2 – Projection d'un graphe sur un espace de fragments de taille 1

	Fragments SCOTT	Fragments ISIDA
Taille du vocabulaire	151	147
Erreur Absolue Médiane	0.450	0.452
Erreur relative moyenne	13.8%	13.9%
Coefficient de détermination $R^2$	0.629	0.627

TABLE 2.1 – Comparaison des vocabulaires de fragments canoniques (SCOTT) et non-canoniques (ISIDA)

minimiser les effets d'un *fold* particulier.

Comme le montrent les métriques très proches d'un modèle de fragmentation à l'autre, l'impact de ces collisions sur la performance des modèles de régression produits reste très réduit : la fragmentation canonique de Scott n'améliore que de 0.1% l'erreur relative moyenne observée sur la prédiction de la propriété cible. Ceci étant, il y a bien un effet lié à la couverture de ce vocabulaire, et il sera par la suite d'autant plus important de ne pas observer de collisions lorsque nous étudierons la distribution de ces fragments.

#### 2.1.4 Dimensionnalité d'un espace de fragments

Comme illustré en Fig. 2.1, la taille des fragments considérés a un impact primordial sur l'information qu'ils transportent. Des fragments de taille 0 ne transportent ainsi pas plus d'information que le label du noeuds (élément chimique), tandis que si l'on augmente la taille des fragments pour obtenir une information plus fine, notamment concernant les cycles, la diversité des fragments est soumise à une explosion combinatoire : il existe plus de 30 éléments que l'on observe régulièrement en chimie organique, combiné à 3 modalités de liaisons courantes. Malgré les contraintes fortes régissant la chimie, et donc les fragments obtenus, on observe un accroissement exponentiel de la taille du vocabulaire à mesure que l'on considère des fragments plus précis (plus grands). On illustre ce phénomène en Tab. 2.2, recensant le nombre de fragments que l'on obtient en fonction de leur taille, sur un corpus de 1'510'000 molécules en libre accès [11].

Taille de fragment <sup>3</sup>	Taille du vocabulaire généré
$t = 1$	3 533
$t = 2$	153 004
$t = 3$	1 386 918

TABLE 2.2 – Taille du vocabulaire généré sur un corpus issu de *PubChem* en fonction de la taille des fragments

Ce nombre de dimensions rapidement très élevé amène à composer avec un phénomène bien connu sous le nom de *malédiction de la dimensionnalité* (*curse of dimensionality*) [73], et complique toute analyse de données portant sur des données à valeur dans un espace dimensionnel élevé, car cet espace devient très rapidement "vide". Une interprétation géométrique de ce problème est la suivante : comparons le volume que couvre la distance euclidienne de valeur arbitraire  $r$  dans un espace

préentissage

euclidien de dimension  $d$ , soit le volume d'une hypersphère de rayon  $r$  et de volume  $\frac{2r^d\pi^{d/2}}{d\Gamma(d/2)}$ <sup>4</sup> par rapport au volume du plus petit hypercube contenant cette dernière, de côté  $2r$  et de volume  $(2r)^d$ .

$$\frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)}$$

Ce ratio tend vers 0 quand  $d$  augmente, et ce très rapidement : si en dimension 2 une sphère inscrite dans un carré occupe environ 78,5% de sa surface, la boule n'occupe plus que 52% du cube, et une hypersphère de dimension 8 seulement 1.5% du volume de l'hypercube. Les données deviennent donc éparques dans un espace très vaste, et les distances classiques sont alors inutilisables. Il devient ainsi très difficile de rapprocher des observations, et par conséquent d'établir des corrélations à la base de tout traitement statistique.

## 2.2 Analogie avec les langages naturels

Nous avons pu remarquer sur la fragmentation de la molécule de caféine un écart important dans les fréquences d'occurrence entre le fragment le plus courant et les fragments les plus rares. Cette molécule étant parmi les molécules les plus classiques que l'on puisse rencontrer en chimie, il serait intéressant de déterminer si ce phénomène se répète sur l'intégralité du corpus de 1'510'000 molécules issues de *PubChem*, ou si au contraire les fragments sont en pratique plutôt équiprobables.

### 2.2.1 Distributions de fragments

Pour répondre à cette interrogation on réalise l'expérience suivante : considérant des fragments de taille  $t \in \{1, 2, 3\}$ , et un corpus extrait de la base de données *PubChem* balayant un espace chimique assez large, nous évaluons les fréquences respectives de chaque fragment, et déterminons l'écart entre les fragments les plus courants et les plus exotiques. Nous souhaitons établir ici si, le cas échéant, cette répartition se rapproche d'une loi statistique connue.

Cette expérience requiert un temps de calcul conséquent, mais présente plusieurs degrés de parallélisation : en effet, la projection d'une molécule sur un espace de fragments n'est fonction que de cette molécule, et peut donc être réalisée indépendamment de tout autre élément du corpus. Ces projections faites, les fréquences globales ne sont qu'une somme de ces projections individuelles. Ce problème est donc tout à fait approprié à une implémentation massivement parallélisée de type *Map-Reduce*, permettant un temps de traitement de l'ordre de la journée sur une grappe de 5 serveurs, pour un volume de données à traiter de l'ordre de 4820 Go.

Nous traçons en Fig. 2.3 la fréquence (nombre d'occurrences) des fragments en fonction de leur rang. Comme en atteste le repère log-log utilisé, la fréquence d'un fragment est fortement décroissante par rapport à son rang, et s'approche dans tous les cas d'une série de fonctions puissance négatives de type *broken power law*. Pour  $t = 1$ , on dénombre ainsi 16'978'587 occurrences pour le fragment (H:1)C (fragment déjà prépondérant sur l'exemple de la caféine), et une seule occurrence pour (Sn:1, Sn:1)O, soit un rapport de l'ordre de  $10^7$ . On dénombre par ailleurs 1220 fragments rencontrés une seule fois, soit environ 35% du vocabulaire, venant confirmer la première tendance donnée par la fragmentation de la caféine. Mis à part une différence sur les ordres de grandeur, on constate le même phénomène en augmentant la taille des fragments, avec la

4.  $\Gamma$  désignant la généralisation de la fonction factorielle, telle que  $\Gamma(n) = (n - 1)! \forall n \in \mathbb{N}$

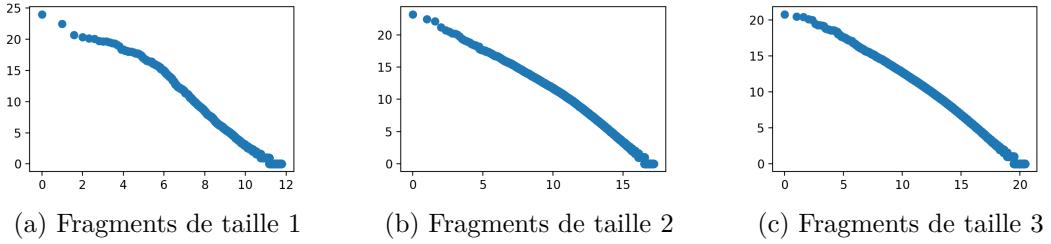


FIGURE 2.3 – Fréquence des fragments en fonction du rang, en repère log-log

nuance toutefois d'une rupture moins marquée entre les différentes fonctions de puissance quand  $t$  augmente, et donc une tendance vers un modèle constitué d'une seule loi de puissance.

Quelle que soit la taille de fragment considérée, nous observons en plus du grand nombre de dimensions la difficulté suivante : il existe une disparité de présence très importante entre les fragments les plus courants, portant finalement assez peu d'information, et les fragments les plus exotiques, potentiellement plus discriminants, mais n'étant pas caractérisés par un nombre d'occurrences suffisant pour en tirer une information fiable (sans induire directement un biais très important lié à l'observation d'un fragment exotique). Enfin, les projections de molécules sur des espaces de fragments aboutiront à des vectorisations creuses (contenant peu de valeurs non nulles) pour caractériser les observations.

## 2.2.2 Rapprochement empirique avec le TALN

Un autre domaine traitant typiquement des observations éparses, en grande dimension et pour lesquelles la couverture des variables explicatives suivent une distribution caractéristique de lois de puissance négatives est le traitement automatique des langages naturels (TALN).

Dans ce cas précis, on parle même de distribution (ou de loi) dite de *Zipf*, ou encore *Zipfienne*, du nom du linguiste s'étant attelé à l'étude de la fréquence des mots (lexèmes) selon leur rang d'utilisation, sur un corpus textuel<sup>5</sup>. Zipf a observé empiriquement que le mot le plus courant était observé 10 fois plus que le dixième mot le plus utilisé, lui-même observé 100 fois plus que le centième mot, etc. La fréquence du mot de rang  $n$ , notée  $freq(n)$  suit donc l'estimation suivante :

$$freq(n) = \frac{K}{n}$$

avec  $K \in \mathbb{N}$  une constante variable selon le langage, conduisant à une droite dans un repère log-log qui traduit bien une dépendance linéaire de la fréquence vis-à-vis du rang [74]. Cette observation empirique, bien qu'assez simpliste, modélise bien l'utilisation des mots sur tous les langages étudiés, à l'exception notable des premiers rangs, qui suivent généralement une distribution assez spécifique. Un ajustement est donc proposé par B. Mandelbrot [75], résumé en [76], introduisant un paramètre  $b$  caractérisant le nombre de termes situés dans la tête de distribution, un paramètre  $a$  permettant de préciser à quel point cette tête de distribution se comporte différemment du corps, et enfin un paramètre  $c$  influant sur le taux de décroissance, et pouvant être vu comme la "profondeur" du vocabulaire.

5. le roman "*Ulysses*" (J. Joyce),  $\simeq 250'000$  mots

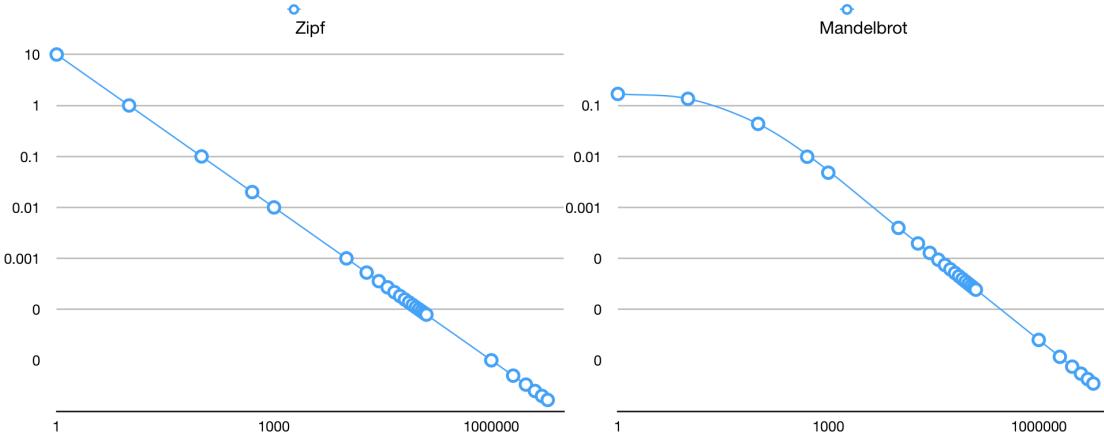


FIGURE 2.4 – Occurrence par rapport au rang pour des distributions Zipf et Mandelbrot

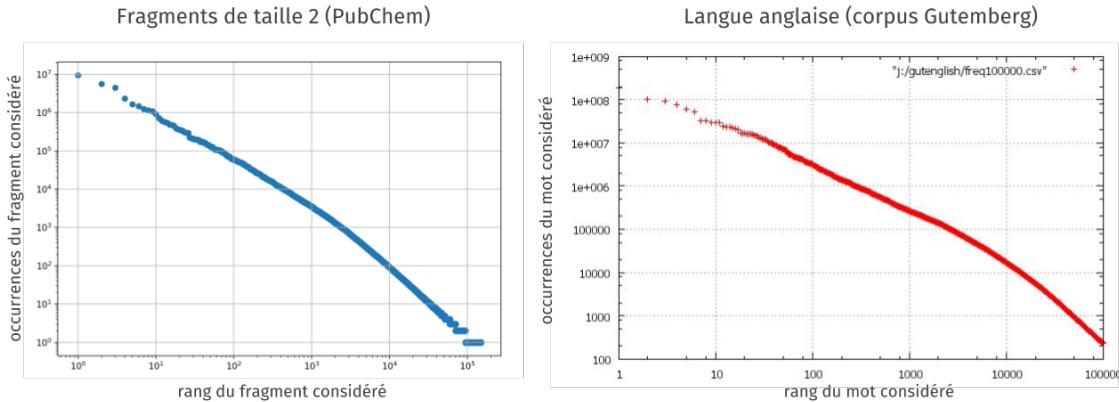


FIGURE 2.5 – Relation entre rang d'utilisation d'un fragment/mot et sa fréquence

$$freq_M(n) = \frac{K}{(an + b)^c}$$

Dans le même repère log-log, cette distribution est quasiment équivalente à une loi de Zipf pour les mots de rang centraux, mais introduit un "coude" caractéristique, permettant de traiter à part la tête de distribution. On donne un exemple de loi de Zipf et de Mandelbrot en Fig. 2.4. Il est à noter que généralement, la queue de distribution diverge également de la loi énoncée à partir d'un certain rang  $n$ , mais les rangs suivants suivent une loi purement Zipfienne, on les ajuste par introduction d'une seconde composante de fonction puissance translatée (on se rapproche ainsi d'une *broken power law*), bien qu'il soit possible d'unifier la totalité de la distribution par des lois nettement plus complexes [77, 79, 78]. Les termes appartenant à cette queue de distribution sont typiquement écartés des études en TALN, en tout cas en l'état, car au delà de capturer la taille du dictionnaire, ces termes ne sont pas directement exploitables du fait de leur rareté.

Nous retrouvons donc pour les langages naturels un comportement similaire à celui que nous venons

d'observer sur les fragments. Pour confirmer cette impression, nous analysons le corpus Gutenberg, et traçons en Fig. 2.5 la fréquence du mot en fonction de son rang. Nous faisons figurer à gauche le même exercice sur les fragments de taille 2 issus du corpus Pubchem, et constatons que l'allure des courbes est bel et bien voisine, aux ordres de grandeur près.

### 2.2.3 Formulation de l'analogie

Cette similarité empirique entre les distributions observées sur des fragments moléculaires et les mots d'un langage naturel nous amène à formuler l'hypothèse suivante : si les fragments moléculaires sont analogues à des mots (lexèmes), tout comme les molécules sont analogues à des phrases, peut-on poursuivre cette analogie en appliquant aux fragments moléculaires certains modèles utilisés en TALN pour contourner les difficultés induites par les projections de type "dictionnaire", en particulier les techniques de réduction de dimensions à base de plongements lexicaux des termes ?

## 2.3 Construction de plongements lexicaux

Nous venons de constater la difficulté de construire des modèles d'apprentissage basés sur des représentations de termes par dictionnaire, du fait des distributions particulières auxquels ces termes se rapportent. Les représentations de termes orientées dictionnaire, si elles font bien entendu partie de la bibliographie du TALN, ont depuis une dizaine d'années laissé place aux représentations apprises, orientées *plongements lexicaux*, que nous allons détailler dans cette section entièrement bibliographique. Nous aborderons dans un second temps les quelques adaptations existantes proposées pour les *plongements de graphe*. Enfin, l'approche développée et utilisée dans le cadre de nos prochaines expériences étant hybride, elle fera l'objet de la prochaine section.

### 2.3.1 Les plongements lexicaux en TALN

Le besoin de pouvoir représenter un terme autrement qu'avec un vecteur creux symbolisant une entrée dans un dictionnaire provient de deux limitations intrinsèques à cette dernière solution :

- les dictionnaires de très grande taille engendrent des plongements en très grande dimension (fléau de la dimension)
- les vecteurs sont de type "*one-hot-encoding*" (une seule valeur non-nulle)

Si nous avons à présent bien compris quels sont les problèmes liés à un nombre de dimensions élevé, il convient de préciser que le fait de n'avoir qu'une seule dimension non-nulle pour représenter un mot pose aussi problème : de cette façon, deux termes d'un dictionnaire ne sont jamais comparables, car représentés par des modalités différentes qui ne sont pas plus informatives qu'un emplacement dans un dictionnaire. Le mot "*chat*" n'est ainsi en aucune façon plus proche du mot "*chien*" ou "*félin*" que du "*pomme*", bien qu'un locuteur français saurait rapprocher ces mots de façon pertinente, car partageant un *concept* commun. Aucun portion de *sémantique* lexicale n'est ainsi conservé par une projection directement basée dictionnaire.

Au contraire, avec une représentation des mots conservant une partie de la sémantique lexicale, deux mots pourraient être comparables de manière plus fine (similarité non binaire), autrement dit il serait possible de définir une ou plusieurs *distances* dans un espace métrique, et une telle représentation serait d'autant plus pertinente que la distance entre "*chat*" et "*chien*" serait plus faible que la distance entre "*chat*" et "*pomme*". Au-delà de la notion de distance, un espace vectoriel idéal ou ces

mots seraient projetés permettrait également de déduire des vecteurs de transformation en additionnant ou soustrayant plusieurs mots, formant ainsi une notion d'arithmétique. On s'attend ainsi à ce que ("roi" – "homme") + "femme" = "reine" et ("Paris" – "France") + "Irlande" = "Dublin".

Pour établir et apprendre le sens, voire les concepts véhiculés par un terme, à défaut de comprendre le langage, il est nécessaire de disposer d'exemples d'utilisation de ce terme, et donc de l'étudier "*en contexte*". Il s'agit là de l'hypothèse fondamentale des plongements lexicaux : deux termes utilisés dans le même contexte, et ayant donc un environnement lexical proche, devraient être porteurs d'une sémantique tout aussi proche. Pour apprendre la meilleure représentation d'un terme, nous l'étudions donc "*plongé*" au sein de son environnement lexical type, autrement dit en tenant compte des autres termes l'accompagnant régulièrement. Le plongement résultant de cet apprentissage devient ainsi un vecteur "dense" et caractérisé par un nombre de dimensions réduit. Nous allons à présent décrire deux méthodes de calcul de plongements parmi les plus répandues dans la bibliographie, qui toutes deux atteignent les objectifs évoqués précédemment, rendant leur utilisation tout à fait pertinente en TALN.

### Modèle *Word2Vec*

La première méthode de vectorisation de mots en plongements lexicaux s'intitule *Word2Vec* [80]. Son principe repose sur le concept général d'auto-encodeur en général implémenté sous la forme de réseaux de neurones : deux variantes existent pour ce modèle. Pour apprendre ces architectures, il est nécessaire de disposer d'un large corpus de phrases exprimées dans un langage naturel, desquelles on extrait toutes les séquences de termes capturées par une fenêtre glissante de taille  $2n$  centrée sur chaque terme. On associe ainsi à chaque terme  $t$  une séquence  $S_t = \langle t_{-n}, \dots, t_{-1}, t_{+1}, \dots, t_{+n} \rangle$  caractérisant son contexte d'occurrence.

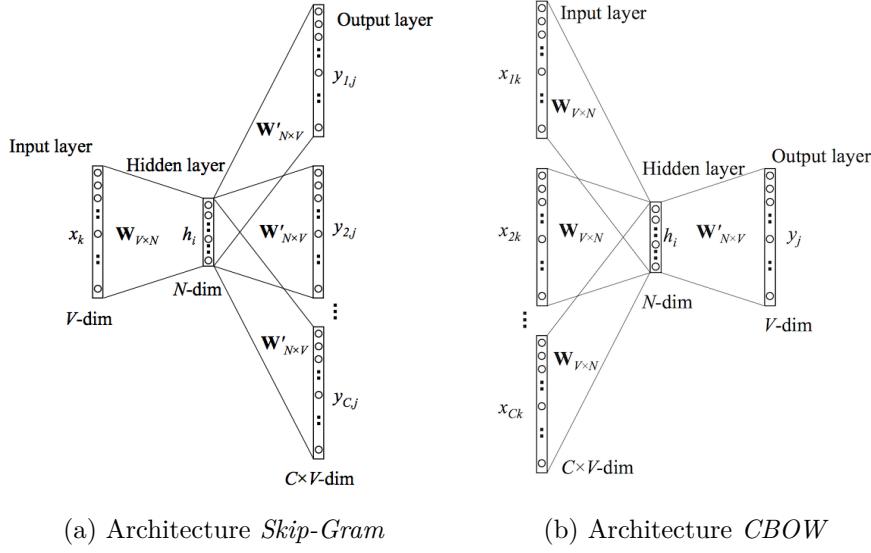
Dans la première variante, nommée *Skip-Gram*, on cherche à prédire cette séquence  $S_t$  à partir de  $t$ . On entraîne pour ce faire un réseau de neurones, constitué d'une couche d'entrée de type "*one-hot-encoding*" permettant de cibler le terme en question sur un dictionnaire, et d'une couche de sortie, séparée en  $2n$  sous-couches donnant  $2n$  prédictions, que l'on présente sous la forme de probabilités (grâce à une fonction *soft-max*), de chaque terme constituant la séquence  $S_t$ .

Une optimisation est faite à l'échelle du corpus à l'aide de méthodes d'optimisation classique des réseaux de neurones (descente de gradient stochastique), tirant éventuellement parti de la nature éparse des couches de sortie (*negative sampling*) [81].

Deux termes observés régulièrement dans des contextes lexicaux similaires vont donc tendre à avoir une projection proche sur la couche cachée, du fait de l'optimisation réalisée. En effet, si le contexte que l'on prédit pour deux termes pourtant caractérisés par des représentations différentes sur la couche d'entrée donnent les mêmes prédictions, c'est que leur projection sur la couche cachée est similaire. Cette couche cachée possède donc les caractéristiques que l'on attend d'un plongement lexical, et permet de représenter un terme tout en captant une partie de la sémantique lexicale qu'il véhicule.

La seconde variante est assez similaire, mais, en quelque sorte, celle-ci appréhende le problème dans le sens contraire : on se sert du contexte  $S_t$  pour prédire le terme  $t$  courant. Cette variante se nomme *CBOW* (*Continuous Bag Of Words*), et donne dans les faits des résultats assez similaires au modèle *Skip-Gram*, bien que l'on observe en pratique qu'il est en général plus adéquat que le modèle *Skip-Gram* pour représenter des mots courants, mais également moins pertinent sur les mots rares. Il nécessite également plus de données d'entraînement, mais s'entraîne plus vite.

On illustre en Fig. 2.6 les deux architectures proposées dans *Word2Vec*.


 FIGURE 2.6 – Les deux types d’architectures décrites dans *Word2Vec* (source : [80])

### Modèle *GloVe (Global Vectors)*

Bien que les plongements issus de *Word2Vec* (ou ses variantes étendant les vectorisations aux documents entiers [82]) démontrent d’excellentes performances lorsqu’on les utilise dans des tâches de classification/régression de texte (analyse de sentiments par exemple), certains aspects de l’entraînement peuvent aller à l’encontre de faits connus en linguistique. En particulier, on ne tient pas compte lors de la construction de ces plongements des nombres d’occurrence des termes dans la langue, autrement dit de la rareté intrinsèque d’un terme. Il y aurait donc une perte d’information liée à la non-utilisation des statistiques lexicales estimées sur le corpus d’entraînement.

Le modèle *GloVe (Global Vectors)* [83] propose de pallier à ce problème en ne construisant pas les plongements à partir d’un modèle prédictif, mais plutôt à partir de la matrice de co-occurrence terme/contexte, préservant ainsi les informations statistiques inhérentes au corpus. Cette matrice étant typiquement gigantesque, celle-ci est factorisée en matrices de dimension arbitraire moindre (typiquement par décomposition en valeurs singulières), projetant ainsi la ligne caractéristique de chaque terme sur un vecteur de cette dimension arbitraire. Le principe est jusqu’ici très proche de l’analyse de sémantique latente (*LSA*) [84]. Dans *GloVe* toutefois, cette réduction de dimension est soumise à un apprentissage, et la meilleure décomposition possible sera apprise par rapport à une tâche de classification ou de régression en cherchant à maximiser la variance expliquée par ces plongements.

Dans les faits, ces modèles démontrent des performances très proches [86, 85], *GloVe* et *Word2Vec* étant tous deux significativement meilleurs que *LSA* dès que la taille du corpus est assez grande (de l’ordre du million de mots).

### 2.3.2 Les plongements adaptés aux graphes

L’émergence de ces techniques de plongement a eu un impact très fort en TALN, les vectorisations produites étant de manière générale très performantes dans les problématiques adressées (correspondance de mots entre langues pour traduction automatique, analyse de sentiments, compréhension de texte, etc.).

C'est ainsi que ces techniques de plongement ont été récemment généralisées aux graphes, toutefois sans toujours se fonder sur les similitudes de distribution observées entre les domaines traités par les graphes et les langages naturels. Nous présentons ici les adaptations existantes, dont nous nous inspirerons par la suite pour proposer notre méthode de plongement spécifiquement dédiée au traitement des fragments moléculaires.

On peut catégoriser ces méthodes selon l'objet ciblé par ces plongements.

## Plongements de vertices

On distingue premièrement les méthodes créant des plongements pour les noeuds d'un graphe. Chronologiquement, *DeepWalk* a constitué le premier modèle proposé [87]. L'idée est la suivante : partant du noeud pour lequel on souhaite calculer un plongement, on génère plusieurs échantillons de marches aléatoires de longueur fixe partant de ce noeud. Ces parcours aléatoires fournissent des séquences de labels issus des noeuds parcourus, qui peuvent alors alimenter un modèle *Skip-Gram*, et permet de s'affranchir du caractère non-linéaire des graphes, pour peu que l'échantillonnage et la longueur des parcours soient suffisants (ceux-ci constituent des paramètres du modèle).

Le comportement purement aléatoire de ce parcours est modifié dans le modèle *Node2Vec* [28], avec l'introduction de paramètres  $p$  et  $q$  influant sur la probabilité de retourner sur ces pas en arrivant sur un noeud, ou au contraire la probabilité de s'éloigner du noeud initiateur du parcours. Il est ainsi possible de favoriser soit un apprentissage du voisinage proche du noeud, soit un voisinage plus global.

Si ces deux méthodes permettent de composer avec le caractère non-linéaire des graphes pour obtenir des séquences, elles ne résolvent en revanche pas le fait que les relations entre noeuds peuvent ne pas être homogènes, comme c'est le cas pour les arêtes colorées. Par ailleurs, la vectorisation effectuée ne porte que sur un noeud unique, pris dans le contexte d'un graphe unique, et ne permettrait donc pas d'acquérir une connaissance sur des fragments. Ces modèles sont donc typiquement utilisés pour des tâches de classification ou de partitionnement de noeuds, et non de graphes entiers.

## Plongements de sous-graphes

Sur une échelle intermédiaire, on dispose du modèle *subgraph2vec* [27], apportant plusieurs innovations par rapport aux méthodes travaillant à l'échelle d'un noeud.

Premièrement, un noeud n'est plus représenté par son label, mais plutôt par le sous-graphe enraciné sur ce noeud. On embarque donc sur chaque noeud l'information de son voisinage, rapporté à ce sous-graphe de taille arbitraire. Pour déterminer les différentes modalités de sous-graphe, une fonction noyau ou *kernel* est utilisée soit par comparaisons deux-à-deux avec les sous-graphes déjà rencontrés, soit par calcul d'un invariant similaire à une fonction hashcoding, par exemple une énumération ordonnée des noeuds, répondant à une grammaire (algorithme *gspan* [67]).

Dans la description de cet algorithme et son implémentation disponible, le *kernel* dit de *Weisfeiler-Lehman* [88] est utilisé. Bien qu'il ne fasse pas partie des *kernels* les plus coûteux<sup>6</sup> à l'usage, celui-ci ne permet pas toutefois de prendre en compte à la fois des noeuds et des arêtes colorés, ce qui constitue sa principale limite [89]. Rien ne laisse toutefois supposer qu'un autre *kernel* ne puisse être utilisé dans la procédure. Cependant, les *kernels* permettant de prendre en compte à la fois les noeuds et arêtes labellisés sont soit coûteux, car tout à fait analogues à un test d'isomorphisme complet, soit approximatifs (typiquement sujets à apprentissage [29]), renforçant par ailleurs l'intérêt évoqué dans §1.6.3 que l'on pourrait avoir à étendre l'usage de l'algorithme

---

6. le résultat est ré-utilisable et calculable en  $O(n^2)$ , avec  $n$  le nombre de noeuds du graphe

SCOTT au calcul de similarité entre deux graphes. Un récapitulatif des *kernels* existants est réalisé dans [90, 91].

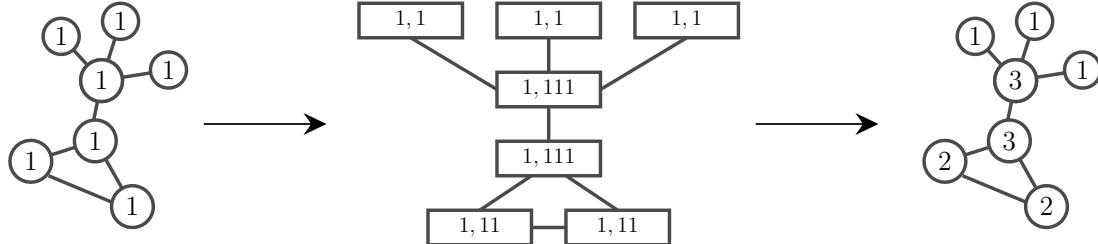


FIGURE 2.7 – Une itération du processus de coloration *Weisfeiler-Lehman*

La seconde innovation apportée par cette méthode concerne l’architecture *Skip-Gram* utilisée par la suite. Celle-ci est en effet adaptée de telle sorte que le contexte relatif à un terme (ici un sous-graphe) ne se présente pas sous la forme d’une séquence de termes (de sous-graphes), mais de l’ensemble du *k-voisinage*, avec  $k$  la distance maximum à considérer depuis le noeud source, autrement dit le *Bag of Fragments* formé par l’agrégation des sous-graphes portés par ses voisins. Cette modification, nommée *Radial Skip-Gram*, apporte une solution alternative aux marches aléatoires pour composer avec la non-linéarité des graphes, mais soulève d’autres questions, telles que la gestion des termes (sous-graphes) observés plusieurs fois dans le *k*-voisinage, et sur la manière de pondérer les termes (sous-graphes) par rapport à leur distance au noeud source. Enfin, la perte d’information locale apparente due au paradigme *Bag of Fragments* pourrait s’avérer préjudiciable comparativement aux approches basées parcours aléatoires. Cette comparaison n’est malheureusement pas traitée dans l’article original. On illustre en Fig. 2.8 la construction d’un contexte de taille 2 d’un fragment  $\chi_6 S_{\chi_6}^2$ , qui sera fourni au *Skip-Gram* lors de l’entraînement. Chaque élément de ce contexte n’est pas un simple terme, mais bien un *Bag of Fragments*.

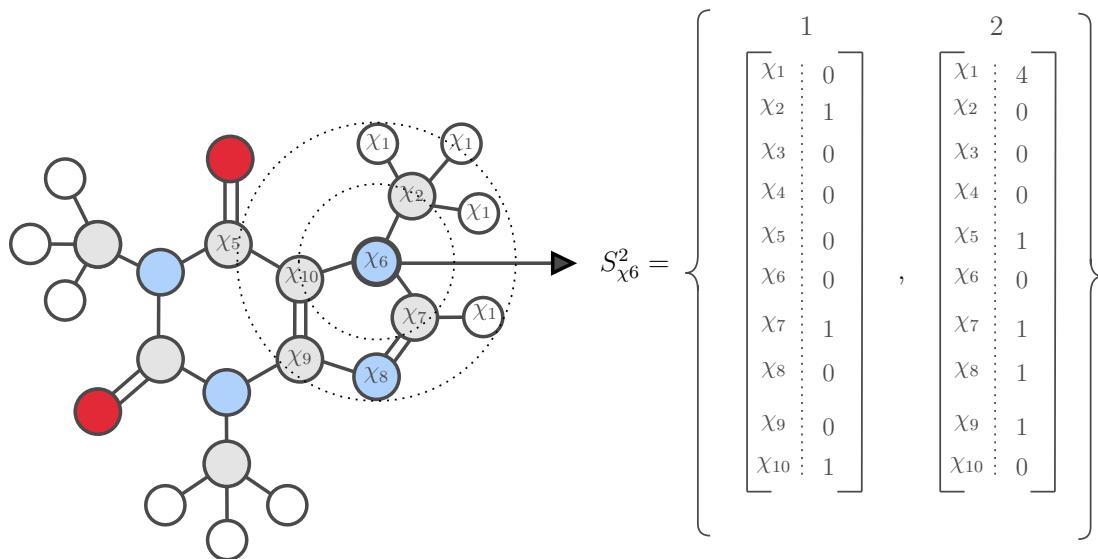


FIGURE 2.8 – Construction d’un contexte de fragment dans le cadre du *Radial Skip-Gram*

## Plongements de graphes

Enfin, il est possible de vectoriser un graphe entier, sur la base des fragments évoqués dans *subgraph2vec* et du principe de *doc2vec*. Il s'agit donc d'un modèle prédictif cherchant à prédire la composition en terme de fragments d'un graphe par rapport à un identifiant artificiel, nommée *graph2vec* [36].

Si cette méthode permet bien d'obtenir une vectorisation du graphe dans son intégralité, et donc de traiter des tâches portant sur ces graphes entiers, elle ne conserve en revanche pas la structure locale du graphe. En effet, la couche de sortie s'apparente à l'ensemble des fragments constituant le graphe, soit un *Bag of Fragments*. Ce modèle est donc équivalent à un *Auto-Encodeur* d'un *Bag of Fragments*. Par ailleurs, comme pour le modèle précédent, la détermination des classes d'isomorphisme pour identifier les modalités de fragments repose sur le test de *Weisfeiler-Lehman*, ne prenant pas en compte les arêtes colorées.

## 2.4 Adaptation aux fragments moléculaires

Nous venons de passer en revue le concept original des plongements lexicaux en TALN, ainsi que leurs plus récentes adaptations aux (sous-)graphes. Aucune ne satisfaisant malheureusement toutes nos contraintes, en particulier la prise en compte des graphes colorés (noeuds et arêtes), nous avons opté pour une approche hybride que nous détaillons ci-après. Nous expliquons par la suite sur quel corpus nous réalisons l'apprentissage, et quelles observations il est possible d'en tirer en amont de la construction de modèles QSAR.

### 2.4.1 Plongements de fragments moléculaires

Malheureusement, aucune méthode existante ne s'est avérée correspondre entièrement à nos attentes, d'autant plus qu'il était impossible de toutes les soumettre à un *benchmark* dans le cadre de cette thèse. Nous attendons en effet d'une méthode de plongement de fragments moléculaires que celle-ci préserve au maximum la structure locale, qu'elle gère les labels portant à la fois sur les noeuds du graphe, mais aussi sur ses arêtes. Nous rappelons que la finalité des modèles exploitant ces plongements est de fournir un modèle de régression/classification de graphes entiers, et non de noeuds ou de sous-graphes. Les corpus de graphes que nous devons traiter s'apparentent également à des ensembles composés de nombreux graphes de taille modeste, et non un immense graphe unique. Les plongements produits doivent donc être ré-utilisables d'un graphe à l'autre, et non centré sur un seul.

L'échelle à laquelle nous choisissons de travailler est celle des sous-graphes, développée par *subgraph2vec*, l'échelle des graphes ne nous paraissant pas significativement différente d'un *Bag of Fragment*. Les plongements de sous-graphes ne seront pas directement exploitables, mais certaines méthodes que nous préciserons dans le chapitre suivant permettent de s'affranchir d'une vectorisation globale du graphe. À la différence du modèle *subgraph2vec*, nous n'utilisons toutefois pas un *kernel* pour déceler les différentes modalités de sous-graphes, mais directement les fragments canoniques issus de l'algorithme SCOTT, évitant un grand nombre de tests deux-à-deux pour déterminer les modalités de sous-graphe, mais surtout permettant de traiter à la fois des noeuds et arêtes colorés. Chaque noeud est ainsi porteur d'un fragment issu de l'algorithme SCOTT de taille arbitraire.

On priviliege des fragments de taille modeste (jusqu'à  $t = 3$ ), pour éviter d'avoir une un dictionnaire constitué de plusieurs dizaines de millions de fragments observés. Si ces fragments sont bien à la base de la structure locale du graphe, on comptera davantage sur les séquences de fragments

obtenues pour faire état de l'information d'échelle intermédiaire, située entre structures locale et globale.

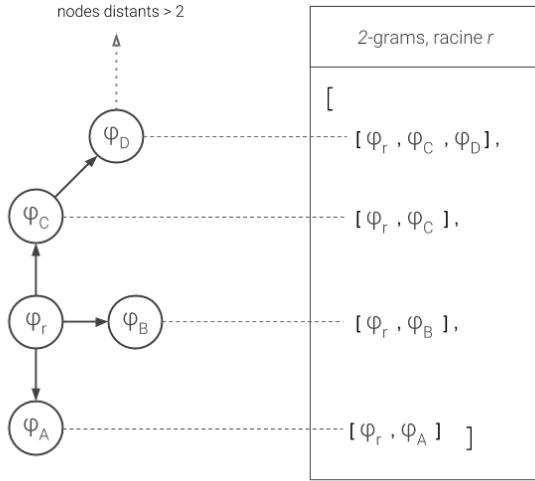


FIGURE 2.9 – Principe de formation de *n-grams* à partir des fragments issus de SCOTT

Enfin, nous n'optons pas pour le *Radial Skip-Gram* à base de *k*-voisinage pour apprendre les plongements, ces méthodes ne nous paraissant pas des plus pertinentes dans le contexte des graphes moléculaires, qui sont finalement des structures assez linéaires aux embranchements à la fois relativement rares à l'échelle d'une molécule, et en tout cas de degré faible (un atome ne pouvant qu'exceptionnellement être associé à un degré supérieur à 4). Il est donc possible d'être exhaustif sur l'énumération des séquences de fragments issus d'un parcours depuis un noeud quelconque. Nous répertorions ainsi les séquences des fragments observés sur un graphe, nommées *n-grams*, comme illustré en Fig. 2.9. La notion d'antécérence n'ayant pas de sens sur un graphe, on veille toutefois à considérer ces *n-grams* dans les deux sens de parcours des séquences (du début vers la fin et réciproquement) au moment de les fournir à un modèle *Skip-Gram* ou un *CBOW*.

## 2.4.2 Protocole d'apprentissage

Nous reprenons le corpus de graphes moléculaires (1'510'000 graphes) issus de la base *PubChem* utilisé précédemment pour l'étude distributionnelle des fragments. Nous en extrayons les *n-grams* décrits ci-dessus, et construisons les plongements à l'aide des modèles *Skip-Gram* et *CBOW* de la librairie *gensim* [92] disponible en Python.

Nous faisons varier les paramètres suivants pour l'apprentissage des plongements :

**fragment\_size** ∈ {1, 2} Taille des fragments considérés

**window\_size** ∈ {1, 2, 3, 4, 5} Étendue du voisinage considérée pour les *n-grams*

**embedding\_size** ∈ {50, 100, 200, 400, 800, 5000} Taille du plongement produit

**model** ∈ {*SkipGram*, *CBOW*} Architecture sur laquelle sont appris les plongements

À l'issue d'un apprentissage constitué de 25 itérations (*epochs*), nous obtenons ainsi des plongements de fragments moléculaires. Les tâches typiques des QSAR portant sur les graphes entiers, ceux-ci ne sont pas directement exploitables : les approches naïves que sont une somme ou une

moyenne des plongements de fragments constituant un graphe ne se sont pas révélés suffisamment "explicateurs" de variance sur une de ces tâches. D'autre-part, la concaténation des vecteurs associés aux fragments est impossible du fait du nombre variable de fragments constituant un graphe.

### 2.4.3 Requêtes par rapport à un fragment de référence

Au préalable de la construction de modèles QSAR qui feront l'objet du prochain chapitre, il est possible de réaliser quelques expériences pour vérifier l'existence des propriétés attendues des plongements obtenus, qui s'expriment notamment en matière de proximité "sémantique".

La première expérience que nous proposons consiste à évaluer les fragments jugés les plus proches (les *top-réponses*), au sens de la *similarité cosinus*, par rapport à un fragment donné comme référence. La similarité cosinus entre deux vecteurs  $u$  et  $v$  de dimension  $n$  s'exprime comme le cosinus de l'angle  $\theta$  entre ces deux vecteurs :

$$\text{similarity}(u, v) = \cos(\theta) = \frac{u \cdot v}{\|u\| \|v\|} = \frac{\sum_{i=1}^n u_i \cdot v_i}{\sqrt{\sum_{i=1}^n u_i^2} \cdot \sqrt{\sum_{i=1}^n v_i^2}}$$

L'utilisation de cette métrique est répandue en TALN, et plus généralement dans les espaces de grande dimension, car on priviliege la direction des vecteurs plutôt que leur norme, cette dernière n'étant que peu significative du fait du fléau de la dimension.

Nous nous intéressons particulièrement aux fragments (de taille 1) incorporant des atomes appartenant à la famille des halogènes, soient les éléments chimiques Fluor (F), Chlore (Cl), Brome (Br) et Iode (I). Ces éléments chimiques ont des propriétés et comportements chimiques très homogènes, du fait de leur forte affinité électronique. Étant donnés quelques-uns de ces fragments, nous cherchons parmi les top-réponses des fragments présentant des substitutions atome pour atome avec un autre élément de la famille des halogènes.

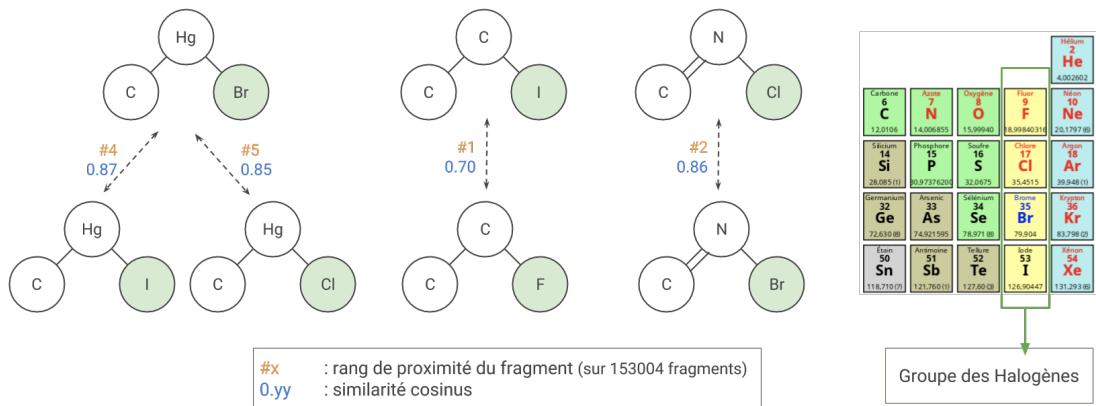


FIGURE 2.10 – Cas de substitution d'halogènes observés sur des fragments proches d'un fragment de référence

Nous présentons en Fig. 2.10 quelques exemples correspondant à cette expérience requête-réponse. Les fragments situés en haut de la figure représentent les fragments de référence, auxquels on fait correspondre les fragments présentant une substitution d'halogène. Nous indiquons à la fois la

similarité cosinus entre les-dits fragments, ainsi que le rang de ce fragment réponse, parmi les 3533 fragments connus de cette taille. On constate que dans chaque cas on trouve dans les dix premiers résultats des fragments impliquant les substitutions attendues. La similarité cosinus de deux de ces fragments respectifs est également d'autant plus forte que les halogènes en question ont un numéro atomique proche (ligne du tableau périodique). Ces plongements de fragments semblent ainsi capter une cohérence chimique.

#### 2.4.4 Densification et projection des plongements

Nous proposons une seconde expérience, toujours avec les mêmes plongements de fragments. Nous utilisons une technique de réduction de dimension probabiliste, la *t-SNE* (*t-distributed stochastic neighbor embedding*) [93], permettant de projeter dans un espace de dimension arbitraire (typiquement le plan de dimension 2 ou l'espace de dimension 3), de façon à préserver au maximum les proximités de l'espace en haute dimension dans ce nouvel espace vectoriel. Il s'agit donc d'appliquer une réduction du nombre de dimensions à des plongements déjà construits dans ce but, mais cette fois dans un but de visualisation (et donc sur un nombre de dimensions très restreint). Cet algorithme est bien connu pour proposer une segmentation dans l'espace pertinente des plongements lexicaux de mots [94] ou même de paragraphes [95].

Nous utilisons cette fois encore la similarité cosinus en tant que métrique de référence dans l'espace original (l'espace des plongements). Nous présentons en Fig. 2.11 l'état de ces projections dans le plan après 500 itérations. Étant donné un fragment de référence ( $C:1:1$ )As comportant du Chlore et de l'Arsenic, nous observons que les 60 top-réponses<sup>7</sup> sont des fragments incluant l'élément Arsenic, mais se situent toutes dans la même région du plan, et forment même un regroupement assez net de fragments incorporant l'élément Arsenic, entouré en bleu sur le schéma.



FIGURE 2.11 – Projection dans l'espace des plongements de fragments réduits par *t-SNE*

En regardant de plus près les regroupements formés dans le plan, on constate que les fragments

7. au sens de la similarité cosinus dans l'espace d'origine, soit la même expérience que précédemment

associés à ces regroupements partagent tous en commun un élément chimique ou un liaison caractéristique. On indique près de chaque groupe sa caractéristique chimique discriminante. L'expérience est concluante, et il semble bien que les plongements produits aient capté la sémantique des fragments contextualisés, ce qui devrait être confirmé lors de l'élaboration de modèles QSAR.

## 2.5 Conclusion

Dans le contexte de ce chapitre nous avons initié nos travaux autour de l'hypothèse selon laquelle une analogie pouvait être établie entre les langages naturels et la description de graphes représentés sous la forme de séquence de fragments, les fragments jouant potentiellement le rôle "d'unités lexicales". Partant de cette hypothèse, nous avons mis en évidence des similitudes distributionnelles entre les fragments chimiques extraits d'un grand nombre de molécules répertoriées et les termes composant un langage naturel (langues européennes).

Sur la base de ces similitudes, nous avons formulé une analogie entre ces deux domaines, faisant l'hypothèse que les fragments sont assimilables à des termes, composant des molécules assimilables à des phrases ou des documents. Après avoir réalisé une étude bibliographique des méthodes dites de plongements lexicaux et de plongements de sous-graphes, nous avons mis au point notre propre méthode de calcul de plongements de fragments moléculaires adaptés aux graphes colorés. L'algorithme Scott que nous avons développé est ici exploité pour identifier les fragments isomorphes et les représenter de manière canonique (unique). Nous avons testé avec succès ces plongements de fragments sur deux expériences typiques connues en TALN, à savoir l'étude de similarité d'un terme devant révéler des termes proches, et une projection des plongements produits dans le plan ou l'espace par *t-SNE*. Les fragments produits semblent donc être porteurs d'une information de plus haut niveau, qui relève d'une forme de sémantique lexicale, qu'il est possible d'acquérir sur un corpus assez large.

Dans le cadre de cette thèse, nous n'avons pas pu en revanche explorer toutes les possibilités de construction de plongements, en jouant notamment sur leur taille, et sommes restés concentrés sur nos hypothèses de départ, les plongements en question présentant de bonnes propriétés dans les expériences décrites, notamment concernant les substitutions d'atomes constatées. Il serait toutefois possible d'augmenter encore la taille des fragments utilisés dans les séquences, de même que la taille de ces séquences. Il serait également intéressant de mesurer l'influence de la méthode *Radial Skip-Gram*, ainsi que d'adapter le modèle *GloVe* plutôt que *Word2Vec*, les statistiques brutes du corpus renfermant peut-être une information supplémentaire qui reste inaccessible pour des fragments de petite taille.

## Références

- [9] Antoine DAINA, Olivier MICHELIN et Vincent ZOETE, « iLOGP : a simple, robust, and efficient description of n-octanol/water partition coefficient for drug design using the GB/SA approach », in : *Journal of chemical information and modeling* 54.12 (2014), p. 3284-3301.
- [11] Sunghwan KIM et al., « PubChem substance and compound databases », in : *Nucleic acids research* 44.D1 (2015), p. D1202-D1213.

- [19] Fiorella RUGGIU et al., « ISIDA Property-Labelled Fragment Descriptors », in : *Molecular informatics* 29.12 (2010), p. 855-868.
- [27] Annamalai NARAYANAN et al., « subgraph2vec : Learning distributed representations of rooted sub-graphs from large graphs », in : *arXiv preprint arXiv :1606.08928* (2016).
- [28] Aditya GROVER et Jure LESKOVEC, « node2vec : Scalable feature learning for networks », in : *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, p. 855-864.
- [29] David K DUVENAUD et al., « Convolutional networks on graphs for learning molecular fingerprints », in : *Advances in neural information processing systems*, 2015, p. 2224-2232.
- [36] Annamalai NARAYANAN et al., « graph2vec : Learning Distributed Representations of Graphs », in : *arXiv preprint arXiv :1707.05005* (2017).
- [67] Xifeng YAN et Jiawei HAN, « gspan : Graph-based substructure pattern mining », in : *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, IEEE, 2002, p. 721-724.
- [72] Karen Spärck JONES, « A statistical interpretation of term specificity and its application in retrieval », in : *Journal of documentation* (2004).
- [73] Christophe GIRAUD, *Introduction to high-dimensional statistics*, Chapman et Hall/CRC, 2014.
- [74] Mark EJ NEWMAN, « Power laws, Pareto distributions and Zipf’s law », in : *Contemporary physics* 46.5 (2005), p. 323-351.
- [75] Benoît B MANDELBROT, *Linguistique statistique macroscopique*, 1957.
- [76] Marc BARBUT, « Note sur l’ajustement des distributions de Zipf-Mandelbrot en statistique textuelle », in : *Histoire & Mesure* (1989), p. 107-119.
- [77] Matthieu CRISTELLI, Michael BATTY et Luciano PIETRONERO, « There is more than a power law in Zipf », in : *Scientific reports* 2 (2012), p. 812.
- [78] Isabel MORENO-SÁNCHEZ, Francesc FONT-CLOS et Álvaro CORRAL, « Large-scale analysis of Zipf’s law in English texts », in : *PloS one* 11.1 (2016), e0147073.
- [79] Steven T PIANTADOSI, « Zipf’s word frequency law in natural language : A critical review and future directions », in : *Psychonomic bulletin & review* 21.5 (2014), p. 1112-1130.
- [80] Tomas MIKOLOV et al., « Efficient estimation of word representations in vector space », in : *arXiv preprint arXiv :1301.3781* (2013).

- 
- [81] Tomas MIKOLOV et al., « Distributed representations of words and phrases and their compositionality », in : *Advances in neural information processing systems*, 2013, p. 3111-3119.
  - [82] Quoc LE et Tomas MIKOLOV, « Distributed representations of sentences and documents », in : *International Conference on Machine Learning*, 2014, p. 1188-1196.
  - [83] Jeffrey PENNINGTON, Richard SOCHER et Christopher MANNING, « Glove : Global vectors for word representation », in : *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, p. 1532-1543.
  - [84] Thomas K LANDAUER, Peter W FOLTZ et Darrell LAHAM, « An introduction to latent semantic analysis », in : *Discourse processes* 25.2-3 (1998), p. 259-284.
  - [85] Marwa NAILI, Anja Habacha CHAIBI et Henda Hajjami Ben GHEZALA, « Comparative study of word embedding methods in topic segmentation », in : *Procedia computer science* 112 (2017), p. 340-349.
  - [86] Edgar ALTSZYLER et al., « Comparative study of LSA vs Word2vec embeddings in small corpora : a case study in dreams database », in : *arXiv preprint arXiv :1610.01520* (2016).
  - [87] Bryan PEROZZI, Rami AL-RFOU et Steven SKIENA, « Deepwalk : Online learning of social representations », in : *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, p. 701-710.
  - [88] Nino SHERVASHIDZE et al., « Weisfeiler-lehman graph kernels », in : *Journal of Machine Learning Research* 12.Sep (2011), p. 2539-2561.
  - [89] Christopher MORRIS et al., « Weisfeiler and leman go neural : Higher-order graph neural networks », in : *Proceedings of the AAAI Conference on Artificial Intelligence*, t. 33, 2019, p. 4602-4609.
  - [90] Swarnendu GHOSH et al., « The journey of graph kernels through two decades », in : *Computer Science Review* 27 (2018), p. 88-111.
  - [91] Nils M KRIEGE, Fredrik D JOHANSSON et Christopher MORRIS, « A Survey on Graph Kernels », in : *arXiv preprint arXiv :1903.11835* (2019).
  - [92] Radim ŘEHŮŘEK et Petr SOJKA, « Software Framework for Topic Modelling with Large Corpora », English, in : *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, <http://is.muni.cz/publication/884893/en>, Valletta, Malta : ELRA, mai 2010, p. 45-50.
  - [93] Laurens van der MAATEN et Geoffrey HINTON, « Visualizing data using t-SNE », in : *Journal of machine learning research* 9.Nov (2008), p. 2579-2605.

- [94] Will Y ZOU et al., « Bilingual word embeddings for phrase-based machine translation », in : *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, p. 1393-1398.
- [95] Andrew M. DAI, Christopher OLAH et Quoc V. LE, « Document embedding with paragraph vectors », in : *NIPS Deep Learning Workshop*, 2015.
- [96] Gerard SALTON et Christopher BUCKLEY, « Term-weighting approaches in automatic text retrieval », in : *Information processing & management* 24.5 (1988), p. 513-523.
- [97] Chris BUCKLEY, Gerard SALTON et James ALLAN, « Automatic retrieval with locality information using SMART », in : *Proceedings of the First Text REtrieval Conference TREC-1*, 1993, p. 59-72.

# Exploitation des plongements de fragments dans les Qsar

---

Nous disposons à présent de plongements de fragments moléculaires qui semblent porteurs d'une connaissance d'ordre chimique intéressante à valoriser. Pour autant, la manière d'exploiter ces fragments dans le cadre d'une tâche de prédiction de propriétés physico-chimiques n'est pas immédiate : il reste en effet à déterminer comment passer d'une information locale (les fragments), à une information globale portant sur un graphe moléculaire entier, l'échelle de référence pour la prédiction de propriétés physico-chimiques.

Nous dressons dans un premier temps un état de l'art des méthodes existantes pour caractériser un graphe, en nous concentrant sur le sous-ensemble des méthodes qui exploitent des fragments (sous-graphes). Nous ferons par la suite une présentation des jeux de données que nous avons à disposition pour élaborer ces modèles structure-activité, pour enfin évaluer la performance comparée des modèles que nous proposons sur ces jeux de données, au regard des meilleurs résultats publiés dans l'état de l'art. Nous verrons que les plongements proposés sont bien exploitables pour ces tâches de prédiction, et se révèlent même très concurrentiels vis-à-vis de l'état de l'art.

## 3.1 Caractérisation de graphe

Nous effectuons dans cette section un état de l'art synthétique des méthodes permettant de caractériser ou de représenter un graphe de telle sorte que les caractérisations ou représentations proposées puissent être exploitées dans un modèle prédictif. Il est possible de dégager trois grands axes dans la manière d'appréhender ces caractérisations :

- sous la forme d'un graphe, la fonction de similarité entre graphes étant appréhendée en général via une distance d'édition, ou d'un noyau,
- sous la forme d'un ensemble de sous-graphes (sac de fragments)
- sous la forme d'un réseau de noeuds

### 3.1.1 Graphes et mesures de similarité définies sur l'ensemble des graphes

La première manière de caractériser un graphe est de le considérer comme tel, l'ensemble des graphes étant muni d'une métrique permettant de comparer des graphes deux à deux. L'apprentissage pour une tâche de prédiction se fera donc sur la base de cette distance, par rapport à d'autres graphes de référence, ces derniers étant porteurs de l'information discriminante utile (constituant une observation).

C'est également le principe des méthodes dites *à noyau*, qui exploitent cette notion de similarité, exprimée sous forme d'une fonction *kernel*, se substituant à une distance. La méthode à noyau la plus classique est la *Support Vector Machine (SVM)*, projetant les observations dans un espace de dimension supérieure dans lequel on espère que le problème sera plus simple à traiter (linéairement par exemple). Si on utilise dans les faits souvent les mêmes noyaux (linéaires, polynomiaux, gaussiens, ...) pour les données numériques, ce degré de liberté rend cette méthode très versatile, et même "sur mesure" si l'on dispose d'un kernel spécifique. [98]

On peut soit calculer ces distances d'édition ou noyau par rapport au graphe brut [99], soit par rapport à un représentant de sa classe d'isomorphisme, par exemple l'arbre calculé par l'algorithme SCOTT. En plus de rendre cette comparaison stable à un isomorphisme près, SCOTT permet d'autre-part de pouvoir considérer les similarités et distances d'éditions définies pour les arbres (dans la mesure où ceux-ci restent de taille raisonnable), moins coûteuses (complexité en  $O(n^3)$  pour les arbres, problème exponentiel pour les graphes quelconques) et plus précises. [102, 100, 101]

Parmi les méthodes existantes spécifiquement adressées aux SAR, la plus connue est la *MMPA*, que nous avons évoqué et illustré en introduction (Fig. 7) [26]. La différence calculée entre un graphe moléculaire et un autre est réalisé grâce à une fonction *kernel*.

### 3.1.2 Graphe comme ensemble de fragments

#### Fragments bruts

La seconde manière d'appréhender un graphe est de le considérer comme une composition d'éléments d'ordre inférieur, tels que des sous-graphes de différentes échelles, jusqu'à une échelle que l'on considérera élémentaire et indivisible. Ces structures de base, nommées fragments, peuvent être formellement identifiées à un isomorphisme près, ce que nous nous sommes efforcés de réaliser, soit admettre des collisions. Ces collisions peuvent être la conséquence d'une notation non-canonical des sous-graphes représentés, ou bien faire partie intégrante du processus. Par exemple, la méthode dite *ECFP* se charge de projeter sur un vecteur de taille arbitraire les fragments rencontrés sur un graphe. Ceci étant, la dimension sur laquelle est projetée une modalité de sous-graphe est donnée par une fonction de *hash*, qui par nature admet un certain taux de collision, d'autant plus fort que le vecteur cible est court. Dans le cas d'une fonction de hash sans collision, cette approche revient à un dictionnaire de fragments. La méthode *Extended Connectivity FingerPrint (ECFP)* [23] est ainsi populaire pour projeter une molécule sur un espace plus compact qu'un dictionnaire, au prix toutefois de collisions non contrôlées.

Une fois caractérisés, ces fragments peuvent être sommés, ou en tout cas agrégés, dans les modèles dits "Bag of Fragments". Leur exploitation, ainsi que leur interprétation est très simple, mais ces modèles se révèlent un peu trop naïfs pour prédire les propriétés les plus complexes. L'hypothèse est en effet faite qu'un graphe entier n'est autre qu'un assemblage linéaire de ces composants fondamentaux. Cette hypothèse peut bien entendu se révéler insuffisante pour de nombreux problèmes.

#### Plongements de fragments

Ces fragments peuvent toutefois être affinés au moyen d'un apprentissage basé sur une étude de leur contexte, comme nous l'avons vu en détail lors du chapitre précédent. L'idée est de capter un niveau de sémantique accessible sur un large corpus, potentiellement externe aux tâches traitées. Cette manœuvre aboutit à des plongements, qui peuvent être exploités de plusieurs façons, avec la difficulté suivante : leur nombre n'est pas constant sur un corpus de graphes, et il est donc compliqué de les utiliser conjointement avec des méthodes travaillant sur des structures de taille fixe, qui constituent malheureusement la majorité des méthodes statistiques.

**Somme/Moyenne** La méthode la plus naïve de fixer ces structures consiste à faire un somme ou une moyenne de ces plongements. Les résultats obtenus par ce biais ne se sont malheureusement pas révélés exploitables, bien que des approches paramétrées puissent être envisagée, par exemple une concaténation des  $n$  fragments que l'on juge les plus intéressants, cet intérêt pouvant être paramétré par une fonction de *scoring*.

**Auto-Encodeurs** On se rapproche alors du domaine des auto-encodeurs entraînés pour une tâche spécifique, permettant d'apprendre la meilleure synthétisation possible. Par exemple, on pourrait placer en entrée et en sortie de cet auto-encodeur le dictionnaire des fragments, dans lequel chaque entrée du dictionnaire ne correspondrait pas à un unique scalaire de décompte, mais aux plongements en question, aboutissant à une matrice de la taille du vocabulaire multiplié par la dimension des plongements. Nous parlons ici d'un auto-encodeur entièrement vectoriel qui se chargerait un *Bag of Embeddings*, ou sac de plongements, mais cela rejoint la thématique des auto-encodeurs de graphes que nous évoquerons dans la sous-section suivante.

**Centroïdes** Enfin, nous avons pu remarquer que des distances ou des fonctions de similarité telles que la similarité cosinus avaient du sens sur ces plongements. Dans le cas où les dictionnaires sont très grands, il pourrait être envisagé de regrouper différentes modalités de fragments avec celles de leurs plus proches voisins, et ainsi construire des *meta-fragments*, avec lesquels il serait potentiellement plus facile de travailler que des fragments bruts, et de toutes les difficultés que l'on leur connaît désormais. Cette approche de *Bag of Meta-Fragments* n'a à notre connaissance pas été explorée.

Dans tous les cas, les plongements de fragments ne sont pas utilisables en l'état, comme en attestent les performances déplorables obtenues sur une moyenne de ceux-ci, mais ouvrent de nouvelles perspectives aux méthodes des autres branches, qui elles bénéficient fortement d'une vectorisation informative.

### 3.1.3 Graphe comme un réseau de noeuds

La troisième (et à ce jour dernière) manière d'utiliser un graphe pour effectuer une prédiction est de le considérer comme un ensemble de noeuds formant un réseau, soit la définition littérale d'un graphe. La nuance permettant de réaliser un apprentissage sur ce réseau est de doter chaque noeud d'un ensemble d'informations qui lui est propre, sous forme de vecteur nommé *node features*, et obtenir de ce fait un réseau de vecteurs contenant chacun une information locale. L'exploitation de ces informations est ensuite déléguée à une couche d'apprentissage dont le concept est variable, mais reposant sur des réseaux de neurones artificiels, d'où le nom de cette famille de méthodes : *Graphs Neural Networks (GNN)*. Cette famille demeurant vaste, elle peut être séparée en différentes catégories, et pourrait faire l'objet d'une bibliographie complète [103].

#### Convolution sur graphe

Le but de cette approche est de combiner ces informations par agrégations successives, d'une échelle locale vers une échelle globale. Chaque noeud a ainsi la charge de synthétiser son information avec celle de ses voisins, à la manière d'un produit de convolution, d'où le nom de cette catégorie de *Graphs Convolution Networks*. Le meilleur moyen de se figurer le fonctionnement de l'opérateur de convolution sur graphes est de faire l'analogie avec la convolution en deux dimensions utilisé en traitement d'images : une image est un maillage régulier de points colorés, les pixels. Lorsque l'on applique une convolution sur ces pixels, chaque pixel devient à tour de rôle le centre d'attention d'un filtre, associant à ce pixel et à ses voisins une valeur (ou un ensemble de valeurs).

Une image n'est finalement qu'un cas spécifique de graphe, dans la mesure où l'on considère un pixel comme un noeud relié aux pixels limitrophes. Pour peu que l'on arrive à définir cette opération pour

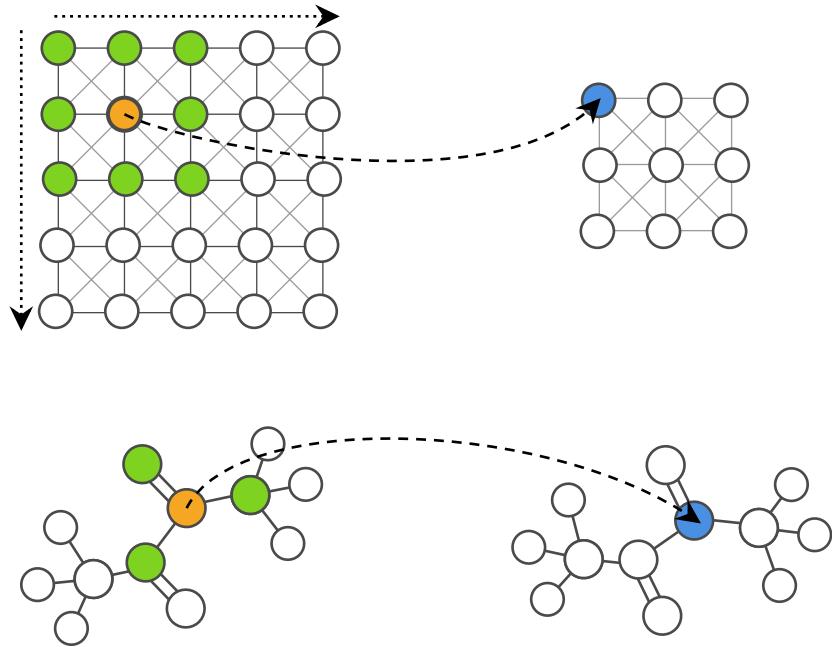


FIGURE 3.1 – Convolution en 2D - Convolution sur Graphe

une structure non rigide telle qu'un graphe quelconque (chaque noeud n'a pas un nombre de voisins constant), on peut généraliser cette opération de convolution aux graphes, et se servir de ce procédé d'agrégation des features voisins pour pondérer progressivement chaque *node feature* avec ceux des voisins. Cette agrégation locale pourra ensuite alimenter une autre couche de convolution, prenant en entrée non pas les *node features* originaux mais les représentations intermédiaires produites par la première couche. On parle alors d'apprentissage profond, en référence au nombre de couches utilisées successivement, dans le but d'apprendre des concepts d'un haut niveau d'abstraction. Cette adaptation des réseaux de neurones convolutifs bien connus en imagerie [105, 104] aux graphes remontent à la fin des années 2010, sous le nom de *Recurrent Graphs Neural Networks*, et se sont répandus notamment pour le calcul d'empreintes similaires à des fonctions de hash (*fingerprint*) [29].

Chaque noeud étant à présent porteur d'informations caractéristiques de plusieurs échelles, il est nécessaire de les traiter en vue d'obtenir une prédiction. Nous illustrons en Fig. 3.2 le principe général de l'architecture permettant cette prédiction. Les noeuds du graphe fourni en entrée du modèle sont porteurs d'un vecteur informatif, les *node features*. Ces *node features* sont mis à jour au regard de ceux des noeuds voisins au cours d'une étape de convolution. Ce nouveau "graphe de vecteurs" est ensuite successivement simplifié par des étapes de *pooling*, afin de se rapprocher d'une structure standardisée. En sortie du modèle, les vectorisations finales sont projetées par concaténation et sommes/moyennes sur la couche d'entrée d'un réseau de neurones artificiel, dite couche *readout*, bien que le terme *fully connected* soit également présent dans la littérature. Cette couche, et d'éventuelles couches cachées, alimentent *in fine* une couche *softmax* apte à fournir une prédiction. De nombreuses variantes existent sur la manière d'effectuer le *pooling* et le *readout*<sup>1</sup>, mais toutes sont basées sur cette architecture générale.

Notons que si l'opération de convolution n'est pas localisée (chaque noeud met à jour ses *features*,

1. Graph Convolutional Network [34], GraphSAGE [106], ChebNet [107], Graph Attention Network [33], etc.

mais la structure du graphe reste identique), cette opération de *pooling* en revanche, nécessite de prendre une décision sur l'identité des noeuds source à agréger dans un noeud cible. Cette prise de décision fait l'objet de nombreuses variantes, mais se base généralement sur un tri des *node features*, selon :

- les valeurs concaténées/sommées de ces *node features*
- un pseudo-tri des noeuds eux-mêmes, par calcul d'invariant (coloration de *Weisfeiler-Lehman* [108]) ou application d'un filtre de graphe [109]

Dans les deux cas, il n'est pas garanti que l'opération soit parfaitement déterministe : il est tout à fait possible d'obtenir des sommes de *features* égales, à fortiori sur un graphe régulier, et les méthodes de calcul d'invariant pour les noeuds ne sont pas injectives. Bien que cela n'empêche pas ces méthodes de remarquablement fonctionner sur des tâches de prédiction, il s'agit d'un aspect qui leur est reproché [110]. Ceci met en évidence que la convolution sur graphe n'est pas aussi bien définie théoriquement que sur des structures régulières comme les maillages en deux dimensions (images). Nous pourrions cependant envisager d'introduire SCOTT à ce niveau pour quantifier l'apport d'un tri fiable sur ces opérations de *pooling*. C'est une perspective que nous envisageons.

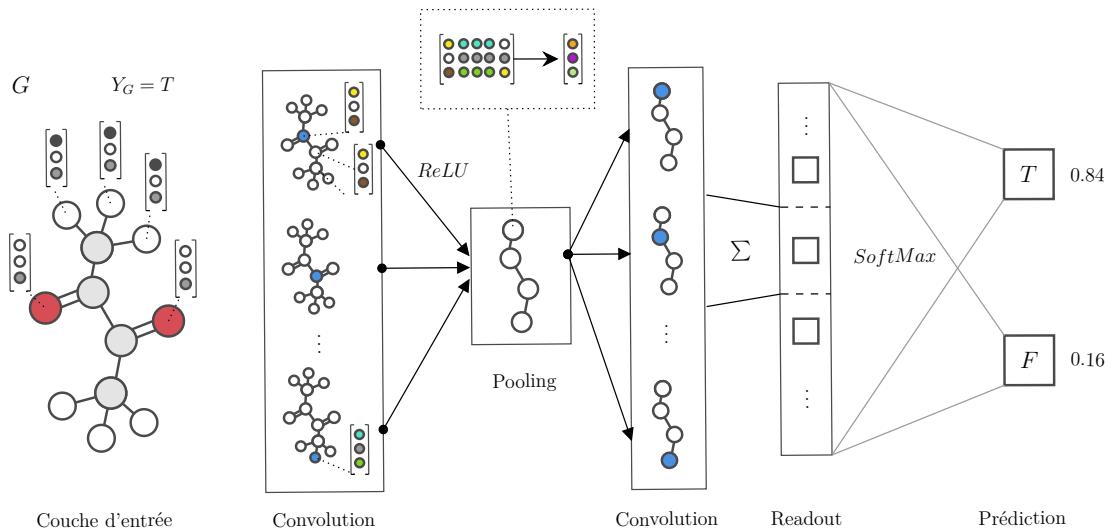


FIGURE 3.2 – Obtention d'une prédiction (classification) à partir d'un graphe. Les neurones sont sur la figure représentés par des carrés pour les différencier de noeuds de graphes

## Auto-encodeurs de graphe

Les Auto-encodeurs que nous avons partiellement évoqués auparavant ont deux cas d'usage. Le premier est de générer un nouveau graphe (en couche de sortie) à partir d'un graphe de référence (en couche d'entrée), en faisant en sorte que le graphe produit soit aussi proche (ou en tout cas "plausible") que le graphe de référence. C'est l'objet des modèles dits adversariaux. Le second cas d'usage découle de la manière dont fonctionne un auto-encodeur : une couche cachée, de dimension moindre aux données d'entrée, est utilisée pour "compresser" au mieux la donnée d'entrée pour la reprojeter avec le moins de perte possible, soit un principe très proche de la couche cachée d'un *Skip-Gram*. Cette couche cachée fournit donc une vectorisation (*embedding*) de taille fixe, supposée embarquer l'essentiel de l'information du graphe. Cette vectorisation peut ainsi être fournie à

des méthodes classiques travaillant sur des observations vectorielles. Contrairement à ce que nous présentons en sous-section précédente, il ne s'agit pas ici de considérer des données d'entrée sous forme de vecteur, mais bien des graphes "bruts".

Si la bibliographie ne contient pas à ce jour d'application de cette méthode pour la construction de SAR, il est intéressant de noter que sa base théorique est un sujet émergent [111, 112, 113]. Les auto-encodeurs variationnels ainsi décrits encodent la matrice d'adjacence et la matrice de *node features* dans un espace latent  $Z$  de dimension moindre, grâce à l'application de convolutions sur graphes successives. Les frontières sont donc poreuses entre ces différentes approches, et les méthodes à base convolution bénéficient directement des progrès réalisés en vectorisation de sous-structures (plongements de fragments).

## Vectorisations

Parmi les 31 variantes de modèles convolutifs et des 12 variantes d'auto-encodeurs répertoriés en [103], toutes ont en commun de nécessiter une vectorisation des noeuds, les *node features*, au moins en tant que valeurs d'initialisation. Au-delà de la variante du modèle utilisé, cette phase de vectorisation est donc cruciale, et très discriminante sur les performances qui découleront du modèle. Les vectorisations utilisées jusqu'ici sont soit des vecteurs aléatoires affinés par agrégations de voisinage, soit des invariants de noeuds dépendant du domaine (par exemple numéro atomique et configuration électronique dans [13] ou le framework Python *Spektral*). À notre connaissance, aucune vectorisation basée sur des plongements appris sur corpus n'a été utilisée dans l'élaboration des SAR, c'est donc le champ de recherche que nous allons privilégier.

### 3.1.4 Récapitulatif

Nous proposons ici un schéma récapitulatif des méthodes évoquées dans cette section. Nous faisons apparaître en vert les méthodes nécessitant une exploitation par des méthodes à noyau, ou en tout cas basées sur des distances (k-NN, kernel-SVM pour citer les plus connues), en orange les méthodes dites classiques, produisant directement des vectorisations de graphes et pouvant ainsi être utilisées dans des méthodes statistiques (régression linéaire, Partial Least Squares, Ridge, Lasso, classificateurs bayésiens, SVM, ...) ou d'apprentissage automatique (Random Forest, réseaux de neurones artificiels). Les méthodes en bleu sont requises une exploitation par des réseaux de neurones profonds prenant en entrée des graphes, conjointement à des attributs de noeuds, les *node features*.

## 3.2 Construction d'un modèle prédictif

Il n'a pas été possible dans le cadre de cette thèse de mener une évaluation sur la totalité des modèles évoqués dans l'état de l'art, bien qu'une proportion notable d'entre eux n'aient pas été testés sur des SAR. Les perspectives futures à ce sujet sont donc larges. Nous avons concentré notre attention sur les méthodes de convolution de graphes, celles-ci pouvant directement tirer parti de vectorisations de sous-graphes apprises sur un corpus. Parmi les modèles référencés, nous privilégions ceux disposant d'implémentations libres et compatibles avec le framework *Keras* [114], directement utilisable conjointement à SCOTT, et utilisé comme support du benchmark existant, *MoleculeNet* [13].

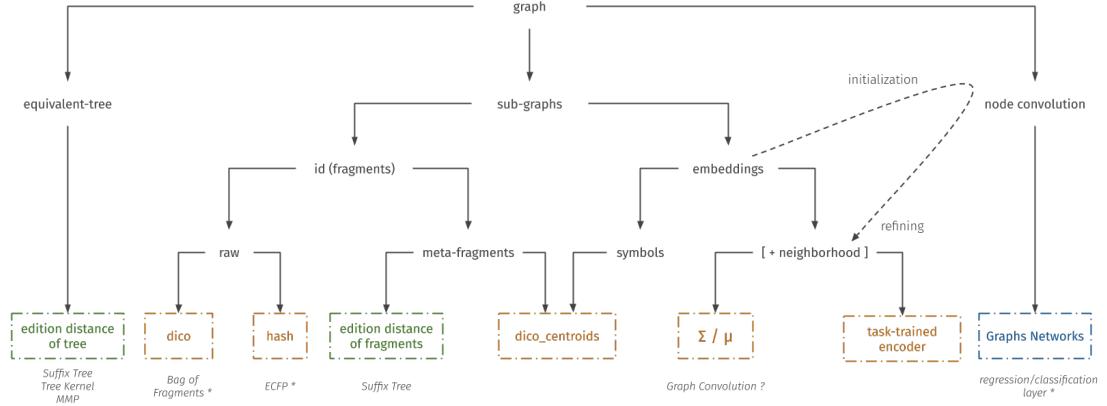
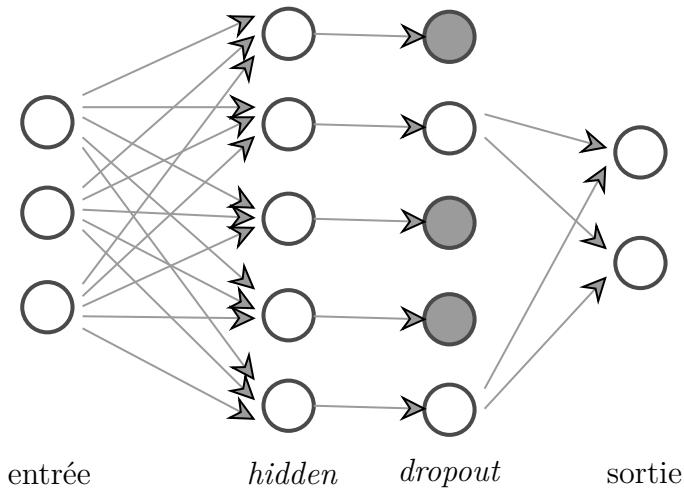


FIGURE 3.3 – Récapitulatif des méthodes de traitements sur graphes

### 3.2.1 Graph Attention Network (GAT)

Notre étude se concentre sur le modèle *Graph Attention Network (GAT)* [115], une des variantes citées précédemment des *Graph Convolutional Network*. Les GAT tirent leur originalité d'un multiplexage des informations échangées entre chaque noeud par différents canaux, permettant une auto-régularisation du modèle lorsque celui-ci est doté de couches *dropout* [116], dont nous illustrons le principe en Fig. 3.4 : en inhibant des liens au regard de la fonction objectif, le modèle apprend à ne pas conserver certaines variables intermédiaires dans le modèle pour éviter un phénomène de sur-apprentissage (*overfitting*), imitant le principe de la régularisation des fonctions objectif exploité en statistiques, par exemple dans le cadre de l'approche Elastic Net [117]. Les GAT sont ainsi réputés bien adaptés au traitement des jeux de données restreints en nombre d'observations exploitables lors de l'apprentissage.


 FIGURE 3.4 – Action d'une couche *DropOut* dans un réseau de neurones profond

Nous incluons dans ce modèle les plongements de fragments faisant l'objet du chapitre précédent, sous la forme de *node features*. Concrètement, cela signifie que les vecteurs associés à chaque noeud, figurant sur la couche d'entrée de la Fig. 3.2 seront les plongements de fragments. Le vecteur caractéristique pour chaque noeud est ainsi initialisé par le plongement associé au fragment enraciné sur ce noeud. Nous introduisons ainsi sur chaque graphe une connaissance plus générale, apprise sur corpus. Nous résumons ce modèle en Fig. 3.5. La taille des vecteurs produits dans la couche *Conv-1* (la taille du filtre) est égale à la taille des plongements, tandis que la couche suivante *Conv-2* divise cette taille par deux.

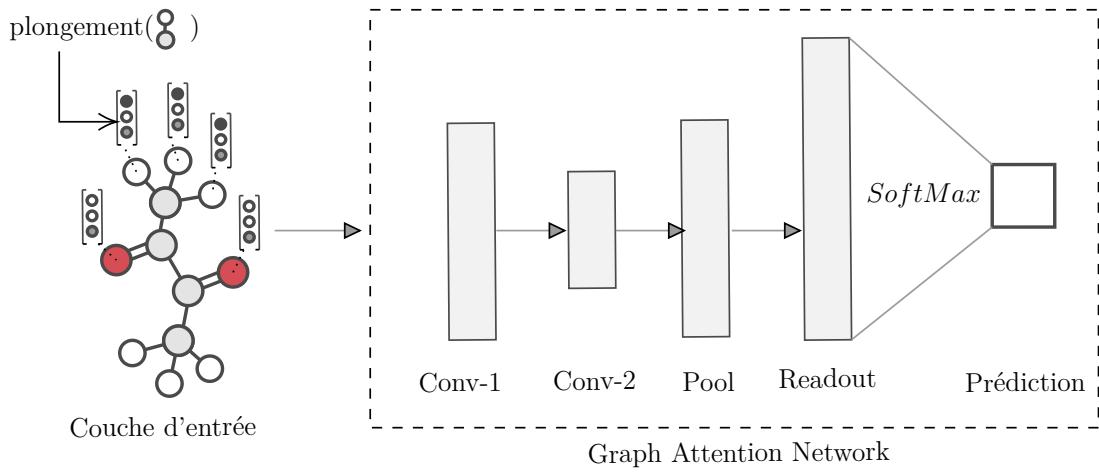


FIGURE 3.5 – Modèle construit et évalué à base de GAT

### 3.2.2 Protocole d'évaluation

Nous exploitons le protocole d'évaluation suivant, sur les tâches de prédiction que nous allons effectuer. Pour quantifier l'apport de l'utilisation des plongements en tant qu'initialisation des *node features*, nous prenons en référence des vecteurs aléatoires de même nombre de dimensions. Nous désignons cette *baseline* sous le nom de *Random-GAT*.

Le jeu de données constitutif d'une tâche de prédiction est séparé en trois parties, nommées *splits*.

Le premier d'entre eux est le jeu d'apprentissage, représentant typiquement 80% du jeu de données. Celui-ci est utilisé pour entraîner le modèle : cette capacité d'apprentissage est portée par les paramètres du modèle, qui sont donc ajustés par un algorithme d'optimisation tels que la descente de gradient stochastique (SGD) ou l'algorithme *Adam* [118]. La mise à jour des paramètres a lieu autant de fois au cours d'un *epoch* que le jeu d'apprentissage est séparé en lots (*batchs*).

Le second *split* est le jeu de validation, constitué de 10% des observations. Celui-ci ne prend pas part à l'apprentissage du modèle, mais sert à évaluer les métriques d'erreur sur un jeu indépendant à la fin de chaque *epoch*. Le but est de pouvoir arrêter prématurément l'apprentissage du modèle si ces métriques s'écartent notablement de celles du jeu d'apprentissage, témoignant d'un sur-apprentissage. Il est également utile pour l'optimisation des hyper-paramètres du modèle.

Category	Dataset	Data Type	Task Type	# Tasks	# Compounds	Rec - Split <sup>a</sup>	Rec - Metric <sup>b</sup>
Quantum Mechanics	QM7	SMILES, 3D coordinates	Regression	1	7160	Stratified	MAE
	QM7b	3D coordinates	Regression	14	7210	Random	MAE
	QM8	SMILES, 3D coordinates	Regression	12	21786	Random	MAE
	QM9	SMILES, 3D coordinates	Regression	12	133885	Random	MAE
Physical Chemistry	ESOL	SMILES	Regression	1	1128	Random	RMSE
	FreeSolv	SMILES	Regression	1	642	Random	RMSE
	Lipophilicity	SMILES	Regression	1	4200	Random	RMSE
Biophysics	PCBA	SMILES	Classification	128	437929	Random	PRC-AUC
	MUV	SMILES	Classification	17	93087	Random	PRC-AUC
	HIV	SMILES	Classification	1	41127	Scaffold	ROC-AUC
	PDBbind	SMILES, 3D coordinates	Regression	1	11908	Time	RMSE
	BACE	SMILES	Classification	1	1513	Scaffold	ROC-AUC
Physiology	BBBP	SMILES	Classification	1	2039	Scaffold	ROC-AUC
	Tox21	SMILES	Classification	12	7831	Random	ROC-AUC
	ToxCast	SMILES	Classification	617	8575	Random	ROC-AUC
	SIDER	SMILES	Classification	27	1427	Random	ROC-AUC
	ClinTox	SMILES	Classification	2	1478	Random	ROC-AUC

FIGURE 3.6 – Récapitulatif des jeux de données présents dans *MoleculeNet*

Enfin, le *split* de test n'est pas évalué avant que le modèle ne soit finalisé, et les métriques obtenues sur ce sous-ensemble constituent le point de comparaison avec les autres modèles mis en concurrence.

La construction du modèle s'opère donc comme suit :

1. Sur  $N$  epochs :
  - (a) Sur  $n$  batchs :
    - i. évaluation de la fonction de coût sur le  $n$ -ième batch du jeu d'apprentissage
    - ii. ajustement des paramètres
  - (b) évaluation de la fonction de coût sur le jeu de validation
  - (c) ajustement des hyper-paramètres
  - (d) [arrêt de l'apprentissage (*early stopping*)]
2. évaluation de la fonction de coût sur le jeu de test

### 3.3 Expérimentations et résultats

Le protocole de construction des modèles prédictifs étant à présent défini, nous pouvons le confronter à des jeux de données, de préférence pour lesquelles nous disposons de points de comparaison récents et reproductibles avec les méthodes de l'état de l'art. Nous accordons de ce fait un grand intérêt au benchmark *MoleculeNet* (2018) [13], qui propose une analyse comparée d'une grande partie des modèles évoqués précédemment, en exploitant des jeux de données accessibles publiquement, de volumétrie et de difficulté diverses. Les jeux de données sont constitués ainsi de quelques centaines d'observations à plusieurs dizaines de milliers d'observations (Fig 3.6). Tous ne sont toutefois pas pertinents dans le contexte de notre étude : typiquement les problématiques de régression de charges électroniques d'atomes nécessitent une caractérisation géométrique en trois dimensions bien plus informative (et performante) qu'une caractérisation sous forme de graphe.

Ces résultats sont par ailleurs reproductibles, grâce aux codes source des méthodes de l'état de l'art mis à disposition sous licence libre. Nous choisissons d'évaluer les modèles produits par association de GAT avec les plongements de fragments sur une partie de ces jeux de données.

Nous effectuons dans cette section un *benchmark* pour évaluer tout à tour l'influence :

- du nombre de dimensions des plongements utilisés
- de la taille des *n-grams* utilisés dans l'apprentissage des plongements
- de la taille des fragments utilisés dans l'apprentissage des plongements
- de la durée d'apprentissage du modèle

Nous nous comparons à la fin de cette section à la fois aux modèles existants, mais aussi aux modèles faisant office de *baseline*, dotés de vectorisations de noeuds aléatoires en lieu et place des plongements, afin de dissocier ceux-ci de la variance naturellement expliquée par l'algorithme par la structure propre du graphe.

### 3.3.1 Tâche de régression

Nous évaluons tout d'abord nos modèles sur une tâche dite de régression, pour laquelle la variable cible, faisant l'objet de la prédiction, est continue. La métrique d'erreur exploitée dans *MoleculeNet* pour ce type de prédiction est la mesure *RMSE* (*Root Mean Square Error*), définie telle que :

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

avec  $\forall i \in [1, n]$ ,  $y_i$  la valeur de la  $i$ -ème observation,  $\hat{y}_i$  la valeur prédictée par le modèle pour cette observation et  $n$  le nombre d'observations. Pour la phase d'optimisation, on lui préfère la *MSE* (*Mean Square Error*), identique à l'exception de la fonction racine appliquée en toute fin, qui est plus utile pour la visualisation que pour l'optimisation. On souhaite en effet pénaliser davantage les écarts importants avec la vérité terrain.

$$MSE = \sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}$$

#### Jeu de données *Lipophilicity*

Ce jeu de données comporte 4200 observations de solubilité (propriété caractérisée par un nombre réel), séparées en jeux d'apprentissage, de validation et de test produits aléatoirement suivant les proportions respectives suivantes : 80-10-10 %. Pour minimiser les effets d'une séparation trop spécifique, nous ré-itérons l'évaluation sur 10 instances de ces séparations aléatoires, que l'on désigne sous le terme de *folds*. La valeur finale que l'on retient est la moyenne de la métrique cible sur ces *folds*.

#### Influence de la taille des plongements de fragments

Sur cette première application, nous examinons l'effet de la taille des plongements, autrement dits la dimension des vecteurs fournis en tant que valeur d'initialisation aux noeuds du graphe. Pour évaluer leur effet, nous fixons tous les autres paramètres de la tâche, à savoir les *splits* du jeu de

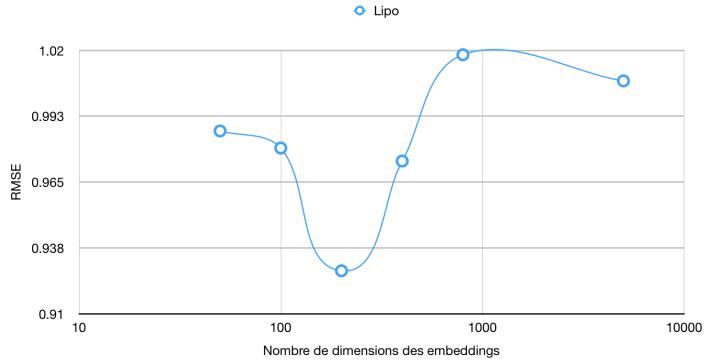
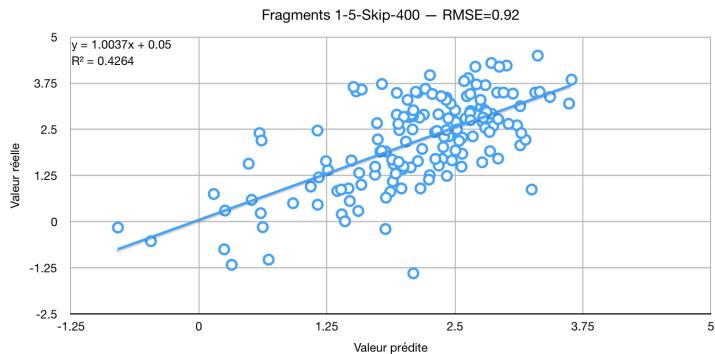


FIGURE 3.7 – Influence du nombre de dimensions des plongements


 FIGURE 3.8 – Scatter-plot des valeurs réelles en fonction des valeurs prédictes,  
*Lipophilicity*

données (apprentissage, validation, test), la durée d'apprentissage<sup>2</sup>, et les autres paramètres<sup>3</sup> de plongements.

Sur cette tâche prédictive, on constate bien un effet du nombre de dimensions de ces plongements, même s'il reste modeste (l'écart maximal sur la *RMSE* constatée est inférieur à 0.1, soit 9%). L'optimal constaté semble se situer aux alentours de 200. Nous verrons sur les autres jeux de données si cette tendance se confirme.

#### Comparaison avec MoleculeNet

Les résultats sur ce jeu de données ne sont pas très favorables à notre modèle comparativement aux résultats obtenus par l'état de l'art. Le meilleur modèle pour notre approche obtient une *RMSE* de 0.92, ce qui ne lui permet pas de se classer parmi les modèles référencés dans *MoleculeNet* (Tab. 3.1). Comme en atteste la Fig. 3.8, le modèle obtenu n'est pas fortement discriminant : une régression linéaire sur le nuage de points obtenu n'obtient un coefficient de détermination *R*<sup>2</sup> de seulement 0.42, on explique ainsi un peu moins de la moitié de la variance observée. Ce n'est donc pas un jeu de données particulièrement bien géré par les modèles élaborés selon notre méthode basée plongements de fragments.

2. 65 epochs

3. taille de fragment de 1, taille des *n-grams* de 5, architecture *Skip-Gram*

	MoleculeNet								GAT	
	RF	MultiTask	XGBoost	KRR	GC	DAG	Weave	MPNN	Scott-GAT	Random-GAT
RMSE - Test	0.876	0.859	0.799	0.899	<b>0.655</b>	0.835	0.715	0.719	0.92	1.22
$R^2$	0.483	0.507	0.577	0.496	0.662	0.524	0.636	<b>0.697</b>	0.426	0.021

TABLE 3.1 –  $RMSE$  obtenue sur la tâche de régression ciblant la propriété *Lipophilicity* selon le modèle

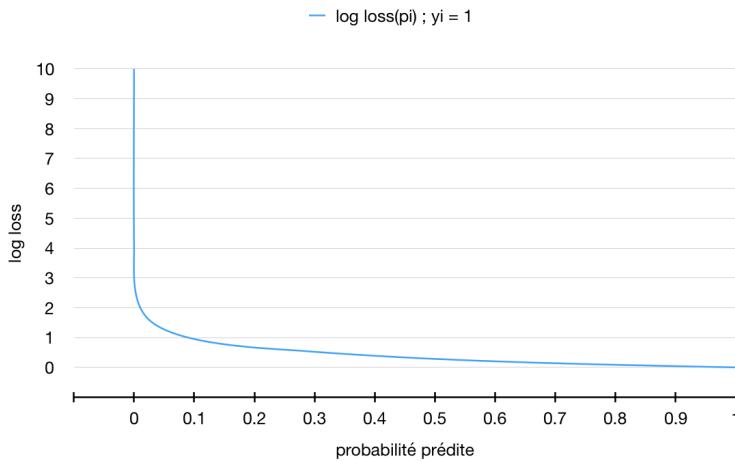


FIGURE 3.9 – Entropie catégorielle obtenue lorsque l'observation  $i$  appartient à la classe 1

### 3.3.2 Tâches de classification

Nous disposons également de tâches de classification, pour lesquelles la métrique adoptée est l'aire sous la courbe sensibilité/spécificité (*ROC-AUC*). La courbe *ROC* donne le taux de vrais positifs (fraction des positifs qui sont correctement détectés, i.e. la *sensibilité* du modèle), en fonction du taux de faux positifs (fraction des positifs non détectés en tant que tel, i.e. la *spécificité* du modèle), et permet d'évaluer le modèle par rapport à une classification aléatoire, qui aurait une *ROC-AUC* 0.5, et une classifieur parfait qui aurait une *ROC-AUC* de 1.0.

Au cours de l'apprentissage, la fonction de coût utilisée pour estimer la performance d'une itération par rapport à la précédente, et ainsi ajuster les paramètres du modèle en conséquence, n'est pas la *ROC-AUC* (qui n'est pas une fonction dérivable, ce qui ne permet pas l'utilisation de nombreux algorithmes d'optimisation de paramètres) mais l'entropie catégorielle, définie dans le cas d'une classification binaire telle que :

$$BCE_i = -(y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i))$$

avec  $y_i \in \{0, 1\}$  la classe réelle de l'observation  $i$  (vérité terrain), et  $p_i \in [0, 1]$ ,  $p_i \in \mathbb{R}$  la prédiction effectuée par le modèle, sous forme de probabilité, que l'observation  $i$  fasse bien partie de la classe  $y$ . Un classifieur parfait annoncera la classe correcte avec une probabilité de 1.0, l'entropie catégorielle vaudra alors 0.0, tandis que dans le pire des cas, on aura un classifieur annonçant une mauvaise classe avec un haut niveau de confiance, soit un score de 1.0. Comme le montre la Fig. 3.9, on pénalise fortement un tel classifieur. Cette fonction est bien entendue à minimiser.

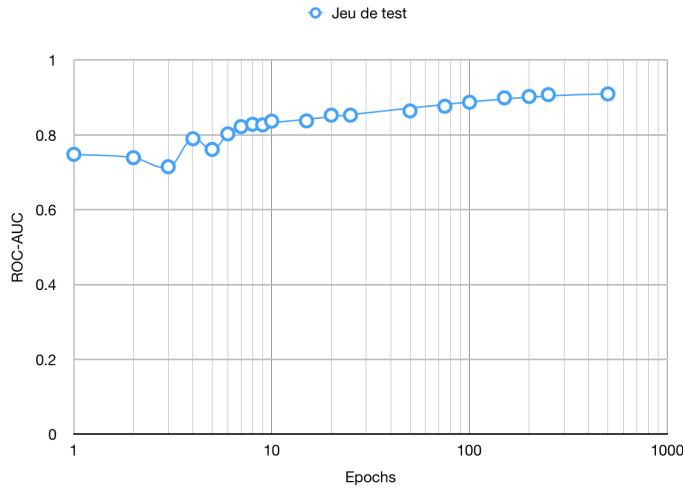


FIGURE 3.10 – Influence du nombre d'epochs sur la performance du modèle

Cette fonction de coût est par ailleurs généralisable à une classification selon  $C$  classes.

$$CE_i = - \sum_{c=1}^C y_{i,c} \cdot \log(p_{i,c})$$

### Jeu de données *Beta-site APP Cleaving Enzyme (BACE)*

Il est question dans ce jeu de données d'estimer la compatibilité ou non d'une substance chimique avec une enzyme spécifique, impliquée dans le développement de la maladie d'Alzheimer. Une substance pouvant inhiber cette enzyme peut ainsi potentiellement avoir des applications thérapeutiques. On dispose ici de 1513 observations réparties équitablement entre deux classes  $T$  et  $F$ . La séparation des jeux d'apprentissage, de validation et de test est ici fixée, il n'est donc pas nécessaire de recourir à des *fold* pour comparer notre modèles avec ceux de l'état de l'art.

#### Influence de la durée d'apprentissage

Nous mesurons ici la faculté du modèle à réellement apprendre depuis les données, en observant l'évolution de la performance du modèle, c'est-à-dire la *ROC-AUC*, au regard du nombre d'*epochs* effectués, c'est-à-dire du nombre d'itérations pour lequel la phase d'apprentissage a été répétée. On suit cette mesure sur le jeu de test.

Nous constatons bien un phénomène d'apprentissage, dans le sens ou au-delà des premières itérations où une optimisation malheureuse peut donner des itérations dégradées par rapport à la précédente, le modèle s'améliore substantivement jusqu'à se stabiliser vers une asymptote. On note également l'absence notable de sur-apprentissage, qui surviendrait si l'on ne recourrait pas à un jeu de validation "pré-test". Les modèles basés sur des architectures d'apprentissage profond ayant un très grand nombre de paramètres (16322 paramètres sujets à apprentissage sur ce modèle), cette pratique est essentielle.

#### Influence de la taille des plongements de fragments

Nous reproduisons ici la précédente expérience concernant la taille des plongements utilisés. La tendance que nous avons observé précédemment semble se confirmer, avec un optimal cette fois-ci autour d'un nombre de dimensions égal à 100. L'impact reste également mesuré, avec un écart

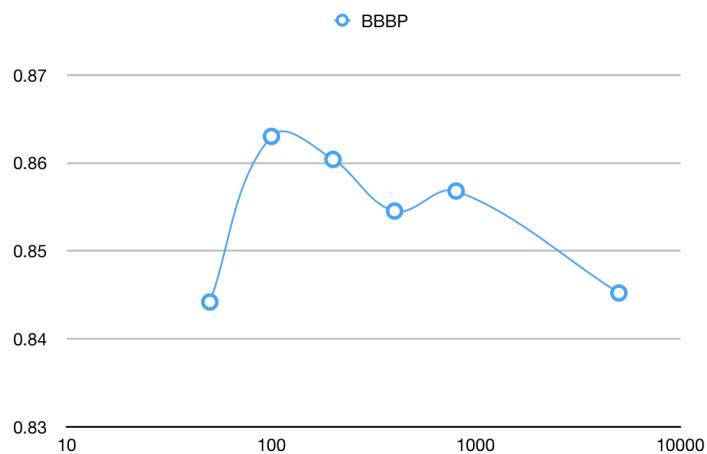


FIGURE 3.11 – Influence du nombre de dimensions des plongements sur la *ROC-AUC*, *BACE*

maximal de 0.02 sur l'aire mesurée sous la courbe *ROC*, soit de l'ordre de 1.5%. Si l'effet semble se confirmer, il n'y a en tout cas aucune contre-indication à préférer une certaine taille de plongement.

#### Influence des paramètres des plongements de fragments

Sur ce jeu de données, nous allons observer l'influence de l'architecture sur laquelle sont appris les plongements. Nous fixons la durée d'entraînement à 65 epochs, et nous constituons des paires de modèles, l'un utilisant des plongements appris sur une architecture *Skip-Gram*, l'autre sur un *CBOW*. Nous faisons varier sur ces paires tout à tour la taille des plongements et la taille des *n-grams*. Les résultats obtenus sont présentés en Fig. 3.12

Sur le diagramme en barres de gauche, nous faisons varier la taille des plongements, comme précédemment. On retrouve bien la tendance que nous venons d'observer pour les *Skip-Gram*, malgré une modification des autres paramètres. Nous remarquons également que le *CBOW* est meilleurs sur les plongements de plus grande taille, avec une valeur optimale décalée par rapport à la première architecture.

L'impact de la longueur des *n-grams* est curieusement moins important, induisant une différence de performance à peine perceptible de moins de 1% dans la majorité des cas, alors que l'on pourrait penser qu'une étude des noeuds voisins plus éloignés pourrait aider la compréhension que l'on a d'un fragment. C'est une observation qu'il serait intéressant de confirmer ou d'infirmer, et pourrait influencer notre façon de construire des fragments.

Par la suite, nous ne poursuivrons pas plus en détail les mesures d'influence de ces paramètres, celles-ci étant finalement assez spécifique au jeu de données considéré. Si la recherche de leur valeur optimale aboutit au meilleur modèle possible, l'espace des solutions doit faire l'objet d'une exploration systématique.

#### Comparaison avec *MoleculeNet*

Enfin, nous comparons le meilleur modèle obtenu par balayage des paramètres avec les modèles répertoriés dans *MoleculeNet*. Sur cette tâche de classification, on constate cette fois une amélioration par rapport aux modèles de l'état de l'art, une grande majorité des modèles testés se classant

### 3.3. Expérimentations et résultats

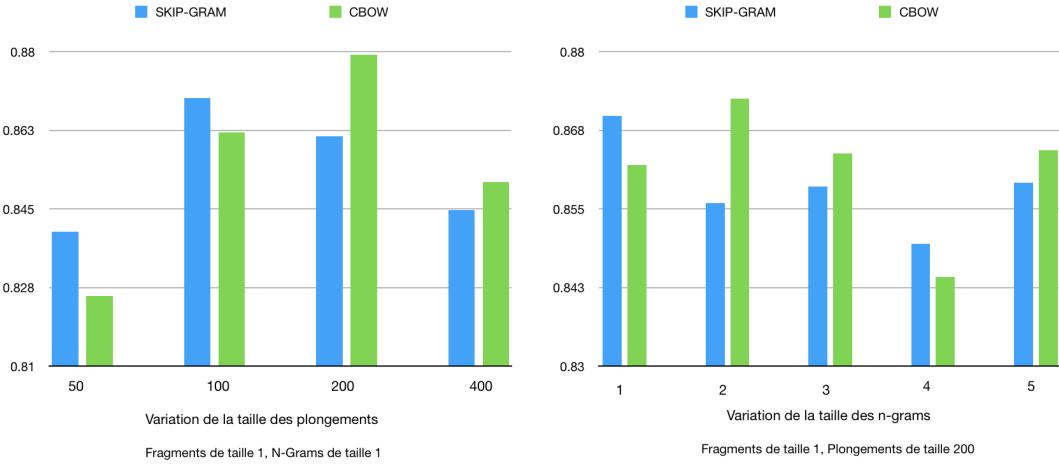


FIGURE 3.12 – Résultats comparés des architectures *Skip-Gram* et *CBOW*, *BACE*

au-dessus du meilleur modèle référencé. Cela vient confirmer que les plongements que nous avons construits peuvent se révéler pertinents sur certaines tâches, et en tout cas ont bien capté la connaissance chimique que l'on semblait percevoir empiriquement lors du chapitre précédent.

	MoleculeNet								GAT	
	LogReg	K-SVM	XGBoost	RF	IRV	MultiTask	Weave	GC	Scott- GAT	Random-GAT
ROC-AUC	0.781	0.862	0.850	<b>0.867</b>	0.838	0.824	0.806	0.783	<b>0.88</b>	0.47

TABLE 3.2 – Performance des modèles évalués sur le jeu de données *BACE*

### Jeu de données *Blood-Brain Barrier Penetration (BBBP)*

On évalue ici la faculté d'une molécule à pouvoir passer à travers une membrane aux propriétés de perméabilité similaires à celles des méninges, membranes protégeant l'encéphale des fluides extérieurs.

Ce jeu de données comporte 2039 observations réparties entre 2 classes (*T/F*). Ces classes ne sont pas équilibrées, aussi nous observons 25% d'observations négatives (*F*) pour 75% d'observations positives (*T*). Les séparations entre jeux d'apprentissage, validation et test sont là encore prédéfinis.

#### Comparaison avec *MoleculeNet*

	MoleculeNet								GAT	
	LogReg	XGBoost	RF	IRV	MultiTask	Weave	GC	Scott- GAT	Random-GAT	
ROC-AUC	0.959	0.956	0.955	0.965	<b>0.988</b>	0.961	0.960	0.962	0.77	

TABLE 3.3 – Performance des modèles évalués sur le jeu de données *BBBP*

On constate tout d'abord que sur ce jeu de données, une classification sur base de plongements aléatoires atteint une précision de 76% pour une *ROC-AUC* similaire, correspondant à l'abondance relative des représentants des deux classes. Ce problème est bien traité par les méthodes statistiques,

	MoleculeNet		GAT
	Meilleur modèle sur vecteurs	Meilleur modèle sur graphe	Scott- GAT
Lipophilicity (RMSE)	0.799	<b>0.655</b>	<i>0.92</i>
BACE (ROC-AUC)	0.867	0.806	<b>0.88</b>
BBBP (ROC-AUC)	<b>0.988</b>	0.961	0.962

TABLE 3.4 – Récapitulatif des résultats obtenus en comparaison avec *MoleculeNet*

et les métriques d’erreur témoignent ainsi d’une grande précision dans tous les cas. Ceci étant, le modèle que nous proposons est tout à fait dans la moyenne des performances constatées, celui-ci se plaçant à la troisième place.

### 3.3.3 Conclusion

Nous proposons un modèle hybride de modèle de convolution de graphe, plus particulièrement un *Graph Attention Network*, auquel nous fournissons des plongements de fragments en tant que vecteurs d’initialisation des *node features*. Nous incorporons donc à chaque graphe une représentation générale, apprise sur un corpus très large de molécules, qui enrichit l’information encodées dans les fragments associés à chaque noeud.

Sur les trois jeux de données explorés, nous constatons que ce gain, s’il n’est pas systématique, peut toutefois être significatif. On résume en Tab. 3.4 l’essentiel des résultats, avec d’une part les modèles les plus performants de l’état de l’art, distinction faite de celui basé sur une vectorisation et de celui basé sur un traitement sur graphe, et d’autre part le modèle proposé.

Sur le premier jeu de données (Lipophilicity), le modèle proposé, sur cette tâche de régression, produit des résultats moins bons que ceux l’état de l’art, en particulier dominé par les méthodes exploitant des graphes. Si les plongements semblent pour autant expliquer une partie de la variance, comme en atteste le score des obtenus par les GAT initialisés par des *node features* vides, ils semblent malgré tout dégrader le modèle, sans que l’on ne sache précisément expliquer pourquoi.

Sur le jeu de données BBBP, en revanche, le modèle se situe exactement au même niveau (compte tenu de la marge d’erreur) que les modèles concurrents, ce qui semble indiquer qu’il n’y a sur ce problème de classification pas d’avantage particulier à utiliser des caractérisations plus larges des fragments. On arrive au même résultat avec des *node embeddings* initiaux décrivant l’atome seul, sans son fragment associé.

Enfin, sur le jeu de données BACE, le modèle proposé est non seulement meilleur que les modèles référencés, mais améliore sensiblement la performance des modèles basés sur graphe, démontrant que ce problème de classification, à l’inverse des précédents, bénéficie grandement d’une vectorisation exploitant une connaissance plus large que celle qui n’est accessible que sur le graphe évalué.

## Références

- [13] Zhenqin WU et al., « MoleculeNet : a benchmark for molecular machine learning », in : *Chemical science* 9.2 (2018), p. 513-530.
- [23] David ROGERS et Mathew HAHN, « Extended-connectivity fingerprints », in : *Journal of chemical information and modeling* 50.5 (2010), p. 742-754.
- [26] Noel M. O'BOYLE et al., « Using matched molecular series as a predictive tool to optimize biological activity », in : *Journal of Medicinal Chemistry* 57.6 (2014), p. 2704-2713, ISSN : 15204804, DOI : 10.1021/jm500022q.
- [29] David K DUVENAUD et al., « Convolutional networks on graphs for learning molecular fingerprints », in : *Advances in neural information processing systems*, 2015, p. 2224-2232.
- [33] Petar VELIČKOVIĆ et al., « Graph attention networks », in : *arXiv preprint arXiv :1710.10903* (2017).
- [34] Thomas N KIPF et Max WELLING, « Semi-supervised classification with graph convolutional networks », in : *arXiv preprint arXiv :1609.02907* (2016).
- [98] Shun-ichi AMARI et Si WU, « Improving support vector machine classifiers by modifying kernel functions », in : *Neural Networks* 12.6 (1999), p. 783-789.
- [99] Xinbo GAO et al., « A survey of graph edit distance », in : *Pattern Analysis and applications* 13.1 (2010), p. 113-129.
- [100] Alessandro MOSCHITTI, « Making tree kernels practical for natural language learning », in : *11th conference of the European Chapter of the Association for Computational Linguistics*, 2006.
- [101] Erik D DEMAINE et al., « An optimal decomposition algorithm for tree edit distance », in : *ACM Transactions on Algorithms (TALG)* 6.1 (2009), p. 2.
- [102] Kaizhong ZHANG et Dennis SHASHA, « Simple fast algorithms for the editing distance between trees and related problems », in : *SIAM journal on computing* 18.6 (1989), p. 1245-1262.
- [103] Zonghan WU et al., « A comprehensive survey on graph neural networks », in : *arXiv preprint arXiv :1901.00596* (2019).
- [104] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON, « Imagenet classification with deep convolutional neural networks », in : *Advances in neural information processing systems*, 2012, p. 1097-1105.
- [105] Yann LECUN, Yoshua Bengio et al., « Convolutional networks for images, speech, and time series », in : *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

- [106] Will HAMILTON, Zhitao YING et Jure LESKOVEC, « Inductive representation learning on large graphs », in : *Advances in Neural Information Processing Systems*, 2017, p. 1024-1034.
- [107] Michaël DEFFERRARD, Xavier BRESSON et Pierre VANDERGHEYNST, « Convolutional neural networks on graphs with fast localized spectral filtering », in : *Advances in neural information processing systems*, 2016, p. 3844-3852.
- [108] Muhan ZHANG et al., « An end-to-end deep learning architecture for graph classification », in : *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [109] Aliaksei SANDRYHAILA et José MF MOURA, « Discrete signal processing on graphs », in : *IEEE transactions on signal processing* 61.7 (2013), p. 1644-1656.
- [110] Keyulu XU et al., « How powerful are graph neural networks ? », in : *arXiv preprint arXiv :1810.00826* (2018).
- [111] Diederik P KINGMA et Max WELLING, « Auto-encoding variational bayes », in : *arXiv preprint arXiv :1312.6114* (2013).
- [112] Thomas N KIPF et Max WELLING, « Variational graph auto-encoders », in : *arXiv preprint arXiv :1611.07308* (2016).
- [113] Daixin WANG, Peng CUI et Wenwu ZHU, « Structural deep network embedding », in : *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, p. 1225-1234.
- [114] François CHOLLET et al., *Keras*, <https://keras.io>, 2015.
- [115] Petar VELIČKOVIĆ et al., « Graph Attention Networks », in : *International Conference on Learning Representations* (2018).
- [116] Nitish SRIVASTAVA et al., « Dropout : a simple way to prevent neural networks from overfitting », in : *The journal of machine learning research* 15.1 (2014), p. 1929-1958.
- [117] Hui ZOU et Trevor HASTIE, « Regularization and variable selection via the elastic net », in : *Journal of the royal statistical society : series B (statistical methodology)* 67.2 (2005), p. 301-320.
- [118] Diederik P KINGMA et Jimmy BA, « Adam : A method for stochastic optimization », in : *arXiv preprint arXiv :1412.6980* (2014).
- [119] Franco SCARSELLI et al., « The graph neural network model », in : *IEEE Transactions on Neural Networks* 20.1 (2008), p. 61-80.

# Conclusion

---

Nous avons au travers des précédents chapitres détaillé les contributions produites au cours de cette thèse. Nous allons résumer, dans cette dernière section, les principaux résultats, en les mettant en perspective avec le sujet de thèse initial.

## Problématique

### Contexte

Cette thèse en informatique s'intègre dans une logique de capitalisation et de valorisation des données disponibles dans un milieu industriel, en particulier dans le domaine de la chimie. Plusieurs cas d'usages industriels peuvent tirer parti d'une estimation rapide et fiable de certaines propriétés physico-chimiques de molécules. Malheureusement ces propriétés ne sont pas toujours aisées à modéliser, les méthodes exactes étant non exhaustives, et souvent complexes. C'est dans ce cadre que les données, publiques ou privées, portant sur ces molécules sont désormais identifiées comme porteuses de valeur, et qu'il peut être pertinent de les valoriser au travers d'analyses statistiques.

### Objectifs et enjeux

Les molécules ne sont toutefois pas des entités faciles à caractériser, et par conséquent à exploiter : celles-ci sont beaucoup plus facilement représentées par des graphes aux noeuds et aux arêtes colorés que par des représentations vectorielles, ces dernières constituant le format de représentation des données le plus courant en statistiques ou en apprentissage automatique. Les modèles exploitant les données sous forme de structures (et donc de graphes) sont appelés modèles structure-activité, que l'on rencontre dans la littérature du domaine de la chimie sous le terme de [Q]SAR (*Structure-Activity Relationship*). Ces modèles constituent la finalité applicative et l'objectif industriel des travaux effectués dans le cadre de cette thèse.

Les moyens explorés pour atteindre cet objectif sont eux fonction de l'abondance relative des données : si l'on trouve dans le domaine public des bases de données répertoriant une quantité impressionnante de substances chimiques (plusieurs dizaines de millions), il est en revanche difficile de constituer un ensemble d'observations portant sur des caractéristiques physico-chimiques particulières, celles-ci étant rarement mises à disposition publiquement. En résumé, si les substances chimiques "viables" sont bien répertoriées, elles ne sont pas pour autant caractérisées par leurs propriétés physico-chimiques.

La problématique abordée s'exprime donc de la manière suivante : est-il possible d'étudier des sous-graphes globalement, i.e. dans un contexte général, pour en obtenir des caractérisations à la fois formelles et porteuses de sémantique, qui puissent être utilisées à des fins prédictives dans des modèles structure-activité, sur des contextes plus restreints d'utilisation (i.e. sur des sous-domaines de la chimie) ?

---

## Réponse apportée

Pour apporter ici une réponse concise, nous avons pu en effet constater que ces sous-graphes peuvent être formellement caractérisés, et qu'il est possible d'apprendre des représentations vectorielles denses et en faible dimension de ces sous-graphes sur un large corpus qui se sont révélés pertinentes dans l'élaboration de modèles prédictifs d'une activité physico-chimique, et même concurrentiels par rapport aux modèles issus de l'état de l'art. D'après nous, la problématique initiale de cette thèse est donc finalement bien couverte par nos travaux.

## Mise en perspective

### État de l'art

Les modèles structure-activité étant populaires en chimie, à fortiori depuis l'émergence de sources de données fiables dans le domaine de la chimie, ceux-ci font l'objet de travaux antérieurs. Le point de départ de cette problématique fut l'étude des modèles *Bag of Fragments*, datant d'une quinzaine d'années, qui sont la première approche statistique avérée s'attachant à ce problème de prédiction : les modèles développés précédemment s'apparentaient davantage à des modèles issus de la simulation physique. Pour ces méthodes, on s'efforce de quitter la caractérisation des observations sous forme de graphes, pour rapidement les vectoriser d'une façon linéaire. Nous avons pu constater dès lors que le problème d'isomorphisme de graphes était un problème sous-jacent récurrent dans le traitement automatique des graphes. En effet, une grande partie des concepts servant à leur analyse, notamment leur décomposition en éléments élémentaires, nécessitent de savoir caractériser un (sous-)graphe "à un isomorphisme près". Dans les faits, si des approximations comme la méthode de *Weisfeiler-Lehman* se révèlent utilisables sur ces méthodes initiales comme sur les plus récentes, cette fragilité théorique interroge, et en tout cas laisse à penser que ces modèles restent perfectibles.

Cet état de l'art plus récent, toutefois, permet la conservation des graphes comme structure porteuse des données beaucoup plus longtemps dans les modèles. Aujourd'hui, on ne cherche plus systématiquement à vectoriser un graphe, les réseaux de neurones profonds permettant de s'affranchir de structures d'entrée très strictes. Ces méthodes, grandement inspirées du domaine de l'analyse d'images, presupposent cependant que, de la même manière que les pixels d'une image, les noeuds d'un graphe soient porteurs d'une information vectorielle riche et généralement appelée *node features*. Dans les faits, ces vectorisations soit sont aléatoires, soit reprennent des attributs triviaux, d'où une vectorisation des noeuds très simple.

En contre-partie, les méthodes issues du traitement automatique des langages naturels permettent de vectoriser des noeuds ou des sous-graphes via des techniques de plongement similaires à des plongements lexicaux. Ces approches, directement applicables à des tâches de classification de noeuds, permettent également d'élaborer des modèles de prédiction plus complexes en exploitant ces vectorisations qui intègrent une forme de sémantique lexicale, dans des architecture de réseau convolutionnel par exemple.

### Apports au sujet

Le rapprochement de ces trois thèmes (décomposition de graphe en fragments, identification des classes d'isomorphisme de sous-graphes, plongement de fragment) a constitué notre voie de recherche pour traiter la problématique abordée dans cette thèse. Nous avons donc cherché à unifier

---

l'utilisation des plongements de fragments avec des modèles de prédiction à partir de graphes, tout en y intégrant des fondements théoriques développés autour du concept d'isomorphisme. Nous distinguons deux axes principaux pour décrire les apports de cette thèse au regard du sujet posé.

## Théorie des graphes

Nous pouvons distinguer d'une part les apports relevant du domaine théorique de la théorie des graphes, plus particulièrement des problèmes d'isomorphisme et de canonisation de graphe. Nous avons proposé l'algorithme SCOTT, qui apporte une solution générale à ces deux problèmes. Le caractère général de la solution concerne l'applicabilité de SCOTT à des graphes quelconques, en particulier des graphes dont les noeuds et les arêtes sont colorées : il n'existe pas à notre connaissance d'algorithme permettant la canonisation de tels graphes. Nous prouvons expérimentalement et théoriquement la validité de cet algorithme, constituant ainsi une base solide permettant d'appliquer certaines opérations sur un graphe "à un isomorphisme près". Notons enfin que SCOTT est parallélisé, tandis que les algorithmes existants demeurent essentiellement séquentiels, ce qui constitue également un apport en matière de passage à l'échelle.

## Modèles [Q]Sar

Concernant l'utilisation de ces graphes pour des tâches de prédiction, nommément des modèles structure-activité, nous apportons quelques considérations ouvrant de nouvelles perspectives. Nous avons en effet vérifié que les tâches de prédiction de noeuds d'un graphe exploitent avec succès le concept de plongement lexical. Celui-ci permet en premier lieu d'obtenir des vectorisations porteuses de sémantique. Il permet dans un deuxième temps d'attaquer des architectures neuro-mimétiques profondes en encodant cette sémantique latente attachée au noeud d'un graphe. Cette démarche nous a permis de proposer un modèle hybride exploitant des plongements de fragments au sein d'un modèle convolutif profond. Nous montrons que si le gain obtenu par notre modèle (dans sa version actuelle) vis-à-vis de l'état de l'art n'est pas systématique et dépend donc du jeu de données considéré, ce gain peut toutefois se révéler très important.

## Limites des travaux

Au delà des apport listés ci-dessus, certains aspects évoqués dans ce manuscrit restent malheureusement en suspens : nous avons fait des choix dans la multitude de modèles à implémenter, tant sur la partie plongements de fragments que sur la partie modèles /Q/Sar. Dans le premier cas, un entraînement à base de *Radial Skip-Gram* permettrait de mesurer directement les apports de cette architecture. Une adaptation du modèle *GloVe* plutôt que *Word2Vec* serait également intéressante à tester. Concernant les modèles [Q]SAR, la grande diversité de modèles envisageables (par ailleurs en constante évolution) laisse entrevoir le travail important qu'il reste à entreprendre pour finaliser un benchmark plus exhaustif.

Enfin, l'algorithme SCOTT développé dans le cadre de cette thèse répond au problème d'isomorphisme de graphe. Les modèles précédents se passent de fonctions purement injectives, bien que cela leur soit reproché pour des considérations théoriques. [110] L'intérêt d'un tel algorithme dans ce contexte est donc bien entendu sujet à débats, spécialement compte tenu du temps que son implémentation a pu représenter. C'est un choix tirant sa motivation initiale du manque de solution existante massivement parallélisable pour la fragmentation de corpus larges de graphes. SCOTT nous a par la suite paru combler un espace théorique intéressant.

---

## Avenir du sujet

Ce sujet actuellement en pleine émergence est amené, à notre avis, à affirmer la nouvelle place des données dans le domaine de la chimie dans un avenir proche. La technologie suit en effet de près la masse de données disponible, d'autant que les initiatives logiciels libres deviennent la norme de diffusion des résultats algorithmiques.

On note par ailleurs que sur les deux derniers chapitres qui concernent les plongements de fragments de graphe et les modèles SAR, l'état de l'art est en constante évolution : une grande partie des ressources bibliographiques des chapitres respectifs sont parues il y a moins de cinq ans, et pour bonne partie au cours de la thèse. C'est donc un sujet difficile à suivre en temps réel, mais paradoxalement de plus en plus facile d'accès. Il ne fait donc aucun doute que les données et leur analyse sont amenées à être des outils centraux de tout laboratoire de chimie dans un avenir proche.

## Perspectives

### Modèles Qsar

De nombreux modèles demandent à être également testés, au delà de la famille des méthodes convolutives déjà explorées. La famille des méthodes à base de distance d'édition de graphes [120] ou de séquences [121], notamment de distance par recouvrement de séquence [122].

### Théorie des Graphes

Enfin, les perspectives concernant le pan plus théorique sont relatives à l'algorithme SCOTT, répondant initialement à un problème bien défini : l'isomorphisme de graphe. Le fait est qu'il permet d'associer à tout graphe un arbre, catégorie de graphes possédant des propriétés intéressantes, sur laquelle des fonctions *kernel* sont bien définies, et qu'il est par ailleurs possible d'ordonner. Il est donc possible d'induire un ordre sur l'ensemble des graphes, et par extension aux sous-graphes composant un graphe, ouvrant de nouvelles perspectives sur des procédures jusqu'ici non-déterministes. Nous envisageons donc les perspectives suivantes :

- définition d'une fonction *kernel* inter-arbres, et par extension inter-graphes, en adaptant une distance par recouvrement de séquences ou d'arbres
  - définition d'une fonction de *hash* pour graphe, par compression de la trace  $\sigma^T$
  - évaluer quels algorithmes ou quelles procédures pourraient bénéficier de graphes ordonnés
- Enfin, une implémentation plus efficace, exploitant tous les axes de parallélisation possibles, constituerait une solution de fragmentation de corpus de graphes, tout en apportant une solution alternative au problème d'isomorphisme de graphes, pouvant tirer parti des architectures multi-coeurs, et ainsi repousser la taille limite des graphes pouvant être traités.

# Bibliographie

---

- [1] Jerry MARCH, *Advanced organic chemistry : reactions, mechanisms, and structure*, John Wiley & Sons, 1992.
- [2] Francis A CAREY et Richard J SUNDBERG, *Advanced organic chemistry : part A : structure and mechanisms*, Springer Science & Business Media, 2007.
- [3] Francis A CAREY et Richard J SUNDBERG, *Advanced Organic Chemistry : Part B : Reaction and Synthesis*, Springer Science & Business Media, 2007.
- [4] Ira N. LEVINE, *Quantum Chemistry, 7th Edition*, Pearson, 2014.
- [5] David J GRIFFITHS et Darrell F SCHROETER, *Introduction to quantum mechanics*, Cambridge University Press, 2018.
- [6] David WEININGER, Arthur WEININGER et Joseph L WEININGER, « SMILES. 2. Algorithm for generation of unique SMILES notation », in : *Journal of Chemical Information and Computer Sciences* 29.2 (1989), p. 97-101.
- [7] Stephen HELLER et al., « InChI-the worldwide chemical structure identifier standard », in : *Journal of cheminformatics* 5.1 (2013), p. 7.
- [8] Charlotte Froese FISCHER, « General hartree-fock program », in : *Computer physics communications* 43.3 (1987), p. 355-365.
- [9] Antoine DAINA, Olivier MICHELIN et Vincent ZOETE, « iLOGP : a simple, robust, and efficient description of n-octanol/water partition coefficient for drug design using the GB/SA approach », in : *Journal of chemical information and modeling* 54.12 (2014), p. 3284-3301.
- [10] Steven ABBOTT et Charles M HANSEN, *Hansen solubility parameters in practice*, Hansen-Solubility, 2008.
- [11] Sunghwan KIM et al., « PubChem substance and compound databases », in : *Nucleic acids research* 44.D1 (2015), p. D1202-D1213.
- [12] Harry E PENCE et Antony WILLIAMS, *ChemSpider : an online chemical information resource*, 2010.
- [13] Zhenqin WU et al., « MoleculeNet : a benchmark for molecular machine learning », in : *Chemical science* 9.2 (2018), p. 513-530.
- [14] JC BRADLEY, A WILLIAMS et A LANG, *Jean-Claude Bradley Open Melting Point Dataset, 2014*, URL : [https://figshare.com/articles/Jean\\_Claude\\_Bradley\\_Open\\_Melting\\_Point\\_Datset/1031637](https://figshare.com/articles/Jean_Claude_Bradley_Open_Melting_Point_Datset/1031637).

- 
- [15] Muthukumarasamy KARTHIKEYAN, Robert C GLEN et Andreas BENDER, « General melting point prediction based on a diverse compound data set and artificial neural networks », in : *Journal of chemical information and modeling* 45.3 (2005), p. 581-590.
  - [16] Sophie MARTEL et al., « Large, chemically diverse dataset of logP measurements for benchmarking studies », in : *European Journal of Pharmaceutical Sciences* 48.1 (2013), p. 21-29.
  - [17] Alexandre VARNEK et al., « Substructural fragments : an universal language to encode reactions, molecular and supramolecular structures », in : *Journal of computer-aided molecular design* 19.9-10 (2005), p. 693-703.
  - [18] Igor BASKIN et Alexandre VARNEK, « Building a chemical space based on fragment descriptors », in : *Combinatorial chemistry & high throughput screening* 11.8 (2008), p. 661-668.
  - [19] Fiorella RUGGIU et al., « ISIDA Property-Labelled Fragment Descriptors », in : *Molecular informatics* 29.12 (2010), p. 855-868.
  - [20] Fiorella RUGGIU, « Property-enriched fragment descriptors for adaptive QSAR », thèse de doct., 2014.
  - [21] Fiorella RUGGIU et al., « Individual Hydrogen-Bond Strength QSPR Modelling with ISIDA Local Descriptors : a Step Towards Polyfunctional Molecules », in : *Molecular informatics* 33.6-7 (2014), p. 477-487.
  - [22] Alexandre VARNEK et al., « ISIDA-Platform for virtual screening based on fragment and pharmacophoric descriptors », in : *Current Computer-Aided Drug Design* 4.3 (2008), p. 191.
  - [23] David ROGERS et Mathew HAHN, « Extended-connectivity fingerprints », in : *Journal of chemical information and modeling* 50.5 (2010), p. 742-754.
  - [24] Ye HU, Dagmar STUMPFE et Jürgen BAJORATH, « Advancing the activity cliff concept. », in : *F1000Research* 2 (2013), p. 199, ISSN : 2046-1402, DOI : 10.12688/f1000research.2-199.v1.
  - [25] D STUMPFE et al., « Follow up : Advancing the activity cliff concept, part II ANRT biblio Chemistry ML NeuralNets QSAR », in : *F1000Research* (2014).
  - [26] Noel M. O'BOYLE et al., « Using matched molecular series as a predictive tool to optimize biological activity », in : *Journal of Medicinal Chemistry* 57.6 (2014), p. 2704-2713, ISSN : 15204804, DOI : 10.1021/jm500022q.
  - [27] Annamalai NARAYANAN et al., « subgraph2vec : Learning distributed representations of rooted sub-graphs from large graphs », in : *arXiv preprint arXiv :1606.08928* (2016).

- 
- [28] Aditya GROVER et Jure LESKOVEC, « node2vec : Scalable feature learning for networks », in : *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, p. 855-864.
  - [29] David K DUVENAUD et al., « Convolutional networks on graphs for learning molecular fingerprints », in : *Advances in neural information processing systems*, 2015, p. 2224-2232.
  - [30] Joseph GOMES et al., « Atomic convolutional networks for predicting protein-ligand binding affinity », in : *arXiv preprint arXiv :1703.10603* (2017).
  - [31] Steven KEARNES et al., « Molecular graph convolutions : moving beyond fingerprints », in : *Journal of computer-aided molecular design* 30.8 (2016), p. 595-608.
  - [32] Mikael HENAFF, Joan BRUNA et Yann LECUN, « Deep convolutional networks on graph-structured data », in : *arXiv preprint arXiv :1506.05163* (2015).
  - [33] Petar VELIČKOVIĆ et al., « Graph attention networks », in : *arXiv preprint arXiv :1710.10903* (2017).
  - [34] Thomas N KIPF et Max WELLING, « Semi-supervised classification with graph convolutional networks », in : *arXiv preprint arXiv :1609.02907* (2016).
  - [35] Jie ZHOU et al., « Graph neural networks : A review of methods and applications », in : *arXiv preprint arXiv :1812.08434* (2018).
  - [36] Annamalai NARAYANAN et al., « graph2vec : Learning Distributed Representations of Graphs », in : *arXiv preprint arXiv :1707.05005* (2017).
  - [37] Palash GOYAL et Emilio FERRARA, « Graph Embedding Techniques, Applications, and Performance : A Survey », in : *arXiv preprint arXiv :1705.02801* (2017).
  - [38] László BABAI et Eugene M LUKS, « Canonical labeling of graphs », in : *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, ACM, 1983, p. 171-183.
  - [39] Viktor N ZEMLYACHENKO, Nickolay M KORNEENKO et Regina I TYSHKEVICH, « Graph isomorphism problem », in : *Journal of Soviet Mathematics* 29.4 (1985), p. 1426-1481.
  - [40] Uwe SCHÖNING, « Graph isomorphism is in the low hierarchy », in : *Journal of Computer and System Sciences* 37.3 (1988), p. 312-323.
  - [41] Vikraman ARVIND et Piyush P KURUR, « Graph isomorphism is in SPP », in : *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*. IEEE, 2002, p. 743-750.
  - [42] Kellogg S BOOTH et Charles J COLBOURN, *Problems polynomially equivalent to graph isomorphism*, Computer Science Department, Univ., 1979.

- 
- [43] Johannes KOBLER, Uwe SCHÖNING et Jacobo TORÁN, *The graph isomorphism problem : its structural complexity*, Springer Science & Business Media, 2012.
  - [44] László BABAI, « Graph isomorphism in quasipolynomial time », in : *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, ACM, 2016, p. 684-697.
  - [45] David WEININGER, « SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules », in : *Journal of chemical information and computer sciences* 28.1 (1988), p. 31-36.
  - [46] Stephen R HELLER et Alan D MCNAUGHT, « The IUPAC international chemical identifier (InChI) », in : *Chemistry International* 31.1 (2009), p. 7.
  - [47] Eugene M LUKS, « Isomorphism of graphs of bounded valence can be tested in polynomial time », in : *Journal of computer and system sciences* 25.1 (1982), p. 42-65.
  - [48] John E HOPCROFT et Jin-Kue WONG, « Linear time algorithm for isomorphism of planar graphs (preliminary report) », in : *Proceedings of the sixth annual ACM symposium on Theory of computing*, ACM, 1974, p. 172-184.
  - [49] Daniel NEUEN et Pascal SCHWEITZER, « Benchmark graphs for practical graph isomorphism », in : *arXiv preprint arXiv :1705.03686* (2017).
  - [50] Yuri GUREVICH et Saharon SHELAH, « On finite rigid structures », in : *The Journal of Symbolic Logic* 61.2 (1996), p. 549-562.
  - [51] Paul T DARGA, Karem A SAKALLAH et Igor L MARKOV, « Faster symmetry discovery using sparsity of symmetries », in : *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, IEEE, 2008, p. 149-154.
  - [52] Paolo CODENOTTI et al., « Conflict analysis and branching heuristics in the search for graph automorphisms », in : *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, IEEE, 2013, p. 907-914.
  - [53] José Luis LÓPEZ-PRESA, Antonio Fernández ANTA et Luis Núñez CHIROQUE, « Conauto-2.0 : Fast isomorphism testing and automorphism group computation », in : *arXiv preprint arXiv :1108.1060* (2011).
  - [54] Tommi JUNTTILA et Petteri KASKI, « Engineering an efficient canonical labeling tool for large and sparse graphs », in : *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*, SIAM, 2007, p. 135-149.
  - [55] Tommi JUNTTILA et Petteri KASKI, « Conflict propagation and component recursion for canonical labeling », in : *International Conference on Theory and Practice of Algorithms in (Computer) Systems*, Springer, 2011, p. 151-162.
  - [56] Brendan D MCKAY et al., « Practical graph isomorphism », in : (1981).

- 
- [57] Adolfo PIPERNO, « Search space contraction in canonical labeling of graphs », in : *arXiv preprint arXiv :0804.4881* (2008).
  - [58] Brendan D MCKAY et Adolfo PIPERNO, « Practical graph isomorphism, II », in : *Journal of Symbolic Computation* 60 (2014), p. 94-112.
  - [59] Jacques NEVEU, « Arbres et processus de Galton-Watson », in : *Annales de l'IHP Probabilités et statistiques*, t. 22, 2, 1986, p. 199-207.
  - [60] Pedro Pablo Perez VELASCO, *Matrix Graph Grammars*, 2008, p. 321, arXiv : 0801.1245, URL : <http://arxiv.org/abs/0801.1245>.
  - [61] G OLSEN, « Gary Olsen's interpretation of the "Newick's 8 : 45" tree format standard », in : URL [http://evolution.genetics.washington.edu/phylip/newick\\_doc.html](http://evolution.genetics.washington.edu/phylip/newick_doc.html) (1990).
  - [62] S Vichy N VISHWANATHAN et al., « Graph kernels », in : *Journal of Machine Learning Research* 11.Apr (2010), p. 1201-1242.
  - [63] Benoit GAÜZERE, Luc BRUN et Didier VILLEMIN, « Two new graphs kernels in chemoinformatics », in : *Pattern Recognition Letters* 33.15 (2012), p. 2038-2047.
  - [64] Reinhard DIESTEL, « Graph theory. 2005 », in : *Grad. Texts in Math* 101 (2005).
  - [65] László BABAI, William M KANTOR et Eugene M LUKS, « Computational complexity and the classification of finite simple groups », in : *24th Annual Symposium on Foundations of Computer Science (Sfcs 1983)*, IEEE, 1983, p. 162-171.
  - [66] Akihiro INOKUCHI, Takashi WASHIO et Hiroshi MOTODA, « An apriori-based algorithm for mining frequent substructures from graph data », in : *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2000, p. 13-23.
  - [67] Xifeng YAN et Jiawei HAN, « gspan : Graph-based substructure pattern mining », in : *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, IEEE, 2002, p. 721-724.
  - [68] Arthur CAYLEY, « A theorem on trees », in : *Quart. J. Math.* 23 (1889), p. 376-378.
  - [69] Frank HARARY et Edgar M PALMER, *Graphical enumeration*, Elsevier, 2014.
  - [70] Pedro Pablo Perez VELASCO et Juan de LARA, « Matrix Approach to Graph Transformation : Matching and Sequences », in : *Lecture Notes in Computer Science* 4178 (2006), p. 122, ISSN : 16113349, DOI : [10.1007/11841883\\_10](https://doi.org/10.1007/11841883_10).
  - [71] H. EHRIG, K. HOFFMANN et J. PADBERG, « Transformations of Petri nets », in : *Electronic Notes in Theoretical Computer Science* 148.1 SPEC. ISS. (2006), p. 151-172, ISSN : 15710661, DOI : [10.1016/j.entcs.2005.12.016](https://doi.org/10.1016/j.entcs.2005.12.016).
  - [72] Karen Spärck JONES, « A statistical interpretation of term specificity and its application in retrieval », in : *Journal of documentation* (2004).

- 
- [73] Christophe GIRAUD, *Introduction to high-dimensional statistics*, Chapman et Hall/CRC, 2014.
  - [74] Mark EJ NEWMAN, « Power laws, Pareto distributions and Zipf's law », in : *Contemporary physics* 46.5 (2005), p. 323-351.
  - [75] Benoît B MANDELBROT, *Linguistique statistique macroscopique*, 1957.
  - [76] Marc BARBUT, « Note sur l'ajustement des distributions de Zipf-Mandelbrot en statistique textuelle », in : *Histoire & Mesure* (1989), p. 107-119.
  - [77] Matthieu CRISTELLI, Michael BATTY et Luciano PIETRONERO, « There is more than a power law in Zipf », in : *Scientific reports* 2 (2012), p. 812.
  - [78] Isabel MORENO-SÁNCHEZ, Francesc FONT-CLOS et Álvaro CORRAL, « Large-scale analysis of Zipf's law in English texts », in : *PloS one* 11.1 (2016), e0147073.
  - [79] Steven T PIANTADOSI, « Zipf's word frequency law in natural language : A critical review and future directions », in : *Psychonomic bulletin & review* 21.5 (2014), p. 1112-1130.
  - [80] Tomas MIKOLOV et al., « Efficient estimation of word representations in vector space », in : *arXiv preprint arXiv :1301.3781* (2013).
  - [81] Tomas MIKOLOV et al., « Distributed representations of words and phrases and their compositionality », in : *Advances in neural information processing systems*, 2013, p. 3111-3119.
  - [82] Quoc LE et Tomas MIKOLOV, « Distributed representations of sentences and documents », in : *International Conference on Machine Learning*, 2014, p. 1188-1196.
  - [83] Jeffrey PENNINGTON, Richard SOCHER et Christopher MANNING, « Glove : Global vectors for word representation », in : *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, p. 1532-1543.
  - [84] Thomas K LANDAUER, Peter W FOLTZ et Darrell LAHAM, « An introduction to latent semantic analysis », in : *Discourse processes* 25.2-3 (1998), p. 259-284.
  - [85] Marwa NAILI, Anja Habacha CHAIBI et Henda Hajjami Ben GHEZALA, « Comparative study of word embedding methods in topic segmentation », in : *Procedia computer science* 112 (2017), p. 340-349.
  - [86] Edgar ALTSZYLER et al., « Comparative study of LSA vs Word2vec embeddings in small corpora : a case study in dreams database », in : *arXiv preprint arXiv :1610.01520* (2016).
  - [87] Bryan PEROZZI, Rami AL-RFOU et Steven SKIENA, « Deepwalk : Online learning of social representations », in : *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, p. 701-710.

- 
- [88] Nino SHERVASHIDZE et al., « Weisfeiler-lehman graph kernels », in : *Journal of Machine Learning Research* 12.Sep (2011), p. 2539-2561.
  - [89] Christopher MORRIS et al., « Weisfeiler and leman go neural : Higher-order graph neural networks », in : *Proceedings of the AAAI Conference on Artificial Intelligence*, t. 33, 2019, p. 4602-4609.
  - [90] Swarnendu GHOSH et al., « The journey of graph kernels through two decades », in : *Computer Science Review* 27 (2018), p. 88-111.
  - [91] Nils M KRIEGE, Fredrik D JOHANSSON et Christopher MORRIS, « A Survey on Graph Kernels », in : *arXiv preprint arXiv :1903.11835* (2019).
  - [92] Radim ŘEHŮŘEK et Petr SOJKA, « Software Framework for Topic Modelling with Large Corpora », English, in : *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, <http://is.muni.cz/publication/884893/en>, Valletta, Malta : ELRA, mai 2010, p. 45-50.
  - [93] Laurens van der MAATEN et Geoffrey HINTON, « Visualizing data using t-SNE », in : *Journal of machine learning research* 9.Nov (2008), p. 2579-2605.
  - [94] Will Y ZOU et al., « Bilingual word embeddings for phrase-based machine translation », in : *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, p. 1393-1398.
  - [95] Andrew M. DAI, Christopher OLAH et Quoc V. LE, « Document embedding with paragraph vectors », in : *NIPS Deep Learning Workshop*, 2015.
  - [96] Gerard SALTON et Christopher BUCKLEY, « Term-weighting approaches in automatic text retrieval », in : *Information processing & management* 24.5 (1988), p. 513-523.
  - [97] Chris BUCKLEY, Gerard SALTON et James ALLAN, « Automatic retrieval with locality information using SMART », in : *Proceedings of the First Text REtrieval Conference TREC-1*, 1993, p. 59-72.
  - [98] Shun-ichi AMARI et Si WU, « Improving support vector machine classifiers by modifying kernel functions », in : *Neural Networks* 12.6 (1999), p. 783-789.
  - [99] Xinbo GAO et al., « A survey of graph edit distance », in : *Pattern Analysis and applications* 13.1 (2010), p. 113-129.
  - [100] Alessandro MOSCHITTI, « Making tree kernels practical for natural language learning », in : *11th conference of the European Chapter of the Association for Computational Linguistics*, 2006.
  - [101] Erik D DEMAINE et al., « An optimal decomposition algorithm for tree edit distance », in : *ACM Transactions on Algorithms (TALG)* 6.1 (2009), p. 2.

- 
- [102] Kaizhong ZHANG et Dennis SHASHA, « Simple fast algorithms for the editing distance between trees and related problems », in : *SIAM journal on computing* 18.6 (1989), p. 1245-1262.
  - [103] Zonghan WU et al., « A comprehensive survey on graph neural networks », in : *arXiv preprint arXiv :1901.00596* (2019).
  - [104] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON, « Imagenet classification with deep convolutional neural networks », in : *Advances in neural information processing systems*, 2012, p. 1097-1105.
  - [105] Yann LECUN, Yoshua BENGIO et al., « Convolutional networks for images, speech, and time series », in : *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.
  - [106] Will HAMILTON, Zhitao YING et Jure LESKOVEC, « Inductive representation learning on large graphs », in : *Advances in Neural Information Processing Systems*, 2017, p. 1024-1034.
  - [107] Michaël DEFFERRARD, Xavier BRESSON et Pierre VANDERGHEYNST, « Convolutional neural networks on graphs with fast localized spectral filtering », in : *Advances in neural information processing systems*, 2016, p. 3844-3852.
  - [108] Muhan ZHANG et al., « An end-to-end deep learning architecture for graph classification », in : *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
  - [109] Aliaksei SANDRYHAILA et José MF MOURA, « Discrete signal processing on graphs », in : *IEEE transactions on signal processing* 61.7 (2013), p. 1644-1656.
  - [110] Keyulu XU et al., « How powerful are graph neural networks ? », in : *arXiv preprint arXiv :1810.00826* (2018).
  - [111] Diederik P KINGMA et Max WELLING, « Auto-encoding variational bayes », in : *arXiv preprint arXiv :1312.6114* (2013).
  - [112] Thomas N KIPF et Max WELLING, « Variational graph auto-encoders », in : *arXiv preprint arXiv :1611.07308* (2016).
  - [113] Daixin WANG, Peng CUI et Wenwu ZHU, « Structural deep network embedding », in : *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, p. 1225-1234.
  - [114] François CHOLLET et al., *Keras*, <https://keras.io>, 2015.
  - [115] Petar VELIČKOVIĆ et al., « Graph Attention Networks », in : *International Conference on Learning Representations* (2018).
  - [116] Nitish SRIVASTAVA et al., « Dropout : a simple way to prevent neural networks from overfitting », in : *The journal of machine learning research* 15.1 (2014), p. 1929-1958.

- 
- [117] Hui ZOU et Trevor HASTIE, « Regularization and variable selection via the elastic net », in : *Journal of the royal statistical society : series B (statistical methodology)* 67.2 (2005), p. 301-320.
  - [118] Diederik P KINGMA et Jimmy BA, « Adam : A method for stochastic optimization », in : *arXiv preprint arXiv :1412.6980* (2014).
  - [119] Franco SCARSELLI et al., « The graph neural network model », in : *IEEE Transactions on Neural Networks* 20.1 (2008), p. 61-80.
  - [120] A. SANFELIU et K. FU, « A distance measure between attributed relational graphs for pattern recognition », in : *IEEE Transactions on Systems, Man, and Cybernetics SMC-13.3* (mai 1983), p. 353-362, DOI : 10.1109/TSMC.1983.6313167.
  - [121] Vladimir Iosifovich LEVENSHTEIN, « Binary codes capable of correcting deletions, insertions and reversals. », in : *Soviet Physics Doklady* 10.8 (1966), Doklady Akademii Nauk SSSR, V163 No4 845-848 1965, p. 707-710.
  - [122] Pierre-François MARTEAU, « Sequence Covering for Efficient Host-Based Intrusion Detection », in : *IEEE Transactions on Information Forensics and Security* 14.4 (avr. 2019), p. 994-1006, DOI : 10.1109/TIFS.2018.2868614, URL : <https://hal.archives-ouvertes.fr/hal-01653650>.



## **Annexes**

## CEMRACS

---

Cette thèse s'est déroulée dans le cadre d'une convention CIFRE, aussi des problématiques de recherche tierces, n'ayant pas un lien fort avec le sujet principal de la thèse, ont été traitées suivant les besoins de l'entreprise. Nous présentons ci-après un de ces travaux annexes, mené dans le cadre de l'édition 2018 du Centre d'Eté Mathématique de Recherche Avancée en Calcul Scientifique (CEMRACS), au Centre International de Recherche en Mathématiques (CIRM), à l'université de Marseille.

Le sujet de cette session était d'approcher les modèles de simulation biologiques avec une approche orientée automate, définis comme un graphe de fonctions biologiques simples.

Cette session de recherche ayant abouti à une publication dans les actes de cette manifestation, nous l'incluons ici dans cette première annexe.

## CONSTRUCTION OF A STATISTICAL LEARNING TOOL BASED ON ORDINARY DIFFERENTIAL EQUATIONS TO MODEL THE DIGESTIVE BEHAVIOUR OF ROSS CHIKENS

NICOLAS BLOYET<sup>1,2</sup>, HÉLÈNE FLOURENT<sup>2</sup>, EMMANUEL FRÉNOD<sup>2,1</sup>, MAROUAN HANDA<sup>3</sup>,  
HAROLD MOUNDOYI<sup>4</sup> AND THUY VO THI PHUONG<sup>5</sup>

**Abstract.** Being able to monitor and forecast farm animal performances is a strategic problem in the agronomy industry. We use a Data-Model Coupling approach to build a biomimetic Statistical Learning tool taking into account some aspects of the biological dynamics of the animal body. The objective is to build a tool which is able to assimilate data about daily feed consumption and measured performances.

The model encompasses several sub-models corresponding to compartments and permitting to mimic a kinetic process divided into several steps. Each sub-model contains parameters which can be learnt by using an optimization algorithm and data.

The goal of the first application of the model on field data was to simulate and predict the growth of chickens. An experiment was performed during 70 days to collect every day the feed consumption and the weight gain of a male and a female chickens. After the learning of the model parameters, the model shows a very good approximation of the chicken's weight evolution over time.

### INTRODUCTION

Being able to monitor and forecast farm animals performances is a strategic problem in the agronomy industry: since several decades, endeavors have been made in this activity sector, with the aim to optimize the breeding-related production, as for example meat, eggs, milk, etc. (See Filipe et al. 2012).

Today, new technologies permit to monitor farm animals and collect a wide range of information as explained in Büchel and Sundrum 2014, Miekley et al. 2012, Suganthi Jemila and Suja Priyadharsini 2018, van der Tol and van der Kamp 2010 and Holman et al. 2011. Yet, those tools are still expensive and their use still infrequent. Furthermore, biological data contain high variability (Noise, heterogeneity, missing and aberrant values, etc.) (See Locke et al. 2005Qi et al. 2006). Therefore, to treat most of biological issues we have to build precise predictive tools from few exploitable data.

Animals are complex living organisms in which intakes induce complex physico-chemical phenomena. Therefore to be able to link inputs and outputs concerning the evolution of some sensed biological factors, we need a mathematical model taking into account some aspects of the animal's body dynamics (See Renzullo et al. 2008).

<sup>1</sup> See-d, 6, rue Henri Becquerel - CP 101, 56038 Vannes Cedex, France

<sup>2</sup> Université Bretagne Sud, Laboratoire de Mathématiques de Bretagne Atlantique, UMR CNRS 6205, Campus de Tohannic, Vannes, France

<sup>3</sup> Université de Picardie Jules Verne, LAMFA UMR 7352, 33 Rue St Leu, 80000 Amiens, France

<sup>4</sup> Sorbonne Université, PR UPMC, Laboratoire de Biologie Intégrative des Modèles Marins, Place Georges Teissier 29680 Roscoff, France

<sup>5</sup> Université Paris 13 - Galilée Institute - LAGA, 99 avenue Jean-Baptiste Clément - 93430 Villetteuse, France

The main objective of this study is to build a tool which is able to predict outputs from inputs concerning farm animals and also able to manage the existence of underlying complex biological phenomena. In this paper we are going to present the construction of a Statistical Learning tool based on a dynamical mathematical model corresponding to a system of Ordinary Differential Equations (ODE). The first application of this Statistical Learning tool consisted to simulate and predict the growth of chickens.

## Purpose

Growth simulation is a well-known problem, and some classical models - such as the Gompertz's and the Verhulst's models - are very well suited to fit some growth data as it is showed in Birch 1999 and Roush et al. 2006. Nevertheless those classical growth models do not permit to perform data assimilation. Indeed, almost no input data can be integrated in those kind of models.

On another hand, some works had already treated biological modeling issues by developing realistic and specific models as in Taghipoor 2012. But the construction of those realistic models is a costly task resulting in models containing a lot of equations and unknown parameters making them difficult to implement.

A model-free approach is also explored, as in Guardabasso et al. 1987 for instance. Machine Learning tool based on Neural Networks were developed to simulate and predict the evolution of biological factors in Gorczyca et al. 2018. This approach, which are only based on data does not need knowledge about the link existing between the used inputs and outputs and permits to develop easier to handle models. But those models need a large amount of data to be fitted and compensate the lack of taken into account knowledge (See Valletta et al. 2017).

In the light of the existing methods for predicting biological responses, we decided to explore an approach which can be defined as being half-way between “model-free” and “full-model” approaches: we explored the Model-Data Coupling theory to construct a tool integrating biological knowledge in a mathematical model, and using data, to optimize the model parameters. Model-Data Coupling is an expanding approach but primarily developed to treat issues in meteorology (See Simmons and Hollingsworth 2002), hydrology (See Kim and Barros 2002, L. Crosson et al. 2002 and Mackay et al. 2003) and biogeochemistry (See Barrett et al. 2005, Barrett 2002, Rayner et al. 2005 and SACKS et al. 2006).

We built our model by splitting the whole complex kinetic into several parsimonious sub-biomimetic-processes performed by a combination of successive compartments. All those compartments compute their respective behaviors, and then exchange messages, assimilable to fluxes. To do that we built several Ordinary Differential Equations integrating the mathematical expressions of biological phenomena (storage, saturation, etc.).

## Scope

In the scope of this study, we focus on meat poultry and specifically two biological strains :

- Ross 308 (Aviagen)
- Cobb 500 (Cobb-Vantress)

Some informations on those animals (an average individual of each strain) are publicly available (REF) and so we know, for each sex of those strains:

- their daily consumption of food, during 70 days
- their daily growth, during 70 days

As a first step, we are going to consider the growth (mass gained each day) and the loss (ratio of food unmetabolized).

In this paper we are going to introduce in Section 1 the applied methodology. Then in Section 2 we will present the developed mathematical model and the results obtained by using this model to simulate the growth of chickens.

## 1. METHODOLOGY

### 1.1. Biological-like function

The basis computational unit used in the following work is what we defined here as *organ-like compartment*. We define it as a tuple :

- $\text{input} \in \mathbb{R}^{in}, in \in \mathbb{N}$
- $\text{output} \in \mathbb{R}^{out}, out \in \mathbb{N}$
- $\text{function} : \mathbb{R}^{in} \rightarrow \mathbb{R}^{out}$
- $\text{state} \in \mathbb{R}^{states}, states \in \mathbb{N}$

where *input* and *output* can be treated as vectors of fixed dimensions *in* and *out*, which are each relative to a flux of a particular kind. There is a mapping between those vectors through *function*, which corresponds to a *biological-like function*. This function models a biological phenomenon in a synthesized way. Finally, *state* is a vector allowing these computational units to have a state, in other words a kind of memory, which can take part in the expression of the said function.

The dimensions of these input / output vectors therefore correspond to information exchanges in the form of messages between the organ-like compartments, the nature of which can be, for example, nutrients, drugs active principles, etc. These functions can for example try to mimic a phenomenon of fixing, convection, diffusion, etc. Figure 1 shows how such a biological-like function can be summarized as a computational unit.

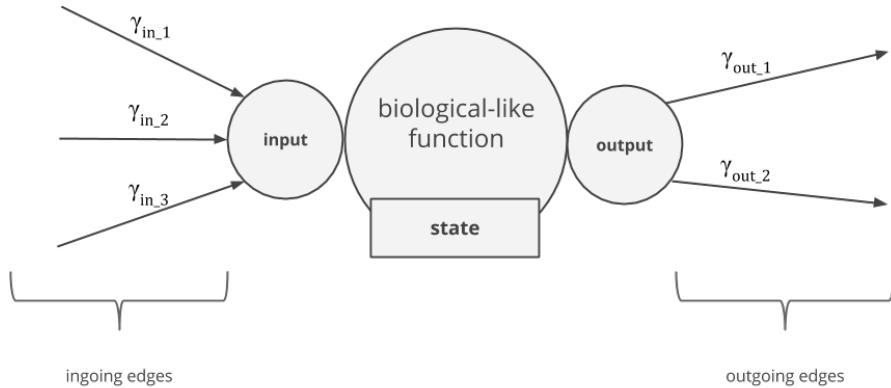


FIGURE 1. A biological-like function

### 1.2. Functions network

These deliberately simple functions are not complex enough to model an organism. However, by making the contact between them, it becomes possible to model a more complex structure. To do this, a communication is established from the output vector of a biomimetic function to the input vector of the following function (Figure 2). Since these functions have states, the network as a whole can be likened to a finite state machine.

### 1.3. Optimization steps

This network acting like a model containing several parameters to optimize. We can distinguish two groups of parameters.

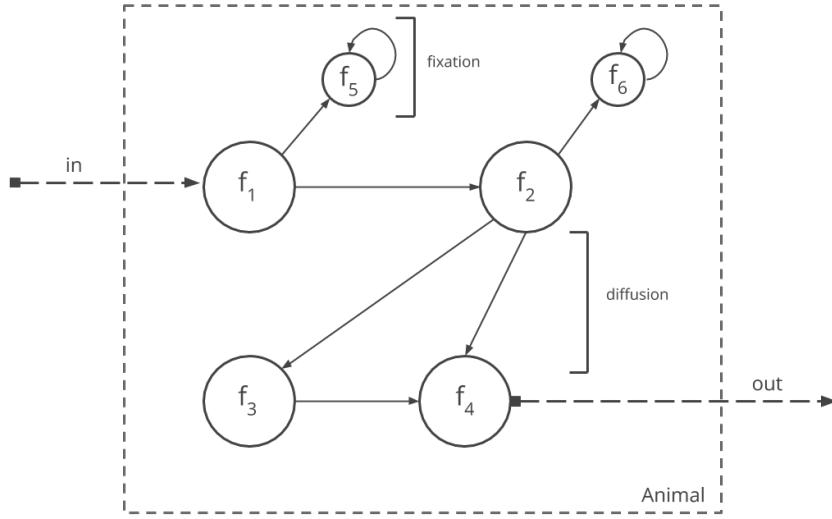


FIGURE 2. Example of a graph-model

### 1.3.1. Biological-like functions (organ-level)

The first step of optimization is at the level of biological-like functions. Most of the phenomena that we are trying to model involve parameters corresponding for example to the part of information exchanged between the compartments, saturation thresholds, etc. These are optimized in order to fit as much as possible the data.

To determine these parameters we use a non-linear optimization algorithm implemented in Matlab (fmincon). This algorithm allows us to find the set of parameters minimizing the model error towards data used.

### 1.3.2. Network structure (organism level)

The nature of the links between these different functions also involves parameters (weighting link, pre-processing of the vector), and is therefore also subject to optimization.

In our study, these parameters essentially correspond to the amount of information exchanged between the compartments. Given the small number of computational units involved at this stage, and so a reasonable dimensionality of the problem, it is possible to use a conventional optimization by just including them in the above set of parameters, but potentially on a complex structure, a heuristic search method (genetic algorithm for example) could be preferred.

We could also favor separated processes of optimization, where each organ would be adjusted independently, before adjusting the whole network.

## 1.4. Main goals

The goal of the following work is to build a Statistical Learning tool that can be used to simulate the food consumption and mass gain of poultry, to obtain results close to what one would obtain with a growth law fitting, with the singular difference that this association of function will not only be a function of time but also of the food consumed, and thus would have an increased biological correspondence.

It therefore seeks to replicate stem growth data, but adjusting a network of biomimetic functions instead of a Gompertz logistic function. The latter will, however, be used as a reference point for the metrics considered (coefficient of determination  $R^2$  and root mean square error  $RMSE$ ).

## 2. CONSTRUCTION OF THE MODEL

Growth requires substrate and complex physico-chemical phenomena to convert substrate into dry weight. Weight formation can be influenced by many factors such that substrate provision, assimilation, digestibility, storage, daily losses, environmental parameters, etc. Modeling of all those processes would lead to the construction of a huge model. Therefore we do not want to reproduce the digestion kinetic but just integrate into the mathematical model information exchanges, delay, fixation, accumulation and saturation effects.

In the model, we use the Gompertz growth equation as in M. C. Kathleen [2017] :

$$W'(t) = A \ln \left( \frac{W_f}{W(t)} \right) W(t), \quad (1)$$

where  $W$  is the weight which varies in time  $t$ ,  $W_f$  is the maximum weight that can be reached with the available nutrients and  $A$  is a constant.

### 2.1. Assumptions

In our model, we assume that the organs participating in the food digestion are the stomach, the small intestine and the large intestine. We built our model under the following assumptions:

- (A1) The ingested food is stored in the compartment named “crop” before moving to the compartment assimilable to the stomach.
- (A2) The flux rates are constant between compartments.
- (A3) There exist some transmission delays between some compartments.
- (A4) The small intestine emptying dynamic starts after some time  $\tau$ .

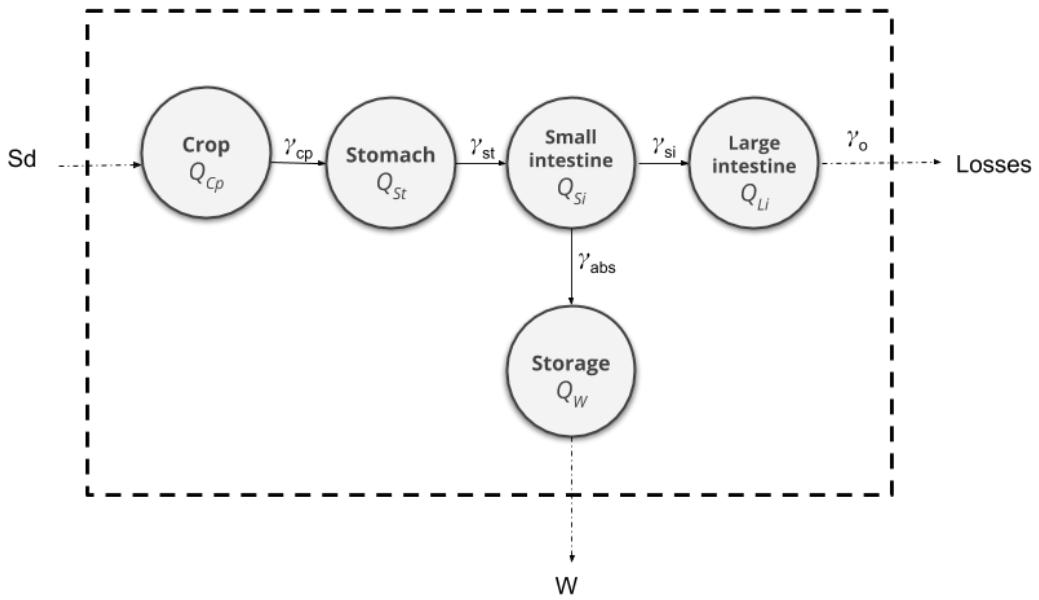


FIGURE 3. Digestive system graph

Following those assumptions, we can draw the graph associated to our model and synthesizing the digestive system of a chicken as in Figure 3, where

- $S_d$  is the amount of food consumed each day;

- $Q_{cp}$  (resp.  $Q_{st}, Q_{si}, Q_{Li}, Q_w$ ) is the information inside the crop (resp. stomach, small intestine, large intestine, storage compartment);
- $\gamma_{cp}$ , (resp.  $\gamma_{st}, \gamma_{si}, \gamma_{abs}$ ) is the rate of nutrients transferred from crop to stomach (rep. from stomach to small intestine, from small intestine to large intestine, from small intestine to storage compartment).
- $\gamma_o$ , corresponds to the non-metabolized part of the nutrients.

The dynamical system of the digestive functions at each day  $d$  are synthesized by the ODEs:

$$Q'_{cp,d} = -\gamma_{cp}Q_{cp,d} \quad (2)$$

$$Q'_{st,d} = \gamma_{cp}(1 - \exp(-\beta_1 t))Q_{cp,d} - \gamma_{st}Q_{st,d} \quad (3)$$

$$Q'_{si,d} = \begin{cases} \gamma_{st}Q_{st,d}, & t \leq \tau \\ \gamma_{st}Q_{st,d} - (\gamma_{abs} + \gamma_{si})Q_{si,d}, & t > \tau \end{cases} \quad (4)$$

$$Q'_{Li,d} = \gamma_{si}Q_{Li,d} - \gamma_{Li}Q_{Li,d}, \quad (5)$$

with the initial conditions

$$Q_{cp,d}(0) = S_d; \quad (6)$$

$$Q_{st,d}(0) = 0; \quad (7)$$

$$Q_{si,d}(0) = 0; \quad (8)$$

$$Q_{Li,d}(0) = 0. \quad (9)$$

Note here that there is some delay in the procedure of information transmission from crop to stomach exponentially with rate  $\beta_1$ . The storage compartment uses a proportion of nutrients going out from small intestine to gain weight and all the information left from the day before is kept to be used in the present day, that gives

$$Q'_{w,d} = \gamma_{abs}Q_{si,d} - \alpha Q_{w,d}; \quad (10)$$

$$Q_{w,d}(0) = Q_{w,d-1}. \quad (11)$$

The quantity of the growth machinery is proportional to the amount of information transmitted to the storage compartment. However, there are some losses during the information transmission corresponding to the part of the nutrient used via physical activities, reproduction needs and the non-metabolizable nutrients. Those losses are proportional to the weight of the animal and nutrients produced from metabolism,

$$\begin{cases} W'_d = \kappa Q_{w,d} - \left( W_{max} \frac{Q_{w,d}}{K+Q_{w,d}} \right) \frac{1}{K} W_d \\ W_d(0) = 0 \end{cases}.$$

The effectiveness of the machinery  $\kappa$  exponentially decays in time with the delay  $\mu$ ,

$$\begin{cases} \kappa' = -\mu\kappa \\ \mu' = \beta_2(\mu_{max} - \mu) \end{cases}. \quad (12)$$

Equation (12) follows that

$$\kappa(t) = \kappa_0 \exp \left[ -\mu_{max} \left( t - \frac{1 - \exp(-\beta_2 t)}{\beta_2} \right) \right]. \quad (13)$$

As a result, the final weight is the total gain up to considering day,

$$W_f = W_0 + \sum_{d=1}^{N_d} W_d(t_f), \quad (14)$$

where  $N_d$  is the number of days and  $t_f$  is the time at the end of a digestive cycle of a day.

## 2.2. Global optimization

The parameters contained into the different biological-like functions are initially unknown. Those parameters which can be learned by using data and optimization algorithms confer to the model a learning capability. Therefore we used the function *directL* (Johnson [2008]) existing in R (R Core Team [2018]), to learn the model parameters. To find the values of the parameters permitting the model to fit the data, this function minimize an objective function. In this application the objective function corresponds to the mean squared difference between the initial and the predicted curves (15),

$$f_{Obj} = \frac{1}{n} \sum_{i=1}^n \left( PredCurve(i) - InitCurve(i) \right)^2 \quad (15)$$

where *PredCurve(i)* and *InitCurve(i)* respectively correspond to the  $i^{th}$  point of the predicted curve and the  $i^{th}$  point of the initial curve, and  $n$  is the total number of point, that is 70 days.

In this application we fixed  $\tau$  and we fitted  $\beta_1$ ,  $\gamma_{st}$ ,  $\gamma_{abs}$ ,  $\gamma_{si}$ ,  $\gamma_o$ ,  $\mu_{max}$ ,  $\beta_2$ ,  $V_{max}$ ,  $K$  and  $\gamma_{cp}$ .

## 2.3. Simulation results

After the fitting we parametrized the model with the obtained values of the parameters (Table 1).

The dynamics of the model throughout one day and throughout two days (Figure 4) were simulated with the Female Model, that is the model simulated the growth of the female chicken. We also simulated the growth of the animals throughout the whole studied period (Figure 5).

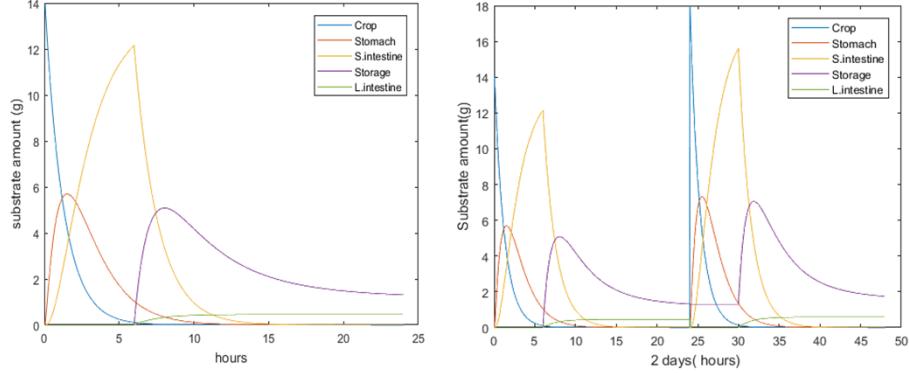


FIGURE 4. Dynamics of the model throughout one day (left) and throughout two days (right).

Parameters	Female Model	Male Model
$\beta_1$	0.405	0.402
$\gamma_{st}$	0.601	0.650
$\gamma_{abs}$	0.650	1.00
$\gamma_{si}$	0.020	0.014
$\gamma_o$	0.424	0.512
$\mu_{max}$	0.008	0.008
$\beta_2$	0.016	0.048
$V_{max}$	0.051	0.050
$K$	24.70	24.90
$\gamma_{cp}$	1.654	0.971

TABLE 1. Parameter values of the Male and Female Models.

### 3. DISCUSSIONS

#### 3.1. Knowledge feedback

First of all, considering for the same strain males on one side, and females on the other, it is interesting to note that some parameters seem to be stable, while only a part of them vary significantly among poultry's sex. In that way, the adjusted model give us back some information, that could be confronted to some business expertise (here zootechnicians).

Indeed, it appears that organism's difference of behavior is concentrated on parameters :

- $\beta_2$ , relative to the effectiveness of the organism <sup>1</sup>
- $\gamma_{abs}$ , the link's strength between the small intestine and the storage compartment
- $\gamma_{si}$ , the link's strength between the small and the large intestine
- $\gamma_{cp}$ , the link's strength between the crop and the stomach

In particular, we observe that male's organism seem to stay highly efficient longer than female's (a fact that we observe in ground truth), characterized by a slightly better  $\beta_2$  on the male. At the same time, we observe that on female's, the amount of material sent from the small intestine to the big intestine (which is going to be lost) is higher than male's, where a bigger part is sent to the storage compartment.

Those observations would need to be confirmed on other strains on to be confornted to a zootechnician expertise, but is by itself a first step of knowledge feedback.

#### 3.2. Accuracy

Figure 5 shows that the built model fits the data with a goodness of fit close to our target. We chose to compare our model with two types of models : a Gompertz growth (17) law usually used to simulate growth phenomena, and a two order Polynomial Model (16) took as a basis with the same number of parameters. The latter one doesn't have a biological likelihood, so the first goal is to perform better.

$$W(t) = p_0 + p_1 t + p_2 t^2 \quad (16)$$

$$W(t) = g_0 \exp(-g_1 \exp(-g_2 t)) \quad (17)$$

We fitted the parameters of the Polynomial Model ( $p_0, p_1, p_2$ ) and the ones of the Gompertz's Model ( $g_0, g_1, g_2$ ) on the data concerning the female and the male chickens.

We calculated for each model the mean of the  $R^2$  and the  $RMSE$  values associated the Female and the Male Models. Figures 5, 6 and 7 and the results contained in Table 2 show that the accuracy of those different models is globally equivalent. Therefore we built a model satisfying in terms of accuracy. But the real advantage of the Biomimetic Model is its capability to integrate biological knowledge and assimilate input data.

Model	$RMSE$	$R^2$
Gompertz's Model	94.2	0.997
Order 2 Polynomial Model	147.4	0.992
Biomimetic Model	102.1	0.996

TABLE 2. Comparison of the models in terms of  $R^2$  and  $RMSE$  values.

---

<sup>1</sup>its ability to metabolize a high ratio of ingested food with respect of time

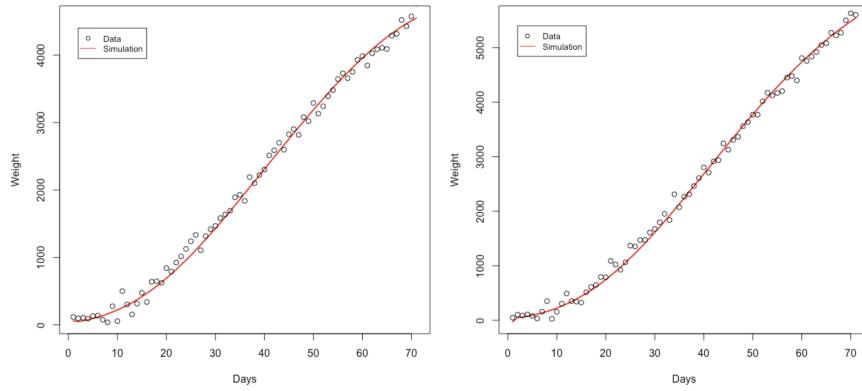


FIGURE 5. Comparison between the initial curve (Data) and the curve simulated via the Biomimetic Model (Simulation) for the Female (left) and the Male Model (right).

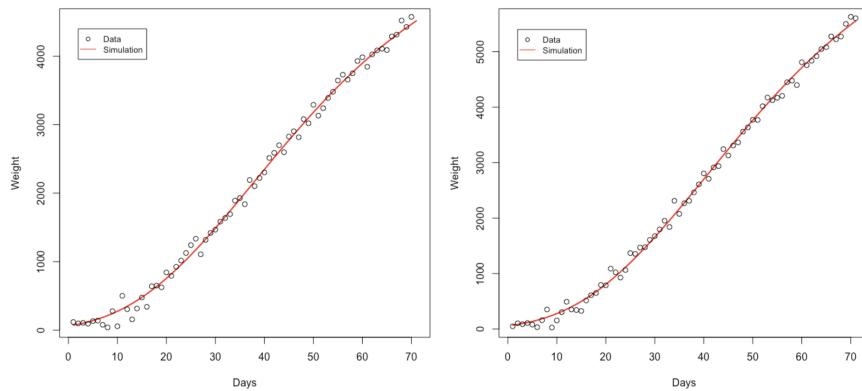


FIGURE 6. Comparison between the initial curve (Data) and the curve simulated via the Gompertz's Model (Simulation) for the Female (left) and the Male Model (right).

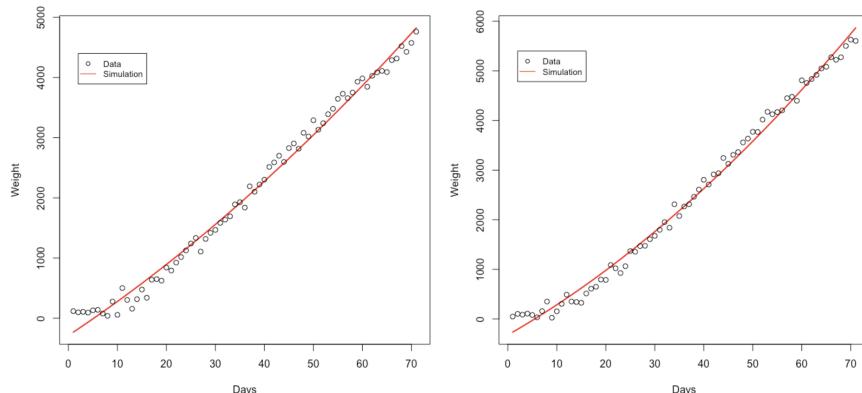


FIGURE 7. Comparison between the initial curve (Data) and the curve simulated via the Polynomial Model (Simulation) for the Female (left) and the Male Model (right).

## 4. PERSPECTIVES AND CONCLUSION

### 4.1. Perspectives

The developed approach allows us to stay very modular with regards to the nature of the simulated model, so it could adapt to a higher level of information for a few adaptations.

#### 4.1.1. Fluxes multiplication

In the scope of this work, the model was only trained to deal with simple weightings data, and so not in a context where it could out-perform classic growth laws. The actual added value to those models would appear as soon as more fine data would be available.

Indeed, the objective at mid-term is to exploit an eventual higher level of knowledge (theoretical or measured) about modeled phenomena, helping to define in a more precise way what happens inside the organism, typically with data centered on an organ's input and output. Unlike classic growth laws based on temporal axis (Gompertz, Verhulst for example), this kind of model could handle data made of several dimensions (time, ingested proteins, drugs, etc.), and among everything, could pretend to follow a biological likelihood.

#### 4.1.2. Model reinterpretation

At a longer term, and in a context where the previous hypothesis about new data would be validated, the goal would be to optimize automatically the weight parameters between biological-like functions.

Knowing the organism's structure to a sufficient level of details, and having a sufficient number of measures, it would be possible to refine the knowledge concerning the nature of the connections between these different virtual organs, by set of parameters optimization (genetical optimization for example).

As those parameters do have a relationship with actual phenomenon due to their biological likelihood, it would be interesting to confront the final set of optimized parameters to a business expertise.

### 4.2. Conclusion

We explored a Data-Model Coupling approach and built a biomimetic dynamic tool containing Ordinary Differential Equations. Those Differential Equations contain parameters conferring a learning capability to the developed tool. Therefore, our tool is a Statistical Learning tool able to learn the value of parameters from field data.

An optimization process permits to learn the parameters and then parametrize the mathematical model by integrating input data concerning the feed consumption of the chickens. After the learning step we obtained a model able to model the growth of Ross chickens to a level of precision similar to a classic growth law.

The number of parameters to determine is important and only one structure of the model was tested. Therefore model selection methods should be used to determine the optimal structure of the model in terms of accuracy and the number of parameters to learn.

## REFERENCES

- Damian Barrett, M Hill, Lindsay Hutley, Jason Beringer, Johnny H. Xu, Garry Cook, John Carter, and Richard J. Williams. Prospects for improving savanna biophysical models by using multiple-constraints model-data assimilation methods. *Australian Journal of Botany*, 53(7), 01 2005. .
- Damian J. Barrett. Steady state turnover time of carbon in the australian terrestrial biosphere. *Global Biogeochemical Cycles*, 16(4):55–1–55–21, 2002. . URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2002GB001860>.
- Colin PD Birch. A new generalized logistic sigmoid growth equation compared with the richards growth equation. *Annals of Botany*, 83(6):713–723, 1999.
- S. Büchel and A. Sundrum. Short communication: Decrease in rumination time as an indicator of the onset of calving. *Journal of Dairy Science*, 97(5):3120 – 3127, 2014. ISSN 0022-0302. . URL <http://www.sciencedirect.com/science/article/pii/S0022030214001684>.
- Patricia Filipe, Carlos Braumann, and Carlos Roquete. Multiphasic individual growth models in random environments. *Methodology and Computing in Applied Probability*, 14:49–56, 03 2012. .

- Michael T. Gorczyca, Hugo Fernando Maia Milan, Alex Sandro Campos Maia, and Kifle G. Gebremedhin. Machine learning algorithms to predict core, skin, and hair-coat temperatures of piglets. *Computers and Electronics in Agriculture*, 151:286 – 294, 2018. ISSN 0168-1699. . URL <http://www.sciencedirect.com/science/article/pii/S0168169918303612>.
- V. Guardabasso, D. Rodbard, and P. J. Munson. A model-free approach to estimation of relative potency in dose-response curve analysis. *American Journal of Physiology-Endocrinology and Metabolism*, 252(3):E357–E364, 1987. . URL <https://doi.org/10.1152/ajpendo.1987.252.3.E357>. PMID: 3826362.
- A. Holman, J. Thompson, J. E. Routly, J. Cameron, D. N. Jones, D. Grove-White, R. F. Smith, and H. Dobson. Comparison of oestrus detection methods in dairy cattle. *Veterinary Record*, 169(2):47–47, 2011. ISSN 0042-4900. . URL <https://veterinaryrecord.bmjjournals.com/content/169/2/47>.
- Steven G. Johnson. The nlopt nonlinear-optimization package. 2008.
- Gwangseob Kim and Ana P Barros. Space-time characterization of soil moisture from passive microwave remotely sensed imagery and ancillary data. *Remote Sensing of Environment*, 81(2):393 – 403, 2002. ISSN 0034-4257. . URL <http://www.sciencedirect.com/science/article/pii/S0034425702000147>.
- William L. Crosson, Charles A Laymon, Ramarao Inguva, and Marius P. Schamschula. Assimilating remote sensing data in a surface flux-soil moisture model. *Hydrological Processes*, 16:1645–1662, 06 2002. .
- James C. W. Locke, Andrew J. Millar, and Matthew S. Turner. Modelling genetic networks with noisy and varied experimental data: the circadian clock in arabidopsis thaliana. *Journal of theoretical biology*, 234 3: 383–93, 2005.
- T. Even M. C. Kathleen. The use of gompertz models in growth analyses, and new gompertz-model approach: An addition to the unified-richards family. *PloS One*, 12(6):e0178691., 2017.
- D.Scott Mackay, Sudeep Samanta, Ramakrishna R. Nemani, and Lawrence E. Band. Multi-objective parameter estimation for simulating canopy transpiration in forested watersheds. *Journal of Hydrology*, 277 (3):230 – 247, 2003. ISSN 0022-1694. . URL <http://www.sciencedirect.com/science/article/pii/S0022169403001306>.
- Bettina Miekley, Imke Traulsen, and Joachim Krieter. Detection of mastitis and lameness in dairy cows using wavelet analysis. *Livestock Science*, 148(3):227 – 236, 2012. ISSN 1871-1413. . URL <http://www.sciencedirect.com/science/article/pii/S187114131200220X>.
- Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins: Structure, Function, and Bioinformatics*, 63(3):490–500, 2006. . URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.20865>.
- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL <https://www.R-project.org/>.
- P. J. Rayner, M. Scholze, W. Knorr, T. Kaminski, R. Giering, and H. Widmann. Two decades of terrestrial carbon fluxes from a carbon cycle data assimilation system (ccdas). *Global Biogeochemical Cycles*, 19(2), 2005. . URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2004GB002254>.
- Luigi J. Renzullo, Damian J. Barrett, Alan S. Marks, Michael J. Hill, Juan P. Guerschman, Qiaozhen Mu, and Steve W. Running. Multi-sensor model-data fusion for estimation of hydrologic and energy flux parameters. *Remote Sensing of Environment*, 112(4):1306 – 1319, 2008. ISSN 0034-4257. . URL <http://www.sciencedirect.com/science/article/pii/S0034425707003227>. Remote Sensing Data Assimilation Special Issue.
- WB Roush, WA Dozier 3rd, and SL Branton. Comparison of gompertz and neural network models of broiler growth. *Poultry Science*, 85(4):794–797, 2006.
- WILLIAM J. SACKS, DAVID S. SCHIMEL, RUSSELL K. MONSON, and BOBBY H. BRASWELL. Model-data synthesis of diurnal and seasonal co2 fluxes at niwot ridge, colorado. *Global Change Biology*, 12(2):240–259, 2006. . URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2486.2005.01059.x>.
- A. J. Simmons and A. Hollingsworth. Some aspects of the improvement in skill of numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 128(580):647–677, 2002. . URL <https://rmets.onlinelibrary.wiley.com/doi/abs/10.1256/003590002321042135>.
- J Suganthi Jemila and S Suja Priyadharsini. A sensor-based forage monitoring of grazing cattle in dairy farming. *International Journal on Smart Sensing and Intelligent Systems*, 11:1–9, 01 2018. .

- Masoomeh Taghipoor. *Modélisation du transport, de la dégradation et de l'absorption des aliments dans l'intestin grêle*. PhD thesis, Université François Rabelais-Tours, 2012.
- John Joseph Valletta, Colin Torney, Michael Kings, Alex Thornton, and Joah Madden. Applications of machine learning in animal behaviour studies. *Animal Behaviour*, 124:203 – 220, 2017. ISSN 0003-3472. . URL <http://www.sciencedirect.com/science/article/pii/S0003347216303360>.
- Rik van der Tol and Arjen van der Kamp. Time series analysis of live weight as health indicator. In *Proceedings of the first North American conference precision dairy management*, pages 230–231, 2010.



**Titre :** Caractérisation et plongement de sous-graphes colorés

**Mot clés :** Isomorphisme de graphes, Apprentissage de plongements, Apprentissage sur graphes, modèles structure-activité (QSAR)

**Résumé :** Dans le domaine de la chimie, il est intéressant de pouvoir estimer des propriétés physico-chimiques de molécules, notamment pour des applications industrielles. Celles-ci sont difficiles à estimer par simulations physique, présentant une complexité temporelle prohibitive. L'émergence des données (publiques ou privées) ouvre toutefois de nouvelles perspectives pour le traitement de ces problèmes par des méthodes statistiques et d'apprentissage automatique. La principale difficulté réside dans la caractérisation des molécules : celles-ci s'apparentent davantage à un réseau d'atomes (autrement dit un graphe coloré) qu'à un vecteur. Or, les méthodes de modélisation statistiques traitent usuellement avec des observations encodées comme telles, d'où la nécessité de méthodes spécifiques, nommées

relations structures-activité, traitant des observations encodées sous forme de graphes. Le but de cette thèse est de tirer parti des corpus publics pour apprendre les meilleures représentations possibles de ces structures, et de transférer cette connaissance globale vers des jeux de données plus restreints. Nous nous inspirons pour ce faire de méthodes utilisées en traitement automatique des langages naturels. Pour les mettre en oeuvre, des travaux d'ordre plus théorique ont été nécessaires, notamment sur le problème d'isomorphisme de graphes. Les résultats obtenus sur des tâches de classification/régression sont au moins compétitifs avec l'état de l'art, voire meilleurs, en particulier sur des jeux de données restreints, attestant des possibilités d'apprentissage par transfert sur ce domaine.

**Title:** Characterization and embeddings of colored subgraphs

**Keywords:** Graph Isomorphism, Embeddings-Learning, Graph-Learning, Structure-Activity Relationships (QSAR)

**Abstract:** In the field of chemistry, it is interesting to be able to estimate the physicochemical properties of molecules, especially for industrial applications. These are difficult to estimate by physical simulations, as their implementation often present prohibitive time complexity. However, the emergence of data (public or private) opens new perspectives for the treatment of these problems by statistical methods and machine learning. The main difficulty lies in the characterization of molecules: these are more like a network of atoms (in other words a colored graph) than a vector. Unfor-

tunately, statistical modeling methods usually deal with observations encoded as such, hence the need for specific methods able to deal with graphs-encoded observations, called structure-activity relationships. The aim of this thesis is to take advantage of public corpora to learn the best possible representations of these structures, and to transfer this global knowledge to smaller datasets. We adapted methods used in automatic processing of natural languages to achieve this goal. To implement them, more theoretical work was needed, especially on the graph isomorphism problem. The results ob-

tained on classification / regression tasks are stricted data sets, attesting some opportunities at least competitive with the state of the art, for transfer learning in this field. and even sometimes better, in particular on re-