UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale Santé, Sciences et Techniques

Laboratoire d'Informatique — EA 6300

THÈSE présenté par :

Nicolas SIDÈRE

soutenue le : 24 Février 2012

pour obtenir le grade de : Docteur de l'université François Rabelais de Tours

Discipline/ Spécialité : Informatique

Contribution aux méthodes de reconnaissance structurelle de formes : approche à base de projection de graphes

DEVANT LE JURY COMPOSÉ DE :

Jean-Yves RAMEL	*Directeur*	Professeur des Universités Université François Rabelais de Tours
Pierre HÉROUX	*Encadrant*	Maître de Conférence Université de Rouen
Jean-Marc OGIER	*Rapporteur*	Professeur des Universités Université de La Rochelle
Christine SOLNON	*Rapporteur*	Professeur Institut National des Sciences Appliquées de Lyon
Luc BRUN	*Examinateur*	Professeur École Nationale Supérieur d'Ingénieur de Caen
Josep LLADÒS	*Examinateur*	Professeur Université Autonome de Barcelone

Remerciements

Je n'aurai jamais pu réaliser cette thèse sans l'aide de nombreuses personnes qui ont contribué, d'une manière ou d'une autre, à sa réalisation. Voici le moment de les remercier.

En premier lieu, je tiens à exprimer toute ma gratitude à Jean-Yves Ramel et à Pierre Héroux qui m'ont fait confiance pour mener à terme cette thèse. Tout au long de cette aventure, ils ont su me guider, me conseiller et m'encourager quand j'en avais besoin.

Je remercie également les personnes qui ont accepté de prendre le temps d'évaluer mon travail. Merci donc à Luc Brun d'avoir présider le jury et à Josep Lladós de s'être déplacé depuis Barcelone pour m'apporter questions et remarques. Je remercie tout autant les rapporteurs : Christine Solnon pour la minutie de sa lecture et ses multiples corrections et Jean-Marc Ogier pour ses retours pertinents mais aussi pour m'avoir donné l'envie de faire une thèse lorsque j'étais étudiant.

Mes plus sincères remerciements vont aux membres du LI de Tours et du LITIS de Rouen pour m'avoir accueilli dans d'excellentes conditions, contribuant ainsi au bon déroulement de ces travaux. J'exprime une reconnaissance identique envers les personnes de Polytech'Tours et de l'Université de Rouen.

J'adresse mes remerciements les plus chaleureux à ma famille : mes parents — Betty et Bernard — pour m'avoir toujours suivi et aidé moralement mais aussi financièrement sans conditions et à chaque instant de mon cursus, à mes grands-parents — Joachim et Michelina — pour leur pragmatisme et leur vision de la vie, à mon frère — Jérémy — pour l'intérêt qu'il a porté sur mon travail ainsi qu'à mes cousins, cousines, oncles et tantes. Je pense tout aussi fort à ma deuxième famille : Claude, Marie-Jo, Manou, Marion, Pierre-Hubert et Juju, sans oublier leurs conjoints et enfants, pour leur soutien sans faille et leurs encouragements sans limite.

Un grand merci à mes amis dont l'aide non-quantifiable m'a été des plus précieuse. Savoir que l'on peut compter sur ses proches est inestimable, mais être entouré de personnes capables de vous changer les idées, de vous remonter le moral, de vous entendre rire est tout aussi important. Dans le désordre : Pepette, le B, Benoit, Marianne, Benji, Yan, Steph, Galou, Marielle, Guinou, Panchen, Pitou, Sabrina, Barlotte, Morgane, Micka, Patrick et tous les autres !

Pour finir, je ne peux oublier Delphine qui m'a accompagné au quotidien tout au long de ces dernières années, que ce soit dans les moments de détente ou de stress, par des petites gestes ou simplement quelques mots. Un tel soutien vaut de l'or. Merci Didi !

Résumé

Par opposition aux méthodes statistiques qui utilisent des vecteurs de caractéristiques pour décrire les formes dans leur globalité, les méthodes structurelles reposent sur une représentation analytique des formes, les décomposant en sous-parties et modélisant les relations existant entre ces parties, la plupart du temps à l'aide de graphes. Les graphes sont, en effet, dotés d'un fort pouvoir de représentation ainsi que d'une capacité d'adaptabilité aux données importante les rendant incontournables dans de nombreuses applications. Cependant, lorsqu'il s'agit de comparer et de reconnaitre des formes, ils sont affectés par les imperfections des outils de comparaison disponibles aujourd'hui. Alors que de nombreux modèles statistiques existent, aucun formalisme clair ne permet, à l'heure actuelle, de définir une mesure de similarité entre deux graphes qui soit efficace et générique.

Les techniques plus récentes de projections de graphes semblent alors constituer une alternative intéressante en cherchant à repositionner les graphes dans un espace vectoriel et permettre ainsi l'utilisation d'outils statistiques éprouvés.

S'inspirant des techniques de sondage de graphes, nous présentons, tout d'abord, une nouvelle méthode de projection de graphes s'appuyant sur la construction d'un lexique générique constitué des sous-graphes non-isomorphes. A la manière des méthodes dites de "sac de graphes", la représentation vectorielle proposée est construite en dénombrant les occurrences de chacun des motifs contenus dans le lexique. Doté de motifs complexes (avec plusieurs sommets et plusieurs arêtes), le lexique permet de conserver l'information topologique qui est portée par le graphe lors de la projection.

Les graphes utilisés en reconnaissance des formes véhiculent souvent une information riche au travers des étiquettes associées aux sommets et/ou aux arêtes. Afin de tenir compte de ces données lors de la projection, un enrichissement de la technique précédente est proposée et constitue la seconde contribution de ce travail. Deux méthodes sont définies pour rendre l'approche proposée compatible aussi bien avec des étiquettes continues (donc non-dénombrables) que discrètes : la première méthode s'appuie sur une discrétisation puis une combinaison des attributs des étiquettes alors que la seconde est basée sur un partitionnement de l'espace des étiquettes.

De nombreuses expérimentations sont également disponibles et largement discutés tout au long du manuscrit afin de mieux cerner les caractéristiques et avancées apportées par les propositions exposées.

Mots clés : Méthodes structurelles de reconnaissance des formes — Projection de graphes — Classification et comparaison de graphes.

Abstract

In contrast to the statistical methods that use feature vectors to describe patterns in a global way, the structural methods are based on a analytical representation of patterns by decomposing them in their constituting sub-parts and modeling the relationships between these subparts, mostly employing graphs. The high representation power and the capacity to adapt to underlying data makes graphs an important data structure for a wide range of application domains.

However, when it comes to comparing and recognizing patterns, graphs suffers from the lack of availability of efficient computational algorithms and tools. While many efficient statistical computational models exists but no formalism today defines an efficient and generic similarity measure for graphs.

Recent graph embedding techniques seem to be an interesting alternative by mapping graphs into vector spaces and enabling the use of mature statistical tools.

Inspired by techniques of graph probing, we first present a new method of graph embedding based on the build of a generic lexicon made of non-isomorphic subgraphs. After the manner of methods called "bag of graphs", the proposed vectorial representation is built by counting the occurrences of each pattern contained in the lexicon. Made of complex patterns (with several vertices and several edges), the lexicon allows to preserve the topological information carried by the graph during the embedding.

Graphs used in pattern recognition, often carry rich information through labels related to vertices and/or edges. To keep these details within the embedding, an update of the previous technique is presented and makes-up the second contribution of this work. Two methods are defined to manage the compatibility either with continuous labels (non countable) or discrete ones : the first method focuses on a discretisation followed by a combination of labels attributes, the second is based on a clustering of the label space.

Also, many experiments are available and discussed through the document in order to figure out features and advances brought by the presented propositions.

Keywords : Structural pattern recognition methods — Graph embedding — Graph classification and comparison

Table des matières

Liste des tableaux

Table des figures

Introduction Générale

> Un problème sans solution est
> un problème mal posé
>
> Albert Einstein

À l'ère du tout numérique, on parle de plus en plus de concept d'intelligence ambiante ; les capteurs se multiplient dans les rues, dans les maisons, partout... La masse de signaux numériques à analyser ainsi que leur diversité devient alors considérable (sons, images, vidéos, séries temporelles...). Les besoins de mécanismes puissants (cloud computing) d'analyse, d'indexation, de recherche voire de prise de décision automatique, ou tout au moins assistée, se généralisent et deviennent un enjeu crucial pour les années à venir. L'intelligence artificielle et plus particulièrement la reconnaissance des formes, dont il est question dans cette thèse, se situe au cœur de cette problématique.

La reconnaissance de formes peut être vue comme la transposition à l'informatique de la faculté humaine d'analyser les signaux qui l'entourent, visuels ou sonores, afin de les comparer, de les classifier ou de les identifier. Aujourd'hui, l'Homme reste un système de reconnaissance des plus parfaits. En effet, nous sommes capables de distinguer sans réfléchir une multitude de formes issues de différentes sources. Par exemple, nous pouvons facilement et naturellement différencier un miaulement d'un aboiement, rechercher Charlie dans une foule ou trier le linge propre du sale. Cette diversité de tâches qu'il peut réaliser est sans doute la plus grande force de l'Homme et du monde animal en général ; il semble aujourd'hui difficilement imaginable de concevoir une machine capable de reproduire l'ensemble de ces capacités. Néanmoins, de nombreux travaux de recherche ont été réalisés et sont actuellement en cours pour tenter de se rapprocher le plus possible des capacités humaines. En effet, si pour l'instant un système capable d'atteindre les performances d'un être humain est impensable, certains cas particuliers méritent d'être automatisés. Par exemple, le recueil puis l'analyse d'un ensemble de signaux sismiques peut se révéler une tâche fastidieuse et rend ainsi la qualité des prévisions de futurs séismes moins fiables. D'un point de vue plus industriel, il est difficile pour un opérateur de repérer un défaut sur une chaîne de production trop rapide, ou à un ouvrier de placer les mêmes vis dans les mêmes trous à longueur de journées sans trouver une certaine lassitude. L'intérêt de la reconnaissance des formes est donc là : assister ou remplacer l'humain dans des tâches trop lourdes à réaliser, afin de gagner du temps ou réduire leurs pénibilités.

Lors d'un processus de reconnaissance de forme, plusieurs principes sont mis en application. Une des étapes nécessaires est la caractérisation d'une forme. Pour introduire

ces approches, prenons l'exemple d'une tâche dont l'objectif serait de décrire des photos d'animaux. Une technique serait de dépeindre leur taille (petit, moyen ou grand), leurs formes ou leurs couleurs ; autrement dit, selon une description globale de l'animal. Il s'agit de l'approche statistique. Une autre méthode serait de représenter ces animaux selon leur morphologie (4 pattes—1 corps—1 tête pour un chien, 2 pattes—2 ailes—1 corps—1 tête pour un oiseau...). On parle alors d'une approche analytique ou structurelle.

Cette thèse s'inscrit dans le contexte de la reconnaissance structurelle des formes et étudie plus précisément l'utilisation des graphes dans ce cadre. Les graphes sont des outils de représentation universels. De par leur pouvoir de représentation et de leur adaptabilité, les graphes sont très largement employés dans les thématiques de la vision par ordinateur, de la chimie, de l'imagerie médicale... Le chapitre 1 présente les concepts de la reconnaissance des formes, le formalisme de la théorie des graphes ainsi que quelques exemples de leur utilisation. Cette partie conduit également à la problématique de la comparaison de graphes.

Outre la caractérisation, une autre étape importante est la classification qui conduit à l'identification d'une forme. La classification est intimement liée aux notions de comparaison et d'apprentissage, et ces notions seront différentes et adaptées aux modèles de représentations. Prenons l'exemple de la lecture pour illustrer nos propos. On oppose habituellement deux types de méthodes d'apprentissage de la lecture : globale ou syllabique. Nous tenons à préciser que notre exemple est avant tout illustratif, objectif et qu'aucun jugement n'est porté sur ces méthodes. La méthode globale a pour ambition de faire acquérir à l'élève une stratégie de déchiffrage des mots, voire des phrases, en tant qu'image visuelle indivisible. Cette approche peut s'apparenter aux représentations statistiques. La lecture se fait par la reconnaissance d'un mot en entier. En effet, tout bon lecteur ne déchiffre pas, mais reconnaît les mots, voire des groupes de mots, visuellement [1]. La méthode syllabique se fonde sur la genèse des sons de la langue parlée par assemblage de syllabes. Elle repose sur les propriétés phonétiques de notre alphabet et a comme base les lettres et les sons. Une fois que ceux-ci sont maîtrisés, la lecture s'effectue en composant des syllabes puis des mots. C'est le fameux "b.a.-ba" (où les lettres "b" et "a" donnent la syllabe "ba"). Cette méthode peut être qualifiée d'analytique. Nous pouvons observer que l'apprentissage diffère selon le type de méthode choisi : d'un coté l'apprentissage est effectué sur les mots entiers, de l'autre coté il est opéré sur des sous-parties de mots (lettres ou syllabes). La lecture, que nous appelons classification, a pour objectif d'identifier les mots lus. Un processus de comparaison doit donc être en place pour évaluer la similarité entre mots. Il est important de remarquer que si la comparaison est plus aisée par une méthode globale (approche statistique), ce qui en fait d'ailleurs un atout pour les défenseurs de cette méthode, elle est peu évolutive et l'apprentissage d'un nouveau mot devra se faire à partir d'un modèle connu. Les méthodes syllabiques (approche structurelle) sont beaucoup plus souples et l'apprentissage de mots nouveaux en est plus facile. En revanche, la reconnaissance d'un mot décomposé en lettres est soumise à un processus plus complexe qui oblige

1. Sleon une édtue de l'uvinertisé de Cmabrigde, l'odrre des ltteers dnas un mtos n'a pas d'ipmrotncae, la suele coshe ipmrotnate est que la pmeirère et la drenèire soit à la bnnoe pclae. Le rsete peut êrte dnas un dsérorde ttoal et vuos puoevz tujoruos lrie snas porlblème. C'est prace que le creaveu hmauin ne lit pas chuaqe ltetre elle-mmêe, mias le mot cmome un tuot.

La peruve...

dans un premier temps à identifier les lettres, puis ensuite les associer pour former un mot que l'on comparera avec les mots appris, lettre par lettre. Sur cette difficulté se basent les jeux de lettres tels que le *Scrabble*® ou les *mots croisés* où le but est de retrouver un mot à partir de lettres tirées aléatoirement ou de lettres manquantes. Ces exemples montrent la problématique de l'utilisation de méthodes structurelles. De la même manière, en utilisant des graphes pour représenter une forme, la problémtique de la comparaison, *a fortiori* de la classification, reste encore ouverte même si de nombreux travaux existent. Ceux-ci sont présentés dans le chapitre 2.

Ces questions peuvent trouver des réponses en utilisant des algorithmes d'appariement, *i.e.* trouver une correspondance entre les sommets d'un graphe et les sommets d'un autre graphe qui satisfasse à certaines contraintes ou critères d'optimalité pour que les graphes soient considérés comme similaires. Il y a selon nous deux grandes catégories de méthodes d'appariement de graphes. La première catégorie inclut les méthodes qui utilisent une approche de mise en correspondance des éléments de deux graphes, par exemple, la recherche d'isomorphisme exact de graphes, de sous-graphes ou encore la recherche de sous-graphes communs similaires. La deuxième catégorie permet la tolérance d'erreur. Elle propose de rechercher des solutions plus approximatives, dans le sens où une tolérance d'erreur est admise. Un compromis entre l'erreur tolérée et la précision de la mesure doit donc être établi selon l'application visée. De plus, si on relaxe les contraintes au niveau de l'appariement des éléments des graphes, les méthodes approximatives deviennent beaucoup plus robustes contre le bruit et les distorsions présentes dans les objets. Toutefois, la mesure d'une dissimilarité définie par un appariement reste trop rigide dans le sens où la réponse sera booléenne : soit deux graphes sont similaires, soit il ne le sont pas. Afin d'affiner l'évaluation de la similarité entre deux graphes, différentes approches ont été conçues pour pouvoir déterminer une distance entre graphes. La plus reconnue est sans doute la distance d'édition de graphes fondée sur les coûts de transformation d'un graphe en un autre. Bien que déterminer ces coûts reste encore aujourd'hui une tâche complexe, elle reste néanmoins indispensable puisqu'elle garantit d'excellents résultats si les coûts sont adaptés aux données du problème. Ces coûts sont néanmoins attachés à une représentation spécifique et nécessitent d'être recalculés pour chaque problème. De plus, ces distances, en général, nécessitent des temps de calcul assez élevés. D'autres travaux proposent de projeter les graphes dans un espace vectoriel pour pouvoir bénéficier d'outils statistiques déjà définis. Malgré ces nombreux travaux, les verrous scientifiques restant à résoudre sont encore nombreux, surtout lorsque l'on s'attache à produire des algorithmes génériques.

Notre contribution porte sur une nouvelle projection de graphes palliant certaines lacunes des méthodes existantes. Notre méthode s'inscrit dans le concept de sondages de graphes qui fondent leurs représentations vectorielles sur une exploration du graphe et sur l'extraction de certaines caractéristiques comme le degré des nœuds ou les plus courts chemins. Nous avons remarqué que ces approches portaient certaines lacunes, par exemple une faible représentativité des informations topologiques ou une capacité de généralisation pauvre. Nous proposons, dans le chapitre 3 une nouvelle méthode de projection de l'information topologique qui s'appuie sur le dénombrement d'occurrences de motifs (avec une information topologique riche) issus d'un lexique indépendant de l'application et donc générique. Nous avons également tenu compte de sa complexité, liée à l'utilisation des graphes, en développant un algorithme performant tout en étant exhaustif.

Le chapitre 4 est consacré à la projection de l'information liée aux étiquettes. Pour poursuivre notre idée directrice, nous présentons une technique permettant d'octroyer une notion d'étiquetage à la projection de la topologie. Nous nous sommes confrontés à deux problèmes liés au nombre d'attributs que pouvaient comporter les étiquettes, leurs différentes natures (numérique ou symbolique) ainsi que leurs nombres. Nous exposons deux méthodes pour répondre à ces contraintes : la première repose sur une discrétisation des attributs numériques suivie d'une combinaison des attributs possibles, la deuxième s'appuie sur un partitionnement de l'espace des étiquettes.

Nous terminons par une conclusion dans laquelle nous dressons un bilan des approches proposées dans cette thèse. Nous y développons également quelques perpectives sur l'optimisation de notre méthode mais également sur sa généralisation à d'autres domaines.

S'il le souhaite, le lecteur peut trouver à la fin de ce mémoire plusieurs annexes qui, si elles ne sont pas impératives à la compréhension des travaux présentés ici, peuvent fournir plus de détails sur certaines notions, et des résultats additionnels.

Chapitre 1

Utilisation des graphes en reconnaissance de formes

Un petit dessin vaut mieux
qu'un long discours

Napoléon

1.1 Objectifs et concepts de la reconnaissance de formes

1.1.1 Présentation générale

De manière générale, la reconnaissance des formes regroupe l'ensemble des méthodes visant à reproduire les capacités de l'Homme à identifier des caractères, des objets, des sons, *etc.* en les comparant à un ou plusieurs modèles. Selon le contexte, l'objectif est donc d'évaluer une similarité ou une dissimilarité entre :

- la forme inconnue et une forme de référence, afin d'estimer si un appariement entre ces deux formes est possible.
- la forme inconnue et une classe de formes, permettant de décider si la forme inconnue possède les caractéristiques de la classe et être considérée comme appartenant à cette classe.
- la forme inconnue et un ensemble de classes, la forme inconnue étant assignée à la classe pour laquelle la ressemblance est la plus forte.

La complexité de la tâche, la diversité des applications et la montée en puissance des ordinateurs font que la reconnaissance des formes est un domaine où les travaux de recherche évoluent depuis des décennies. De ce fait la littérature est abondante. Parmi tous ces travaux, nous pouvons lister quelques ouvrages qui font figure de référence :

- [Watanabe, 1985]
- [Weiss et Kulikowski, 1991]
- [Nadler et Smith, 1993]
- [Jain *et al.*, 2000]
- [Duda *et al.*, 2001]

– [Kuncheva, 2004]

En introduction, nous avons vu que ces méthodes pouvaient être classées en plusieurs catégories selon le formalisme de représentation des formes choisi. Nous présentons dans la sous-section suivante les approches statistiques et les approches structurelles. Il existe néanmoins d'autres méthodes basées sur des approches floues ou encore neuronales par exemple. Ces représentations étant beaucoup moins communes et hors du cadre de cette thèse, elles ne seront pas abordées.

1.1.2 Les approches structurelles et statistiques

En reconnaissance des formes dite statistique, les formes sont représentées par un ensemble de propriétés. Concrètement, une forme est caractérisée par un nombre fini de n mesures, formant un vecteur de taille n. Plus formellement, une forme peut être considérée comme un point dans un espace à n dimensions, *i.e.* $\mathbf{x} = (x_1, x_2, ..., x_n)$. D'un point de vue mathématiques, un point et un vecteur ont des définitions différentes. Les deux termes sont néanmoins utilisés indifféremment dans la littérature. En fait, les formes sont représentées comme des points dans un espace à n dimensions (aspect géométrique) mais considérées comme des vecteurs, permettant l'utilisation des propriétés d'un espace euclidien (aspect statistique).

Afin d'illustrer la représentation statistique d'une forme, nous considérons l'image de la figure 1.1. Nous nous sommes inspirés des travaux de [Novak et Shafer, 1992] pour extraire un vecteur numérique caractérisant la couleur de cette image. Le vecteur est composé des intensités moyennes des 3 couleurs primaires (rouge, vert et bleu) calculées sur l'ensemble de l'image.

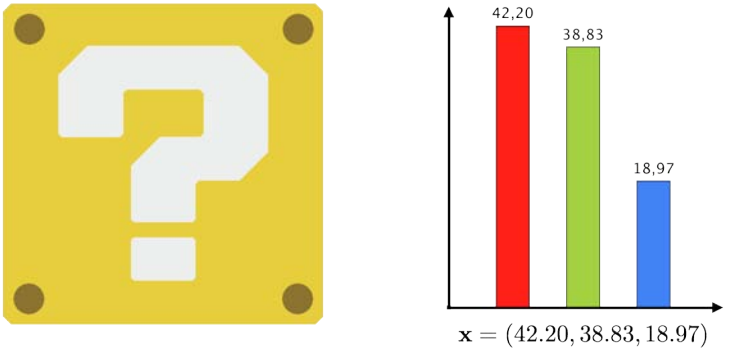


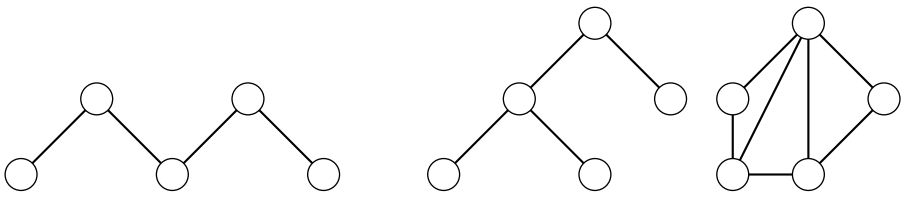
FIGURE 1.1 – Exemple de la représentation d'une image par l'histogramme des 3 couleurs primaires.

Par opposition, dans l'approche structurelle, une forme est représentée par une décomposition en sous-parties et par la description de la relation entre ces sous-parties. Plusieurs structures de données peuvent être alors utilisées pour modéliser une forme. Les chaînes permettent d'encapsuler des informations de façon séquentielle et sont souvent utilisées par les méthodes de lecture automatique pour modéliser un mot (un sommet représente une

lettre et le chaînage définit l'ordre de ces lettres) ou une liste de mots (un sommet = un mot). Les chaines peuvent être également utilisées dans des systèmes de reconnaissance moins triviaux. Par exemple, dans [Ros *et al.*, 2005], les auteurs s'appuient sur le parcours visuel de lecture d'une image pour la caractériser sous forme de chaîne. En analyse d'image, la méthode décrite dans [Freeman, 1961] s'appuie sur le parcours du contour d'un objet pour le décrire suivant la direction d'un pixel à son voisin. La similarité est alors calculée par une distance d'édition ([Levenshtein, 1966]) ou bien à l'aide de mesures telles que la moyenne ou l'écart-type définis dans [Jolion, 2003].

Parmi les structures utilisées pour représenter une forme, nous pouvons également citer les arbres. Les exemples d'applications à la reconnaissance de forme sont nombreux. Dans [Nagy et Seth, 1984, Cesarini *et al.*, 1999], les arbres sont utilisés pour représenter de façon hiérarchique la mise en page de documents numérisés pour ensuite les comparer. Des procédés identiques sont également exploités pour les images naturelles([Haj et Aghbari, 2008]). Les arbres sont également très utilisés en recherche d'informations sur le Web. Par exemple, dans [Reis *et al.*, 2004], les auteurs s'appuient sur les une repésentation XML (langage permettant une représentation semi-structurée d'un document sous forme d'arbre) pour comparer deux documents. Cette comparaison peut se faire au moyen de la distance d'édition entre arbres ([Selkow, 1977]).

D'un point de vue algorithme, ces structures de données peuvent être généralisées comme des cas particuliers des graphes définis comme un ensemble fini de sommets connectés par des arêtes. Une chaîne est un graphe dont les sommets sont consécutivement connectés par une arête. L'arbre est un graphe pour lequel il n'existe qu'un seul parcours possible entre deux sommets. De manière générale, un graphe permet de représenter une forme en se basant sur une segmentation de cette forme en sous-parties et sur la description de la relation entre ces objets. La figure 1.2 illustre ces trois représentations.



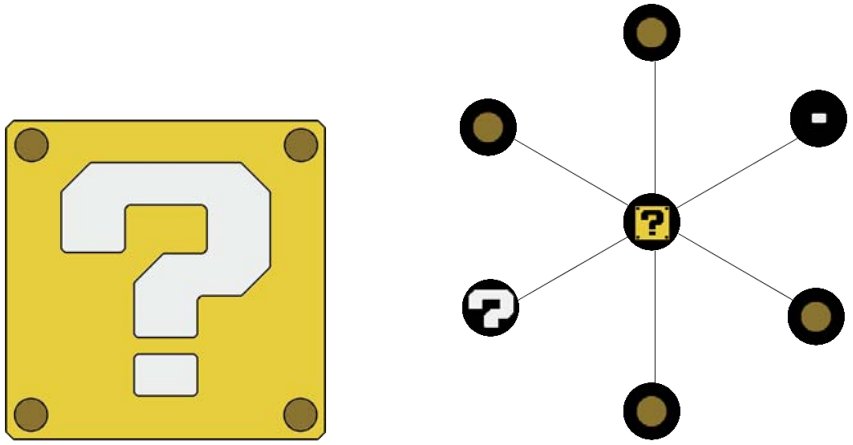
(a) Une chaîne. (b) Un arbre. (c) Un graphe.

FIGURE 1.2 – Illustration des différentes structures de données utilisées pour représenter une forme.

D'autres modèles existent pour caractériser une information structurelle ; nous pouvons par exemple citer les cartes combinatoires. Dans [Damiand *et al.*, 2004], les auteurs proposent leurs utilisation pour représenter des images naturelles. Cependant, nous nous focaliserons, dans cette thèse, sur les méthodes à base de graphes.

Nous présentons un exemple illustré dans la figure 1.3. La représentation de l'image originale (1.1(a)) sous forme de graphe en se basant sur une segmentation en régions (1.3(a)), [Rosenfeld, 1974]. Chaque sommet v_i du graphe représente alors une région de l'image et les arêtes représentent les relations d'adjacence entre les régions (1.3(b)). Notons que des informations supplémentaires peuvent être ajoutées sur les sommets et/ou sur les arêtes.

Ces informations sont appelées des étiquettes. Dans chaque sommet, les étiquettes peuvent contenir des valeurs numériques ainsi que des valeurs symboliques pour décrire la région correspondante. Les arêtes peuvent elles aussi être étiquetées pour préciser et affiner les relations entre les régions.



(a) (b)

FIGURE 1.3 – Exemple de la représentation d'une image par un graphe d'adjacence.

L'utilisation des vecteurs de caractéristiques pour la représentation des formes (approches statistiques) donne accès à un ensemble d'opérateurs qui permettent le calcul de distances, de moyennes ou de médianes en un temps linéaire par rapport à la dimension des vecteurs et au nombre de vecteurs. Comparer des formes représentées par des vecteurs est donc très facile. De plus, ces différentes mesures statistiques, en plus d'être faciles à calculer, sont mathématiquement bien définies. Il devient possible de mettre en place des techniques d'apprentissage et classification, d'analyse de données, de statistiques pour produire des modèles de reconnaissance de formes très (trop ?) adaptés aux caractéristiques des données à traiter. En contrepartie, la rigidité de la représentation apporte quelques limitations. En effet, les vecteurs doivent être de taille fixe et identique pour pouvoir être comparés. Or, les caractéristiques au sein des formes à reconnaître peuvent être fluctuantes au sein d'un même problème. La représentation ne sera donc pas idéalement adaptée à toutes les formes à décrire. La taille de ce vecteur devient donc un paramètre pouvant être complexe à fixer. Nous pouvons supposer que si le vecteur comporte beaucoup de caractéristiques alors la description sera meilleure. Aux vues des capacités de mémoire dont nous pouvons facilement disposer dans un ordinateur, la limite de cette taille pourrait être très élevée. Malheureusement, parmi ces caractéristiques, certaines peuvent être erronées car inadaptées au problème ou perturbées par un bruit. Ce problème est qualifié de fléau de la dimension ([Bellman, 1961]). De plus, la description globale d'une image peut être un frein dans certaines applications lorsque l'objectif est de reconnaître une partie d'image. La méthode de Viola et Jones ([Viola et Jones, 2001]) pourtant reconnue comme une des plus performante pour la détection de visage reste moyennement efficace en cas d'occultation ou de bruit trop important. Outre la taille du vecteur caractéristique, le choix des des-

cripteurs est également une contrainte forte et fluctuante aux vues des propriétés exigées : indépendance, corrélation, discrimination, séparabilité des classes...

Contrairement aux vecteurs, les graphes n'ont aucune contrainte de taille leur donnant ainsi plus de flexibilité. Cela permet donc une plus grande souplesse de représentation et rend ce modèle adaptable à la diversité des objets au sein d'une même collection. De plus, les graphes bénéficient d'une richesse de représentation en offrant la possibilité d'une description analytique, par opposition à une description globale. Ainsi, un graphe peut représenter un objet en le décomposant en sous-parties et permet de décrire les relations entre ces sous-parties. L'étiquetage des sommets permet une description des sous-parties et celui des arêtes de préciser la nature des relations. Il est donc possible de combiner l'usage des graphes avec une approche statistique ; les outils statistiques étant individuellement utilisés sur les étiquettes de nœuds ou des arcs. La réciproque n'est pas possible. En revanche, aucun cadre formel n'est *a priori* disponible actuellement. Bien que la théorie des graphes possède une longue histoire derrière elle, il n'existe pas d'opérateur générique, pour comparer des formes représentées par des graphes attribués, telles que celles utilisées dans les approches statistiques. Cette contrainte oblige alors une spécificité des outils. La représentation est meilleure mais la reconnaissance est plus difficile. De plus, les opérations de comparaison sont souvent complexes, et augmentent les coûts de traitements les rendant parfois incompatibles avec les contraintes du monde réel.

1.1.3 Exemples

Nous allons maintenant illustrer les différences que ces représentations engendrent dans un système de reconnaissance des formes. Nous présentons en figure 1.4, deux briques, ou tétrominos, issues du jeu mondialement connu *Tetris*®. Les tétrominos sont des pièces possédant des motifs différents, ici les tétrominos I (1.4(a)) et O(1.4(b)), qu'il faut emboîter à la manière d'un puzzle. Le joueur doit donc dans un premier temps identifier la forme pour pouvoir la positionner au meilleur emplacement possible.

(a) Tétromino I. (b) Tétromino O.

FIGURE 1.4 – Tetrominos.

La tâche de reconnaissance est donc réduite à sa plus simple expression, *i.e.* déterminer si les deux tétrominos sont similaires ou différents. Pour une approche statistique, nous avons choisi de caractériser les tétrominos par un vecteur de deux caractéristiques : la longueur et la largeur exprimées en pixels. Le tétromino I est caractérisé par $\mathbf{p}_I = \binom{160}{40}$ et le tétromino par $\mathbf{p}_O = \binom{80}{80}$. Pour mesurer leur similarité, nous pouvons facilement utiliser la distance entre deux points ou comparer leur ratio $\frac{Longueur}{Largeur}$.

Pour une approche structurelle, les tétrominos sont représentés par des graphes. Chaque brique composant une pièce est un sommet et la connexité entre deux briques est modélisée par la présence d'une arête. La figure 1.5 fournit les graphes représentant le tétromino I en 1.5(a) et le tétromino *0* en 1.5(b), respectivement G_I et G_0. Si visuellement une distinction peut-être faîte de façon intuitive, les outils formels pour comparer ces deux graphes sont plutôt complexes à mettre en œuvre. Par exemple, une similarité peut être vérifiée si les sommets d'un graphe peuvent être appariés avec les sommets de l'autre graphe. Cet appariement doit évidemment conserver la connexité des briques représentée par les arêtes des graphes. Dans notre exemple, un appariement est impossible puisque deux sommets de G_I sont connectés à un autre sommet (les deux autres portent chacun deux arêtes) tandis que pour le graphe G_O tous les sommets sont reliés à deux autres sommets. Nous pouvons finalement en conclure que les tétrominos sont différents mais après une analyse plus complexe et plus longue.

(a) Graphe G_I du tétromino I

(b) Graphe G_O du tétromino O

FIGURE 1.5 – Graphes représentant les tétrominos de la figure 1.4.

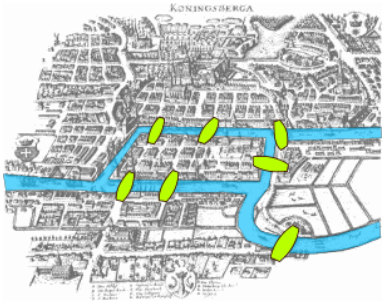
Construire et proposer des méthodes robustes, simples et rapides pour mesurer la similarité entre graphes constitue donc toujours actuellement le coeur de la reconnaissance structurelle des formes. Cette tâche est d'autant plus complexe que les modèles peuvent fluctuer. En effet, des bruits dits structurels peuvent venir perturber l'identification en modifiant la structure ou les étiquettes des graphes. Il est d'autant plus difficile de corriger ces détériorations que leurs natures peuvent être différentes. Un dysfonctionnement du scanner, un mauvais réglage d'appareil, une modélisation inadaptée sont autant de sources de bruit. Il est donc peu pensable d'obtenir à chaque fois une représentation parfaite dans des conditions réelles. En revanche, l'exploitation de mesures de similarités ou de méthodes d'appariement, tolérantes aux bruits, est une solution envisageable et engendre une thématique de recherche étudiée par beaucoup d'équipes. La très riche littérature se focalise donc sur la robustesse des algorithmes d'appariement (ou de calcul de distance entre graphes) qui prennent en considération les erreurs dues aux bruits tout en réduisant leurs complexités. Nous présentons un état de l'art sur la comparaison de graphes dans le chapitre 2.

Dans la section suivante, nous présentons une introduction à la théorie des graphes ainsi que certaines définitions et certains formalismes que nous utiliserons dans ce mémoire.

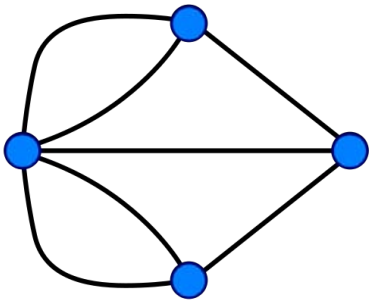
1.2 Les graphes

1.2.1 Histoire des graphes

L'histoire veut que la théorie des graphes débute avec les travaux de Léonard Euler au XVIII^{e} siècle alors qu'il visitait la ville de Königsberg en Prusse orientale. Cette ville est notamment remarquable par sa typologie autour de deux îles reliées entre elles par un pont ; six autres ponts relient le continent à l'une des îles ou l'autre. Les habitants se demandent alors s'il est possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer deux fois par le même et de revenir au point de départ. Euler choisit alors de représenter ce problème sous forme d'un diagramme illustré par la figure 1.6. Les 4 rives terrestres sont alors imagées par des sommets et les ponts par des arêtes reliant ces sommets.



(a) Plan schématique de la ville de Königsberg.



(b) Représentation de la ville de Königsberg sous forme de graphe.

FIGURE 1.6 – Plan de la ville de Königsberg et sa représentation par un graphe. [2].

On peut trouver une multitude de problèmes ayant contribué au développement de la théorie des graphes. Par exemple, le problème du facteur chinois ou problème du voyageur de commerce cherchant à optimiser leur tournée. Le problème du cavalier où un cavalier positionné au hasard sur un échiquier doit visiter l'ensemble de l'échiquier sans passer deux fois par la même case. Le problème du coloriage de carte où chaque région doit être

2. Illustrations extraites de Wikipedia : `http://fr.wikipedia.org/wiki/Problème_des_sept_ponts_de_Königsberg`.

d'une couleur distincte de celle de ses voisines. Certains ressortent même dans la cour de récréation comme la fameuse maison qu'il faut dessiner sans lever le crayon ou encore le jeu du loup, de la chèvre et du chou.

Tous ces problèmes ont trouvé une solution en les modélisant sous forme de diagramme avec des points représentants des concepts et des liens symbolisant une relation entre ces concepts. Par exemple, le problème des sept ponts de Königsberg initial est alors traduit en une modélisation plus théorique autour des propriétés du graphe : « Est-il possible de parcourir tout le graphe en partant d'un point et en empruntant chaque lien une et une seule fois ? ». Cet exemple montre bien le pouvoir de modélisation des graphes mais aussi le fait qu'ils sont le plus souvent utilisés pour modéliser un système où des flux sont à optimiser plutôt que des formes à reconnaître comme nous l'avions déjà noté auparavant.

C'est de ce principe de modélisation d'une entité mais aussi de l'ensemble des relations entre elles sous une unique structure qu'est née la théorie des graphes. Cependant, il a fallu attendre jusqu'au milieu du XIX^{e} pour que les graphes quittent la récréation mathématique pour connaître leur essor en tant que branche entière des mathématiques sous l'impulsion de travaux de chercheurs tels que Arthur Cayley, Peter Guthrie Tait, Percy John Heawood, Frank Ramsey ou Hugo Hadwiger. Plus tard, la théorie des graphes sera réellement étendue par le français Claude Berge ([Berge, 1967],[Berge, 1973]) qui introduit de nouvelles notions telles que les hypergraphes ou les graphes parfaits et par les mathématiciens hongrois Paul Erdõs et Alfréd Renyi par l'introduction de probabilités.

Ces graphes vont connaître un intérêt croissant. En effet, la possibilité de caractériser une grande variabilité de problèmes sous forme de graphes a permis leur utilisation dans des domaines aussi nombreux que variés :

- Chimie : modélisation de molécules
- Électrique : lois de Kirchoff
- Génie logiciel : diagramme état-transition
- Gestion de projet : réseau PERT
- Reconnaissance des formes

Dans cette thèse, nous n'aborderons qu'une petite partie de la théorie des graphes. Nous nous focaliserons sur les aspects d'analyse d'images et de reconnaissance des formes utilisant comme outils des graphes. Avant de faire le panorama des principales méthodes exploitant les graphes dans ce domaine, nous rappelons quelques concepts et définitions de la théorie des graphes.

1.2.2 Définitions et notations

Il est possible de trouver, dans la littérature, plusieurs définitions adoptant différents vocabulaires. Bien sur, toutes ces définitions partagent toutes les concepts permettant de caractériser les aspects structurels et informatifs des graphes. Dans un soucis de clarté, nous allons, dans cette section, définir les objets et leurs notations tels que nous les utiliserons tout au long de ce mémoire.

Dans [Rosenstiehl, 2002], Rosenstielh rapporte une définition informelle mais intuitive de Claude Berge :

Il faut imaginer dans sa tête des trucs qu'on appelle sommets, et pour toute

paire de sommets soit une arête qui les joint, sois une non-arête qui les laisse sans joint : ceci est un graphe.

De cette définition imagée découle un formalisme (Définition 1) que nous adopterons.

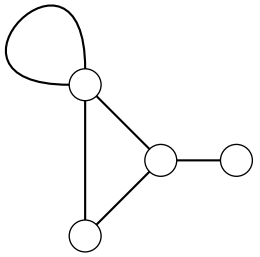
Définition 1 (Graphe) *Étant donnés deux ensembles finis L_V et L_E, un graphe est un 4-tuple tel que $G = (V, E, \mu, \nu)$ avec :*
- *V est un ensemble fini de nœuds.*
- *$E \subseteq V \times V$ est un ensemble d'arcs.*
- *$\mu : V \to L_V$, la fonction d'attribution des étiquettes aux nœuds.*
- *$\nu : E \to L_E$, la fonction d'attribution des étiquettes aux arcs.*

La généricité de cette définition permet de prendre en considération la totalité des types de graphes qui seront utilisés par la suite. Nous allons, dans un premier temps, étudier la topologie des graphes. Pour cela, nous définissons un graphe G comme un couple (V, E) tel que $V = \{v_1, v_2, \ldots, v_n\}$ et $E = \{e_1, e_2, \ldots, e_m\}$ soient deux ensembles finis disjoints et nous omettons les fonctions μ et ν.
Un élément de l'ensemble V est appelé nœud. On notera v_i le i$^{\text{e}}$ nœud du graphe. L'ensemble des nœuds d'un graphe G est noté $V(G)$ ou V_G. La cardinalité de cet ensemble ($|V|$) est appelée **ordre** du graphe.
Un élément de l'ensemble E est appelé arc. L'ensemble des arcs d'un graphe G est noté $E(G)$ ou E_G. Un arc e est défini par un couple de nœuds appelés extrémités. L'ensemble E est donc un sous-ensemble de $V \times V$ tel que $e = (v_i, v_j)$ avec $v_i \in V(G)$ et $v_j \in V(G)$, est un arc dont les extrémités sont les nœuds v_i et v_j. La cardinalité de cet ensemble ($|E|$) est appelée **taille** du graphe.
Dans le cas général, un graphe est dit **orienté** puisque les éléments appartenant à E sont des couples et donc présentent un ordre. Pour $e = (v_i, v_j)$, v_i sera nommé le nœud initial ou source de l'arc et v_j le nœud terminal ou cible. Toutefois, si pour tout $e = (v_i, v_j) \in E$, il existe $(v_j, v_i) \in E$ tel que $(v_i, v_j) = (v_j, v_i)$, autrement dit si E est une relation symétrique sur V, alors le graphe est dit non-orienté. En effet, dans ce cas, la direction d'un arc peut être ignorée. E est alors un ensemble de paire $\{v_i, v_j\}$. On parlera alors de sommets et d'arêtes plutôt que de nœuds et d'arcs. Un graphe **simple**, est un 1-graphe sans boucle.

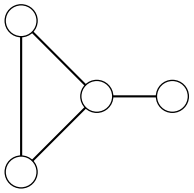
Un arc dont les extrémités coïncident est appelé une **boucle**. Deux arcs sont dits *adjacents* s'ils ont au moins un nœud commun.
Le *degré* du nœud v_i, noté $d_G(v_i)$, est le nombre d'arcs ayant v_i comme extrémité. Le demi-degré *extérieur* du nœud v_i, noté $d_G^+(v_i)$, est le nombre d'arcs ayant v_i comme extrémité initiale. Le demi-degré *intérieur* du nœud v_i, noté $d_G^-(v_i)$, est le nombre d'arcs ayant v_i comme extrémité finale.
Un p-graphe (non-orienté), ou multi-graphe (orienté), est un graphe dans lequel il existe jusqu'à p arcs de la forme (v_i, v_j) entre les sommets v_i et v_j. Dans ce cas là, l'ensemble E devient un multiensemble.

Des exemples de graphes sont illustrés par la figure 1.7. 1.7(a), 1.7(b) et 1.7(c) sont des graphes non-orientés, 1.7(d), 1.7(e) et 1.7(f) sont orientés, 1.7(b) et 1.7(e) sont des graphes simples, 1.7(c) et 1.7(f) sont des p-graphes.

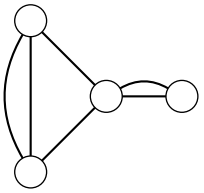
Indépendamment de ces considérations, des informations sont portées sur les nœuds et



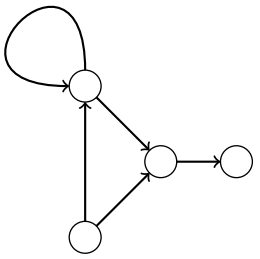
(a) Un graphe non-orienté.



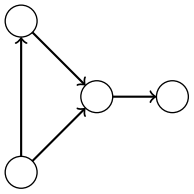
(b) Un graphe non-orienté simple.



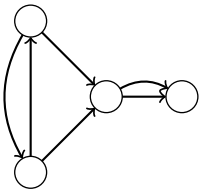
(c) Un multi-graphe.



(d) Un graphe orienté.



(e) Un graphe orienté simple.



(f) Un 2-graphe orienté.

FIGURE 1.7 – Exemples de graphes.

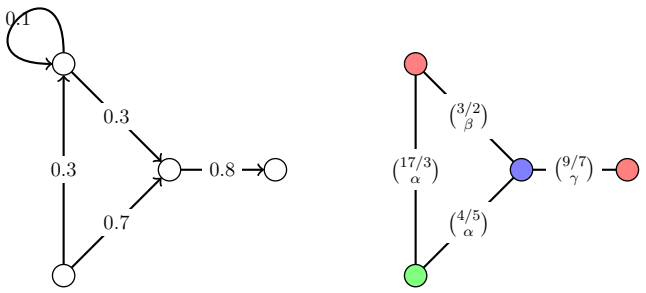
sur les arcs par les fonctions μ et ν avec $\mu : V \to L_V$ et $\nu : E \to L_E$. μ et ν sont les relations associant les nœuds, respectivement les arcs, à leurs étiquettes. L_V et L_E sont les ensembles d'étiquettes possibles respectivement pour un nœud ou pour un arc. Ces étiquettes composées d'attributs. Un attribut est défini par un triplet (identifiant, type, littéral) tel que :

- l'identifiant désigne de façon unique un attribut au sein d'une étiquette.
- le type désigne l'alphabet A que peut prendre l'attribut. Cet alphabet peut être numérique ($A = \mathbb{R}$, $A = \mathbb{N}$, ...) ou nominal ($A = \alpha, \beta, \gamma, \ldots$).
- Le littéral désigne la valeur de l'attribut.

Ainsi, $\mu(v_i) = l_{v_i}$ sera l'étiquette affectée au nœud v_i et $\nu(e_i) = l_{e_i}$ sera l'étiquette affectée à l'arc e_i. Si $L_V = L_E = \emptyset$, alors aucune étiquette n'est portée par les nœuds ou par les arcs : le graphe est dit **non-étiqueté**. Tout graphe non-étiqueté peut être représenté par un graphe étiqueté. Il suffit pour cela d'associer la même étiquette de nœuds à tous les nœuds et la même étiquette d'arcs à tous les arcs.
Si $L_V = \{\emptyset\}$ et que tous les arcs sont étiquetés avec un unique attribut entre 0 et 1, alors le graphe est dit **pondéré** Dans tous les autres cas, le graphe est appelé **étiqueté**.

La figure 1.8 montre deux graphes portant des étiquettes : 1.8(a) est un graphe orienté pondéré avec des étiquettes sur les arcs avec un unique attribut compris entre 0 et 1, 1.8(b)

est un graphe non-orienté dont les sommets sont étiquetés avec 3 attributs (valeurs RGB représentées par une couleur) et les arêtes par 2 attributs (un réel et un symbolique).



(a) Un graphe orienté pondéré.

(b) Un graphe étiqueté non-orienté.

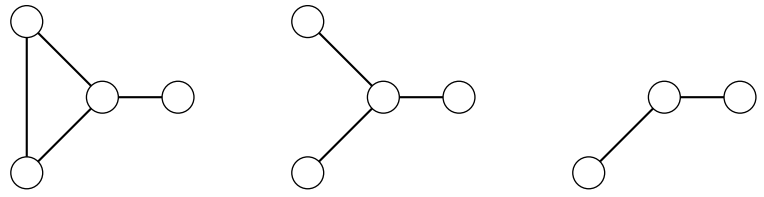
FIGURE 1.8 – Exemples de graphes pondérés et étiquetés.

Nous allons maintenant introduire la notion de **sous-graphe**, définie comme une équivalence de la relation de sous-ensemble dans la théorie des ensembles.

Définition 2 (Sous-graphe) *Soient les graphes g_1 et g_2 tels que $g_1 = (V_1, E_1, \mu_1, \nu_1)$ et $g_2 = (V_2, E_2, \mu_2, \nu_2)$. Le graphe g_1 est un **sous-graphe**, noté $g_1 \subseteq g_2$ si :*

- $V_1 \subseteq V_2$
- $E_1 \subseteq E_2 \cap V_1 \times V_1$
- $\forall v \in V_1, \mu_1(v) = \mu_2(v)$
- $\forall e \in E_1, \nu_1(e) = \nu_2(e)$

Un sous-graphe est obtenu d'un graphe en lui retirant des nœuds et leurs arcs adjacents. Dans ce cas là, on obtient $V_1 \subseteq V_2$ et $E_1 = E_2 \cap V_1 \times V_1$. G_1 est alors appelé **sous-graphe induit** de G_2. Dans le cas contraire, *i.e.* si G_1 est un graphe qui ne contient pas tous les arcs de G_2 ayant leurs extrémités dans V_1, alors il s'agit d'un **sous-graphe partiel**.



(a) Un graphe non-orienté simple.

(b) Un sous-graphe partiel.

(c) Un sous-graphe induit.

FIGURE 1.9 – Exemple de sous-graphes partiel et induit.

1.2.3 Représentations

Si les ensembles sont les structures de données les plus génériques pour représenter un graphe, il en existe néanmoins d'autres. La représentation sous forme de matrices est

couramment utilisée pour sa manipulation plus aisée. De plus, ces matrices sont utilisées dans la théorie spectrale des graphes qui s'intéresse aux rapports entre le spectre d'une matrice d'un graphe et ses propriétés.

La représentation par **matrice d'adjacence** consiste en l'utilisation d'une matrice carrée ayant pour taille l'ordre du graphe. Les coefficients peuvent prendre deux valeurs. A_{ij} vaut 1 si l'arc (i, j) existe, *i.e.* il existe un arc reliant le sommet i au sommet j, 0 s'il n'y a pas d'arc. Si le graphe est non-orienté, alors la matrice d'adjacence est symétrique.

Définition 3 *Soit G un 1-graphe non-étiqueté tel que $G = (V, E)$ d'ordre n. La* ***matrice d'adjacence*** *de G est une matrice de dimension $n \times n$ telle que :*

$$A_{ij} = \begin{cases} 1 & si\ e = (v_i, v_j) \in E \\ 0 & sinon \end{cases}$$

La représentation par **matrice laplacienne** consiste en l'utilisation d'une matrice carrée ayant pour taille l'ordre du graphe. Les coefficients peuvent prendre des valeurs dans $\mathbb{N}$. L_{ij} vaut -1 si l'arc (i, j) existe, i.e. il existe un arc reliant le sommet i au sommet j, le degré du nœud si $i = j$, 0 sinon.

Définition 4 *Soit G un graphe simple non-étiqueté tel que $G = (V, E)$ d'ordre n. La* ***matrice laplacienne*** *d'un graphe d'ordre n est une matrice de dimension $n \times n$ telle que :*

$$L_{ij} = \begin{cases} d(v_i) & si\ i = j \\ -1 & si\ i \neq j\ et\ (v_i, v_j) \in E \\ 0 & sinon \end{cases}$$

1.3 Des données aux graphes

Depuis la fin des années 70, les graphes soulèvent un intérêt croissant dans le domaine de la reconnaissance des formes ([Conte *et al.*, 2007]). Opposées aux méthodes statistiques qui se basent sur un ensemble de caractéristiques pour décrire un objet dans sa globalité, les méthodes structurelles de reconnaissance de formes exploitent une représentation analytique dans laquelle l'objet est représenté par un graphe modélisant les relations existants entre ses sous-parties. De multiples articles traitent de la modélisation sous forme de graphes et différentes techniques permettent de définir les sommets, représentant des composantes de l'objet, et les arêtes, les relations entre ces composantes. Cette structure de graphe est souvent enrichie d'attributs, portés aussi bien par les sommets que par les arêtes afin de caractériser les sous-parties de l'objet ainsi que les relations existants entre ces sous-parties. La structure de données d'un graphe permet ainsi de véhiculer simultanément des informations sur les sous-parties qui composent un objet (nature, caractéristiques) que sur leurs agencements topologiques. De par leur pouvoir de représentation, les graphes sont utilisés, en tant qu'outils de modélisation, dans de nombreux domaines : chimie, biologie, réseaux, analyse d'images, recherche d'informations...

Cette section vise à exposer un échantillon des méthodes de modélisation des des images et de formes par un graphe. Leur pouvoir de représentation fait que les graphes sont utilisés

dans beaucoup d'applications. Ainsi, les techniques pour extraire les informations varient en fonction de la nature des données mais également selon le contexte et les objectifs du domaine applicatif. Nous avons sélectionné des travaux issus de différentes thématiques de recherche où les graphes sont utilisés comme outils de représentation dans un système de reconnaissance des formes ; nous les présentons dans les sous-sections suivantes.

1.3.1 Vision articielle et images

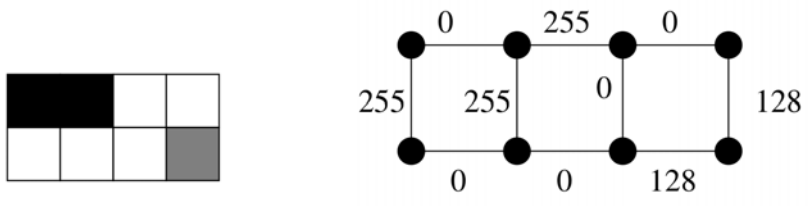
De nombreux travaux sont portés vers la réalisation des graphes à partir d'images numériques ([Pavlidis, 1972]). La grande flexibilité des graphes et leur pouvoir descriptif plus complet que les représentations statistiques favorisent la multiplication des méthodes de caractérisation par les graphes, malgré les contraintes de complexité. Afin de vous donner un aperçu des différentes approches, nous mentionnons ci-dessous quelques contributions représentatives de l'état de l'art sur la techniques de caractérisation des images par les graphes, que l'on peut subdiviser en quatre catégories :

- utilisation des pixels.
- utilisation de points particuliers.
- utilisation de primitives de bas niveau.
- utilisation de régions.

Dans sa thèse [Jouili, 2011], Jouili propose une comparaison de ces différentes représentations et met en évidence l'impact de leurs qualités en tant que descripteurs dans un système de reconnaissance des formes.

Graphes de pixels

La représentation la plus intuitive est certainement le graphe de pixel. Les sommets représentent les pixels d'une image (1.10(a)) et les arêtes caractérisent la connexité entre les pixels (il existe une arête entre deux sommets si les pixels sont voisins) et sont étiquetées avec la valeur absolue de la différence de niveaux de gris (1.10(b)).



(a) Une image. (b) Graphe de pixel étiqueté.

FIGURE 1.10 – Une image et sa représentation par un graphe de pixel[3]. Les sommets représentent les pixels. La connexité entre deux pixels est modélisée par la présence d'une arête étiquetée par la valeur absolue de la différence des niveaux de gris entre les pixels.

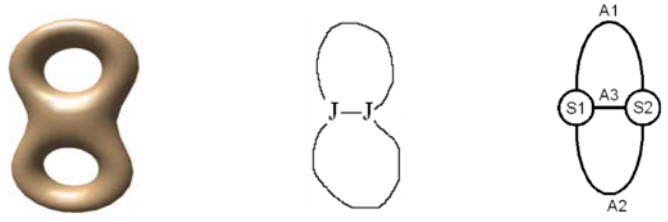
Il est également possible de représenter une image par un graphe dont les sommets sont étiquetés par la valeur du pixel, par exemple le niveau de gris, et la connexité des pixels

3. Illustration extraite de [Tupin, 2006].

est représentée par la présence d'une arête non-étiquetée. L'ordre du graphe est égal à la taille de l'image et la taille du graphe est bornée. Dans [Franco *et al.*, 2003], les auteurs utilisent les graphes de pixels pour comparer des caractères ou des symboles. Pour réduire leur taille, seule la composante de forme est représentée. Néanmoins, de manière générale, la complexité des algorithmes étant un frein à leur utilisation en reconnaissance des formes, les graphes de pixels sont difficilement utilisables sur des images de trop grande résolution. En analyse d'images, ce type de graphes est souvent utilisé pour simplifier la segmentation en calculant l'arbre couvrant de poids minimal pour en déduire les arêtes à supprimer ([Morris *et al.*, 1986]).

Graphes de points particuliers

Certaines méthodes basent leurs extractions sur la recherche de points particuliers pour construire les graphes. Dans [Bunke, 1982], l'auteur utilise les points de jonction des segments dans la description de schémas électriques. Les sommets sont alors étiquetés selon la nature de cette jonction (point terminal, jonction en X, en T, en L...). Cette méthode nécessite néanmoins des prétraitements de binarisation et de squelettisation pour extraire ces points. La figure 1.11 illustre un exemple d'une représentation d'une image 3D (1.11(a)) par un graphe de points particuliers (1.11(c)) construit après une squeletisation (1.11(b)) selon [Brunner et Brunnett, 2004].



(a) Une image 3D.

(b) Squelette avec 2 points particuliers (Jonctions).

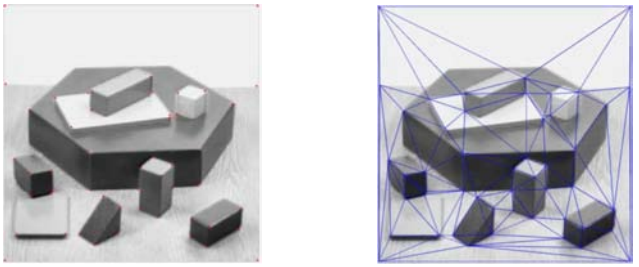
(c) Graphe de points particuliers.

FIGURE 1.11 – Une image 3D et sa représentation par un graphe de points particuliers[4]. L'image est squelettisée et les points de jonctions sont ensuite identifiés. Ces points sont les sommets S1 et S2 du graphe et les arêtes A1, A2 et A3 représentent les traits du squelette entre les jonctions.

D'autres méthodes utilisent la notion de point d'intérêt introduite par Moravec [Moravec, 1977]. Ces techniques permettent de localiser les points où le signal de l'image est riche en information. La figure 1.12 illustre la méthode décrite dans [Wilson *et al.*, 2005] pour comparer des images. Les auteurs isolent les points d'intérêt (1.12(a)) en utilisant le détecteur de Harris [Harris et Stephens, 1988]. Ensuite, les arêtes sont ajoutées (1.12(b)) en utilisant la méthode de la triangulation de Delaunay ([Delaunay, 1934]). Dans [Sanromà *et al.*, 2010], les auteurs proposent d'exploiter les points clefs du descripteur SIFT ([Lowe, 2004]) comme

4. Illustration extraite de [Qureshi, 2008].

les sommets du graphe et de les étiqueter avec les caractéristiques associées. Les arêtes sont définies selon la distance entre les points clefs.



(a) Une image.

(b) Graphe de ppoints particuliers.

FIGURE 1.12 – Une image et sa représentation par un graphe de points d'intérêts [5]. Le détecteur de points d'intérêts de Harris (marqués en rouge) est appliqué sur l'image. Le graphe non-étiqueté est issu de la triangulation de Delaunay.

Graphes de primitives

Les graphes de primitives, ou de traits, se focalisent non plus sur des points caractéristiques, mais sur les primitives de l'objet à décrire, le plus souvent le trait. Par exemple, dans [Ramel *et al.*, 2000], les auteurs effectuent, dans un premier temps, une vectorisation des contours afin d'obtenir un ensemble de quadrilatères représentatifs des formes fines présentes dans l'image. La figure 1.13 fournit l'exemple d'une image de symbole (1.13(a)) et sa représentation par un graphe de primitives (1.13(b)) . Les quadrilatères et leurs relations de voisinage sont utilisés pour construire un graphe étiqueté. Dans cette représentation structurelle, les quadrilatères constituent les sommets du graphe tandis que les arêtes décrivent la nature de la relation topologique.

(a) Une image de symbole.

(b) Graphe de primitives.

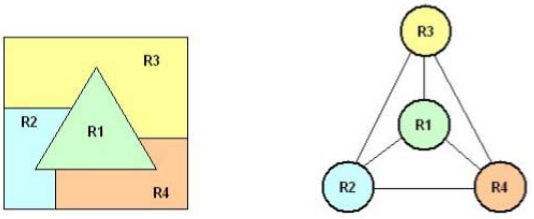
FIGURE 1.13 – Une image de symbole et de sa représentation par un graphe de primitives [6]. L'image de symbole est décomposée en primitives qui sont identifiées par les sommets du graphe, étiquetés par la couleur de la primitive. La relation entre deux primitives proches est représentée par une arête étiquetée par la nature de la relation : P pour parallèle, T pour une jonction en "T".

5. Illustration extraite de [Stoica, 2011].
6. Illustration extraite de [Ramel, 2009].

Ces graphes peuvent aussi être utilisés dans l'identification et la reconnaissance de caractère. Par exemple, dans [Liu *et al.*, 2004] les auteurs choisissent de représenter les caractères chinois par des graphes dans lesquels les sommets caractérisent les traits et les arêtes les relations topologiques.

Graphes d'adjacence de régions

Le graphe d'adjacence de régions, introduit par Rosenfeld [Rosenfeld, 1974] est un graphe planaire qui représente les régions d'une image préalablement segmentée par des sommets. Les arêtes sont définies par les relations d'adjacence existant entre ces régions. Les régions peuvent être le résultat d'une phase de segmentation [Suk et Oh, 1986]. Nous ne présenterons pas en détail ces algorithmes de segmentation dans ce manuscrit mais nous renvoyons le lecteur vers la consultation de la littérature, riche en ce domaine. Les graphes d'adjacence des régions sont utilisés pour différents types d'images (e.g. scènes, 3D, graphiques). Néanmoins, dans certains cas, la segmentation préalable d'images en régions n'est pas toujours une solution efficace pour définir un graphe avec une représentation de bonne qualité. Certains travaux ont contourné cette phase de segmentation par d'autres techniques mieux adaptées au domaine d'application. Lladòs *et al.* [Lladós *et al.*, 2001] proposent une formulation particulièrement intéressante au regard de la problématique de détection de symboles. Ceux-ci ne sont plus modélisés par un graphe d'adjacence de régions obtenues par segmentation mais par une vectorisation de l'image. Les régions sont alors définies par les contours, chaînes cycliques de vecteurs successifs. La figure 1.14 illustre une image (1.14(a)) et son graphe d'adjacence de régions (1.14(b)).



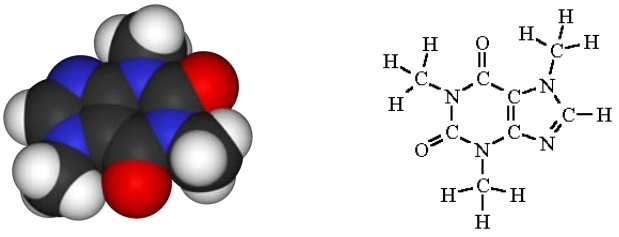
(a) Une image.

(b) Graphe d'adjacence de régions.

FIGURE 1.14 – Une image et sa représentation par un graphe d'adjacence de régions. L'image est préalablement segmentée en régions. Chaque sommet étiqueté du graphe représente une région et sa couleur. Une arête est présente entre deux sommets si les deux régions représentées sont adjacentes.

1.3.2 Chimie organique et biologie

Depuis longtemps, les graphes sont utilisés en chimie organique pour représenter les molécules [Fischer et Meinl, 2004]. Il est usuel de caractériser les atomes par des sommets étiquetés et les valences par les arêtes. La figure 1.15 illustre la molécule de la caféine par une représentation 3D (1.15(a)) et par une représentation en graphe (1.15(b)). Les graphes, qui permettent une modélisation plus complète que la formule brute ($C_8H_{10}N_4O_2$), sont ensuite utilisés pour rechercher des motifs, appelés fragments, qui apparaissent dans les molécules grâce à des techniques dites de recherche d'isomorphisme de sous-graphes. Une autre application est de différencier les molécules actives et inactives en se focalisant sur la présence de fragments fréquents ou non ([Borgelt, 2002]). En biologie, les problématiques sont élevées à l'échelle des macromolécules (protéines, acides nucléiques...) : les sommets représentent alors un acide aminé et les arêtes correspondent aux peptides. Une application, présentée dans [Cook et Holder, 2000], est de classifier les protéines selon leurs structurations. Dans [Dooms *et al.*, 2005], les auteurs s'intéressent au problème de la découverte d'interactions dans des réseaux biochimiques. Les réseaux sont modélisés par des graphes dont chaque sommet représente un composant d'une cellule et à chaque arc correspond une interaction entre ces composants lors d'une réaction chimique. Les graphes sont comparés afin de trouver de nouvelles interactions.



(a) Représentation 3D de la molécule de caféine.

(b) Graphe de la molécule.

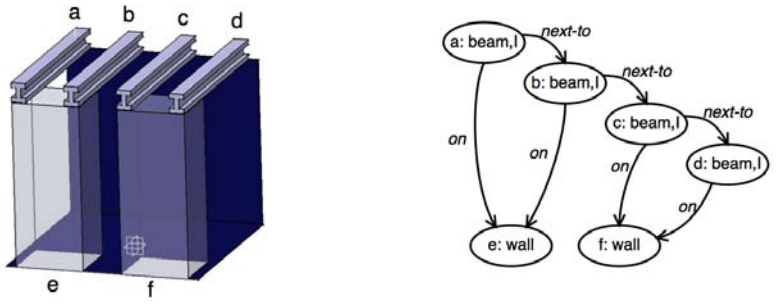
FIGURE 1.15 – Une molécule ($C_8H_{10}N_4O_2$) symbolisée sous forme 3D [7]et sa représentation par un graphe. Les sommets du graphes sont les atomes, étiquetés par leur symbole (C, H, N ou O). Une arête non-étiquetée et non-orientée représente une liaison covalente ; il peut y avoir plusieurs liaisons et dans ce cas de multiples arêtes entre deux sommets.

1.3.3 Conception assistée par ordinateur

Dans [Champin et Solnon, 2003], les auteurs proposent une application de reconnaissance d'objets de conception assistée par ordinateur à partir d'une modélisation par des graphes. L'objectif voulu est de comparer, de façon automatique, un objet à tous les objets déjà conçus. Les objets de conception sont décrits en terme de composants (murs, boulons, poutres, ...) et de relations entre ces composants (une poutre est sur un mur, une poutre est à coté d'une autre poutre, ...). Leur modélisation sous forme de graphes est donc triviale en représentant les composants par les sommets et les arêtes décrivent la relation

7. Illustrations extraites de Wikipedia : `http://fr.wikipedia.org/wiki/Caféine`.

entre les composants. La figure 1.16 fournit un objet de conception (1.16(a)) et le graphe qui le représente selon les modalités décrites (1.16(b)).



(a) Objet de conception assistée par ordinateur.

(b) Graphe de l'objet.

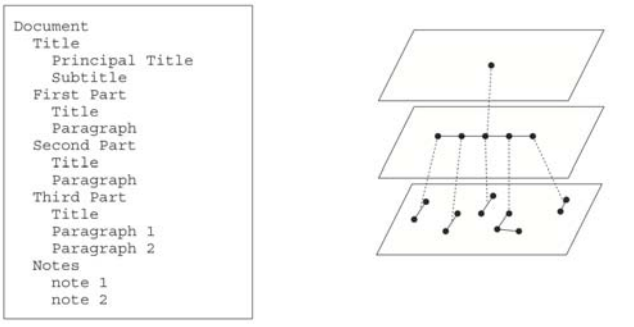
FIGURE 1.16 – Un objet de conception et sa représentation par un graphe [8]. Un composant de l'objet et symbolisé par un nœud étiqueté (*wall*, *beam*). Un arc étiqueté définit la relation entre deux composants (*next-to*, *on*).

1.3.4 Documents

Structures de document

Dans [Héroux *et al.*, 2000], les auteurs proposent un processus de rétroconversion de documents basée sur la classification des structures de document. Ils s'appuient sur deux types de structures représentées par des graphes : la structure physique et la structure logique. La structure physique décrit la mise en page d'un document. Sa description par un graphe peut alors être rapprochée des représentations d'images par régions. L'image de document est préalablement segmentée et les régions (bloc de texte, graphique, note de marge...) sont décrites par les noeuds et les relations entre ces régions par les arêtes. La structure logique du document s'appuie sur une description du rôle et de la nature de chaque élément et sur l'ensemble des liens hiérarchiques et/ou logiques entre eux. La structure logique est donc souvent représenté par un arbre dont les sommets caractérisent les éléments (titre, section, texte, note...) et d'arêtes représentant les liens hiérarchiques. De nombreux formats de fichiers, tels que XML, SGML ou LaTeX, sont orientés vers une spécification de la structure logique des documents et permettent de s'affranchir de la segmentation puis de l'analyse de l'image d'un document. La figure 1.17 illustre un document (1.17(a)) et un graphe représentant sa structure logique (1.17(b)).

8. Illustration extraite de [Champin et Solnon, 2003].



(a) Document.

(b) Graphe de la structure logique du document.

FIGURE 1.17 – Un document et une représentation de sa structure logique par un graphe [9]. Le document est organisé selon la structure logique décrite. Le graphe est un arbre dont la racine est *Document*, ses fils représentent les sections *Title*, *First Part*, *Second Part*, *Third Part* et *Notes*. Des feuilles leurs sont ajoutées selon la décomposition des sections.

Analyse textuelle

Dans [Schenker *et al.*, 2005], plusieurs méthodes de description de contenus de document Web sont proposées. À partir de l'analyse des mots présents dans le document, les auteurs construisent un graphe où les nœuds représentent un mot et sa fréquence d'apparition. Afin de réduire l'ordre du graphe, les mots dits d'arrêts, peu importants à la compréhension d'un texte, sont supprimés au préalable. Les arcs décrivent les relations de successions entre ces mots (si elles existent) et sont étiquetés selon leur type : TX signifie une succession entre deux mots dans un bloc de texte, L indique la présence d'un lien hypertexte et TI notifie une relation présente dans un titre. La figure 1.18 illustre un tel graphe.

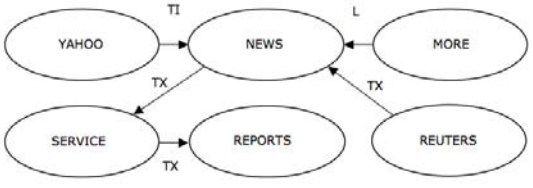


FIGURE 1.18 – Graphe d'une analyse textuelle d'un document Web [10]. Tous les mots présents dans le document sont représentés par les nœuds du graphe et un arc définit la relation de succession entre deux mots et est étiqueté selon la localisation de cette relation : dans un bloc de texte (TX), un lien hypertexte (L) ou un titre (T).

9. Illustration extraite de [Héroux *et al.*, 2000].
10. Illustration extraite de [Schenker *et al.*, 2005].

1.4 Conclusion

Dans ce chapitre, nous avons tout d'abord introduit les principaux aspects de la reconnaissance des formes, notamment les méthodes dites structurelles à base de graphes. Nous avons aussi présenté les notations et concepts que nous utiliserons dans la suite de cette thèse. Une sélection de aspects sur les représentations sous forme de graphes a également été présentée. Tout ce travail nous permet de mettre en évidence quelques points importants dont nous faisons le bilan.

Le pouvoir de description des graphes fait de ces derniers un outil de modélisation utilisé dans beaucoup de travaux. La théorie des graphes, tout d'abord développée pour la modélisation d'un problème ou d'un système unique en recherche opérationnelle, connaît depuis quelques décennies un intérêt dans d'autres thématiques. La possibilité d'une description analytique, *i.e.* de représenter une forme en sous-parties et les relations entre ces sous-parties au sein d'une même structure, a permis aux graphes de connaître un véritable essor en proposant une alternative aux problèmes rencontrés avec l'utilisation de méthodes statistiques (recherche d'une sous-partie d'une forme, aide à la segmentation...).

En reconnaissance des formes, l'utilisation des graphes est venue enrichir le domaine en proposant une modélisation structurelle que ne permettent pas les vecteurs de caractéristiques. Des travaux se sont portés sur des applications multiples où les structures sont explicites comme les réseaux, qu'ils soient physiques (par exemple, Internet ([Dickinson *et al.*, 2004])), conceptuels ([Montes-y Gomez *et al.*, 2000]) ou sémantique ([Ehrig *et al.*, 1992]). En donnant un sens plus large au terme *réseau*, nous pouvons également inclure la classification d'empreintes digitales ([Maio et Maltoni, 1996]), la reconnaissance de symboles techniques ([Cordella *et al.*, 2000]) ou de caractères ([Rocha et Pavlidis, 1994]) ou l'analyse de silhouette ([Shokoufandeh et Dickinson, 2001]) en plus de ceux que l'ont a évoqués au cours de ce chapitre.

Le verrou scientifique de l'utilisation des graphes dans ce contexte est de déterminer si deux graphes représentent une même forme ou non. Le processus pour évaluer une similarité fait alors référence aux méthodes de comparaison de graphes qui doivent prendre en considération les bruits structurels pouvant intervenir suite aux fluctuations des formes ou aux imprécisions des algorithmes de modélisation. De nombreux travaux sont dédiés à cette problématique et nous en présentons un état de l'art dans le chapitre suivant.

Chapitre 2

Etat de l'art sur les méthodes de comparaison de graphes

> Je n'ai pas échoué. J'ai simplement trouvé 10 000 solutions qui ne fonctionnent pas.
>
> Thomas Edison

Les représentations des formes par un graphe apportent deux avantages par rapport aux vecteurs : l'adaptation aux données due à la flexibilité des graphes et le pouvoir de description analytique d'un objet. Nous avons vu que les méthodes statistiques de reconnaissance des formes sont basées sur la mesure de similarité entre les objets. Si la comparaison de deux vecteurs est facilitée par un cadre formel et un panel d'outils statistiques disponibles, la comparaison de deux objets représentés par des graphes est quant à elle plus complexe. En utilisant de telles représentations, le verrou est de déterminer si deux graphes représentent chacun un objet de même type ou de deux types différents ; plus formellement, si deux graphes décrivent des objets appartenant à la même classe.

Entre les premières méthodes de comparaison de graphes en reconnaissance des formes ([Winston, 1970, Pavlidis, 1972, Fischler et Elschlager, 1973]) qui utilisent des algorithmes non tolérants aux variations et les travaux plus récents tels que les machines à noyaux [Bunke et Riesen, 2007] ou la programmation par contraintes [Solnon, 2010], la littérature sur les méthodes de comparaisons de graphes devient de plus en plus abondante. Une difficulté supplémentaire concernant l'étude des méthodes de comparaison de graphes provient de la diversité et de la multiplicité des domaines et donc des communautés produisant des contributions très intéressantes à cette problématique (recherche opérationnelle, recherche d'information, fouille de données...). Nous proposons dans ce chapitre une synthèse des techniques les plus représentatives en privilégiant celles adaptées aux domaines de l'analyse d'images et de la reconnaissance des formes. Nous conseillons également la lecture de deux états de l'art, [Conte *et al.*, 2004] et [Bunke et Riesen, 2011], pour une revue très complète de ces travaux.

Contrairement aux vecteurs, déterminer une similarité entre deux graphes n'est donc

pas facile. Ce processus d'évaluation de similarité fait premièrement référence à l'appariement de graphes. On désigne par **appariement** la mise en correspondance des nœuds et/ou des arcs des graphes. Si un appariement est possible entre deux graphes, alors ces graphes sont considérés comme similaires, sinon ils sont différents. Les méthodes d'appariement entre graphes sont alors divisées en plusieurs catégories. La première, dite appariement exact, impose des contraintes dures et n'autorise qu'un appariement au sens strict du terme. Ces méthodes sont introduites dans 2.1. Par opposition, la deuxième catégorie, dite appariement inexacte, induit des contraintes plus souples pour permettre un appariement avec une tolérance d'erreur et il est ainsi possible de comparer des données que l'on sait bruitées. La section 2.2 présente ces techniques. Suivant les domaines d'application, il est intéressant de pouvoir quantifier une similarité. Dans ce cas, les algorithmes d'appariements ne sont plus pertinents dans le sens où ils déterminent si deux graphes sont similaires ou non (réponse booléenne). Certains travaux proposent plutôt de définir des distances entre graphes afin de mesurer leur similarité. Ces distances sont introduites dans la section 2.3. L'inconvénient majeur de toutes ces techniques est la complexité associée aux algorithmes. Afin d'améliorer les temps de calcul, certains travaux proposent de projeter les graphes dans un espace vectoriel. La comparaison de graphe est ainsi ramenée à une comparaison de vecteurs. Cces méthodes permettent d'utiliser des outils statistiques connus et plus rapides. Nous les introduirons en 2.4. Pour finir, nous nous intéresserons aux méthodes utilisées en classification de graphes dans la section 2.5

2.1 Appariements exacts

Afin de déterminer si deux graphes sont strictement identiques, il faut prouver qu'il existe une mise en correspondance de tous leurs nœuds, de tous leurs arcs ainsi que de leurs fonctions d'étiquetage. Contrairement aux éléments d'un vecteur, il est n'est pas possible d'ordonner un graphe, ce qui rend l'égalité entre deux graphes complexe à établir puisqu'il faut au préalable déterminer l'appariement entre leurs nœuds en tentant compte des arcs et des fonctions d'étiquetages. Cette égalité entre deux graphes est donc communément définie par une fonction bijective appelée **isomorphisme de graphe** [10].

Définition 5 *Soient $G = (V, E, \mu, \nu)$ et $G' = (V', E', \mu', \nu')$ deux graphes. Un isomorphisme de graphes est une fonction bijective $f : V \to V'$ telle que*
- *Pour tout nœud $v \in V$ il existe un nœud $v' = f(v) \in V'$ tel que $\mu(v) = \mu'(v')$*
- *Pour tout arc $e = (v_i, v_j) \in E$, il existe un arc $e' = (f(v_i), f(v_j)) \in E'$ tel que $\nu(e) = \nu'(e')$*
- *Pour tout arc $e' = (v'_i, v'_j) \in E'$, il existe un arc $e = (f^{-1}(v'_i), f^{-1}(v'_j))$ tel que $\nu(e) = \nu'(e')$*

*Si un isomorphisme existe entre deux graphes, alors ils sont dits **isomorphes**.*

Nous définissons donc deux graphes isomorphes comme deux graphes identiques. Pour déterminer un isomorphisme de graphe, une mise en correspondance entre les nœuds du

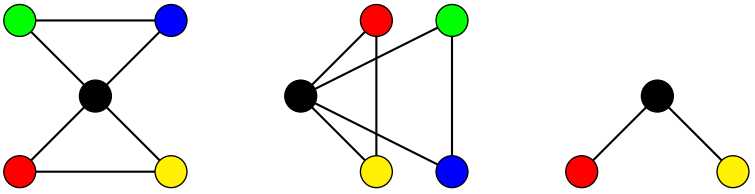
10. *Stricto sensu*, l'isomorphisme de graphe est uniquement défini sur la topologie d'un graphe. En reconnaissance des formes, il est usuel d'élargir cette définition aux étiquettes.

premier graphe et du second devra être déterminée en respectant la structure des arcs et en conservant l'étiquetage des nœuds et des arcs. La relation d'isomorphisme de graphe satisfait les conditions de réflexivité, de symétrie et de transitivité. Elle est à ce titre considérée comme la relation d'équivalence du domaine des graphes [Matousek et Nesetril, 2004].

Nous définissons maintenant le concept d'**isomorphisme de sous-graphe**, très proche de l'isomorphisme de graphe. Si l'isomorphisme de graphe est considéré comme la notion formelle d'égalité de graphes, alors l'isomorphisme de sous-graphe peut être vu comme l'égalité de sous-graphes.

Définition 6 *Soient $G = (V, E, \mu, \nu)$ et $G' = (V', E', \mu', \nu')$ deux graphes. Un **isomorphisme de sous-graphe** est une fonction injective $f : V \to V'$ avec $|V_1| \leq |V_2|$ s'il existe un graphe $G'' \subseteq G'$ tel que f soit un isomorphisme entre G'' et G.*

Un isomorphisme de sous-graphe existe entre deux graphes G et G' si le graphe de plus grande taille G' peut être transformé en un graphe G'' isomorphe de G en retirant des nœuds et des arcs. En d'autres mots, un isomorphisme de sous-graphe indique qu'un petit graphe est contenu dans un plus grand graphe. La figure 2.1 présente un graphe G, un graphe isomorphe de G et un sous-graphe isomorphe de G.



(a) un graphe G (b) un graphe G' (c) un graphe G''

FIGURE 2.1 – Soient 3 graphes G, G', G'' : G' est un isomorphe de G et G'' est un sous-graphe isomorphe de G et G'. Les appariements sont dénotés par les couleurs des sommets.

De manière naïve, l'algorithme présenté dans [Reingold *et al.*, 1977] permet de déterminer des isomorphismes exacts entre deux graphes en procédant de manière combinatoire. Il consiste à rechercher toutes les fonctions f valides, et à tester pour chacune de ces correspondances s'il y a isomorphisme ou non. Cette recherche d'isomorphisme entre deux graphes nécessite de générer $n!$ fonctions dans le pire des cas (avec n la taille des graphes). Bien que la complexité de ce problème soit considérée dans la classe NP ([Garey et Johnson, 1979]), il n'existe aucun algorithme capable de le résoudre en temps polynomial et son appartenance à la classe NP-complet n'est pas encore démontrée ([Köbler *et al.*, 1993]), contrairement à celle de la recherche d'isomorphisme de sous-graphe ([Read et Corneil, 1977]). Les travaux se sont donc concentrés sur la réduction de cette complexité. Nous présentons dans cette sous-section quelques travaux phares, organisée selon les différents approches qu'ils exploitent.

2.1.1 Recherche arborescente

La majorité des méthodes de recherche d'isomorphisme se basent sur la notion d'arbre de recherche et de retour-arrière. Ce principe, utilisé dans différents domaines, autorise

une remise en cause d'une décision d'appariement afin de sortir d'un blocage. Un retour en arrière est alors effectué dans l'espace de résolution du problème. Nous présentons ici le fonctionnement général de tels algorithmes tel qu'il est défini dans [Berztiss, 1973] :
Soient $G_r = (V_r, E_r)$, un graphe requête et $G_c = (V_c, E_c)$ un graphe cible . Les sommets de G_r sont appariés avec les sommets de G_c progressivement, un à un. À chaque mise en correspondance, la topologie de l'ensemble est vérifiée, *i.e.* les arêtes existantes dans G_c sont présentes dans le sous-graphe formé par l'appariement des sommets. Si cette étape est validée, alors l'algorithme continue en appariant un nouveau sommet ; sinon, la dernière mise en correspondance est annulée et un nouvel appariement est recherché. Ceci constitue la première phase de retour en arrière proposée par l'algorithme. Un deuxième retour peut aussi être exécuté lorsqu'un appariement est validé mais conduit à l'incompatibilité des mises en correspondance lors des étapes suivantes. L'appariement est alors annulé. Une tentative sur de nouveaux candidats est alors effectuée. L'algorithme peut remonter jusqu'au premier appariement afin de tester toutes les combinaisons possibles. Cette recherche d'appariement est alors exhaustive. Lorsque l'ensemble des sommets de G_r sont appariés, alors un isomorphisme existe entre les deux graphes en entrée, trois solutions sont alors possibles :

- $|G_r| = |G_c|$ alors un isomorphisme de graphes existe entre G_r et G_c
- $|G_r| < |G_c|$ alors G_r est isomorphe d'un sous-graphe de G_c
- $|G_r| > |G_c|$ alors G_c est un isomorphe d'un sous-graphe de G_r

L'inconvénient majeur de cet algorithme est que de nombreuses configurations ne donnant pas d'isomorphisme sont testées vainement. De manière à résoudre ce problème, cette catégorie d'algorithmes utilise des heuristiques pour élaguer l'arbre de recherche en détectant les branches non fructueuses. Nous détaillerons deux approches qui sont aujourd'hui des références pour ces méthodes.

Ullman [Ullmann, 1976] propose un algorithme reconnu pour être un des plus efficaces en terme de réduction de la taille de l'arbre de recherche. L'idée de base consiste à ne pas tester certains appariements dont on sait qu'ils ne fourniront pas d'isomorphisme. Dans un premier temps, le problème est reformulé en utilisant les matrices d'adjacence. La méthode est construite autour d'une matrice M synthétisant les appariements possibles entre les sommets de deux graphes G_1 et G_2. La matrice de taille $|V_1| \times |V_2|$ est définie telle que :

$$\begin{cases} 1 & \text{si } d_{G_1}(v_i) \geq d_{G_2}(v_j) \\ 0 & \text{sinon} \end{cases}$$

Ainsi, un 0 présent dans la matrice révèlera une information sûre sur l'impossibilité d'un isomorphisme entre deux sommets d'un graphe, tandis qu'un 1 mettra en évidence la possibilité d'un appariement. L'innovation de cet algorithme est une deuxième matrice des associations futures possibles pour une liste d'associations donnée par la première matrice. On parlera alors de "Forward checking". La liste des associations est ensuite mise à jour par les 2 procédures.

Cordella *et al.* proposent successivement les méthodes VF [Cordella *et al.*, 1999] et VF2 [Cordella *et al.*, 2001]. Ces algorithmes permettent de travailler sur des graphes de grande taille dans la mesure où ses besoins en mémoire sont moindres que la plupart des autres algorithmes déterministes grâce à une fonction dite de faisabilité qui optimise l'arbre de recherche en se basant sur l'analyse des nœuds adjacents à ceux déjà appariés. Cette heu-

ristique est facile à calculer dans la plupart des cas et améliore de manière significative les performances de l'algorithme d'Ullmann et de ses dérivés [Foggia *et al.*, 2001] permettant d'être efficace sur des graphes de grandes tailles.

D'autres heuristiques d'améliorations ont été proposées comme [Levi, 1974] basées sur le partitionnement des arcs, [Rota Bulò *et al.*, 2009] issu d'idées de la théorie des jeux. Un des travaux les plus récents de cette catégorie est proposé dans [Larrosa et Valiente, 2002]. Les auteurs reformulent l'isomorphisme de graphes comme un problème de satisfaction de contraintes, un problème largement étudié dans le domaine de la recherche opérationnelle. Ainsi, les auteurs appliquent les heuristiques issues de cette littérature ([Apt, 2003, Dechter, 2003]) pour déterminer rapidement un appariement de graphe.

2.1.2 Entités canoniques

Un des premiers algorithmes basé sur des entités canoniques est développé par Corneil et Gotlieb [Corneil et Gotlieb, 1970]. La détection d'isomorphisme de graphes ou de sous-graphes, n'est alors non plus basée sur une méthode de recherche exhaustive de tous les appariements sommets à sommets possibles, mais sur la transformation des graphes en de nouvelles entités canoniques. Cette représentation est basée sur une conjecture pour laquelle le calcul d'isomorphisme est plus facile. Mais, il a été démontré dans [Mathon, 1978] que la conjecture qui en est à la base n'est pas toujours vraie. Néanmoins, il utilise des méthodes intéressantes de partitionnement et de raffinement, similaire à [Weisfeiler, 1976], pour établir les bases de la plupart des algorithmes utilisant les entités canoniques. Le principe de ces algorithmes n'est pas de comparer directement un graphe à un autre, mais de travailler séparément sur un graphe d'abord puis sur un autre, pour générer deux entités canoniques qui pourront être comparables. Des détails et explications sur l'implémentation de cette classe d'algorithmes sont donnés dans [Kreher et Stinson, 1999].

Dans [Berztiss, 1973], Berztiss propose une méthode basée sur la notion de k-formule et du retour-arrière. Cette méthode consiste à représenter chaque graphe, particulièrement les graphes orientés, par une chaîne qui contient toutes les informations pertinentes associées au graphe.

Actuellement, la méthode utilisant le concept d'entité canonique la plus performante, et donc la plus populaire est *nauty* [McKay, 1981, McKay, 2004]. L'algorithme utilisé par *nauty* (et par la plupart des méthodes similaires) est un algorithme à retour-arrière qui parcourt le graphe pour rechercher les entités canoniques. Les nœuds de chaque graphe sont ordonnés en se basant sur un étiquetage canonique. En fait, pour chaque nœud v_i d'un graphe l'algorithme calcule une étiquette unique en se basant sur un ensemble de caractéristiques décrivant les relations entre v_i et les autres nœuds du graphe. Ensuite, les étiquettes sont utilisées pour l'ordonnancement des nœuds de chaque graphe. Enfin, l'isomorphisme entre deux graphes est calculé en vérifiant l'égalité de leurs représentations canoniques (*i.e.* les étiquettes canoniques ordonnées). Cet étiquetage canonique permet à l'algorithme *nauty* d'être encore à ce jour un des plus performant de sa catégorie et reste une référence pour la recherche d'isomorphisme avec VF2 suivant la topologie des graphes à analyser ([Foggia *et al.*, 2001]).

Il existe d'autres nombreuses méthodes que nous ne décrirons pas dans ce mémoire.

Nous en indiquons cependant quelques références sur les techniques de représentations canoniques ([Babai et Luks, 1983], [Cook et Holder, 2006]).

2.1.3 Algorithmes avec contraintes sur les graphes

Pour finir cette première section, nous pouvons noter qu'il existe des algorithmes d'une complexité plus faible qui exploitent les restrictions que peuvent comporter certaines classes de graphes. Les exemples classiques sont, premièrement, les graphes planaires (un graphe qui a la particularité de pouvoir se représenter sur un plan sans qu'aucun arc n'en croise un autre) pour lesquels un algorithme linéaire existe [Hopcroft et Wong, 1974] et pour les arbres [Aho *et al.*, 1974] (un arbre est un graphe non-orienté qui ne possède aucune une suite d'arêtes consécutives dont les deux sommets extrémités sont identiques, et dont la taille égale l'ordre-1). Pour la classe plus large des graphes pour lesquels le degré est borné, un algorithme polynomial existe [Luks, 1980]. Enfin, Babai *et al.* proposent deux algorithmes polynomiaux pour des graphes dont la multiplicité des valeurs propres est bornée [Babai *et al.*, 1982].

2.1.4 Conclusion

Les algorithmes décrits ci-dessus sont considérés suffisamment performants et la recherche d'appariement exact n'est plus une thématique de recherche forte. De plus, en reconnaissance des formes, les représentations issues de données réelles présentent systématiquement des différences par rapport au modèle parfait. Ces imperfections rendent l'utilisation d'algorithmes de recherche d'isomorphisme difficile. En effet, les contraintes sont trop fortes, puisqu'il faut que les deux graphes soit strictement identiques. Le moindre bruit (suppressions, ajouts ou modifications d'attributs sur les nœuds ou les arcs) induit une réponse négative des algorithmes. Les techniques d'appariements exacts sont lourdes en temps de calcul et ne sont donc pas forcément très pertinentes en reconnaissance des formes. Pour contourner ce problème, la stratégie, définie comme appariement inexact, vise à rendre l'étape de comparaison de graphes plus tolérante aux différences entre les graphes.

2.2 Appariements inexacts

Les principaux avantages des méthodes d'appariements exacts de graphes sont leurs définitions rigoureuses et leurs fondements mathématiques solides. Hélas, il s'avère que la comparaison de graphes basée sur l'isomorphisme de (sous-)graphes est difficilement applicable en reconnaissance des formes. En pratique, deux graphes sont rarement exactement égaux, même s'ils ont été extraits du même objet. Des variations peuvent être causées par l'incertitude pendant le processus d'extraction de caractéristiques, par des environnements bruités, *etc.* Pour cette raison, un nombre important de techniques d'appariement inexact de graphes ont été proposées. L'idée est d'évaluer la ressemblance de deux graphes au sens large du terme. L'appariement inexact est donc défini comme un appariement qui est capable de tolérer ou de corriger, sous conditions, les différences entre un graphe et le graphe idéal. Cet appariement est aussi dénommé à correction d'erreurs ou tolérant aux erreurs. Si

le problème exact est un problème de décision, le problème inexact est donc un problème d'optimisation combinatoire. Il est donc NP-difficile dans le cas général. Notons bien que ce type de méthodes cherche, comme les précédentes à associer les sommets d'un graphe avec les sommets d'un autre graphe mais en offrant cette fois ci la possibilité d'apparier deux sommets légèrement différents. Ainsi, différentes contraintes fortes présentes dans les méthodes d'appariements exacts de graphes peuvent être relaxées :

- Tolérance sur la structure : des sommets et/ou des arêtes peuvent ne pas trouver d'appariement ; au-dessous d'un seuil, l'appariement entre deux graphes est tout de même accepté.
- Tolérance sur les étiquettes : deux sommets (ou deux arêtes) sont appariés même si leurs étiquettes présentent de faibles différences.
- Appariements multivoques : deux sommets (respectivement deux arêtes) peuvent être appariés à un seul autre sommet (respectivement une autre arête). Nous consacrons une sous-section à cette dernière famille.

2.2.1 Recherche arborescente

On trouve en premier lieu des approches basées sur l'exploration arborescente de l'espace des solutions comme dans le cas d'isomorphisme exact. On effectue une recherche exhaustive de l'espace contenant la totalité des affectations possibles entre les nœuds des graphes [Tsai et Fu, 1979, Sanfeliu et Fu, 1983, Eshera et Fu, 1984]. On commence par une affectation vide. Ensuite on ajoute des paires de nœuds une par une. L'espace des solutions prend la forme d'un arbre dont un niveau k est composé de toutes les affectations partielles contenant exactement k paires de nœuds. Le choix de la paire à ajouter est effectué par un algorithme glouton. Puis une affectation partielle est évaluée par la somme de la distance qu'elle impose sur les graphes, prenant en compte uniquement les nœuds déjà fixés, et une estimation du coût minimal nécessaire pour la compléter. Si l'on regarde une branche quelconque, la fonction d'évaluation est croissante avec le niveau de l'arbre. Comme l'estimation du coût d'extension est une borne inférieure de la distance réelle, chaque sous-arbre dont la racine a une valeur supérieure à la distance maximale acceptable peut être ignoré sans risque d'omission d'une solution valable. L'algorithme $A\star$ [Hart *et al.*, 1968] effectue cette recherche vers l'optimum local. Au cas où l'extension n'est plus possible, une stratégie de retour arrière est appliquée. Quand la recherche arrive dans une feuille, une solution candidate a été trouvée. Ici l'optimum global n'est pas garanti non plus, mais la distance imposée par cette solution candidate est ensuite utilisée comme borne plus stricte et la recherche continue jusqu'à ce que toutes les branches aient été évaluées. Cette méthode a été étendue dans [Tsai et Fu, 1983] pour prendre en compte d'autres opérations d'édition (*i.e.* insertion et suppression) sur les nœuds et les arcs. Dans [Wong *et al.*, 1990], Wong *et al.* ont proposé une amélioration de l'heuristique pour qu'elle prenne en compte les appariements des arcs.

De fait, l'algorithme initial ne tient pas compte du coût de l'extension. Plusieurs approches proposent des heuristiques admissibles pour ajouter une borne inférieure au coût d'extension à la fonction d'évaluation. La méthode de [Berretti *et al.*, 2001] utilise un algorithme de relaxation discrète pour la résolution d'un problème d'isomorphisme de graphes bipartites dont le résultat détermine l'estimation du coût. En partant d'une affectation

partielle de k sommets, ils cherchent à établir une estimation de la distance finale en prenant en compte les coûts d'affectation des k sommets déjà appariés mais aussi des $n-k$ sommets encore non-appariés ainsi que leurs relations. La complexité du calcul de l'extension optimale vers un appariement complet équivaut à celle du problème d'isomorphisme de graphes de $n-k$ sommets. Afin de rendre le calcul polynomial et d'obtenir une borne assez stricte en même temps, Berretti *et al.* abandonnent la contrainte que les extensions élémentaires soient cohérentes entre elles. Pour chaque niveau entre k et n, ils choisissent l'appariement de coût minimum qui est cohérent avec l'appariement d'origine. Toutefois, les appariements utilisés pour l'estimation du coût de l'extension ne sont pas forcément cohérents entre eux. Cette astuce permet d'estimer ce coût en temps polynomial. Basée sur cette valeur, les affectations partielles, dont l'extension vers une solution complète avec une similarité acceptable est impossible, peuvent être ignorées plus tôt. Ceci accélère l'algorithme en conservant l'optimalité du résultat. Cette méthode est performante et compte toujours parmi les algorithmes les plus efficaces disponibles. Par contre, les tailles des graphes que l'on peut traiter en un temps acceptable sont plutôt de l'ordre de la dizaine que du millier (comme dans le problème exact).

2.2.2 Méthodes d'optimisation continues

Les algorithmes d'optimisation continue forment un second grand groupe. Le problème est transformé dans le domaine continu et devient un problème d'optimisation non-linéaire. Les méthodes dans cette catégorie sont en général approchées et elles nécessitent la transformation de la solution dans le domaine d'origine. Il existe des méthodes basées sur la relaxation probabiliste, par exemple celle de Wilson et Hancock [Wilson et Hancock, 1997], ou sur l'appariement pondéré de graphes. Ce dernier est lié au problème du plus grand sous-graphe commun. Une matrice d'appariements avec des poids compris entre 0 et 1 est utilisée et cette matrice est ensuite optimisée dans le domaine continu. Les algorithmes [Almohamad et Duffuaa, 1993] et [Gold et Rangarajan, 1996] suivent cette approche. Une troisième transformation se base sur le théorème de [Motzkin et Straus, 1965] lié à la recherche de cliques ([Pelillo, 1999]). L'appariement de graphes est alors transformé en un problème d'optimisation quadratique. Cesar *et al.* [Cesar jr *et al.*, 2005] présentent une comparaison des différentes méthodes d'optimisation dans le domaine d'application de la reconnaissance des formes. Ils considèrent notamment des approches de recherche d'arbre, des algorithmes génétiques et des algorithmes à estimation de distribution. De manière générale, ces algorithmes tolèrent tout type de graphes et aucune contrainte sur les étiquettes n'est imposée, mais la taille de ceux-ci peuvent rendre complexe le processus d'appariement nécessaire à cette technique.

2.2.3 Méthodes spectrales

Les matrices d'adjacence ou laplaciennes offrent une représentation intéressante au vu des outils mathématiques associés aux matrices. Néanmoins, l'appariement de deux graphes en utilisant les matrices est complexe. En effet, les matrices sont construites selon l'ordre donné par les numéros des nœuds. Or, ces numéros sont fixés aléatoirement. Les lignes et les colonnes peuvent donc être permutés entre deux matrices représentant le même graphe.

Le problème devient alors combinatoire puisqu'il nécessite une étape pour affectation des lignes d'une matrice à une autre. Une des méthodes d'affectation est l'algorithme hongrois qui est de complexité polynomiale. Dans ce cas, l'intérêt d'utiliser les matrices est donc limité.

En revanche, les méthodes basées sur la théorie spectrale des graphes ([Chung, 1997, Godsil et Royle, 2001]) analysent les valeurs et les vecteurs propres des matrices d'adjacence ou laplaciennes. Les valeurs propres sont indépendantes des permutations de sommets. De fait, si deux graphes sont isomorphes, alors leurs spectres sont égaux sans avoir recours à une affectation au préalable. L'inverse par contre n'est pas vrai : des graphes non-isomorphes peuvent avoir le même spectre. On les dénomme alors co-spectraux. Une des premières méthodes est proposée par Umeyama [Umeyama, 1988]. L'auteur utilise la décomposition en valeurs et vecteurs propres des matrices d'adjacence pour en déduire la matrice orthogonale qui est optimisée par la suite. Pour cela il suppose *a priori* que les graphes sont isomorphes. S'ils le sont, alors la méthode trouve la permutation optimale. Si les graphes sont proches de l'isomorphisme, alors la solution est sous-optimale. Toutefois, pour des graphes non-isomorphes la qualité des résultats diminue. Cette idée est utilisée en combinaison avec une approche de partitionnement ([Jain *et al.*, 1999]). D'une part, un partitionnement des sommets avant l'appariement permet d'associer d'abord les clusters et ensuite les sommets [Carcassoni et Hancock, 2001]. D'autre part, [Caelli et Kosinov, 2004] proposent de projeter les sommets dans l'espace propre des graphes et d'utiliser le partitionnement dans cet espace afin de trouver les sommets à mettre en correspondance. Les méthodes spectrales donnent lieu également aux méthodes exploitant les caractéristiques des parcours aléatoires ([Gori *et al.*, 2005]). Le point faible de ces approches est la non prise en compte des étiquettes par les matrices de graphes, les limitant à une certaine catégorie restreinte d'applications.

2.2.4 Appariements multivoques

En analyse d'images, il est fréquent qu'un nœud dans un graphe modèle correspondent à plusieurs nøeuds dans un graphe à traiter. Par exemple, si l'on considère un graphe qui modélise des régions d'une image, on est confronté aux problèmes de sous- et sur-segmentation. Selon l'algorithme (et ses paramètres) de segmentation choisi, on obtient des graphes différents pour la même image. Dans le but d'établir un appariement entre de tels graphes, il s'avère utile de permettre un appariement multivoque, *i.e.* une relation où chaque nœud peut être associé avec plusieurs nœuds de l'autre graphe.

Dans [Boeres *et al.*, 2004] et [Deruyver *et al.*, 2005], les auteurs utilisent une relation non-bijective. Plus précisément, dans leur application plusieurs sommets d'un graphe peuvent être affectés à un seul sommet dans le deuxième graphe. Leur application est l'appariement des images à des modèles prédéfinis. Les images sont (automatiquement) sur-segmentées tandis que les modèles sont créés manuellement.

2.2.5 Conclusion

L'utilisation des algorithmes d'appariements inexacts apportent une réponse à l'intolérance aux bruits des méthodes d'appariements exacts. Excepté les méthodes spectrales,

ces méthodes sont utilisables sur des graphes comportant des étiquettes avec soit des attributs symboliques, soit des attributs numériques ou une combinaison des deux, bien que cela augmente considérablement la difficulté. Cette dernière contrainte reste donc le principal défaut de cet ensemble de méthodes. De plus, elles ont l'avantage de produire en sortie un appariement sommets à sommets des graphes contrairement aux méthodes que nous présentons dans la suite. Grâce à cette propriété, ces techniques peuvent être utilisées pour localiser précisément les différences entre deux graphes permettant ainsi d'identifier une source de bruit par exemple. De plus, la recherche d'isomorphisme de sous-graphe est souvent exploitée dans le cas d'une recherche d'un motif dans une forme (recherche d'un symbole dans un plan, par exemple).

2.3 Distances entre graphes

Les méthodes de comparaison de graphes citées précédemment s'appuient toutes sur des appariements de graphes, exacts ou inexacts. Ces techniques permettent de mettre en évidence que deux graphes sont identiques ou respectent des contraintes sur une tolérance éventuelle. Par conséquent, deux graphes présentant certaines différences, suffisamment pour ne pas être appariés, sont jugés dissimilaires. La similarité entre ces deux graphes sera jugée nulle et la distance qui les sépare devient infinie. Dans de nombreuses applications et notamment en reconnaissance de formes, la comparaison se fait de manière moins booléenne ou plus continue en exploitant des mesures de similarité permettant de graduer cette ressemblance. De ce point de vue, les méthodes d'appariement (exact ou inexact) ne répondent donc pas toujours efficacement aux problématiques de la reconnaissance des formes. Pour palier cette lacune, des méthodes proposent la mise en place de distance entre graphes.

De manière formelle, en mathématiques la distance est une application sur un ensemble $\mathbf{E}$, appelé espace métrique, tel que :

$$d : \mathbf{E} \times \mathbf{E} \rightarrow \mathbb{R}^+$$

De plus, cette application doit vérifier les propriétés suivantes :

symétrie : $\forall x, y \in \mathbf{E}, d(x, y) = d(y, x)$

séparation : $\forall x, y \in \mathbf{E}, d(x, y) = 0 \Leftrightarrow x = y$

inégalité triangulaire : $\forall x, y, z \in \mathbf{E}, d(x, z) \leq d(x, y) + d(y, z)$

Cette définition de distance est directement applicable pour comparer deux formes représentées par des vecteurs. L'espace euclidien dans lequel sont définis les vecteurs des méthodes statistiques autorise, par définition, à utiliser toutes les métriques définies dans cet espace, dont les distances ([Verley, 2002]). Dans les sections qui suivent, nous introduisons deux distances souvent citées comme références : la distance d'édition de graphe et les distances à partir du plus grand sous-graphe commun. Pour compléter cet état de l'art, nous présentons également des distances basées sur des appariements multivoques.

2.3.1 La distance d'édition de graphes

La distance d'édition de graphe est une méthode inspirée par les travaux sur les distances de chaînes de caractères de Levenshtein ([Levenshtein, 1966]) généralisées aux cas des graphes. L'idée directrice de cette distance est de mesurer une dissimilarité en s'appuyant sur le nombre et la force des transformations à appliquer sur un ensemble — une chaîne — pour le transformer en un autre. Le concept de distance d'édition a été étendu des chaînes de caractères à des structures plus complexes comme les arbres dans un premier temps ([Selkow, 1977, Tai, 1979]) puis aux graphes ([Tsai et Fu, 1979, Eshera et Fu, 1984]). L'idée générale d'une distance d'édition de graphe est donc de définir la dissimilarité de deux graphes par la distorsion nécessaire pour transformer un graphe en un autre.

Concrètement, la distance d'édition de graphe se modélise par le chemin d'édition de coût minimum représentant la succession des opérations de transformation en lui associant un coût. Le coût sera faible si les graphes présentent beaucoup de similarités, élevé sinon. Formellement, soient $G = (V', E', \mu', \nu')$ le graphe source et $G' = (V', E', \mu', \nu')$ le graphe cible. La distance d'édition est définie par :

$$d_{GED}(G, G') = \min_{(ed_1,\ldots,ed_k)\in\Upsilon(G,G')} \sum_{i=1}^{k} c(ed_i)$$

avec $\Upsilon(G, G')$ l'ensemble des chemins d'édition transformant G en G' et c la fonction de coût mesurant la distortion $c(ed)$ de l'opération d'édition ed.

La fonction c englobe tous les coûts d'édition possibles. Les distorsions sont généralement l'ajout ou la suppression d'un nœud, l'ajout ou la suppression d'un arc et la modification d'une étiquette (qu'elle soit portée par un nœud ou par un arc). Les points forts et les points faibles de cette distance sont donc concentrés autour de cette fonction. Elle permet de pouvoir adapter les coûts en fonction du problème posé en pénalisant plus ou moins les différentes distorsions à appliquer. Mais par conséquent, ces fonctions de coûts restent des paramètres sensibles et difficiles à régler ([Bunke, 1999]). Des travaux portent toutefois sur l'apprentissage automatique de ces coûts, facilitant ainsi le paramétrage de la fonction ([Neuhaus et Bunke, 2004, Neuhaus et Bunke, 2007]). L'un des points forts de cette technique est sa généricité. Cette distance constitue donc souvent une référence pour évaluer la qualité d'une mesure de similarité entre deux graphes.

Le calcul de la distance d'édition est basé sur un arbre de recherche représentant toutes les solutions d'appariements ; chaque nœud de l'arbre représente un appariement et un chemin — parcours de la racine à une feuille — correspond à un appariement possible, mais pas forcément optimal, entre deux graphes. Cet arbre est construit dynamiquement — à la manière des appariement à l'aide d'arborescence — en optimisant la recherche des problables appariements sommets à sommets avec une heuristique, $A\star$ par exemple ([Hart *et al.*, 1968]). La complexité de l'algorithme est néanmoins exponentielle au nombre de sommets du graphe. La souplesse de la distance d'édition permet donc de l'utiliser sur une grande variété de graphes, sans contraintes sur les étiquettes ou sur la topologie, mais son application est restreinte à de petits graphes. Dans [Neuhaus *et al.*, 2006], les auteurs proposent une solution pour réduire cette complexité au prix d'un appariement sous-optimal.

2.3.2 Les distances à partir du plus grand sous-graphe commun

La recherche du plus grand sous-graphe commun est aussi une méthode couramment utilisée pour le calcul de distance entre graphes. D'autres méthodes basées sur son complémentaire, la recherche du plus petit supergraphe, existent également. Le plus grand sous-graphe commun de deux graphes G et G' est le plus grand graphe qui soit à la fois sous-graphe de G et sous-graphe de G'. Il peut donc être vu comme une intersection de G et de G'. Une approche standard pour extraire le plus grand graphe commun de deux graphes est liée à la notion de la recherche de clique maximum. Des méthodes sur cette approche et leur comparaison sont présentées dans [Bunke *et al.*, 2002]. L'approche de McGregor [McGregor, 1982] est aussi une référence reconnue.

Le plus grand sous-graphe commun permet de mesurer la similarité entre deux objets. Intuitivement, un plus grand sous-graphe commun de G et G' sera de taille importante si les deux graphes présentent beaucoup de similarité, plus faible dans le cas contraire. À partir du plus grand sous-graphe commun déterminé, plusieurs calculs de distance sont envisageables. La distance ainsi déterminée permet de traduire la similarité entre graphes. Dans [Pelillo, 2004], Pelillo formalise l'espace métrique et la définition de distance à partir de plus grand sous-graphe commun. À titre d'exemple, nous vous présentons les deux distances les plus couramment utilisées dans la littérature. Dans [Bunke, 1997], Bunke définit une distance telle que $d = |G| + |G'| - 2.|mcs(G, G')|$ et établit une relation entre la distance d'édition et la taille du plus grand sous-graphe commun. Dans [Bunke et Shearer, 1998], les auteurs définissent la distance $d(G, G') = \frac{|mcs(G,G')|}{\max(|G|,|G'|)}$ normalisée entre 0 et 1, contrairement à la première.

2.3.3 Les distances basées sur des appariements multivoques

Dans [Ambauen *et al.*, 2003], les auteurs proposent une extension de la distance d'édition de graphe. Ils ajoutent la subdivision d'un sommet et la fusion de sommets à l'ensemble des opérations. La subdivision est définie de telle sorte qu'elle remplace un sommet par un ensemble de nouveaux sommets ; la fusion représente l'opération inverse. Bien qu'il soit désormais possible de traiter certains cas d'appariements multivoques, l'approche est soumise à des contraintes. De fait, si l'on veut mettre en correspondance les sommets v_1 et v_2 du premier graphe avec le sommet v'_1 du deuxième graphe, il faut fusionner les deux. Il devient alors impossible de mettre en correspondance le sommet v_1 avec un autre sommet v'_2 sans aussi mettre en correspondance v_2 avec v'_2.

Dans [Champin et Solnon, 2003], les auteurs suivent une approche plus générique appliquée aux graphes étiquetés. Leur mesure met en rapport le nombre d'attributs communs (dénommées Π) entre deux graphes, le nombre de divisions de sommets et le nombre total d'attributs. Tandis que cette mesure de similarité ne traite pas le problème d'isomorphisme inexact, une distance générique est proposée dans [Sorlin *et al.*, 2007]. Les paramètres génériques sont, non seulement les distances élémentaires (*i.e.* entre les sommets et entre les arêtes), mais aussi la fonction d'agrégation qui reste à définir. Ceci rend l'approche très générique, mais la nécessité d'implémenter tous ces éléments rend l'utilisation moins confortable.

2.4 Projections de graphes

Parallèlement aux méthodes de noyaux de graphes, qui sont présentées ultérieurement, qui proposent une projection implicite dans un espace vectoriel, des travaux connexes se focalisent sur les méthodes de projection (ou plongement) faisant référence à des projections explicites dans un espace vectoriel. Ces deux approches suivent la même idée directrice qui consiste à pouvoir utiliser les outils statistiques pour mesurer une similarité de graphes. Cependant, la différence majeure est que les méthodes implicites utilisent des distances de graphes, donc complexes, pour définir un noyau qui sera ensuite utilisé pour la classification. Les techniques de projection explicite cherchent quant à elles à encapsuler les informations portées par un graphe dans un vecteur numérique. Formellement, soit un graphe $G = (V, E, \mu, \nu)$, la fonction $\phi : \mathcal{G} \rightarrow \mathbb{R}^n$ est une projection explicite de G si $\phi(G) = (f_1, \ldots, f_n)$. Ce vecteur numérique peut ensuite être utilisé pour la classification avec le formalisme des techniques de la reconnaissance des formes statistique. Parmi les outils disponibles figure, par exemple, la distance euclidienne. La mesure de similarité pourra donc être effectuée en un temps linéaire, ou encore des techniques de sélection de caractéristiques ([Guyon et Elisseeff, 2003]). La projection assure que si deux graphes sont isomorphes alors la distance entre leurs deux vecteurs sera égale à 0. Il existe différentes approches pour réaliser cette projection. Nous présentons dans cette section quelques travaux issus de l'état de l'art illustrant les différentes approches.

2.4.1 Les sondages de graphes

La première approche des projections explicites de graphes est liée à la recherche de motifs topologiques ou sondage. Cette transformation est basée sur l'extraction de caractéristiques relatives aux données structurelles. Ces caractéristiques numériques capturent aussi bien des informations sur la topologie (nombre de nœuds, d'arcs, degré des nœuds...) que sur le contenu du graphe (histogramme des étiquettes...). De ce fait, le graphe initial est projeté dans un espace euclidien.

Un des premiers sondages de graphes remonte aux années 40 ([Wiener, 1947]). Dans cette méthode, chaque graphe est représenté par un indice dit *indice de Wiener*. Cet indice est un descripteur topologique défini par la somme de tous les plus courts chemins dans le graphe, tel que si $G = (V, E)$ est un graphe, alors l'indice de Wiener $W(G)$ de G est : $W(G) = \sum_{v_i \in G} \sum_{v_j \in G} l(v_i, v_j)$ avec $l(vi, vj)$ une fonction qui définit la longueur du plus court chemin entre le nœud v_i et et le noeud v_j dans le graphe G.

Une autre approche est introduite dans [Papadopoulos et Manolopoulos, 1999]. Sur des graphes non-orientés et non étiquetés, les auteurs calculent le degré de chaque nœud pour ensuite constituer un histogramme trié des degrés. Ils obtiennent donc le vecteur $\phi(G) \in \mathbb{R}^n$ tel que si $G = (V, E)$ est un graphe alors $\phi(G) = (n_0, n_1, n_2, \ldots)$ avec $n_i = |\{v \in V | d_G(v) = i\}$. Les attributs de ce vecteur résultant sont donc le nombre de nœuds de degré 1 pour la première caractéristique, le nombre de nœuds de degré 2 pour la deuxième, etc ... Une extension de cette technique est proposée dans [Lopresti et Wilfong, 2003]. Les auteurs suggèrent également une représentation basée sur des descripteurs locaux aux nœuds et la généralise pour tous les types de graphes. Cette représentation repose sur le degré des

sommets dans le cas de graphes non orientés et non étiquetés. Dans le cas où les graphes sont orientés, la représentation distingue les demi-degrés intérieur et extérieur. La représentation peut être adaptée pour prendre en considération les graphes étiquetés, prenant en compte, en plus du degré des nœuds, les étiquettes des arcs connectés. D'autre part, les auteurs présentent une relation intéressante liant la distance d'édition et la distance entre projection de graphe. Ils montrent que, à un facteur 4 près, la distance entre la projection de graphe d_{GP} est un minorant de la distance d'édition d_{GED} quels que soient les graphes à comparer : $d_{GP}(G, G') \leq 4.d_G ED(G, G')$.

Les algorithmes présentés dans les travaux cités permettent une projection d'un graphe en un temps linéaire, donc très rapide. Leur simplicité de mise en œuvre est également un atout. En revanche, les caractéristiques utilisées sont très localisées (nœuds ou arcs) et la projection renferme assez peu d'informations topologiques pouvant affaiblir les résultats de classification.

2.4.2 Les méthodes spectrales

De nombreux travaux pour la projection de graphes exploitent la théorie spectrale des graphes ([Chung, 1997]) et s'intéressent aux propriétés des spectres d'un graphe et à la caractérisation de la topologie des graphes en utilisant les valeurs et les vecteurs propres des matrices d'adjacence ou laplaciennes.

Dans [Harchaoui, 2007], une projection utilisant les approches spectrales est proposée. Les auteurs utilisent les vecteurs propres principaux de la matrice d'adjacence pour définir les espaces propres des matrices d'adjacence. Des propriétés spectrales sont ensuite calculées pour chaque mode propre. Dans [Luo *et al.*, 2003], les auteurs utilisent aussi les matrices d'adjacence des graphes comme support pour ensuite calculer leurs vecteurs propres et obtenir ainsi des modes pour définir l'espace vectoriel (périmètre, volume, nombre de Cheeger, ...). La construction du vecteur est finalisée par l'application de la réduction de la dimensionnalité à l'aide d'une ACP (Analyse en Composantes Principales), d'une ACI (Analyse en Composantes Indépendantes) ou d'un MDS (*MultiDimensional Scaling* ou Positionnement Multidimentionnel).

D'autres projections sont réalisées à partir de la matrice laplacienne. Par exemple, dans [Robles Kelly et Hancock, 2007], une approche spectrale a été proposée en rapprochant les opérateurs de Laplace-Beltrami afin de projeter les graphes dans une variété Riemannienne où une métrique permet de définir la longueur d'un chemin (appelée une géodésique) entre deux points de la variété. Cette longueur est alors utilisée pour calculer une similarité entre graphes. Dans [Kosinov et Caelli, 2002], la projection des nœuds d'un graphe est effectuée sur un sous-espace propre défini par les premiers vecteurs propres. Dans [Wilson *et al.*, 2005], les auteurs proposent d'utiliser une décomposition spectrale, toujours de la matrice laplacienne, et construisent des polynômes symétriques. Les coefficients de ces polynômes sont alors utilisés pour projeter les graphes.

Comme pour les techniques d'appariements, la théorie spectrale des graphes montre des propriétés intéressantes qui peuvent être utilisées pour la projection de graphes dans un espace vectoriel. Son principal atout étant la complexité linéaire associée aux opérations sur les matrices. Néanmoins ces approches restent limitées. Les méthodes spectrales sont

sensibles aux bruits ; la suppression d'un nœud, par exemple, modifie les matrices (d'adjacence ou laplaciennes) et ces erreurs se répercutent sur les fonctions de projection. De plus, leur usage limité aux graphes non-étiquetés rendent ces approches difficiles à exploiter dans la majorité des applications de reconnaissance de formes.

2.4.3 Les représentations par dissimilarités

Contrairement aux approches que nous avons présentées, les méthodes de projection par une représentation de dissimilarités ne se focalisent pas sur l'extraction d'un vecteur numérique à partir du graphe. Au contraire, l'idée directrice est de construire un vecteur caractéristiques en comparant un graphe à une sélection de graphes dits prototypes. Dans [Pekalska et Duin, 2005], les auteurs présentent la représentation par dissimilarité comme une alternative à l'utilisation des vecteurs de caractéristiques. Partant du postulat que la dissimilarité permet de séparer une classe d'objets d'une autre, ils proposent non plus de caractériser un objet par des attributs absolus mais de le définir comme un vecteur de différence aux objets d'autres classes. Un objet est donc défini par sa projection dans un espace de dissimilarité.

Dans [Riesen *et al.*, 2007, Bunke et Riesen, 2008], les auteurs proposent d'adapter cet espace de dissimilarités pour la construction d'une fonction de projection de graphes. L'idée directrice de ces travaux est de construire un vecteur de distances d'édition de graphes entre le graphe que l'on veut projeter et un ensemble de k graphes prototypes sélectionnés dans une base. La projection du graphe est donc un vecteur de k distances. Formellement, soit $\Gamma = G_1, \ldots, G_N$ un ensemble de graphes et $P = P_1, \ldots, P_k \subseteq \Gamma$ un sous-ensemble de prototypes sélectionnés de Γ. La projection est définie par la fonction $\phi : \Gamma \to (R)^k$ telle que $\phi(G) = [d(G, P_1), \ldots, d(G, P_k)]$ avec $d(G, p_i)$ la distance d'édition entre le graphe G et le i-ème graphe prototype dans P. Dans [Bunke et Riesen, 2010], les mêmes auteurs proposent ensuite d'améliorer la projection en utilisant des méthodes de sélection de caractéristiques issues de la reconnaissance des formes statistiques.

Cette catégorie de projection est très performante. La difficulté du paramétrage de la distance d'édition évoquée précédemment se retrouve bien sûr dans cette méthode. De plus, le choix des graphes comme prototypes est aussi un paramètre non négligeable puisqu'il détermine la taille du vecteur et sa qualité à bien représenter le graphe dans l'espace vectoriel. De plus, il reste très dépendant de l'application et de sa base d'apprentissage. L'auteur donne quelques indicateurs sur le choix de ces prototypes dans [Riesen, 2009]

2.4.4 Conclusion

Les méthodes de projection de graphes dans un espace vectoriel permettent de lier les reconnaissances statistiques et structurelles de formes. Ces projections explicites transposent le calcul de la similarité entre objets dans un espace vectoriel. Les outils statistiques peuvent donc être utilisés sur les représentations vectorielles des graphes. Cependant, certains inconvénients peuvent être notés. Les méthodes spectrales qui utilisent les matrices d'adjacence ou les matrices laplaciennes interdisent la projection de graphes étiquetés, ce qui exclue l'exploitation de ces techniques dans la majorité des cas. Les représentations par dissimilarités offrent des performances excellentes. Mais les différents paramètres à

régler peuvent être un frein dans leur utilisation. En revanche, les sondages de graphes présentent une plus grande facilité d'adaptation aux données mais leur pouvoir représentatif est moindre. La description de la topologie étant limitée aux nœuds et aux arcs, une partie de l'information est perdue lors de la projection.

2.5 Quelques mots sur la classification de graphes

De manière générale, la classification a pour objectif d'affecter une classe à un individu inconnu. La fonction de classification, le classificateur, aura pour rôle de prendre une décision en vue de l'affectation. Elle doit être capable de modéliser au mieux les frontières qui séparent les classes les unes des autres et se décline en deux catégories :

- Une classification **non supervisée** est utilisée lorsqu'aucune connaissance *a priori* n'est disponible. Le classificateur doit alors répartir les objets sans connaître le nom, le nombre ou les propriétés des classes. Il s'agit alors d'un regroupement d'individus selon un critère de compacité, *i.e.* une minimisation de la dissimilarité de individus au sein d'une même classe et une maximisation de la dissimilarité entre les classes. La classification non-supervisée est également appelée partitionnement (*clustering* en anglais). Pour plus d'informations sur les méthodes utilisées, nous renvoyons le lecteur aux états de l'art : [Xu et Wunsch, 2005] et [Jain *et al.*, 1999].
- Une classification **supervisée** n'est possible que si des connaissances *a priori* sont apportés, par l'intermédiaire d'une base d'apprentissage : une série d'individus dont les classes sont connues. Cette connaissance permet au classificateur d'apprendre les propriétés des classes à partir des modèles fournis. À partir de cette base, le classificateur va extraire de l'information sur les fonctions délimitant les classes. L'algorithme de classification va pouvoir ainsi être capable de prédire la classe inconnue d'un nouvel objet. Nous proposons aux lecteurs un état de l'art de [Kotsiantis, 2007] sur les techniques de classification supervisée.

Dans cette thèse, nous nous sommes focalisés sur la classification supervisée de graphes. Dans ce cadre, un algorithme de classification de graphes a pour but d'affecter une classe à un graphe inconnu. Plus formellement, on peut définir l'apprentissage d'une telle fonction de la façon suivante :

Définition 7 *Soit χ un ensemble de graphes étiquetés et C un ensemble de N classes. Soit un ensemble d'apprentissage $L = \{(g_i, c_i)\}_{i=1}^{N}$, où les $g_i \in \chi$ sont des graphes étiquetés et où $c_i \in C$ est la classe de g_i parmi les N classes présentes dans la base d'apprentissage. L'apprentissage d'un classificateur de graphes consiste à induire de L une fonction $f(g) : \chi \setminus L \rightarrow C$ attribuant une classe à un graphe g inconnu.*

Il existe dans la littérature différentes approches pouvant être distinguées en trois catégories. La première famille de techniques consiste à utiliser les projections de graphes dans un espace vectoriel pour ensuite tirer parti des outils de classification issus de la reconnaissance statistique des formes. Une seconde famille repose sur la définition de noyaux de graphes. La troisième famille s'appuie sur le classificateur des plus proches voisins avec une optimisation de la base d'apprentissage aux moyens de graphes médians ou de graphes prototypes.

2.5.1 Classification à partir de projections de graphes

Nous avons vu, dans la section 2.4, qu'il existait des méthodes permettant de projeter les graphes dans un espace vectoriel, pour pouvoir ensuite bénéficier d'outils statistiques. Ces outils permettent d'évaluer une mesure de similarité entre deux graphes en temps linéaire en utilisant une distance euclidienne sur les représentations vectorielles de ces graphes. De la même manière, ces vecteurs peuvent être utilisés pour l'apprentissage d'un classificateur statistique. La richesse des méthodes de classification utilisés en reconnaissance statistique des formes apportent, par ce biais, un avantage considérable à la représentation sous forme de graphes en associant la diversité et l'adaptabilité de techniques bien définies au pouvoir de représentation des graphes. Parmi l'abondance de la littérature dans ce domaine, nous exposons trois classificateurs reconnus, représentatifs de l'état de l'art, que nous avons utilisés.

Les k plus proches voisins

L'algorithme des k plus proches voisins (ou k-ppv) [Cover et Hart, 1967] est certainement un des algorithmes les plus utilisés en classification supervisée. Le principe de cet algorithme est de classifier un individu i selon les classes des k individus dont il est le moins distant. La première étape consiste à calculer toutes les distances entre i et les individus de la base d'apprentissage. Ces individus sont ensuite classés par ordre croissant et les k premiers sont sélectionnés. La classe de l'individu i est ensuite déterminée selon un vote majoritaire parmi les classes des k individus choisis. Si $k = 1$, alors la classe sera celle de l'individu le plus proche de i. Le k-ppv fait partie des méthodes à apprentissage qualifié de paresseux (*lazy learning*) puisqu'ils n'induisent pas de fonction de classification, celle-ci étant directement effectuée à partir de l'ensemble d'apprentissage. Cet algorithme est défini par trois paramètres principaux : le nombre de voisins k, la mesure de distance et l'ensemble d'apprentissage. L'algorithme 1 décrit le pseudocode des k plus proches voisins.

Algorithme 1: Algorithme des k plus proches voisins

Entrées : i : l'individu que l'on veut classifier.
Entrées : k : Entier positif définissant le nombre de voisins.
Entrées : L : Ensemble d'apprentissage.
Sorties : c_i : La classe de i.
début
  pour chaque $(x, c_i) \in L$ **faire**
    Calculer la distance $D(x, i)$;
  fin
  Ajouter les k individus les plus proches à l'ensemble **K**.
  pour chaque $x \in$ **K faire**
    Compter le nombre d'occurences de chaque classe;
  fin
  Attribuer à c_i la classe la plus fréquente;
fin

L'un des avantages majeurs de cet algorithme est sa simplicité. En effet, seule une fonction de distance doit être définie pour pouvoir l'appliquer. De ce fait, il est généralisable à beaucoup de domaines. Nous verrons un peu plus tard qu'il est utilisable directement en classification de graphes. Choisir une grande valeur pour k permettra de réduire l'influence du bruit sur la classification mais ajoutera en contrepartie un flou au niveau de la frontière entre les classes. Le paramètre k étant un paramètre sensible, il peut être optimisé sur une base dite de *validation* Plusieurs valeurs de k sont testées sur un ensemble d'individus dont la classe est connu, indépendant de l'ensemble d'apprentissage. La valeur de k selectionnée correspond au minimum de l'erreur empirique observée.

Le perceptron multicouche

Le deuxième classificateur que nous présentons est issu de la famille des réseaux de neurones. Introduits dans [Rosenblatt et Laboratory, 1958], les réseaux de neurones sont des modèles inspirés de la biologie utilisant une approche connexioniste afin d'interconnecter des neurones formels (ou artificiels). L'un des modèles les plus populaires est le perceptron multicouche (ou MLP pour *Multi Layer Perceptron*) développé dans [Rumelhart *et al.*, 1986], qui est connu comme approximateur universel de fonctions.

Le perceptron multicouche est un réseau à propagation avant composé de plusieurs couches de neurones. La première couche est la couche d'entrée : aucun calcul n'est effectué, une cellule d'entrée ne fait que copier son entrée vers sa sortie. Les entrées correspondent aux attributs (ou leurs codages) du problème considéré. Ensuite, viennent une ou plusieurs couches cachées constituées de neurones élémentaires. Tous les neurones d'une couche sont les entrées de chaque neurone de la couche suivante et de elle seulement : autrement dit, il n'y a pas de retour arrière et on passe d'une couche à la suivante. Enfin, la dernière couche est la couche de sortie qui contient un ou plusieurs neurones. Les sorties de ces neurones correspondent aux valeurs à estimer (ou leurs codages) pour le problème considéré.

Pour le neurone biologique, lorsque l'activité en entrée dépasse un certain seuil, sa sortie est activée. Pour le neurone formel, une fonction de transfert H est appliquée à l'activité d'entrée. Cette fonction doit être telle que sa valeur de sortie soit 1 lorsque l'activité est suffisante et 0 sinon (parfois -1). Un premier choix possible est la fonction de Heaviside définie par $H(x) = 1$ si $x \geq 0$ et $H(x) = 0$ sinon. Cette fonction n'est pas dérivable. De ce fait, on utilise le plus souvent la fonction sigmoïde, une approximation dérivable de la fonction de Heaviside, définie par :

$$s(x) = \frac{1}{1 + e - x}$$

Dans le perceptron multicouche, les neurones d'une couche sont reliés à la totalité des neurones des couches adjacentes. Ces liaisons sont soumises à un coefficient (appelé poids synaptique) altérant l'effet de l'information sur le neurone de destination. Ainsi, le poids de chacune de ces liaisons est l'élément clef du fonctionnement du réseau : la mise en place d'un perceptron multicouche pour résoudre un problème passe donc par la détermination des meilleurs poids applicables à chacune des connexions interneuronales. Les poids synaptiques sont initialisés aléatoirement et l'apprentissage est réalisé par l'algorithme de rétroprapagation du gradient de l'erreur ([Lippmann, 1987], un algorithme itératif qui a

pour objectif de trouver le poids des connexions minimisant l'erreur quadratique moyenne commise par le réseau sur l'ensemble d'apprentissage.

La figure 2.2 illustre un perceptron multicouche.

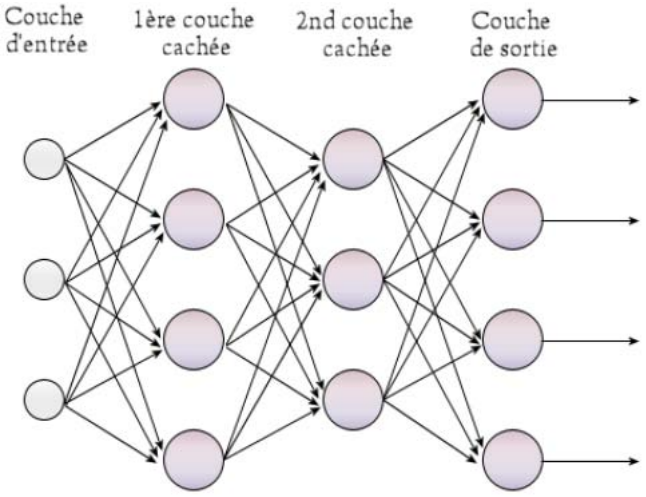


FIGURE 2.2 – Un perceptron multicouches à 4 couches. [11]

Les machines à vecteurs de supports

Introduits dans [Vapnik, 1995], les machines à vecteurs de supports (ou SVM pour *Support Vector Machine*) sont issus de la famille des classificateurs à noyau. Ils présentent deux avantages :

- de très bonnes performances en généralisation (*i.e.* quand les exemples de test sont légèrement éloignés en terme de caractéristiques des individus de l'ensemble d'apprentissage)
- une capacité à résoudre des problèmes où l'ensemble d'apprentissage χ ne possède pas nécessairement de séparation linéaire dans l'espace des caractéristiques.

Le principe théorique des SVM comporte deux points fondamentaux :

1. la transformation non linéaire (Φ) des exemples de l'espace d'entrée vers un espace dit de redescription de grande dimension muni d'un produit scalaire (espace de Hilbert).
2. la détermination d'un hyperplan permettant une séparation linéaire optimale dans cet espace de grande dimension.

L'intérêt est que dans l'espace de redescription la classification s'avère plus aisée : en effet, intuitivement, plus la dimension de l'espace de redescription est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée.

Du point de vue mathématique, la transformation non linéaire (Φ) est réalisée via une fonction noyau κ (ou noyau de Hilbert-Schmidt) facile à calculer. Ainsi l'espace de

11. Illustration extraite de Wikipédia : `http://fr.wikipedia.org/wiki/Perceptron_multicouche`

redescription reste virtuel, jamais explicité. Il existe une multitude de noyaux, le plus utilisée étant le noyau gaussien :

$$\kappa(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2.\sigma^2})$$

La figure 2.3 illustre l'idée directrice des machines à vecteurs de supports. L'espace du problème n'est pas séparable linéairement. La projection dans un espace de Hilbert permet la définition d'un hyperplan séparant les individus linéairement.

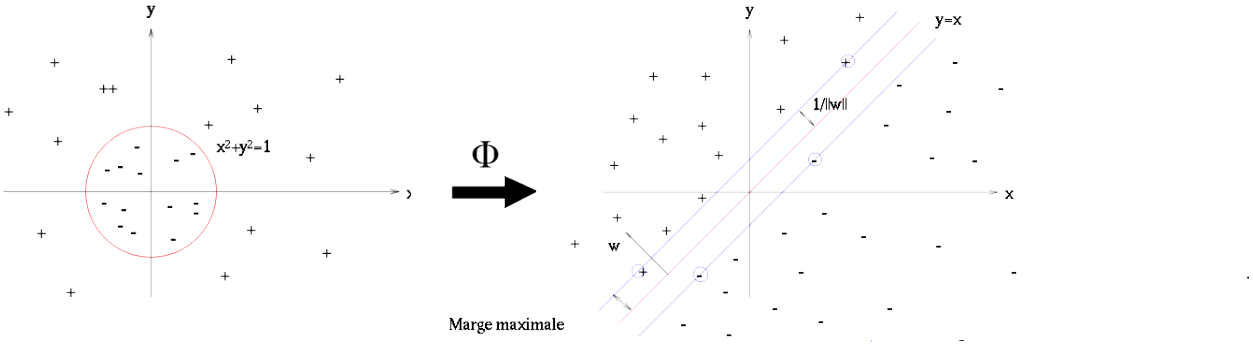


FIGURE 2.3 – Illustration du principe des SVM : transformer un problème de séparation non linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de redescription de grande dimension. [12]

2.5.2 Noyaux de graphes

Les méthodes à noyau sont une classe d'algorithmes utilisés en reconnaissance des formes. Bien que leurs fondements mathématiques soient assez anciens ([Vapnik et Chervonenkis, 1971, Vapnik, 1982, Berg *et al.*, 1984]), leurs applications en reconnaissance des formes sont plutôt récentes ([Scholkopf et Smola, 2002, Shawe-Taylor et Cristianini, 2004, Vapnik, 1998]). La principale motivation de ces méthodes, appelées machines à noyaux, n'est plus de définir un objet comme un vecteur caractérisant des propriétés de l'objet mais comme un vecteur de distances à d'autres objets. De manière plus formelle, soit un espace d'objets χ avec n objets $\{x_1, ..., x_n\} \subseteq \chi$ et soit $\kappa : \chi \times \chi \rightarrow \mathbb{R}$ la fonction de similarité dans χ. La fonction κ définit un noyau qui représente tout l'espace d'objet χ d'une manière implicite par les valeurs du noyau par paires $\kappa_{ij} = \kappa(x_i, x_j)$. Les machines à noyaux exposent alors des propriétés théoriques permettant leurs utilisations en reconnaissance des formes statistiques, notamment sur la définition de nouveaux classificateurs en utilisant par exemple les SVM.

La transposition des méthodes à noyaux aux graphes tente donc d'apporter une réponse au problème majeur de l'utilisation des graphes qu'est la comparaison. Le fil conducteur de l'utilisation de noyau de graphes résulte dans la projection implicite dans un espace vectoriel. Cet espace est défini par des produits internes selon des mesures de similarité

12. Illustration extraite de Wikipédia : `http://fr.wikipedia.org/wiki/Machine_à_vecteurs_de_support`

entre graphes, en respectant un certain nombre de propriétés ; ce sont les noyaux de graphes. L'apport le plus important de ces méthodes est le support des structures mathématiques existantes (distances, moyennes, ...) qui manquent aux domaines des graphes. Cette section présente donc une synthèse des travaux portant sur les noyaux utilisés pour la comparaison de graphes suivant leurs différents philosophies.

Kashima *et al.* ([Kashima *et al.*, 2003]) proposent une fonction de noyau sur des parcours aléatoires. L'idée directrice est de définir une similarité entre graphes en comparant des marches aléatoires dans deux graphes. Une idée intéressante sur ce principe est fondée sur le calcul du nombre de marches aléatoires que deux graphes ont en commun ([Gärtner *et al.*, 2003]). En effet, des parcours de longueur k sont calculables en temps polynomial par l'utilisation de la matrice d'adjacence A élevée à la puissance k. $A^k(i,j) = c$ signifie qu'il existe c parcours de longueur k entre les sommets i et j. Pour trouver les parcours en commun, entre 2 graphes, les auteurs utilisent les produits de graphes $G_X = (V_X, E_X)$(paires de sommets et arêtes identiques entre G et G') définis tels que :

- $V_X(G,G') = \{(v_1, v_1') \in V \times V' : \mu(v_1) = \mu(v_1')\}$
- $E_X(G,G') = \{((v_1,v_1'),(v_2,v_2')) \in V_X \times V_X : (v_1,v_2) \in E \wedge (v_1',v_2') \in E' \wedge \nu(v_1,v_2) = \nu(v_1',v_2')\}$

Il en découle que les parcours communs des deux graphes peuvent être déduits des matrices A_X^k d'où

$$\kappa(G,G') = \sum_{i,j=1}^{|V_X|} [\sum_{n=0}^{\infty} \lambda^n A_X^n]_{i,j}$$

Une autre approche des noyaux de graphes répandue est celle s'appuyant sur les noyaux de convolution. L'idée de base des noyaux de convolution est fondée sur le principe qu'un objet peut être décomposé en sous-parties pour lesquelles une fonction de similarité peut-être définie ou moins complexe à calculer. En utilisant une opération de convolution, ces similarités deviennent une fonction de noyau ([Haussler, 1999]). Neuhaus et Bunke ([Neuhaus et Bunke, 2006]) proposent un noyau de convolution sur graphe basé sur les chemins d'édition. En considérant un graphe $G = (V, E, \mu, \nu)$ et un paramètre $s_{max} \in (N)$, la décomposition est définie telle que $R^{-1} = \{(s, v_1, \ldots, v_s) : 1 \leq s \leq s_{max}, v_i \in V$ et $v_i \neq v_j$ pour $i \neq j\}$. Le noyau de convolution pour les graphes $G = (V, E, \mu, \nu)$ et $G' = (V', E', \mu', \nu')$ est alors défini par :

$$\kappa(G,G') = \sum_{\substack{(s,v_1,\ldots,v_s)\in R^{-1}(g) \\ (s',v_1',\ldots,v_s')\in R^{-1}(g')}} \kappa_\delta(s,s') \prod_{i=1}^{s} \kappa_{RBF}(v_i, v_i')$$

avec $\kappa_\delta(s,s') = 1$ si $s = s'$ et κ_{RBF} le noyau gaussien couramment utilisé par les machines à noyaux.

Dans un espace euclidien, la distance communément utilisée est la distance euclidienne. Dans l'espace des graphes, la distance d'édition de graphe nous donne une mesure de dissimilarité flexible et pertinente. En utilisant cette mesure de dissimilarité en mesure de similarité entre deux graphes et des graphes dit prototypes ([Neuhaus et Bunke, 2006]), la

distance d'édition peut donc dériver sur des noyaux tels que :

$$\kappa_X(G, G') = \sum_{G_0 \in I} \frac{1}{2}(d(G, G_0)^2 + d(G', G0_0)^2 - d(G, G')^2)$$

avec I un ensemble de graphes choisi dans l'ensemble d'apprentissage.

Issues de fondements mathématiques, les méthodes à noyaux de graphes proposent une alternative intéressante en utilisant une projection implicite des graphes dans un espace vectoriel. Nous n'avons rapporté ici que les méthodes phares des travaux de noyaux de graphes. Des états de l'art plus complets sont proposés dans la littérature [Gärtner, 2003] ou [Saunders et Demco, 2007]. Comme les machines à noyaux éprouvées en reconnaissance des formes statistiques, ils donnent de bons résultats en classification mais leurs complexités reste néanmoins leur principal inconvénient. À ce jour, ils sont tous positionnés dans la classe des problèmes polynomiaux ($O(n^6)$ pour les marches aléatoires par exemple) ce qui peut être contraignant pour une utilisation sur de grands graphes.

2.5.3 Graphes prototypes et Graphes médians

Souvent choisi pour sa simplicité, l'algorithme des k-ppv est souvent exploité pour la classification de graphes. Le classificateur est alors utilisé avec une mesure de dissimilarité entre graphes, telle que la distance d'édition de graphes. Ces méthodes souffrent néanmoins de certaines limitations, intrinsèques au k-ppv : sa sensibilité aux bruits (déformations structurelles, étiquettes fluctuantes...) et sa complexité combinatoire.

Certaines méthodes visent toutefois à pallier ces contraintes en réduisant l'ensemble des graphes utilisés par la technique des k-ppv en s'appuyant sur un processus de choix de représentants. On parle alors de méthode des k plus proches prototypes.

Il existe plusieurs méthodes pour obtenir ces graphes prototypes : les travaux présentés dans [Jia et Abe, 1998] exploitent des prototypes basés sur la présence de sous-graphes communs, les approches proposées dans [Bonev *et al.*, 2007] et [Bunke *et al.*, 2003] créent des représentations appelées *super-graphs* ou les travaux décrits dans [Marini *et al.*, 2007] qui consistent à générer des *creative prototypes* en appliquant à un graphe germe une série d'opérations d'édition pour générer les prototypes.

Une des approches à mentionner, probablement la plus utilisée, est celle consistant à exploiter les graphes médians en tant que prototypes. Étant donné un ensemble d'éléments, le médian peut être un concept très utile pour avoir un représentant qui accumule une information globale de l'ensemble. Dans le domaine des graphes, le graphe médian, introduit dans [Jiang et Bunke, 2002], a pour objectif de porter au mieux l'information essentielle à partir d'un ensemble de graphes. Autrement dit, le graphe médian d'un ensemble de graphes est un graphe qui représente l'ensemble de la meilleure manière possible. Étant donné un ensemble de graphes, le graphe médian est défini comme le graphe ayant la plus petite somme des distances à tous les graphes dans l'ensemble ([Jiang *et al.*, 2001]) . Dans la littérature, nous distinguons deux types de graphe médian : le graphe médian d'ensemble et le graphe médian généralisé. La différence principale entre ces deux types est l'ensemble des graphes sur lequel le graphe médian est calculé. Si le graphe médian appartient à l'ensemble de graphes de départ, alors on parle du graphe médian d'ensemble.

En revanche, si le graphe médian n'appartient pas à l'ensemble de départ (*i.e.* un nouveau graphe), alors on parle du graphe médian généralisé.

Définition 8 *Soit $d(.,.)$ une distance ou une mesure de dissimilarité entre deux graphes. Soit $S = g_1, g_2, \ldots, g_n$ un ensemble de graphes. Le graphe médian d'ensemble (*set median graph - smg *) de S est défini par :*

$$smg = \arg\min_{g \in S} \sum_{i=1}^{n} d(g, g_i)$$

Définition 9 *Soit $d(.,.)$ une distance ou une mesure de dissimilarité entre deux graphes. Soit $S = g_1, g_2, \ldots, g_n$ un ensemble de graphes. Soit U l'ensemble infini des graphes qui peuvent être construits à partir des labels de S. Le graphe médian généralisé (*generalized median graph - gmg *) du sous-ensemble S est défini par :*

$$gmg = \arg\min_{g \in U} \sum_{i=1}^{n} d(g, g_i)$$

2.6 Conclusion

Dans cette section, nous avons abordé la vaste thématique des techniques de comparaison de graphes en proposant une synthèse des méthodes représentatives de l'état de l'art. Nous avons classé ces méthodes en 5 catégories selon les concepts qu'elles mettent en œuvre.

Les méthodes d'appariements exacts ont été les premières à être proposées. Sur les fondements de la théorie des graphes tels que l'isomorphisme de graphes et de sous-graphes, ces techniques visent à vérifier la similarité entre deux graphes. Néanmoins, leurs caractères rigides pénalisent la moindre distorsion. Leurs performances sont donc liées à la qualité de la représentation des données sous forme de graphe. L'utilisation des méthodes d'appariements (exacts ou inexacts) permet, en plus de l'évaluation de la similarité entre ces graphes, de conserver une information sur la comparaison des topologies avec en sortie une proposition d'association sommets à sommets de deux graphes. Cette information se révèle être très utile dans un système de recherche d'information où la requête est une forme partielle que l'on cherche à localiser dans une forme plus grande. Une représentation des formes par des graphes autorise ce type de recherche directement (via l'utilisation des appariements), là où les approches statistiques nécessitent un prédécoupage *a priori* de la forme, ou l'utilisation d'une fenêtre glissante, impliquant un paramétrage difficile influant sur la qualité des résultats. Les algorithmes de recherche d'appariement fournissent en sortie la liste des associations sommets à sommets uniquement lorsque l'isomorphisme est constaté. La moindre distorsion renverra une réponse négative même si les graphes diffèrent très peu.

Afin de répondre à cette problématique, les méthodes d'appariements inexacts suggèrent d'assouplir les contraintes sur l'appariement des nœuds et des arcs. Elles permettent de rendre la recherche d'isomorphisme tolérante aux bruits leur permettant d'être applicable

aux données réelles comportant des distorsions. Les méthodes d'appariement inexact proposent une plus grande tolérance afin d'apparier deux sommets (ou deux arêtes) présentant de légères différences mais intrinsèquement, ces méthodes restent complexes à mettre en place. Nous avons vu que les autres méthodes proposées permettent une plus grande tolérance aux bruits mais perdent les appariements entre sommets des graphes comparés pour la plupart d'entre elles.

Les distances entre graphes sont définies comme la manière la plus intuitive de quantifier les similarités entre graphes. Elles apportent une mesure qui est efficace lorsqu'elle est utilisée en classification et robuste aux différents bruits. En revanche, les algorithmes sont complexes et cette complexité peut être un frein non négligeable dans certains cas. Issus de la reconnaissance des formes statistiques, les noyaux sur graphes sont des méthodes très performantes. En projetant implicitement les graphes dans un espace vectoriel, elles lient l'efficacité des outils de classification statistiques à la flexibilité et au pouvoir de description des graphes. En revanche, les algorithmes utilisés sont très complexes car la projection étant implicite, le calcul de la distance effectué dans l'espace des graphes est lui même complexe. La représentation vectorielle des graphes par une projection explicite semble donc être la bonne voie pour faire face aux verrous techniques de la reconnaissance structurelle puisqu'elle permet, en théorie, d'allier efficacement le pouvoir de représentation et la flexibilité des graphes avec la diversité et la complexité des outils statistiques. Cependant, aucune méthode efficace et simple à adapter aux données n'est encore disponible.

Nous allons maintenant présenter nos travaux sur une nouvelle projection explicite de graphes qui répond aux deux inconvénients. Notre méthode s'appuie sur les méthodes de sondage de graphe en enrichissant le sondage par la capture de motifs moins localisés autour du nœud mais sur l'extraction de motifs plus larges.

Chapitre 3

Méthode de projection de l'information topologique d'un graphe

> Rien ne se perd, rien ne se crée,
> tout se transforme.
>
> Lavoisier

3.1 Introduction

Dans le chapitre 2 de ce mémoire, nous avons étudié certaines méthodes visant à comparer les graphes. Nous avons pu observer que depuis les années 70, les techniques exploitant ces graphes dans un contexte de reconnaissance des formes sont en constantes évolutions permettant ainsi de multiplier les domaines d'applications et donc de palier certaines lacunes. Les premiers travaux sur la comparaison des graphes portaient sur la recherche d'isomorphismes parfaits. Ces algorithmes, pertinents sur des données parfaites se révèlent en difficulté lors de leurs applications en analyse d'images et de reconnaissance des formes lorsque les données à comparer subissent des dégradations et des déformations. En effet, une unique perturbation (ajout/suppression d'un sommet ou d'une arête, modification d'un attribut) conduit alors à une réponse négative. Malgré l'évolution des techniques de numérisation, d'analyse d'images et des postraitements correctifs, la modélisation des données à l'aide de graphes reste fluctuante et induit des variations dans les graphes. L'application des méthodes d'appariement reste donc difficile et implique d'introduire de la tolérance lors de la comparaison des graphes.

Dans la partie précédente, nous avons vu que certains travaux proposent alors de définir et d'exploiter une distance entre graphes. Moins stricte qu'un algorithme d'appariement exact dont la réponse est booléenne, ce type de distance propose de mesurer une similarité (ou une dissimilarité) entre deux graphes. On parle alors d'approches inexactes. Plus flexibles, ces techniques apportent une première réponse à la problématique de la déforma-

tion des données dans les problèmes de reconnaissance de formes. Nous avons cependant aussi noté quelques limitations pour ces approches.

Une des difficultés majeures dans ce domaine est également la réduction de la complexité, et donc du temps d'exécution des algorithmes. La littérature fournit d'un coté des approches performantes en termes de taux de reconnaissance mais très coûteuses en temps, et d'un autre coté, des méthodes sont certes plus rapides mais dédiées à un problème particulier, interdisant leur utilisation dans un autre contexte. Ces récentes techniques sont regroupées dans la littérature sous la dénomination *projection de graphes*. Actuellement très étudiées, elles restent encore à améliorer sur de nombreux points car projeter un graphe dans un espace de caractéristiques, conduit à une importante perte d'information et donc à de nombreuses limitations comme expliqué dans la conclusion de la section 2.4. Notre proposition s'inscrit dans ce cadre en proposant de palier certaines lacunes de ces méthodes tout en conservant leurs avantages.

Dans la section 3.2, nous introduisons tout d'abord le concept de l'information topologique d'un graphe et des informations structurelles des formes représentées. Nous rappelons aussi brièvement le positionnement des méthodes de projection existantes par rapport à cette information. Ensuite, nous présentons en détail la méthode que nous proposons, fondée sur un lexique issu du réseau des graphes non-isomorphes. L'algorithme d'extraction des motifs est exposé et illustré avec l'aide d'un exemple. Dans l'ordre naturel des choses, la section 3.3 montre les expérimentations que nous avons menées pour caractériser notre méthode dans un premier temps, puis dans le but d'évaluer ses performances dans un second. Nous finirons en 3.4 en dressant un bilan sur la méthode que nous proposons.

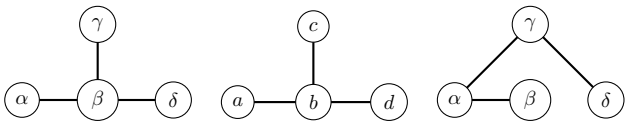
3.2 Projection de l'information topologique d'un graphe dans un espace vectoriel

3.2.1 Idées directrices

Appliqué dans un contexte de reconnaissance de formes, le graphe autorise, de par sa nature, une description analytique d'un objet. Tandis qu'une représentation statistique conduit à décrire un objet dans sa globalité, un graphe permet une représentation en sous-parties et une description des relations entre ces sous-parties grâce aux deux informations véhiculées au sein de la même structure :

- une information topologique, portée par les arêtes, qui définit les relations structurelles entre les formes (ou parties de formes) représentées par les sommets.
- une information étiquetée qui peut correspondre soit aux caractéristiques de la forme (ou d'une partie de forme) si elle est portée par les étiquettes de sommets, soit à affiner la relation entre deux sommets quand elle est véhiculée par les étiquettes d'arêtes.

La figure 3.1 montre un exemple didactique sur la complémentarité des informations topologiques et relatives à l'étiquetage portées par un graphe. Nous présentons ici trois graphes : un graphe dit requête (G_{req}) et deux graphes auxquels nous voulons le comparer (G_1 et G_2). Ces graphes présentent deux jeux d'étiquettes (a, b, c, d et $\alpha, \beta, \gamma, \delta$) et deux topologies différentes, que nous appellerons *étoile* pour G_1 ou G_{req} et *chaîne* pour G_2. Nous



(a) G_{req}. (b) G_1. (c) G_2.

FIGURE 3.1 – Illustration de la complémentarité des informations topologiques et étiquetées. Le graphe requête G_{req} est comparé aux graphes G_1 et G_2. Si seule l'information topologique est prise en compte alors G_{req} et G_1 sont similaires ; en tenant compte uniquement des étiquettes G_{req} et G_2 sont identiques. Si l'ensemble des informations est considéré, alors les graphes G_1 et G_2 sont équidistants de G_{req}.

pouvons constater que les graphes G_1 et G_{req} sont isomorphes. En effet, en négligeant les étiquettes, chaque sommet trouve un appariement univoque, ce qui n'est pas réalisable avec le graphe G_2 puisque G_{req} comporte un sommet de degré 3 et G_2 n'en contient aucun. En revanche, en ne considérant que les étiquettes, alors seuls les graphes G_{req} et G_2 peuvent être appariés, le graphe G_1 ayant des étiquettes différentes. Lorsque l'ensemble de l'information portée par le graphe est considéré, il est possible d'utiliser une distance d'édition de graphes en fixant, par exemple, les coûts de transformations des étiquettes (c_e) et de la structure(c_s) à 1.

- Nous avons vu que les deux graphes G_{req} et G_1 étaient isomorphes, donc $c_s(G_{req} \to G_1) = 0$. Pour obtenir un appariement entre ces deux graphes, les transformations des étiquettes nécessaires sont : $a \to \alpha, b \to \beta, c \to \gamma, d \to \delta$, donc $c_e(G_{req} \to G_1) = 4$. On obtient donc la distance d'édition suivante entre G_{req} et G_1 :

$$c(G_{req} \to G_1) = c_e(G_{req} \to G_1) + c_s(G_{req} \to G_1) = 0 + 4 = 4$$

- Les étiquettes de G_{req} et G_2 sont identiques, aucune modification n'est nécessaire sur les sommets : $c_e(G_{req} \to G_2) = 0$. À l'inverse, certaines opérations sur les arêtes le sont : la suppression de $e(a, c)$ et $e(c, d)$ ainsi que l'ajout de $e(b, c)$ et $e(b, d)$, d'où $c_e(G_{req} \to G_2) = 4$. le coût final est donc :

$$c(G_{req} \to G_2) = c_e(G_{req} \to G_2) + c_s(G_{req} \to G_2) = 4 + 0 = 4$$

Cet exemple montre la double information portée par un graphe. En effet, s'il n'existe pas d'appariement exact entre G_{req} et les graphes G_1 et G_2, nous avons démontré qu'en attribuant des poids identiques aux informations topologiques et physiques, G_1 et G_2 sont à égale distance de G_{req}. Il n'est donc pas possible de déterminer le plus proche de G_{req}. Ce résultat témoigne de l'importance de conserver au maximum cette double information lors de la projection d'un graphe dans un espace vectoriel dans le cas de graphes étiquetés.

Nous avons vu, dans le chapitre 2, que les méthodes de projection de graphes apportaient une contribution intéressante en créant un lien entre les graphes, et leur pouvoir de représentation, et les outils statistiques. Selon nous, pour atteindre une projection de graphes optimale, la méthode doit répondre au maximum aux contraintes énoncées ci-après :

- **Généricité** : la projection doit rester indépendante du contexte d'utilisation. En d'autres termes, la portabilité de la méthode d'un ensemble de graphes à un autre

devra s'effectuer à moindre coût et s'adapter aux données représentées et contenues dans les graphes.

- **Représentativité** : de manière générale, la projection d'un graphe sous forme vectorielle engendre des pertes d'informations. Celles-ci induisent alors des erreurs lors de la phase de classification. En effet, une relation univoque entre un graphe et sa signature permettrait de conserver un pouvoir discriminant optimal en évitant de se retrouver dans le cas où un même vecteur représente deux graphes différents.
- **Complexité** : Nous avons vu que les méthodes de calcul de distances entre graphes étaient onéreuses en temps de calcul. En projetant les graphes dans un espace de dimension N, il est alors possible d'utiliser les méthodes statistiques définies pour des espaces vectoriels qui sont plus efficaces avec une complexité en temps linéaire. Ce gain se fait en contrepartie de la projection qui, elle, peut être coûteuse. Pour la projection des graphes de la base d'apprentissage, une forte complexité de l'algorithme choisi peut être tolérée dans le sens où cette phase s'effectue en temps masqué (hors-ligne). En revanche, il est important de limiter l'impact du temps de calcul nécessaire à la projection du graphe requête qui, elle, est effectuée en ligne.

Le tableau 3.2.1 montre les trois catégories de projection de graphes — les sondages de graphes, les méthodes spectrales et les approches par dissimilarités — et catégorisent les points faibles et les points forts de chacune de ces approches selon les trois contraintes définies ci-dessus. Nous remarquons, que selon nos critères, chaque catégorie de méthode comporte au moins un point faible. Les méthodes de sondages de graphes semblent intéressantes dans le sens où elles présentent une bonne généricité, en s'adaptant facilement à tous types de graphes, et avec une faible complexité. Le point faible des méthodes spectrales réside dans le fait qu'elles ne sont seulement applicables sur des graphes non-étiquetés ou valués (une seule étiquette sur les arêtes) et qu'elles ne tolèrent que peu les bruits structurels. Les approches par dissimilarités présentent de bonnes performances en comparaison mais leur adaptabilité passe par un long processus — choix des prototypes, paramétrage de la distance — et leur complexité est plutôt élevée.

	Généricité	Représentativité	Complexité
Sondages de graphes	++	-	++
Méthodes spectrales	-	-	++
Approches par dissimilarités	-	++	-

TABLE 3.1 – Tableau récapitulatif des points forts et des points faibles des méthodes de projection de graphes.

Nos travaux portent sur une projection et tentent de répondre à ces trois contraintes en s'inspirant du sondage de graphes. Ces méthodes effectuent une projection de la topologie d'un graphe en analysant uniquement le degré des sommets. Nous pensons donc qu'elles peuvent être améliorées en donnant plus de poids et de précision à la représentation de l'information topologique au sein de la projection en dénombrant des motifs encapsulant une information topologique plus riche.

3.2.2 Exploitation du réseau des graphes non-isomorphes

Pour construire une projection adaptée à l'utilisation des graphes en reconnaissance des formes, nous nous sommes inspirés des représentations à base de sac de caractéristiques utilisées en recherche d'images par le contenu [Lazebnik *et al.*, 2006]. La technique des sacs de caractéristiques est une technique éprouvée pour la classification et la catégorisation d'objets visuels et de textures. L'idée de base consiste à procéder à l'échantillonnage de l'image en sous-fenêtres, chacune étant utilisée comme une caractéristique de l'image initiale. Cette approche est très générale car elle ne nécessite aucune hypothèse sur la nature des images. Il devient alors aisé de classer les images en les considérant comme une collection de sous-régions.

Des modèles similaires, appelés sacs de mots, ont été utilisés dans la communauté de la recherche de textes. Les documents textuels sont décrits par une distribution des mots provenant d'un lexique fixe. Cette distribution des termes du lexique au sein des documents, appelée modèle vectoriel [Salton, 1971], [Salton et Lesk, 1968], est exploitée dans des tâches d'indexation et de classification. La description vectorielle est construite en calculant une pondération des termes du lexique, aussi bien pour les documents que pour les requêtes. Ces descriptions vectorielles sont ensuite utilisées pour calculer un degré de similarité entre chaque document et une requête ou entre deux documents. À la différence des sacs de caractéristiques, une dimension sémantique est implicitement encapsulée dans la description à base de sac de mots puisqu'il est possible d'attacher un sens (ou plusieurs) à chaque terme du lexique. Cette considération est d'ailleurs exploitée par des méthodes permettant d'extraire des thèmes cohérents du point de vue sémantique et de façon non supervisée. Citons par exemple l'analyse sémantique latente probabiliste proposée dans [Hofmann, 1999] ou l'allocation latente de Dirichlet décrite dans [Blei *et al.*, 2003]. Les auteurs de [Fei-Fei et Perona, 2005] et [Sivic *et al.*, 2005] ont appliqué des techniques semblables dans le domaine de la vision. Dans le domaine des graphes, les premiers travaux reprenant ce type de concepts sont les approches de [Barbu *et al.*, 2005] visant à utiliser l'idée de sacs de sous-graphes fréquents pour l'indexation de documents graphiques.

La projection de graphe que nous présentons s'appuie sur le dénombrement, au sein du graphe à représenter, de sous-graphes, que nous appellerons motifs. L'ensemble de ces sous-graphes est appelé le lexique. De manière plus formelle, soient $G = (V, E)$ un graphe non-étiqueté et Λ un lexique tel que $\Lambda = \{\gamma_1, \ldots, \gamma_n\}$ avec $\gamma_i = (V_i, E_i)$ le graphe i — appelé i-ème motif — et $1 \leq n \leq N$ avec $n \in \mathbb{N}$. Nous proposons de définir la fonction $\Phi : G \to \mathbb{R}^N$ comme la projection du graphe G tel que :

$$\Phi(G) = (|SG_{\gamma_1}(G)|, \ldots, |SG_{\gamma_n}(G)|)$$

avec $SG_{\gamma_i}(G)$, l'ensemble des sous-graphes de G isomorphes à γ_i.

Nous avons choisi de baser notre approche sur un lexique capable de répondre aux trois critères nécessaires à la mise en place d'une bonne représentativité de la projection du graphe. Pour cela, nous prenons comme référence le réseau des sous-graphes non-isomorphes, présenté dans [Jaromczyk et Toussaint, 1992]. Ce réseau présente l'ensemble des graphes composés de n arêtes jusqu'au rang de taille N (N étant le nombre d'arcs maximum) de manière totalement exhaustive. Ce réseau est construit de façon itérative à partir du graphe constitué d'un unique sommet. À chaque itération, il est possible de

construire un graphe de taille n en ajoutant une arête à un graphe de taille $n-1$ avec au besoin l'ajout d'un sommet supplémentaire. Toutes les solutions sont envisagées ce qui rend le réseau complet. Le graphe de taille n issu d'un graphe de taille $n-1$ est appelé successeur. Réciproquement, le graphe de taille $n-1$ est appelé prédécesseur. Un graphe de ce réseau peut avoir plusieurs successeurs. De même, plusieurs graphes de taille $n-1$ peuvent donner lieu à un même successeur. Les chemins de construction de ce réseau des graphes non isomorphes peuvent être conservés pour facilement reconstituer l'ensemble des prédécesseurs et des successeurs d'un graphe donné.

Par exemple, la figure 3.2 illustre le réseau des graphes non-isomorphes jusqu'au rang de taille 4. Les flèches en pointillé indiquent les chemins de construction du réseau, les flèches étant orientées du prédécesseur vers le successeur.

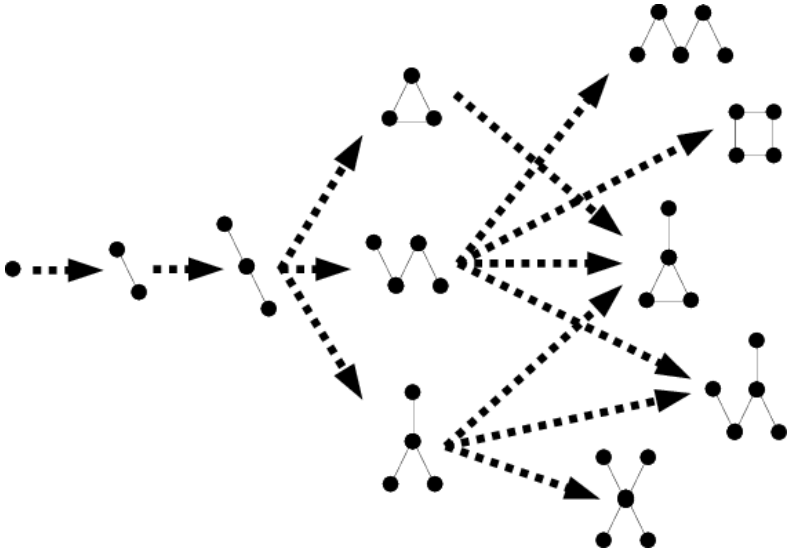


FIGURE 3.2 – Réseau des sous-graphes non isomorphes.

Comme nous le présentions précédemment, le choix d'utiliser ces graphes générés par ce réseau est guidé par la vérification des hypothèses émises pour une projection de graphe pertinente.

- la **généricité** du réseau est induite par la construction même de celui-ci. En effet, aucune connaissance *a priori* sur le contexte de l'utilisation des graphes n'est requise pour construire le réseau. Le lexique sera donc indépendant des graphes à caractériser. Notre lexique est unique mais néanmoins adapté au contexte d'application sans qu'aucune modification ne soit nécessaire.
- La **représentativité** entre un graphe et sa forme vectorielle est maximale lorsque la taille des graphes du dernier rang du lexique est égale ou supérieure à la taille du graphe à encapsuler. En effet, dans ce cas précis, le vecteur caractéristique des motifs de taille N comporte un 1 et une suite de 0 certifiant le rapport univoque entre le graphe et sa représentation vectorielle. La projection du graphe est, dans ce cas, bijective.
- La **complexité** de cette méthode peut être importante. Nous avons vu que l'isomor-

phisme de sous-graphe était un problème NP-Complet. Cependant, nous proposons un algorithme qui permet de réduire cette complexité en se basant sur la forme itérative de la construction du réseau ; nous le détaillons en 3.2.2.1.

Nous observons d'ores et déjà que la taille du lexique est un paramètre important qu'il faudra fixer suivant différents critères. Le tableau 3.2 montre l'évolution du nombre d'éléments du lexique en fonction du rang maximal du réseau des graphes non-isomorphes.

Rang	Taille
0	1
1	2
2	3
3	6
4	11
5	23
6	51
7	117
8	276
9	669
10	1629

TABLE 3.2 – Effectif du lexique en fonction du rang des graphes non isomorphes.

La complexité de l'algorithme de la projection d'un graphe en une représentation vectorielle est directement dépendante du nombre de motifs à dénombrer. La taille des motifs est liée à la taille du lexique déterminant la richesse de la représentation vectorielle en découlant. Elle contient alors une information topologique plus ou moins riche. Il s'agit donc de trouver un compromis entre l'expressivité et la complexité.

Nous pouvons observer que le lexique présenté ici ne permet pas de projeter des graphes contenant des boucles. Afin d'étendre la projection à ce type de graphe, il est toutefois possible de rajouter un motif *boucle*, comportant un seul sommet et une seule arête. Cependant, les graphes utilisés en reconnaissance structurelle des formes ne comportent généralement pas de boucles, nous avons donc préféré ne pas exposer ce cas. D'une manière générale, la méthode permet de ne projeter que des graphes simples non-orientés. Dans le cas contraire (le graphe est orienté), la représentation associée à un graphe est extraite après un prétraitement qui ramène le graphe à un graphe simple en intégrant les informations d'orientation au sein de l'étiquette de l'arc. Les boucles, comme les arêtes multiples, peuvent également être facilement remplacées par des étiquettes (de sommet pour les boucles, d'arête pour les multiarêtes) (Nous verrons comment projeter les étiquettes dans le chapitre 4). Nous pouvons également remarquer que les motifs sont des graphes connexes ; la projection permet toutefois la description de graphes non connexes.

3.2.2.1 Algorithme de projection

La construction du vecteur numérique représentant le graphe consiste à déterminer la fréquence de chacun des motifs du lexique présents dans le graphe à décrire pour obtenir une représentation histogrammique. Nous utiliserons le terme *fréquence*, issu du vocabulaire

utilisé en recherche d'information, pour désigner le nombre d'occurrences d'un motif. La dimension du vecteur correspond à la taille du lexique. Le lexique étant trié en suivant l'ordre du réseau des sous-graphes (parcouru en largeur), la première valeur du vecteur correspondra alors au nombre de sommets (le premier motif du lexique est un graphe connexe sans la moindre arête, soit un graphe constitué d'un seul sommet), la deuxième au nombre d'arêtes (le second motif du lexique est un graphe connexe composé de deux sommets reliés par une arête), la troisième au nombre de sous-graphes présentant deux arcs, ... Les automorphismes, injections différentes possibles d'un motif dans le graphe, ne seront dénombrés qu'une seul fois.

Nous présentons ici l'algorithme que nous avons mis en place pour projeter un graphe

dans un espace vectoriel à partir d'un lexique.

Algorithme 2: Algorithme d'extraction de la méthode de projection.

```
    Entrées : G : Graphe que l'on veut projeter
    Entrées : N : Entier positif définissant la taille maximale des motifs du lexique
    Sorties : motif : Vecteur d'entiers
    début
(1) |  RepVectorielle ← ConstruireLexique(N);
    |  /* L'ensemble des motifs de taille 0 jusqu'à N est généré          */
(2) |  pour i = 1 a |V_G| faire
    |  |  Ajouter v_i(G) à set;
    |  fin
    |  /* On recherche les occurences du premier motif : les sommets      */
    |  temp ← set;
    |  /* Les occurences de ce motif serviront comme supports pour la
    |     suite. Ce sont les prédécesseurs des occurrences suivantes      */
    |  pour i = 1 a N faire
    |  |  pour chaque g_i ∈ temp faire
(3) |  |  |  successeurs ← GénérerSuccesseurs(g_i, G);
    |  |  |  /* Les successeurs candidats sont générés                    */
    |  |  fin
    |  |  pour chaque g_i ∈ successeurs faire
    |  |  |  temp ← ∅;
    |  |  |  si g_i ⊂ G alors
    |  |  |  |  Ajouter g_i à temp ;
    |  |  |  fin
    |  |  |  Fusionner(temp, set) ;
    |  |  fin
    |  |  /* Si un successeur candidat est un isomorphe d'un sous-graphe
    |  |     de G, alors il est rajouté à l'ensemble des occurrences d'un
    |  |     motif                                                        */
(4) |  |  pour chaque g_i ∈ set faire
    |  |  |  motif ← IdentifierMotif(g_i);
    |  |  |  /* Pour chaque occurrence dans l'ensemble, on identifie le
    |  |  |     motif correspondant ...                                   */
    |  |  |  IncrémenterNbOccurencesMotif(motif, RepVectorielle);
    |  |  |  /* ... et on incrémente le nombre d'occurrence de ce motif   */
    |  |  fin
    |  fin
    fin
```

La projection est décrite par le pseudocode de l'algorithme 2. Cet algorithme prend en paramètre d'entrée un graphe, que l'on désire projeter, et un entier qui définit la taille maximum des motifs du lexique. La construction de la représentation vectorielle est progressive : en partant de chacun des sommets d'un graphe, un successeur potentiel est construit en ajoutant un arc. Si ce sous-graphe synthétique est présent dans le graphe à extraire,

Algorithme 3: La fonction *Successeurs()*.

Entrées : g : graphe
Entrées : G : graphe
Sorties : set : ensemble de graphes
$set \leftarrow \varnothing$
pour chaque v_i *dans* g **faire**
  $g_{temp} \leftarrow$ AjouterArc(v_i, g)
  si $g_{temp} = SousGraphe(G)$ **alors**
    si $g_{temp} \notin set$ **alors**
      Ajouter g_{temp} à set;
    fin
  fin
fin

alors il est ajouté à l'ensemble des prédécesseurs ainsi qu'à l'ensemble des sous-graphes présents dans le graphe. Ainsi, l'occurrence d'un successeur n'est recherchée qu'à partir de l'occurrence d'un prédécesseur ; limitant ainsi la complexité de la recherche. Cette boucle continue tant que l'ordre maximum des graphes présents dans l'ensemble des successeurs est inférieur à l'ordre maximum du lexique. Pour finir, l'ensemble des sous-graphes présents est exploré pour identifier les motifs et incrémenter la valeur de ce motif dans la représentation vectorielle. La section suivante illustre le déroulement de l'algorithme sur un exemple concret.

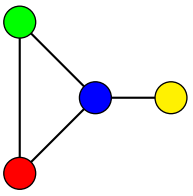
3.2.2.2 Exemple d'une projection

Nous avons choisi de présenter un notre extraction sur le graphe 3.3(a) pour un lexique de 6 motifs. Nous rappelons que ce lexique est issu du réseau des graphes non isomorphes (cf. figure 3.2) dont le paramètre N fixe la taille maximum des graphes. Pour obtenir un ensemble de 6 motifs, il faut $N = 3$. Le tableau 3.3(b) présente le lexique que nous utiliserons pour cet exemple. Ce tableau correspond à la sortie de l'étape (1) de l'algorithme 2 utilisé

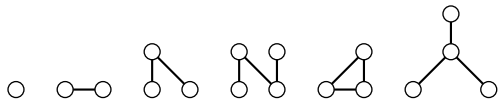
L'étape suivante (2) de notre algorithme correspond à l'identification des sommets. Cette étape, présentée en figure 3.4, correspond également au dénombrement des occurrences du premier motif du lexique. Les sous-graphes issus de cette étape sont stockés dans l'ensemble *set* des occurrences des motifs du lexique afin d'être dénombrés par la suite.

Durant l'étape (3), les successeurs de chaque motif de l'ensemble *set* issu de l'étape précédente sont générés. Cette synthèse est réalisée par la fonction 3. Chaque sommet du graphe passé en paramètre est soumis à la génération d'un successeur ou plusieurs successeurs. Nous rappelons que la définition d'un successeur dans le réseau des graphes non-isomorphes est conditionné par l'ajout d'une arête. Il existe alors deux possibilités :

- une connexité avec un autre sommet du graphe. L'arc est alors ajouté entre les deux sommets. L'ordre du successeur est alors égal à celui du prédécesseur.
- aucune connexité n'est possible. Un sommet est alors créé pour permettre l'ajout



(a) Un graphe G.



(b) Lexique issu du réseau des graphes non-isomorphes pour $N = 3$.

FIGURE 3.3 – Un graphe g et Un lexique L. g est un graphe non-orienté et non-étiqueté ; les couleurs sont utilisés comme identifiants de sommets. Le lexique L est généré à partir du réseau des graphes non-isomorphes pour $N = 3$.

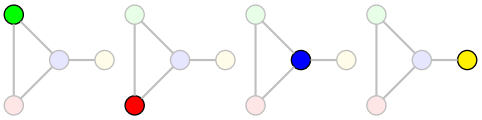
γ_1 (*entrée*)	G (*entrée*)	*sortie*
		*set* = {  }

FIGURE 3.4 – Extraction des occurrences du premier motif γ_1 du lexique présentes dans le graphe G.

d'un arc. L'ordre du successeur est alors égal à l'ordre du prédécesseur + 1.
Une recherche du nouveau motif est ensuite effectuée localement autour du prédécesseur (cf. figure 3.5). La localisation du prédécesseur étant faîte à l'étape précédente, cette recherche est d'une complexité linéaire puisqu'il suffit de constater la (les) présence(s) de l'arc ajouté dans le graphe, ainsi que le sommet ajouté le cas échéant. La complexité totale de l'algorithme est donc fortement correlée au nombre de sommets du graphe à projeter et à leur degré moyen. Les isomorphes trouvés sont alors stockés, dénombrés et utilisés comme prédécesseurs lors du bouclage suivant. La taille des occurrences détermine l'arrêt de la boucle lorsque N est atteint.

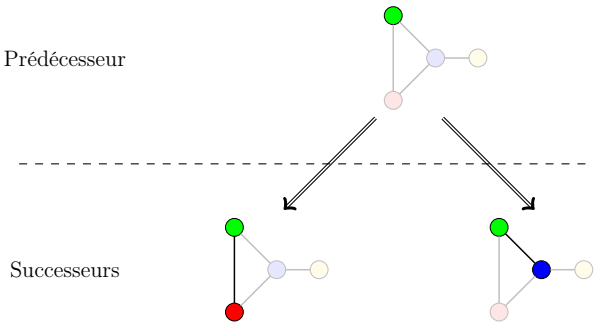
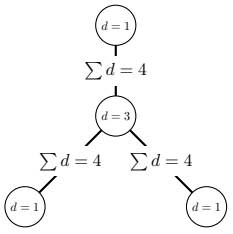


FIGURE 3.5 – Génération des successeurs pour une occurrence issue de l'ensemble généré (motif *sommet*) lors de la précédente boucle de l'algorithme.

La dernière étape consiste à construire le vecteur de sortie de l'algorithme. Ce vecteur est de taille égale à celle du lexique et les valeurs sont initialisées à 0. Chaque élément du vecteur correspond à un motif du lexique. Il est ensuite facile d'incrémenter le nombre d'occurrences d'un motif pour constituer la représentation vectorielle d'un graphe à partir de chaque sous-graphe stocké aux étapes (2) et (3). Dans le but d'accélérer l'identification, nous avons mis en place une signature des motifs afin de les indexer pour rendre cette étape plus rapide. Chaque sommet est étiqueté par son degré et chaque arête par la somme des degrés des deux sommets connectés. La signature est un vecteur de l'ensemble des étiquettes, trié par ordre décroissant. L'ensemble des signatures des motifs est ensuite répertorié dans une table de hachage. Ainsi, la recherche du motif correspondant à l'occurrence que l'on souhaite identifier est très rapide. Une illustration de la signature est fournie en figure 3.6.



(a) Le motif *étoile*.

$$Sig(etoile) = \langle 4, 4, 4, 3, 1, 1, 1 \rangle$$

(b) Sa signature.

FIGURE 3.6 – Exemple de signature d'un motif.

À titre indicatif, nous présentons en figure 3.7 les onze premiers motifs du lexique et leurs signatures attitrées.

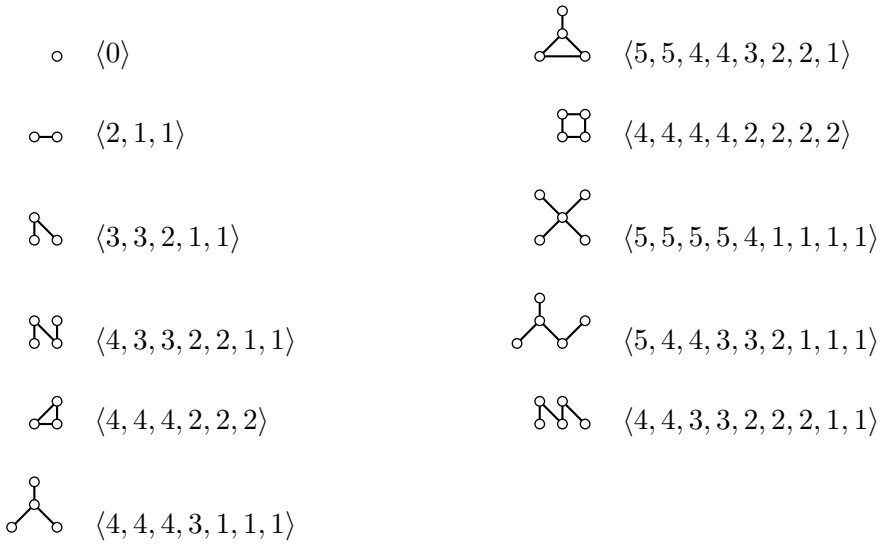
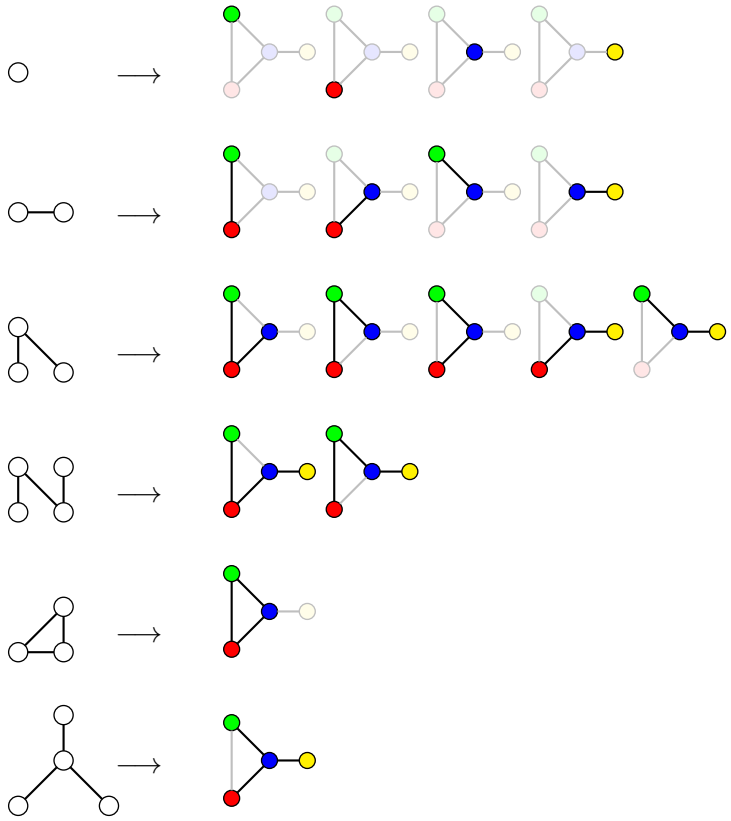


FIGURE 3.7 – Les 11 premiers motifs du lexique et leurs singatures.

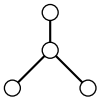
La figure 3.8 illustre l'extraction de l'information topologique du graphe 3.3(a). La figure 3.8(a) présente l'ensemble des motifs du lexique en colonne, à gauche. Pour obtenir la représentation vectorielle du graphe G, chaque occurrence d'un motif est décomptée. Nous présentons donc, pour chaque motif, l'ensemble des occurrences présentes dans le graphe G. Ces occurrences apparaissent en noir, le reste du graphe n'appartenant pas à un isomorphe de ce motif est grisé. Par exemple, la première ligne met en évidence les occurrences du premier motif, composé d'un unique sommet. Quatre sommets sont décomptés dans le graphe. De la même manière, les quatre occurrences du deuxième motif, les arêtes, sont identifiées et énumérées. Notons que les deux premières caractéristiques de notre représentation vectorielle correspondent respectivement à l'ordre et à la taille du graphe. Dans cet exemple, nous dénombrons ensuite cinq occurrences de type *chaîne de 2 arêtes* , deux *chaîne de 3 arêtes*, un motif *triangle* et un *étoile* (les termes utilisés sont purement illustratifs et sont utilisés uniquement pour aider à la compréhension de la figure).

Le tableau 3.8(a) présente un résumé du dénombrement des occurrences, correspondant à notre représentation vectorielle de l'information topologique du graphe G . Nous remarquons que les valeurs de l'histogramme diminuent puisqu'un même sous-graphe contribue à plusieurs motifs durant l'énumération. Cela ne constitue cependant pas de redondance inutile car supprimer l'énumération des motifs de taille faible correspond à une perte d'information topologique dans la représentation vectorielle obtenue. Rappelons que l'objectif n'est pas de détecter un isomorphisme mais de produire une signature vectorielle permettant une mesure de similarité entre graphe exploitant l'information topologique portée par les graphes.

Afin d'avoir une représentation plus graphique, et pour aider les lecteurs dans la comparaison de ces vecteurs numériques, nous illustrerons notre représentation vectorielle d'un graphe sous forme d'histogramme. La figure 3.9 illustre de manière visuelle l'exemple 3.8.



(a) Description des occurrences du premier motif du lexique présents dans le graphe G.

Motifs						
$\Phi(G)$	4	4	5	2	1	1

(b) Représentation vectorielle du graphe G.

FIGURE 3.8 – Exemple de la projection du graphe G.

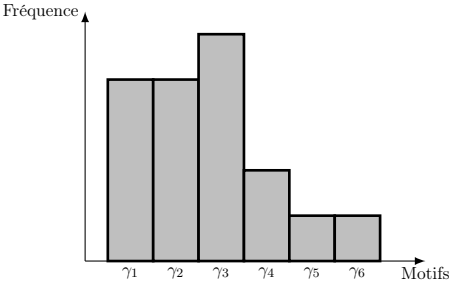


FIGURE 3.9 – Histogramme de la représentation vectorielle du graphe G.

3.2.2.3 Exemple de comparaison de projections

La sous-section 3.2.2.2 illustrait la projection de la topologie d'un graphe de manière pédagogique afin d'aider le lecteur dans la compréhension de la signature et le déroulement de l'algorithme. Nous allons maintenant voir un cas d'utilisation sur trois graphes présentés figures 3.10(a),3.10(b) et 3.10(c). Les représentations vectorielles extraites avec un lexique de 11 motifs, soit $N = 4$, sont synthétisées dans le tableau 3.10(d) et les histogrammes respectifs en 3.10(e), 3.10(f) et 3.10(g). Cet exemple a pour objectif d'utiliser notre méthode pour quantifier une différence entre ces graphes, pour affecter un graphe à une classe par exemple ou pour la recherche du graphe le plus proche. Ces trois graphes sont issus de la base *GREC*, introduite ultérieurement dans 3.3.2.1. Il s'agit donc d'une situation issue d'un problème réel où G_1 et G_3 appartiennent à la même classe et G_2 à une classe différente.

L'un des avantages d'exploiter une signature de graphes est l'utilisation d'outils issus de la reconnaissance des formes dites statistiques. Rappelons que, dans ce domaine, les objets d'une image sont définis par des vecteurs caractérisant un ensemble d'attributs. Le plus souvent des vecteurs numériques sont utilisés. Il est par exemple facile de décrire une couleur grâce aux différents modèles de représentation [Swain et Ballard, 1991] ou bien une forme par des moments invariants [Belkasim *et al.*, 1991]. La projection de l'information topologique est ainsi réalisée et un vaste choix de métriques permettant de définir une distance entre deux individus est ensuite disponible.

Dans $\mathbb{R}^n$, la distance entre deux points peut être définie de plusieurs manières. La distance euclidienne est la plus utilisée et la plus intuitive car elle correspond à ce que nous appelons couramment la distance à vol d'oiseau. Appelée également 2-distance, la distance euclidienne est définie par :

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

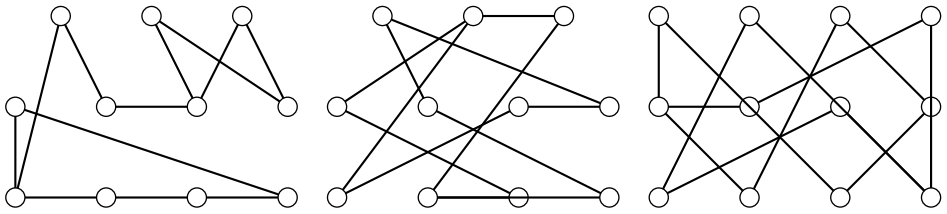
avec X et Y, 2 points de $\mathbb{R}^{\aleph}$ tels que $X = (x_1, x_2, \ldots, x_n)$ et $Y = (y_1, y_2, \ldots, y_n)$.

Le calcul de la distance euclidienne donne pour résultats :
- $d(\Phi(G_1), \Phi(G_2)) = \sqrt{26} \approx 5,10$
- $d(\Phi(G_1), \Phi(G_3)) = \sqrt{5} \approx 2,24$
- $d(\Phi(G_2), \Phi(G_3)) = \sqrt{21} \approx 4,58$

Cette distance se révèle être pertinente au vu de la vérité terrain associée avec ces graphes. En effet, G_1 et G_3 qui appartiennent à la même classe sont à la fois à une distance faible ($\approx 2,24$) tandis que leurs distances à G_2, est élevée avec un rapport 2.

Cet exemple nous permet également d'illustrer l'importance du choix des motifs du lexique, dans notre cas il s'agit de sa taille N. Alors que nous venons de conclure positivement sur la classification de ces 3 éléments, nous pouvons aussi noter que pour un lexique réduit à 6 motifs (pour $N = 3$) les distances obtenues sont les suivantes :
- $d(\Phi(G_1), \Phi(G_2)) = \sqrt{0} = 0$
- $d(\Phi(G_1), \Phi(G_3)) = \sqrt{4} = 2$
- $d(\Phi(G_2), \Phi(G_3)) = \sqrt{4} = 2$

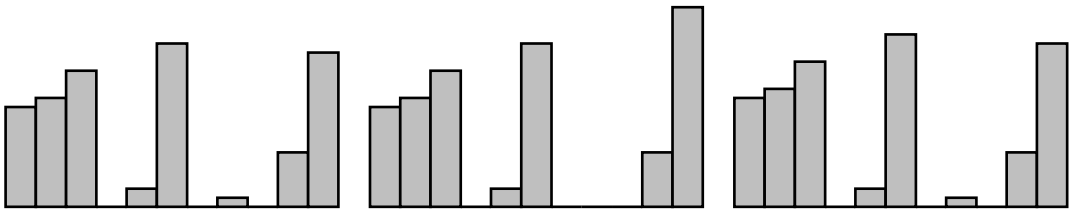
Ce résultat peut laisser croire que G_1 et G_2 sont similaires alors que cela n'est pas le cas. Il est donc essentiel que la taille du lexique soit adaptée à la taille des graphes.



(a) Un graphe $G1$. (b) Un graphe $G2$. (c) Un graphe $G3$.

Lexique	○	○—○	○—○—○								
$\Phi(G_1)$	11	12	15	0	2	18	0	1	0	6	17
$\Phi(G_2)$	11	12	15	0	2	18	0	0	0	6	22
$\Phi(G_3)$	12	13	16	0	2	19	0	1	0	6	18

(d) Représentation vectorielle des graphes G_1, G_2 et G_3.



(e) Histogramme du graphe G_1. (f) Histogramme du graphe G_2. (g) Histogramme du graphe G_3.

FIGURE 3.10 – Trois représentations vectorielles et leurs histogrammes correspondants aux graphes G_1, G_2 et G_3.

3.3 Évaluation de la pertinence de la projection des informations topologiques

3.3.1 Introduction

Afin de valider le respect des hypothèses énoncées dans la section précédente, nous avons effectué des expérimentations sur l'exploitation de notre projection de l'information topologique. Dans cette section, nous présentons des résultats d'expérimentations faites dans un contexte de classification de graphes utilisés en reconnaissance de forme. Le choix des bases de graphes et du processus de classification a été réalisé en fonction des critères auxquels nous voulions répondre :

– **Généricité** : Nous avons sélectionné 3 bases de natures variées avec des paramètres très différents. Les graphes utilisés sont tous issus de données réelles liées à des problèmes de classification. Elles sont présentées par Riesen et Bunke dans [Riesen et Bunke, 2008]. Les graphes sont initialement dotés d'attributs sur leurs arêtes et sur leurs sommets. Ceux-ci ont été supprimés lors d'un prétraitement afin de ne conserver que l'information topologique. Nous présentons ces bases en détail dans la sous-section suivante.

– **Représentativité** : Afin de vérifier que notre projection basée sur la distribution de motifs topologiques étendue (avec des motifs pouvant être constitués de plusieurs sommets et plusieurs arêtes) améliore la représentativité de la représentation vectorielle par rapport au graphe initial, nous évaluons la qualité de la représentativité de notre approche en la comparant aux méthodes de la littérature sur des tâches de classification.
– **Complexité** : L'extraction des projections de graphes est de complexité exponentielle puisque la recherche des occurrences des motifs est exhaustive. Cependant, nous avons proposé un algorithme limitant la recherche des motifs autour des occurrences des prédécesseurs. Nous avons donc également conduit des expérimentations sur les temps de calcul nécessaires pour la projection d'un ensemble de graphes illustrant les performances de notre projection.

3.3.2 Présentation des bases

3.3.2.1 GREC

Cette base de données est issue du concours de reconnaissance ([Dosch et Valveny, 2005]) organisé en 2005 lors du Workshop Graphics Recognition (GREC) . Elle se compose de graphes représentant les symboles issus de dessins architecturaux et électroniques. Les symboles subissent cinq différents niveaux de distorsion. En figure 3.11 des exemples de dessin de symboles sont donnés.

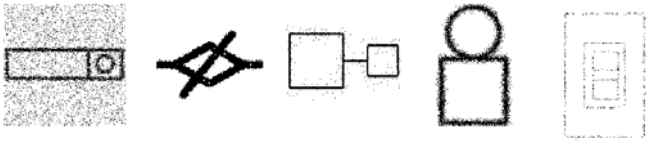
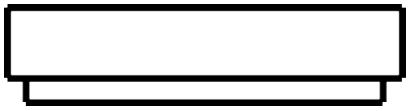
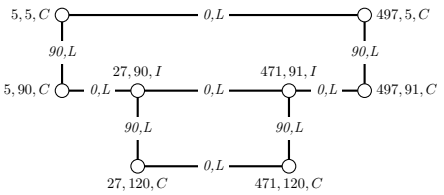


FIGURE 3.11 – Exemples de symboles issus de la base GREC.

Différentes opérations morphologiques sont appliquées : érosion, dilatation ou autres. Une étape de squelettisation est ensuite effectuée afin d'obtenir des lignes d'un pixel de largeur. Les prétraitements faits, les graphes sont extraits de ces images après un débruitage. Les points terminaux, les coins, les intersections et les milieux sont représentés par des sommets et portent une étiquette possédant deux attributs — coordonnées x et y — définissant leurs positions en plus d'un attribut symbolique caractérisant leurs types. Si les points sont reliés par un trait, alors une arête non-orientée est créée entre les deux sommets. Les arêtes sont étiquetées par un attribut définissant leurs types — *arc* ou *ligne* — et un attribut numérique précisant l'angle par rapport à l'horizontale. De la base de données GREC originale, 22 classes sont retenues. Pour obtenir un ensemble complet et consistant, tous les graphiques sont déformés neuf fois afin d'obtenir un ensemble de données contenant 1100 graphes uniformément répartis sur les 22 classes. L'ensemble est divisé en une sous-partie entrainement et une sous-partie validation de la taille 286 chacune, et une de test de 528 graphes. L'ordre moyen des graphes est de 11 tandis que la taille moyenne est 12. La figure 3.12 illustre la représentation d'un symbole par un graphe.



(a) Symbole issu de la base du concours de reconnaissance lors du Workshop GREC'05.



(b) Graphe du symbole.

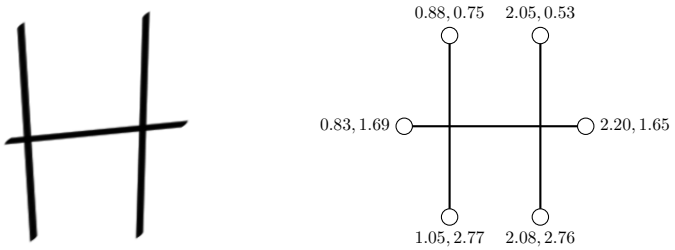
FIGURE 3.12 – Un symbole et sa représentation par un graphe. Le graphe est construit à partir des points particuliers. Les sommets sont étiquetés avec 3 attributs : ses coordonnées (x et y) et son type (I pour *intersection* , c pour *corner*). Les arêtes portent une étiquette à 2 attributs : un angle et un type (pour ce cas, seul l'atttribut *line* (L) est présent).

3.3.2.2 Letter

Cette base comporte des graphes représentant des distorsions de dessins de lettres. 15 lettres majuscules de l'alphabet romain sont repertoriés (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z). Pour chaque classe, un prototype de dessin est construit manuellement. La figure 3.13 présente le prototype de la lettre A, ainsi que des exemples de distorsions.

FIGURE 3.13 – Exemples de la lettre A soumise à différentes distortions

Ces prototypes sont ensuite convertis en graphes, en représentant les lignes par des arêtes non-orientées et les points terminaux des lignes par des sommets. Chaque sommet est étiqueté par deux attributs (x et y) informant sur leurs positions relatives à un système de coordonnées. Les arêtes ne sont pas étiquetées. L'ensemble se compose de 2250 graphes uniformément répartis sur les 15 classes. Afin de tester les classificateurs dans des conditions différentes, des distorsions sont appliquées sur le prototype graphique. Par conséquent, l'ensemble de données total comprend 6750 graphes. Les graphes ont en moyenne 5 sommets et 3 arêtes. La figure 3.14 fournit l'exemple, issu de cette base, d'un dessin de la lettre H et de sa représentation par un graphe.



(a) Dessin de la lettre H

(b) Graphe du dessin

FIGURE 3.14 – Un dessin de la lettre H et sa représentation par un graphe. Le graphe est construits à partir des points terminaux. Les sommets sont étiquetés avec 2 attributs : ses coordonnées (x et y) exprimés par la distance par rapport à l'origine (coin supérieur gauche). Les arêtes ne portent pas d'étiquettes.

3.3.2.3 Mutagenicity

Cette base est composée de graphes représentant des molécules chimiques. La mutagénicité est l'une des nombreuses propriétés indésirables d'un composé qui entrave sa capacité à devenir un médicament commercialisable. Afin de convertir des composés moléculaires en graphes, les atomes sont représentés par les sommets et les liaisons covalentes par les arêtes non-orientées. Les sommets sont étiquetés avec un attribut symbolique pour définir le nom de l'atome (C, N, H...) et les arêtes portent un attribut numérique caractérisant le nombre de liaisons covalentes entre deux atomes. Il est intéressant de noter que cet attribut numérique peut être considéré comme un attribut symbolique au vu du faible intervalle dans lequel il est défini, 8 valeurs possibles au maximum. L'ensemble de données a été préparé initialement par les auteurs de [Kazius *et al.*, 2005]. L'ensemble des données est divisé en deux catégories : mutagène et non mutagène. Nous utilisons un ensemble d'apprentissage de 1500 graphes, un ensemble de validation de taille 500, et un ensemble de test de 2337. Ainsi, il y a 4337 éléments au total (2401 éléments mutagènes et 1936 éléments non mutagènes). Les graphes sont en moyenne d'ordre 30 et de taille 30. La figure 3.15 illustre un graphe issu de cette base.

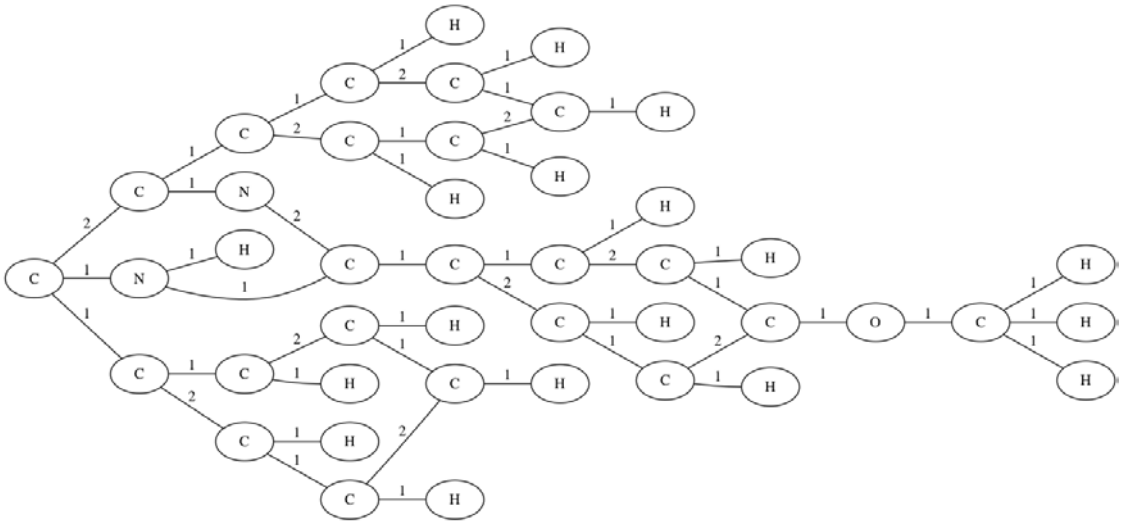


FIGURE 3.15 – Graphe de la molécule $C_{22}H_{18}N_2O_1$. Les sommets portent une étiquette corespondant à l'élément chimique représenté (C, H, N ou O). Les arêtes modélisent les liaisons covalentes ; leurs étiquettes portent leurs nombres.

3.3.2.4 Résumé

Le tableau 3.3 introduit un récapitulatif des différentes caractéristiques importantes.

	♯ classes	♯ graphes (app., valid., test)	♯ att. (sommets)	♯ att. (arêtes)	μ ordres	μ tailles
GREC	22	286, 286, 528	2 num., 1 symb.	1 num, 1 symb.	11	12
Letter	15	750, 750, 750	2 num	0	5	3
Mutagenicity	2	1500, 500, 2337	1 symb.	1 symb.	30	30

TABLE 3.3 – Tableau récapitulatif des caractéristiques des bases. Pour chaque ensemble de graphes sont présentés le nombre (♯ classes) de classes, le nombre d'attributs (♯ att.) que comportent les étiquettes de sommets et d'arêtes, l'ordre et la taille moyens (μ) des graphes.

3.3.3 Caractérisation du paramètre N de notre méthode

3.3.3.1 Représentativité de l'information topologique

Dans cette première expérimentation, nous voulons mettre en évidence l'apport de notre projection de graphe avec un lexique proposant des motifs complexes, *i.e.* constitué de plusieurs arêtes et/ou sommets. Le sondage de graphes, la projection proposée dans [Lopresti et Wilfong, 2003], à laquelle nous comparons notre méthode, n'explore qu'une information topologique très locale et réduite. Nous rappelons que cette méthode utilise le dénombrement des sommets de degré égal comme signature de graphes, engendrant une perte d'information importante lors de l'extraction de la signature d'un graphe. Cette im-

précision est alors répercutée lors de l'utilisation du vecteur, dans une tâche de classification par exemple.

Dans cette expérimentation, nous avons suivi deux objectifs. Dans un premier temps, il s'agit de comparer notre projection avec le sondage de graphe, et donc, valider l'intérêt d'exploiter une information topologique non plus localisée sur les sommets mais sur des motifs plus larges. Dans le même esprit, nous avons aussi voulu évaluer l'impact de l'augmentation du nombre de motifs du lexique. Cette taille est contrainte par le paramètre N qui fixe la taille maximale des motifs du lexique. Ici, compte-tenu de la taille et de l'ordre des graphes des bases choisies, nous avons fait varier N tel que $N \in \{2, 3, 4, 5\}$ (respectivement, la taille du lexique est de 3, 6, 11, 23 motifs)

Les approches sont comparées sur une tâche de classification par plus proches voisins ou $k - ppv$. L'idée sous-jacente de cet algorithme est de faire reposer la classification d'un élément x sur l'ensemble des k éléments les plus proches de x. La décision est faîte selon un vote majoritaire dans cet ensemble. On parle alors de "lazy learning" (apprentissage paresseux) car il n'y a pas d'apprentissage proprement dit ; les exemples d'apprentissage sont simplement mémorisés pour être utilisés lors de la classification. La base de validation permet d'optimiser la valeur de l'hyperparamètre k. La classification est effectuée sur les échantillons d'une base de test indépendante des deux bases d'apprentissage et de validation.

Les tableaux suivants montrent les taux de classification obtenus suivant le processus décrit. Nous avons donc reporté, pour chaque base de graphes, les résultats d'une description locale, le sondage de graphes, et ceux avec la projection que nous proposons, avec un lexique de 3, 6, 11 et 23 motifs.

	GREC	*Letter*	*Mutagenicity*
Sondage de graphe	86.10	48.81	67.26
Projection topologique (3 motifs)	85.99	48.73	65.82
Projection topologique (6 motifs)	90.35	49.55	68.34
Projection topologique (11 motifs)	93.16	49.64	69.89
Projection topologique (23 motifs)	93.36	49.60	70.85

TABLE 3.4 – Tableau de comparaison de résultats de classification (en %) sur les bases *GREC*, *Letter* et *Mutagenicity* par l'utilisation de sondage de graphe et de la projection topologique.

Le tableau 3.4 résume les résultats de classification. En ligne sont présentées les méthodes utilisées, le sondage de graphe et les représentations vectorielles obtenues avec notre méthode de projection pour plusieurs valeurs de N et en colonne les trois bases.

Dans un premier temps, en étudiant la différence entre le sondage de graphe et notre représentation vectorielle en exploitant peu de motifs (N=2) nous pouvons remarquer, pour les bases *GREC* et *Letter*, des taux de classification très proches, environ 0,10%. Nous pouvons en conclure que pour une valeur de N faible notre méthode obtient des résultats quasi équivalent à ceux obtenus par le sondage de graphes. Pour la base *Mutagenicity*, l'écart est plus important, soit environ 2%. Ces scores proches prouvent le manque de prise

en compte de la topologie concernant la méthode de Lopresti et Wilfong.

Le tableau 3.4 montre également l'évolution positive des taux de classification lorsque le nombre de motifs du lexique augmente. En effet, pour les 3 bases, les résultats s'améliorent avec la taille du lexique et dépassent ceux du sondage de graphes lorsque $N \geqslant 3$. Les motifs sont alors plus complexes et permettent de mieux préserver l'information sur la topologie des graphes. Les résultats tendent vers une valeur limite, lorsque la taille des graphes approche celle des motifs du lexique. La représentation vectorielle encapsule alors la quasi totalité de l'information topologique portée par le graphe. L'erreur est alors induite non plus par une perte de l'information topologique lors de la projection des graphes mais par le bruit engendré lors de l'extraction de ces graphes et la caractérisation des objets analysés.

Ces expérimentations révèlent l'importance de prendre en considération l'information topologique dans une méthode de projection de graphes. Nous remarquons que neuf graphes sur dix sont bien classifiés pour la base *GREC*, sept graphes sur dix pour *Mutagenicity* et seulement un sur deux pour la base *Letter*. Ces résultats montrent également que l'information topologique apporte plus ou moins d'informations selon les bases.

3.3.3.2 Évaluation du temps d'extraction des motifs

Nous avons mentionné, dans la sous section 3.2.2.1 l'influence du paramètre N, et donc de la taille du lexique, sur les résultats et sur les temps de calcul. Au vu des résultats précédents, il paraît nécessaire d'étudier les coûts de projection des graphes durant la phase d'indexation.

Nous présentons donc une série d'expérimentations pour analyser les temps de projection sur différents types de graphes (différentes bases). Ceux-ci ont été mesurés pour des lexiques de taille 3, 6, 11 et 23 motifs, afin de conclure sur l'impact de l'évolution de N. De plus, l'utilisation des trois bases présentées précédemment permet aussi de considérer d'autres facteurs, comme la taille et l'ordre des graphes. Les résultats sont issus d'une expérimentation effectuée sur un ordinateur personnel (Intel Core2Duo 2,4 Ghz, 4 Go de RAM), à partir d'une implémentation de l'algorithme en Java, sur les ensembles d'apprentissage des bases *GREC*, *Letter* et *Mutagenicity*

	GREC	Letter	Mutagenicity
Projection topologique (3 motifs)	4.53	6.88	22.43
Projection topologique (6 motifs)	5.56	7.92	50.40
Projection topologique (11 motifs)	7.93	9.15	165.75
Projection topologique (23 motifs)	13.03	9.82	770.01

TABLE 3.5 – Tableau de comparaison des temps d'extraction en secondes sur les bases *GREC*, *Letter* et *Mutagenicity* en fonction de la taille du lexique.

L'analyse du tableau 3.5 et du graphique 3.16 associé nous mène à plusieurs conclusions. Il existe bien évidemment une relation entre le temps d'extraction et la taille du lexique. Nous pouvons cependant observer que cette évolution est également dépendante de la structure des graphes. En effet, nous constatons une certaine stabilité entre les temps

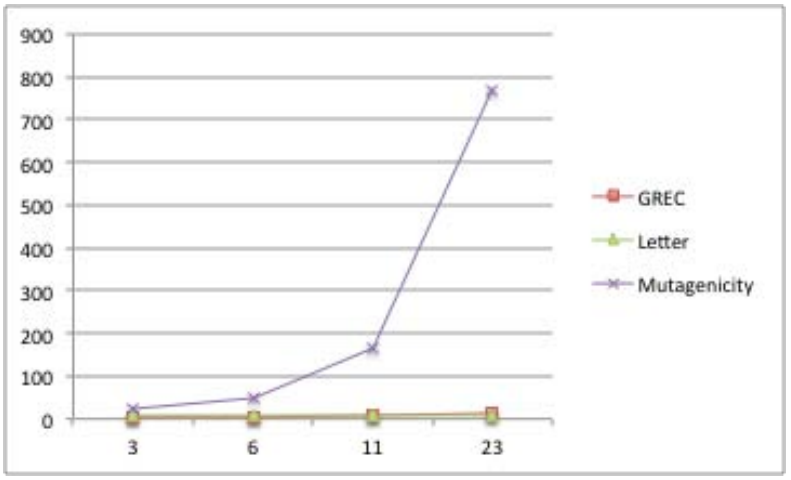


FIGURE 3.16 – Évolution des temps d'extractions de la projection de l'ensemble des bases *GREC*, *Letter* et *Mutagenicity* en fonction de la taille du lexique.

d'extraction pour un lexique de taille 11 et de taille 23 pour la base *Letter*. Les graphes composant cette base, sont d'ordre et de taille faible ; les occurrences de grands motifs sont donc peu nombreuses, contribuant ainsi à stabiliser le temps d'extraction. Les temps sur les bases *GREC* et *Mutagenicity* évoluent de façon comparable dans leur forme de type exponentielle. Les échelles de grandeur sont tout de même différentes, conséquence directe du nombre de graphes plus important dans *Mutagenicity* que dans *GREC*. De plus, il existe un rapport 3 entre la taille et l'ordre moyen des graphes constituant ces bases. Cela se répercute également sur la durée de cette phase d'indexation.

3.3.3.3 Bilan de la caractérisation

Il est intéressant de noter que les deux expérimentations précédentes sont influencées par le paramètre N définissant la taille du lexique ainsi que la complexité des motifs. En effet, les premiers résultats nous montrent que les taux de classification augmentent en même temps que N selon une croissance asymptotique. De plus, le temps d'indexation des graphes, *i.e.* le temps de la phase d'extraction des représentations vectorielles, varie également en fonction de N. Cependant, cette évolution tendrait plus vers une croissance exponentielle. La problématique de ces deux observations résulte alors en un ajustement du ratio rapidité/efficacité de la méthode. Pour l'utilisateur, ce ratio consiste à trouver le compromis entre l'expressivité de notre représentation vectorielle et les temps de calcul qui lui sont associés. Nous notons donc que le paramètre N va être la variable d'ajustement pour optimiser le système de classification en fonction des bases, des objectifs et des contraintes.

3.3.4 Performance de notre projection de graphes

3.3.4.1 Étude des performances de la representation vectorielle en classification

Afin d'évaluer les performances de la représentation vectorielle que nous présentons par rapport à l'état de l'art, nous proposons de comparer les résultats d'une classification en utilisant une distance d'édition de graphe et notre méthode. Nous rappelons que la distance d'édition de graphe est une mesure de similarité entre deux graphes basée sur les coûts de modifications nécessaires pour transformer le premier graphe afin qu'il corresponde au deuxième. Un coût est affecté à chaque opération de suppression ou d'ajout d'une arête ou d'un sommet. La distance entre les deux graphes est une somme de ces coûts. Afin d'assurer une certaine parité entre notre approche entièrement topologique et la méthode de calcul de la distance d'édition, l'opération de modification des étiquettes des graphes n'est pas effective. Les paramètres (coût d'ajout et de suppression) sont choisis de manière empirique sur une base de validation afin d'assurer le meilleur taux de bonne classification. Le tableau 3.6 présente ces résultats.

	GREC	Letter	Mutagenicity
Distance d'édition de graphe	88.00	58.67	65.80
Projection topologique ($N = 2$)	85.99	48.73	65.82
Projection topologique ($N = 5$)	93.36	49.60	70.85

TABLE 3.6 – Tableau de comparaison de résultats de classification (en %) sur les bases *GREC*, *Letter* et *Mutagenicity* par l'utilisation de la distance d'édition de graphe et de la projection proposée.

Le tableau 3.6 nous montre de meilleurs résultats pour les bases *GREC* et *Mutagenicity*. Nous supposons que le mauvais résultat observé sur la base *Letter* est engendré par la taille et l'ordre des graphes de cette collection. Les différences de topologie entre ces graphes sont alors plus restreintes et entraînent une sensibilité accrue aux bruits divers et variés.

3.3.4.2 Étude de l'influence du classificateur

Dans un système de reconnaissance de formes, le choix du classificateur est important et de ce fait peut influencer la qualité des résultats [Caruana et Niculescu-mizil, 2006]. En effet, selon les caractéristiques des données, certaines méthodes obtiendront d'excellents résultats et d'autres non. Il est ainsi difficile d'affirmer qu'un classificateur sera le meilleur dans tous les contextes. Afin de vérifier la neutralité de ce choix vis-à-vis des résultats obtenus précédemment, nous soumettons notre approche à différents classificateurs. Notre choix s'est porté sur des outils reconnus et représentatifs de la littérature dans le domaine de la classification que nous avons présenté en 2.5 :

- Le k-ppv est un classificateur simple utilisé avec une distance euclidienne.
- Un Perceptron Multicouche est un classificateur de type réseaux de neurones. Le MLP (pour MultiLayer Perceptron) est une extension du perceptron unicouche.

– Une Machine à Vecteurs Support ou SVM (Support Vector Machine) est un classifieur à noyaux. Nous avons choisi pour cette étude le noyau gaussien
– Une forêt aléatoire ou RF est un agrégat de classifieurs simples, les arbres de décisions, générés en injectant de l'aléatoire avant ou pendant la construction de l'arbre.

Pour cette expérimentation, nous avons suivi le processus de classification appelé validation croisée. Les bases d'apprentissage, de validation et de test sont unifiées au sein d'une unique base. Dans un deuxième temps, cette base est découpée aléatoirement de la façon suivante : 80% utilisés pour l'apprentissage, 20% pour le test. Pour un résultat, ce processus est appelé 5 fois afin d'obtenir, pour chaque classificateur et pour chaque base, 5 résultats de classification.

	GREC	Letter	Mutagenicity
k-ppv	93.18 (2.11)	49.64 (1.84)	70.00 (1.39)
MLP	89.45 (2.93)	50.09 (0.54)	68.76 (1.97)
SVM	93.27 (1.91)	50.62 (0.81)	71.50 (2.51)
RF	93.18 (2.34)	49.96 (1.50)	71.82 (1.05)

TABLE 3.7 – Tableau de comparaison de résultats (moyennes et écarts-types) de classification (en %) sur les bases *GREC*, *Letter* et *Mutagenicity* avec différents classifieurs : k-ppv, MLP, SVM et RF.

Le tableau 3.7 montre les moyennes du taux de bonne classification et les écarts-types pour chaque base. Nous signalons également que les résultats affichés sont obtenus après optimisation des paramètres de chaque classificateur. Les expérimentations effectuées pour cette optimisation sont disponibles dans l'annexe A. Comme nous l'avions supposé, notre approche présente une certaine stabilité. Aucun des classificateurs proposés ne montre un retrait important. Nous pouvons donc en conclure que, pour ces graphes, notre projection est adaptée à tous les types de classificateur.

3.3.4.3 Étude de l'influence de la distance

En étudiant le classificateur k-ppv (*cf.* section 2.5), nous avons remarqué qu'un des paramètres de l'algorithme était la distance choisie pour évaluer la dissimilarité des individus. Ce critère peut donc influer sur les taux de classification. Une distance inadaptée aux données provoque une baisse des résultats. Pour vérifier que les bons résultats présentés précédemment ne dépendent pas d'un choix de distance heureux, nous présentons plusieurs résultats obtenus par une classification par un k-ppv avec différentes distances suivant un protocole de validation croisée. Nous avons vu que les projections de graphes permettent l'utilisation des outils définis dans un espace vectoriel, notamment les mesures de similarités. Les distances utilisées sont donc issues de la littérature :

– distance de Manhattan (1-distance) : $\sum_{i=1}^{n} |x_i - y_i|$
– distance euclidienne (2-distance) : $\sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$

– distance de Manhattan normalisée : $\sum_{i=1}^{n} \frac{|x_i - y_i|}{\sigma_i^2}$

– distance euclidienne normalisée : $\sqrt{\sum_{i=1}^{n} \frac{|x_i - y_i|^2}{\sigma_i^2}}$

– distance de Tchebychev (∞-distance) : $\lim_{p \to \infty} \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} = \max_{1 \leq i \leq n} |x_i - y_i|$

		GREC	Letter	Mutagenicity
Manhattan	non-normalisée	92.83 (1.84)	49.55 (1.42)	69.42 (1.24)
	normalisée	93.07 (1.61)	49.60 (1.40)	70.06 (1.16)
Euclidienne	non-normalisée	93.04 (1.80)	49.55 (1.42)	69.13 (1.35)
	normalisée	93.16 (1.55)	49.64 (1.41)	69.89 (1.29)
Tchebychev		93.04 (1.57)	49.62 (1.42)	69.76 (1.36)

TABLE 3.8 – Tableau de comparaison de résultats (moyennes et écarts-types) de classification (en %) sur les bases *GREC*, *Letter* et *Mutagenicity* avec différentes distances : Manhattan (normalisée ou non), euclidienne (normalisée ou non) et Tchebychev.

Le tableau 3.3.4.3 démontre la faible influence du choix de la distance sur les résultats de classification obtenus.

3.4 Synthèse

Pour finir ce chapitre, nous dressons un bilan reprenant les points clefs de notre méthodes pour projeter la topologie des graphes. Notre méthode s'inspire à la fois de la philosophie des techniques de sondage de graphes ([Lopresti et Wilfong, 2003]) et de l'utilisation de sacs de sous-graphes ([Barbu *et al.*, 2005]). Nous proposons d'intégrer l'information topologique du graphe en dénombrant des motifs issus d'un lexique. Ces motifs peuvent être constitués de plusieurs sommets et/ou de plusieurs arcs.

– Ces motifs sont issus d'un lexique construit indépendamment des graphes à projeter. De ce fait, la méthode peut être appliquée sans modifications lourdes. L'unique paramètre de notre méthode est la taille maximum des motifs N. Les résultats des expérimentations 3.3.3.1 et 3.3.3.2 nous ont montré que la performance de notre méthode ainsi que le temps d'extraction d'un graphe découlait de ce paramétrage. La **génericité** de la méthode que nous proposons est donc effective. Nous avons également constaté que notre projection n'était pas attachée à l'utilisation d'un type de classificateur ou à une distance.
– Notre hypothèse concernant la **représentativité** est elle aussi validée. En effet, les résultats obtenus montrent l'interêt de ne pas négliger l'information topologique dans une méthode de projection. Notre choix de s'appuyer sur un lexique contenant des motifs complexes, *i.e* des graphes de taille et d'ordre supérieur à 1, est donc pertinent. Dans la majorité des cas, cela se traduit par des taux de classification meilleurs qu'une méthode référence telle que la distance d'édition de graphes.

– La projection a une **complexité** importante, due à la recherche des motifs du lexique dans le graphe à représenter. Toutefois, notre projection est optimisée grâce au support du réseau utilisé pour définir le lexique. Il permet de limiter la recherche des occurences des motifs en testant uniquement la présence de quelques arcs à chaque itération. De plus, l'augmentation du temps de calcul n'est pas critique dans notre contexte puisque les projections des graphes de la base d'apprentissage peuvent être effectuées au préalable lors d'une phase d'apprentissage hors-ligne.

Nous avons donc proposé une nouvelle méthode pertinente pour projeter l'information topologique d'un graphe. Or, nous avons pu observer que les graphes utilisés en reconnaissance structurelle des formes comportent également beaucoup d'informations contenues dans les étiquettes portées par les sommets ou par les arêtes. En suivant la ligne directrice de la technique exposée dans ce chapitre, nous avons enrichi notre projection en offrant la possibilité d'intégrer, en plus de l'information topologique, une description de l'information véhiculée par l'étiquetage des graphes. Nous exposons maintenant cette proposition dans le chapitre suivant.

Chapitre 4

Intégration de l'information relative à l'étiquetage des graphes

> Tous les progrès sont précaires, et la solution d'un problème nous confronte à un autre problème.
>
> Martin Luther King

4.1 Introduction

L'information que doit véhiculer le graphe va au-delà de ce que permet de représenter sa topologie et il devient alors nécessaire d'ajouter des données supplémentaires pour caractériser plus finement les objets. Cela est possible en associant des étiquettes aux arêtes et aux sommets du graphe.

Ainsi, dans la suite, nous définirons une étiquette comme un vecteur de caractéristiques englobant plusieurs informations descriptives, comparables aux vecteurs de caractéristiques utilisés par les approches statistiques. Nous rappelons que nous notons respectivement l_{v_i} et l_{e_i} les étiquettes associées au sommet (resp. l'arête) i. Chaque étiquette est définie dans un ensemble A d'attributs, et chaque attribut $a \in A$ prend une valeur dans un ensemble de valeurs W. Chaque étiquette peut donc être décrite par un vecteur tel que $l = (a_1, \ldots, a_p)$ est une étiquette qui possède p attributs. Un sommet peut donc avoir un ou plusieurs attributs et de la même manière, une arête peut également comporter plusieurs attributs.

Les informations portées par les attributs sont souvent variées. Elles peuvent être de nature statistique (observations de points, distributions, ...), géométrique (distance, angle, courbures) ou bien caractériser des positions (absolues ou relatives) ou des indices visuels (couleur). Deux types d'attributs peuvent être distingués. Une information dite **qualitative** est une information qui définit une donnée observée non mesurable et non hiérarchisable. Elle caractérise une donnée en définissant une propriété. A l'inverse, une information qui provient de mesures ou de calculs est dite **quantitative**, la donnée évalue alors une

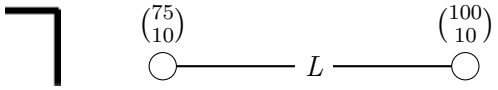
grandeur, au sens mathématique du terme. En fonction des informations qu'ils véhiculent, les attributs peuvent être de natures numériques ou symboliques :

- les attributs **numériques** sont par définition non dénombrables lorsqu'ils appartiennent à un ensemble infini, tel que $\mathbb{R}$ ou $\mathbb{N}$, rendant impossible tout dénombrement. Ils servent à caractériser une mesure physique, un calcul d'angle, un ratio, etc.
- les attibuts **symboliques** sont issus d'un alphabet fini. Le terme nominal est aussi employé pour les désigner. Il sont utilisés pour décrire une information qualitative telle qu'une couleur, une lettre, la nature d'une jonction, etc.

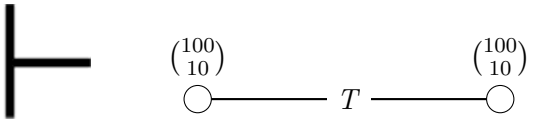
La définition de la nature est liée au contexte de l'utilisation de ces graphes et à l'objet à décrire. Suivant la problématique, la description d'une texture peut être symbolique, et dans ce cas caractérisée par un adjectif parmi n (*lisse, marbrée, tachetée, régulière...*), ou numérique (*filtre de Gabor, matrice de co-occurence...*). Toutefois, les attributs numériques peuvent, dans la majorité des cas, être transformés en attributs symboliques à l'aide de méthodes de partitionnement et de discrétisation par exemple. Notre approche utilise ces techniques, que nous analyserons ultérieurement dans ce chapitre. Notons également que certains attributs numériques sont en réalité nativement composés de valeurs discrètes, il s'agit alors de valeurs **quantiques**. Dans ce cas là, ils seront traités comme des valeurs symboliques.

En analyse d'images et reconnaissance des formes, les graphes étiquetés sont très souvent utilisés et posent problème aux méthodes de comparaison de graphes classiques (recherche d'isomorphismes) dès lors qu'elles rencontrent plus qu'un attribut symbolique ou des attributs numériques continus. Seules, les méthodes inexactes restent alors pertinentes mais leur mise en place devient d'autant plus difficile que le nombre d'attributs augmente. De la même manière, la diversité de types d'attributs (mélange de symboliques et de numériques) sur les arêtes et les sommets augmente la difficulté et donc très peu de méthodes sont capables de traiter de tels graphes à notre connaissance (cf. chapitre 2).

Les figures 4.1(a) et 4.1(b) montrent deux dessins, respectivement un symbole L et un symbole T, représentés par deux graphes étiquetés.



(a) Symbole L et sa représentation sous forme de graphe étiqueté.



(b) Symbole T et sa représentation sous forme de graphe étiqueté.

FIGURE 4.1 – Deux extraits de symboles de la base *GREC* et leurs représentations sous forme de graphe.

Le choix a été fait de représenter les traits par des sommets, avec une étiquette comportant deux attributs numériques : la longueur et la largeur du trait. Les arcs représentent les liens existants entre ces traits avec pour étiquette un attribut symbolique : le type de jonction existant entre les traits. Nous pouvons donc voir sur ces exemples didactiques la variété des étiquettes, autant dans leur type (symbolique ou numérique) que dans leur dimension.

De manière générale, un graphe dépend largement de la méthode d'extraction utilisée. Il existe alors une multitude de représentations possibles pour une même forme. Dans cette thèse, nous proposons une méthode de projection de graphe se voulant générique. Il est alors important qu'en plus de l'information topologique, notre méthode permette la projection de l'information portée par les étiquettes des sommets et des arêtes en imposant un minimum de contraintes sur leurs types et leurs dimensions.

4.2 Nos propositions

4.2.1 Introduction

Afin de compléter notre représentation et de pouvoir l'utiliser sur des graphes étiquetés, nous avons étendu le dénombrement de la fréquence des motifs topologiques au dénombrement de la fréquence des motifs étiquetés. La conservation de la relation entre l'information structurelle et l'information portée par les étiquettes est primordiale, puisque l'essence même de l'utilisation des graphes en reconnaissance de formes. Afin de conserver cette relation, nous avons choisi de ne pas les séparer lors de la projection (comme le font la plupart des méthodes). La solution retenue consiste à compléter notre représentation topologique par un dénombrement des étiquettes présentes pour chacun des motifs préalablement extraits selon l'algorithme vu dans le chapitre précédent.

Conserver les étiquettes lors de la projection n'est pas chose simple. Les aspects suivants constituent les principales difficultés à résoudre lorsque l'on tente de mettre en place une méthode suffisamment générique :

- la variabilité des valeurs que peut prendre chaque sommet et chaque arête est difficilement représentable à l'aide d'un ensemble fini de valeurs numériques.
- la répartition, disposition relative des valeurs prises par les étiquettes dans les graphes devrait être traduite dans la projection.
- le simple dénombrement des valeurs prises par les étiquettes n'est donc pas suffisant selon nous, même si cela constitue une première technique souvent exploitée dans les systèmes existants.
- le nombre de combinaisons possibles lorsque l'on analyse les motifs étiquetés rend ce type d'approches inenvisageable surtout lorsque les étiquettes comportent des attributs numériques continus.

Les techniques que nous proposons tentent d'apporter des réponses à ces difficultés en utilisant un compromis entre les méthodes se basant sur un dénombrement des valeurs prises par les étiquettes sans tenir compte de la topologie et les méthodes d'appariement de graphe classiques de trop forte complexité car réalisant une analyse trop localisée de chaque partie des graphes.

La projection que nous proposons repose alors sur la mise en place d'une matrice au lieu d'un vecteur numérique. Les colonnes de la matrice correspondent comme auparavant aux motifs topologiques possibles dans les graphes tandis que les lignes sont dédiées à la description de l'information liée aux étiquettes. Un élément de cette matrice donne alors le nombre d'occurrences du motif portant l'étiquetage spécifié sur la ligne correspondante. La figure 4.2 illustre la structure de la matrice ainsi obtenue.

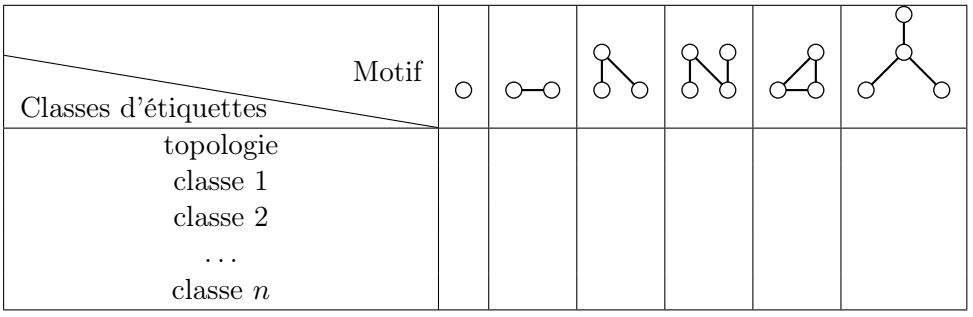


FIGURE 4.2 – Exemple d'un lexique avec un vecteur de motifs topologiques de taille 6 et n classes d'étiquettes.

Comme nous l'avons vu dans le chapitre précédent, les éléments topologiques contenus dans les lexiques sont connus *a priori* et définissent les colonnes de la matrice. Le nombre de colonnes dépendra donc du nombre de motifs que l'on désire considérer (*cf.* chapitre 3). La ligne topologie correspond au vecteur décrit précédemment, *i.e.* une projection de l'information topologique. Indépendante des étiquettes, elle apporte donc une information complémentaire indifférente aux bruits pouvant faire fluctuer les valeurs des attributs. Notons également que la complexité de la projection d'un dans une représentation matricielle (topologie + information des étiquettes) sera du même ordre que la projection dans une représentation vectorielle (topologie uniquement) puisque nous suivrons le même algorithme en ajoutant seulement le dénombrement des étiquettes qui se fera en même temps que le dénombrement des motifs.

La détermination des lignes de la matrice est en revanche beaucoup plus complexe puisque les étiquettes ne sont pas connues *a priori*. Elle n'intervient cependant qu'une seule fois, lors de la phase dite indexation qui est effectuée hors-ligne. Pour définir les lignes de la matrice, il est donc nécessaire d'explorer au préalable le contenu des graphes pour analyser les étiquettes des sommets et arêtes, et finalement en déduire ce qui constituera les lignes de la matrice. La prise en compte des étiquettes lors du dénombrement des éléments (que nous appelons, pour l'instant classes d'étiquettes (figure 4.2)) du lexique nécessite donc de résoudre une problématique supplémentaire qui consiste à définir tous les éléments à prendre en compte lors de la projection. Il peut s'agir par exemple d'inventorier les étiquettes qui sont présentes au moins une fois dans les graphes, pour les arcs et pour les sommets. La détermination des lignes est effectuée à partir de l'examen des étiquettes sur un ensemble d'apprentissage, il est possible qu'une étiquette apparaisse dans un ensemble de test. Il faudra alors tenir compte de cette contrainte en généralisant les classes d'étiquettes.

Différentes solutions sont envisageables pour définir les éléments constituant les lignes de la matrice représentant l'information étiquetée portée par les sommets, les arêtes voire

les motifs (un motif étant constitué de plusieurs sommets et arêtes). Remarquons que les classes d'étiquettes seront différentes selon qu'elles soient portées par les sommets ou par les arêtes. En effet, les attributs ne contenant pas les mêmes informations, il convient de proposer plusieurs classes d'étiquettes pour les sommets et également plusieurs classes pour les arêtes. Notons aussi que toutes ces méthodes nécessitent de disposer d'une ''base d'apprentissage" permettant d'obtenir des informations sur les étiquettes présentes dans les graphes à encapsuler. Nous présentons dans les sous-sections suivantes les différentes techniques que nous avons mises en place pour générer ces classes d'étiquettes et réaliser leurs dénombrements pour chaque motif topologique.

4.2.2 Détermination des classes d'étiquettes

La première méthode que nous avons envisagée pour déterminer les classes d'étiquettes consiste à tenter d'énumérer toutes les étiquettes possibles pour ensuite les dénombrer dans les différents motifs correspondant aux colonnes de la matrice. Cette énumération aurait été envisageable si les graphes ne comportaient que des attributs symboliques, ce qui n'est malheureusement que très rarement le cas en analyse d'images ou en reconnaissance des formes.

Pour rendre néanmoins possible cette approche la première méthode que nous avons mise en place procède à une discrétisation des attributs numériques composant les étiquettes numériques des sommets et des arêtes pour ensuite dénombrer pour chaque motif la manière dont se combinent les valeurs prises par les attributs. Cette technique sera décrite dans la section 4.2.3.

Nous avons également envisagé une seconde technique utilisant une vision plus globale des étiquettes. Cette approche propose de réaliser un partitionnement des étiquettes afin d'obtenir des classes d'étiquettes pour les sommets et des classes d'étiquettes pour les arêtes. Cette approche est présentée section 4.2.4.

4.2.3 Approche basée sur la discrétisation des attributs numériques et la combinaison

L'idée sous-jacente de cette méthode est de dénombrer toutes les étiquettes potentiellement présentes dans un graphe en listant toutes les combinaisons des valeurs possibles d'attributs possibles pour les sommets et les arêtes. Dans ce cas, le nombre de classes sera défini par le nombre d'attributs et par la taille du domaine des valeurs possibles de ces attributs. Si une étiquette possède p attributs et que chaque attribut peut prendre k_p valeurs, alors le nombre de classe sera $\prod_{i=1}^{p} k_i$.

En partant du principe qu'aucune connaissance *a priori* (hormis la base d'apprentissage) n'est disponible, la première étape consiste à lister toutes les valeurs d'attributs qui sont présentes au moins une fois dans la base d'apprentissage, pour les sommets et pour les arêtes. À la fin de cette étape, nous obtenons donc p listes de valeurs possibles ; chacune de ces listes est de taille k_p.

Ce dénombrement des valeurs possibles est envisageable pour les attributs symboliques

et revient à constituer l'alphabet des valeurs nominales que peut prendre l'attribut. En revanche, pour les attributs numériques, une étape de discrétisation est indispensable pour ramener les attributs numériques à des attributs symboliques correspondant aux intervalles obtenus. La manière de discrétiser les attributs numériques puis de les combiner pour construire les classes d'étiquettes (lignes de la matrice) est décrite dans les sections suivantes.

4.2.3.1 Techniques de discrétisation

En mathématiques, la discrétisation fait référence aux fonctions permettant un passage de données continues en données discrètes. Utilisée dans le domaine informatique, cette définition est élargie. En effet, les technologies numériques ne permettent pas, par définition, de traiter des éléments continus. La discrétisation permet alors d'attribuer une valeur symbolique, appartenant à un ensemble fini, à une valeur numérique. Ces techniques sont alors très utilisées pour passer d'une information analogique à une information numérique. Il est aussi possible de parler de quantification. Sur un ensemble de données réelles, la discrétisation se traduit par un étiquetage de l'ensemble des valeurs. Cet étiquetage est réalisé par un découpage de l'ensemble en n classes dont les bornes définissent les limites d'une étiquette. Les valeurs sont ensuite affectées à une unique étiquette.

L'état de l'art sur la discrétisation, utilisé comme pré-traitement des données nous montre que peu d'études ont été initiées pour évaluer son impact sur les algorithmes d'apprentissage [Weiss et Kulikowski, 1991]. Pourtant, l'importance de cette étape est réelle. En effet, elle permet à partir d'un ensemble d'apprentissage d'estimer les bornes de discrimination entre les individus. Elle conditionne alors le résultat final, un mauvais choix entraînant automatiquement une perte d'information [Celeux et Robert, 1993]. Il existe néanmoins quelques travaux de référence, [Dougherty *et al.*, 1995] et [Rabaséda *et al.*, 1996] sur lesquels Rakotomalala [Rakotomalala, 1997] s'appuie pour distinguer 3 thèmes autour de la discrétisation :

- **supervisé / non-supervisé** : dans le cas de la discrétisation supervisée, une vérité terrain est disponible avec un ensemble d'étiquettes connues pour certaines valeurs. La discrétisation est donc effectuée selon cette connaissance *a priori* en tenant compte des valeurs et de leur distance. Pour une discrétisation non-supervisée seules les similarités entre les valeurs sont considérées. Plus précisément, la discrétisation supervisée privilégie les découpes entre valeurs que l'on sait appartenir à des intervalles différents. La discrétisation non-supervisée se base sur d'autres critères pour le découpage, comme par exemple la proximité des objets.
- **global / local** : pour la discrétisation globale, les intervalles sont construits et fixés avant de réaliser la construction de la discrétisation. Au contraire, en discrétisation locale, le découpage est réalisé au fur et à mesure de sa construction. Ainsi à chaque étape de construction, on divise les ensembles de valeurs concernées en sous-ensembles.
- **statique / dynamique** : la discrétisation statique est la stratégie la plus utilisée. Il s'agit de traiter les attributs indépendamment les uns des autres. A l'inverse, la discrétisation dynamique intègre l'information de toutes les valeurs simultanément pour construire les intervalles, ce qui permet la prise en compte d'une éventuelle

corrélation entre elles.

Actuellement, la méthode décrite dans [Fayyad et Irani, 1993] apparaît comme une des plus efficaces. Elle n'est cependant pas applicable à notre problème puisqu'il s'agit d'une approche supervisée et nécessite donc une base d'apprentissage avec une vérité terrain. Dans notre cas, la discrétisation doit être non-supervisée puisque nous ne disposons pas d'exemple de partition d'attributs.

4.2.3.2 Méthode choisie

Il semble qu'une méthode globale correspondrait au mieux à notre besoin car il est important de pouvoir définir le nombre d'intervalles désirés pour chaque attribut. Nous avons vu que la taille de la matrice dépendait du nombre de classes d'étiquettes, donc du nombre d'intervalles obtenus. En fixant ce nombre *a priori*, nous pourrons ainsi contrôler directement la taille de la matrice.

Nous proposons de définir les intervalles en nous basant sur l'histogramme des valeurs que peuvent prendre les attributs. Nous rappelons que cet histogramme est l'ensemble ordonné des valeurs d'un attribut. La discrétisation est réalisée par la subdivision de l'histogramme en k intervalles équivalents en terme de population. La taille d'un intervalle n est obtenue en divisant la cardinalité de cet ensemble par k. Le découpage est donc, dans un premier temps, effectué dans l'histogramme de toutes les n valeurs. Par cette technique, l'affectation d'une valeur à un intervalle ne tient pas compte de la distance entre une valeur et ses voisines dans l'histogramme. Il se peut que des valeurs soient très distantes des valeurs de l'intervalle auquel elle a été affectée. Au contraire, la subdivision peut intervenir entre deux valeurs très proches, les séparant dans deux intervalles différents. La discrétisation ne sera alors pas optimale dans le sens où il est préférable d'avoir des partitions homogènes avec des valeurs proches. Pour palier cela, nous proposons dans un second temps d'affiner ces intervalles en maximisant la vraisemblance dans ces subdivisions en utilisant la technique dite "leave-one-out". Chaque élément va être classifié en mesurant sa distance par rapport à ses voisins. Il sera alors affectée dans la classe des voisins les plus proches. Cette opération est effectué plusieurs fois selon le principe de la validation croisée. Certaines classes peuvent être alors supprimées si leur population devient nulle. De plus, deux classes pourront fusionner si leur population sont ressemblantes et qu'une union est réalisable en conservant le maximum de vraisemblance. Ainsi, l'entropie entre deux classes, *i.e.* l'empattement d'une classe sur l'autre, sera minimisée. La difficulté de cet algorithme réside dans le choix de k nécessitant d'être fixé au préalable. En effet, ce critère détermine le nombre de subdivisions initiales et influence la recherche des classes voisines d'une valeur, il va donc aussi influencer la distance entre cette valeur et ses voisins.

Les bornes des intervalles sont alors définies pour permettre une discrétisation des attributs numériques d'un graphe. Ces bornes peuvent être conservées et réutilisées pour permettre la projection d'un graphe inconnu suivant la même représentation matricielle. Par exemple, dans un contexte de classification supervisée de graphes, il s'agit des graphes appartenant aux bases de validation et de test.

Afin d'illustrer cette approche, nous proposons de détailler les étapes décrites précédemment pour la projection du graphe représentant le symbole d'une résistance (*cf.* figure 4.3).

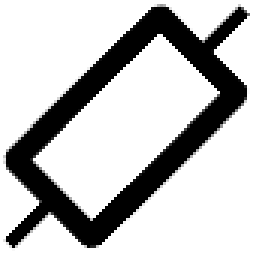


FIGURE 4.3 – L'image d'un symbole (une résistance).

Le graphe (figure 4.4) est construit à partir de l'extraction des primitives bas niveau inspirée de [Ramel *et al.*, 2000]. Le symbole est décomposé en segments qui sont les sommets du graphe étiquetés par leurs longueurs et leurs épaisseurs. Une arête représente l'existence d'une jonction entre deux segments et caractérise le type de cette jonction "T" ou "L". Nous avons donc deux attributs numériques (a_1, a_2) avec $a_1, a_2 \in \mathbb{R}$ qui composent une étiquette l_v des sommets et une étiquette symbolique $b \in \{T, L\}$ pour une étiquette l_e d'arête.

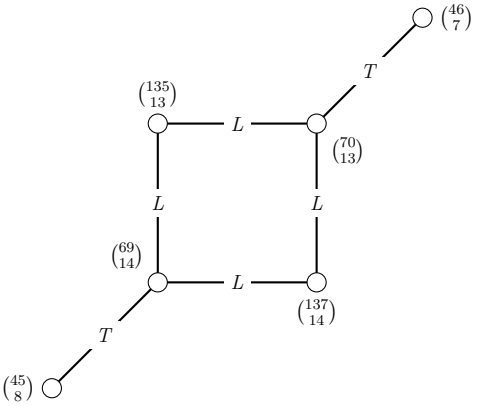


FIGURE 4.4 – Représentation du symbole 4.3 sous forme du graphe étiqueté G.

La première étape consiste donc à lister l'ensemble des valeurs possibles pour tous les attributs. Dans notre exemple, A_1 le multiensemble des longueurs possibles vaut $\{45, 69, 135, 137, 70, 46\}$, A_2 le multiensemble des largeurs vaut $\{8, 14, 14, 13, 13, 7\}$ et B le multiensemble des types de jonction vaut $\{T, L\}$. B étant un attribut nominal, seuls A_1 et A_2 nécessitent une discrétisation. Par notre méthode, deux intervalles sont définis pour chaque attribut tels que $C(A_1) = \{c_1(A_1) = [0, 100], c_2(A_1) =]100, +\inf[)\}$ soit l'ensemble des intervalles pour A_1 et $C(A_2) = \{c_1(A_2) = [0, 10], c_2(A_2) = ([10, +\inf[)\}$ l'ensemble des intervalles pour A_2.

4.2.3.3 Combinaison des attributs

Une fois les attributs numériques discrétisés, la dernière étape de cette approche est la construction des classes d'étiquettes en listant toutes les combinaisons des valeurs attributs à partir des listes apprises à partir de la base d'apprentissage. Une classe d'étiquette correspond alors à une combinaison obtenue.

Prenons par exemple, une étiquette comportant un attribut symbolique dont les valeurs possibles sont A, B, C ou D et un attribut réel discrétisé en trois classes α, β et γ. Pour cette étiquette, nous obtiendrons donc douze combinaisons telles que $\{A, B, C, D\} \times \alpha, \beta, \gamma$. La matrice sera donc constituée de douze lignes qui correspondront aux classes d'étiquettes.

Sur l'exemple utilisé précédemment, la combinaison des attributs discrétisés et symboliques permet de transformer le graphe initial en sa version discrétisée tel que décrit sur la figure 4.5.

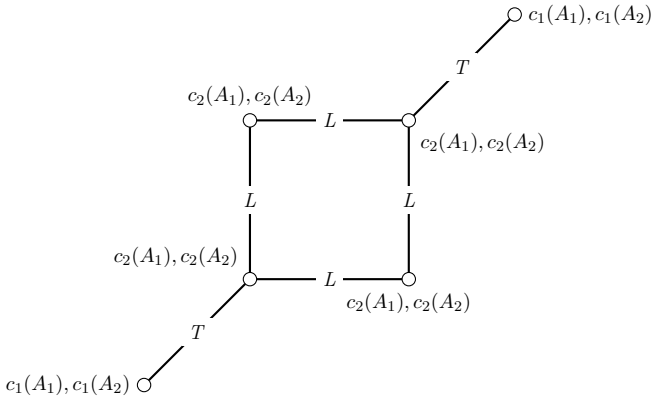


FIGURE 4.5 – Graphe G' isomorphe de G, possédant des étiquettes symboliques obtenues par une combinaison des attributs symboliques et/ou discrétisés.

Puisque tous les attributs des étiquettes sont maintenant de nature symbolique, une projection est alors réalisable. Le résultat de cette projection est reporté par la matrice 4.6.

La matrice illustrée en 4.6 comporte sept lignes : une pour la représentation de l'information topologique (identique à la représentation vectorielle présentée dans le chapitre 3, quatre pour les classes d'étiquettes des sommets et deux pour les classes d'étiquettes des arêtes. Elle est constituée également de six colonnes présentant les six motifs du lexique (avec la taille maximale des graphes N égale à 3). Le premier constat est l'absence de motif *triangle* dans le graphe. Les valeurs de la colonne représentant ce motif sont donc égales à 0. Nous constatons aussi que le graphe ne comporte pas d'étiquette de type $c_2(A_1), c_1(A_2)$, la ligne correspondante ne comporte alors que des 0. Nous observons également que pour un motif donné, la somme des occurrences des étiquettes possibles pour les sommets (respectivement pour les arêtes) est égal au nombre d'occurrences du motif multiplié par le nombre de sommets (respectivement d'arêtes) que possède ce motif. Par exemple, le graphe contient 8 motifs de type *étoile*, le nombre d'étiquettes de sommets est égale à 4 (nombre de sommets dans le motif *étoile*) multiplié par 8 (nombre d'occurrences du motif) ce qui correspond à

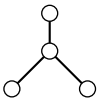
	Étiquettes \ Motif						
	topologie	6	6	8	8	0	2
sommets	$c_1(A_1), c_1(A_2)$	2	2	4	4	0	2
	$c_1(A_1), c_2(A_2)$	2	6	10	16	0	2
	$c_2(A_1), c_1(A_2)$	0	0	0	0	0	0
	$c_2(A_1), c_2(A_2)$	2	4	10	12	0	4
arêtes	T	0	2	4	4	0	2
	L	0	4	12	20	0	4

FIGURE 4.6 – Représentation matricielle de la projection du graphe G'.

la somme des occurrences des étiquettes de type *sommet* pour ce motif $(8 + 4 + 16)$, soit 32.

4.2.4 Approche basée sur le partitionnement de l'espace des étiquettes

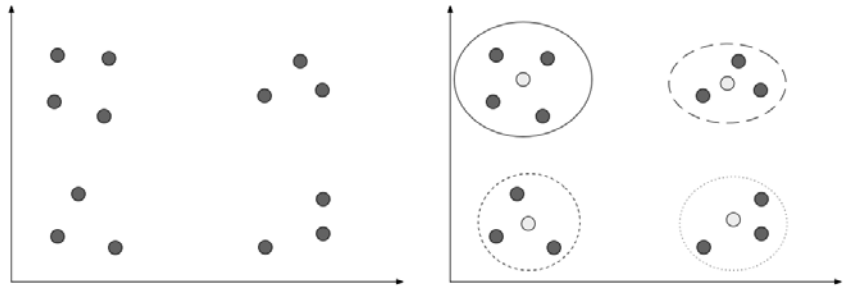
Cette méthode est basée sur une vision plus globale des étiquettes en les considérant comme un vecteur de caractéristiques de taille n. Une étiquette est donc considérée comme un point défini par n coordonnées. La génération des classes est alors possible en utilisant une technique de partitionnement (clustering). Ces méthodes visent à créer des sous-ensembles partageant une homogénéité en divisant l'espace initiale suivant des critères de similarités.

4.2.4.1 Méthodes de partitionnement

Il existe de nombreuses techniques de clustering. Nous indiquons deux références ([Jain *et al.*, 1999], [Xu et Wunsch, 2005]) qui présentent un état de l'art sur les principales méthodes utilisées. Dans ces approches, deux problématiques apparaissent couramment : le choix du nombre de classes et leurs frontières. Une méthode performante aura pour objectif de minimiser une distance intra-classe pour obtenir une homogénéité forte entre les individus de cette classe et de maximiser les distances inter-classes pour atteindre une différenciation optimale entre les classes. Une distance intra-classe faible indique une variance également faible au sein de la classe. Ainsi, tous les points lui appartenant seront proches, leur regroupement sous une même étiquette est donc judicieux. La distance inter-classe régie la variance entre les centres des classes et donc la discrimination entre ces classes. Lorsque la distance inter-classe est faible, le risque de mal classer un point est fort.

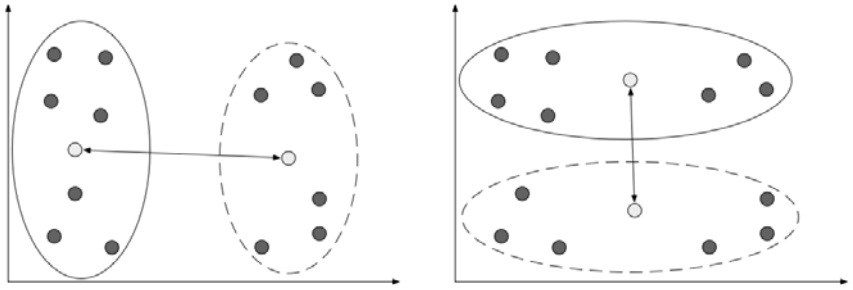
La figure 4.7 illustre ces propos. La figure 4.7(a) nous montre un ensemble de points dans un espace à deux dimensions que nous cherchons à partitionner. La première problématique est le choix du nombre de partitions. Nous présentons des cas où le partitionnement a été effectué visuellement. Les figures 4.7(b) et 4.7(c) présentent respectivement un partitionnement à 4 classes et 2 classes. Pour chacune des figures, les partitions sont représentées par leurs frontières (tracés en pointillés) et leurs centres de gravité. Nous pouvons donc

observer que les partitions semblent compactes (distance intra-classe faible) et les centres de gravité suffisamment éloignés pour éviter toute confusion de classe (distance inter-classe forte). Ces deux partitionnement semblent donc satisfaire les propriétés évoquées précédemment. Le choix entre 2 et 4 partitions sera donc dépendant de son utilisation et il est impossible de considérer le meilleur d'entre eux sans d'autres critères comme la précision ou la performance. Les figures 4.7(c) et 4.7(d) montrent deux partitionnement en 2 classes différents. Nous voulons ici insister sur le choix des frontières. Contrairement à la première figure, la deuxième montre une distance inter-classe faible et une distance intra-classe forte. la partitionnement est alors moins pertinent que le premier exemple.



(a) Un ensemble de points.

(b) Exemple de partitionnement en 4 classes.



(c) Exemple de partitionnement correct.

(d) Exemple de partitionnement incorrect.

FIGURE 4.7 – Exemple de clustering sur un ensemble de points dans $\mathbb{R}$.

4.2.4.2 Méthode choisie

L'objectif de notre partitionnement est d'atteindre un regroupement des étiquettes afin de constituer des classes d'étiquettes homogènes ou tout du moins présentant une certaine similarité. Comme pour le cas de la discrétisation des attributs, afin de pouvoir maitriser la taille de la matrice nous avons opté pour une méthode nous permettant de définir manuellement le nombre de classes désirées. Nous avons donc choisi l'algorithme des k-moyennes. Dans [Min, 2009], les auteurs éprouvent l'utilisation de cette technique pour effectuer une discrétisation globale. En effet, l'algorithme permet de créer des classes dans un ensemble de données continues et ensuite d'affecter un attribut symbolique à chacune

de ces classes ; les données définies dans $\mathbb{R}$ sont donc réduites à un espace d'attributs symboliques. Dans le même esprit, nous allons utiliser l'algorithme des K-moyennes en l'appliquant successivement sur les étiquettes des sommets puis sur les étiquettes des arêtes.

L'algorithme des k-moyennes L'algorithme des k-moyennes est un algorithme couramment utilisé en analyse de données décrit dans [MacQueen, 1967]. Il permet de partitionner une collection d'objets en K classes, K étant un nombre fixé par l'utilisateur. On supposera dans la suite que nos étiquettes l_i ($1 \leq j \leq N$ avec N le nombre d'étiquettes) sont représentés sous forme de vecteurs. L'algorithme des k-moyennes se déroule de la façon suivante :

1. Choisir K étiquettes au hasard parmi les étiquettes des graphes de la base d'apprentissage. Soient $(\lambda_1, \ldots, \lambda_K)$ les étiquettes ainsi obtenues. $(\lambda_1, \ldots, \lambda_K)$ sont les représentants de K classes $(C_1, ..., C_K)$ qui sont pour l'instant vides.
2. Affecter chaque étiquette de la collection à l'une des classes en fonction du représentant le plus proche :
$$argmin_{k,1\leq k\leq K} d(l_i, \lambda_k)$$
où d est une distance ou une similarité entre étiquettes.
3. Calculer de nouveaux représentants pour les classes. Ces nouveaux représentants correspondent à la moyenne des objets de la classe :
$$\forall k, 1 \leq k \leq K, \lambda_k = \frac{1}{|C_k|} \sum_{i, l_i \in C_k} l_i$$
4. Retourner en 2 tant que la différence entre les anciens et les nouveaux représentants est supérieure à un seuil fixé.

Nous remarquons que cet algorithme repose sur le calcul d'une distance entre deux étiquettes. Nous avons vu que les attributs composant ces étiquettes pouvaient être de nature différente et donc de possibles différences d'échelles entre les valeurs d'attributs (la différence d'échelle peut être caractérisée comme la différence entre la plus petite et la plus grande valeur de l'attribut). Cette contrainte est à prendre en compte afin de ne pas introduire de biais lors du calcul de distance. Nous utilisons donc la distance euclidienne normalisée, *i.e.* la distance euclidienne divisée par l'écart-type de l'attribut, afin de réduire les différences entre les échelles des différents attributs. La distance entre les attributs symboliques est égale à 0 si les valeurs sont égales, 1 sinon.

Nous avons utilisé une implémentation des K-moyennes présentée dans [Arthur et Vassilvitskii, 2007]. Les auteurs proposent d'optimiser la vitesse d'exécution de l'algorithme en choisissant des étiquettes initiales de manière probabiliste afin d'accélerer la convergence des centroïdes.

Contrairement à la technique de discrétisation, les classes d'étiquettes sont ici définies par leurs centroïdes. Nous rappelons que l'apprentissage des classes est faît sur des exemples connus mais que notre méthode de projection de graphes peut également être destinée à être appliquée sur des graphes inconnus, par exemple un graphe requête que l'on voudrait identifier à un graphe (ou une classe de graphes). Les centroïdes sont donc conservés pour pouvoir effectuer un prétraitement identique sur les étiquettes des sommets et des arêtes et ainsi effectuer une projection selon la même matrice. La classe d'étiquette n'est plus

obtenue par un partitionnement mais par une recherche de la classe étiquette de sommet ou d'arête la plus proche.

A titre d'exemple, nous allons maintenant appliquer cette technique pour projeter le graphe G (*cf.* figure 4.4). Nous rappelons que les étiquettes sur les arêtes sont de nature symbolique ($L_E = T, L$) et ne nécessitent donc aucun prétraitement. Ces étiquettes sont directement utilisées comme classes d'étiquette. En revanche, les étiquettes de sommets sont numériques. Nous allons donc appliquer un partitionnement de ces étiquettes afin de créer des classes intégrables dans notre matrice. Pour rappel, nous listons ici l'ensemble de ces étiquettes :

- $l_{v_1} = \binom{31}{8}$
- $l_{v_2} = \binom{79}{14}$
- $l_{v_3} = \binom{107}{14}$
- $l_{v_4} = \binom{105}{13}$
- $l_{v_5} = \binom{80}{13}$
- $l_{v_6} = \binom{31}{7}$

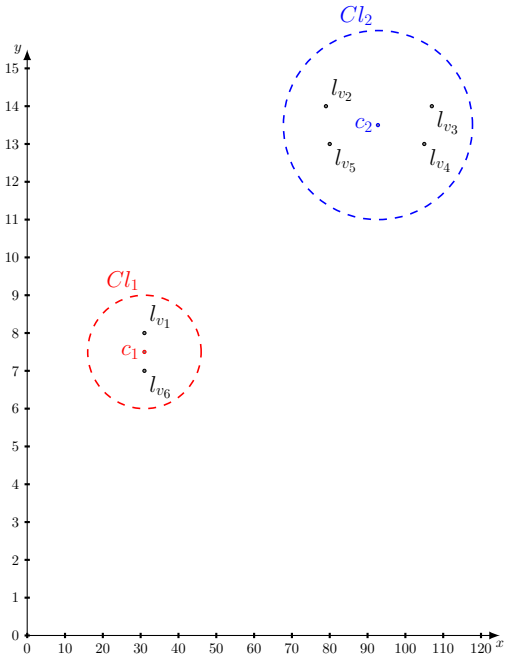


FIGURE 4.8 – Les étiquettes des sommets de G dans un système à deux dimensions.

La figure 4.8 illustre les étiquettes projetées dans un système à deux dimensions. Nous avons reporté le résultat du partitionnement par l'algorithme des k-moyennes. Les cercles rouge et bleu représentent visuellement les classes Cl_1 et Cl_2 dont les centres respectifs sont $c_1(31, 7.5)$ et $c_2(92.75, 13.5)$. Ainsi, les étiquettes sont classées telles que $Cl_1 = \{l_{v_1}, l_{v_6}\}$ et $Cl_2 = \{l_{v_2}, l_{v_4}, l_{v_5}, l_{v_5}, l_{v_1}\}$. Nous pouvons noter qu'un partitionnement à 3 classes aurait également était possible en séparant Cl_2 en deux classes (l_{v_2}, l_{v_5} dans une classe et l_{v_4}, l_{v_5} dans l'autre). La distance intra-classe aurait alors été moins importante mais la distance inter-classe plus grande. Nous notons donc l'importance du choix de K et l'influence

de ce paramètre sur les classes d'étiquettes. Dans le cas d'un processus de classification supervisée, le choix de K sera effectué sur une base de validation afin d'en optimiser sa valeur.

La figure 4.9 fournit le graphe étiqueté après partitionnement des étiquettes.

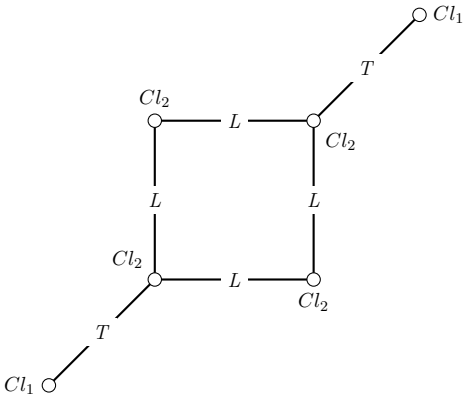


FIGURE 4.9 – Graphe G' isomorphe de G, possédant des étiquettes symboliques obtenues par un partitionnement.

Le résultat de la projection de graphe après partitionnement des étiquettes est présenté par la matrice 4.10.

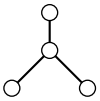
	Motif / Étiquettes	○	○—○				
	topologie	6	6	8	8	0	2
sommets	Cl_1	2	2	4	4	0	2
sommets	Cl_2	4	10	20	28	0	6
arêtes	T	0	2	4	4	0	2
arêtes	L	0	4	12	20	0	4

FIGURE 4.10 – Représentation matricielle de la projection du graphe G'.

Comme pour la méthode de discrétisation présentée en 4.2.3.2, le graphe ne comporte pas de motif *triangle*, la colonne correspondante ne comporte alors que des 0. Nous pouvons notamment remarquer que contrairement à la représentation matricielle obtenue par la méthode précédente, toutes les lignes sont non nulles, ce qui signifie que la classe d'étiquette représentée par cette ligne est présente au moins une fois dans le graphe. Par cette méthode, nous excluons par définition, les lignes composées uniquement de 0. En utilisant une combinaison, l'ensemble des classes d'étiquettes est obtenu en générant de manière synthétique les étiquettes possibles mais pas forcément présentes dans le graphe, contrairement à cette méthode par partitionnement où les étiquettes sont analysées pour extraire les classes d'étiquettes nécessaires à la projection du graphe. La matrice est ainsi plus petite.

4.3 Expérimentations

Comme pour l'encapsulation de l'information topologique, nous avons conduit des expérimentations en classification. Ces résultats ont pour objectif de comparer les deux approches de projection des étiquettes dans un premier temps puis de caractériser notre représentation matricielle dans un second temps. Enfin d'autres expérimentations permettront de situer son efficacité en la comparant avec des approches de la littérature. Les tests ont été menés sur les bases présentées en section 3.3.2. Pour les tests présentés en section 4.3.2 et 4.3.4 , nous avons choisi d'utiliser la méthode du partitionnement des données par un algorithme des k-moyennes, optimisée en suivant les résultats présentés en section 4.3.1.

4.3.1 Comparaison des deux approches

Afin de comparer ces deux approches des projections d'étiquettes des graphes, nous avons réalisé une première expérimentation sur les bases présentées en section 3.3.2. Nous avons également étudié l'influence des paramètres de la discrétisation k ou du partitionnement K . Pour la base *GREC*, nous rappelons que les étiquettes des sommets comportent deux attributs numériques (coordonnées x et y) et un attribut symbolique (type : *corner*, *circle*, *intersection* ou *endpoint*) et que les étiquettes sur les arêtes comportent un attribut numérique (*angle*) et un symbolique (type : *arc* ou *line*). Deux prétraitements seront donc nécessaires pour créer les classes d'étiquettes nécessaires à la projection des graphes, un pour les étiquettes des sommets et un autre pour les arêtes. Les graphes de la base *Letter* ne comportant que des étiquettes sur les sommets (coordonnées x et y), un seul prétraitement est donc nécessaire. La base *Mutagenicity* ne figure pas ici puisque les étiquettes des graphes sont uniquement symbolique et ne nécessitent donc aucun prétraitement. Nous avons fixé arbitrairement le lexique à 11 motifs. Nous avions vu que ce choix montrait de bonnes performances tout en conservant un temps de calcul acceptable.

Nous avons, dans un premier temps, utilisé le première méthode présentée, *i.e.* une discrétisation sur les attributs numériques suivie d'une combinaison des classes d'intervalles obtenues. Nous avons ensuite évalué cette approche dans le cadre d'une classification par plus proches voisins. L'ensemble des bases a été séparé en trois ensembles. Les deux premiers sont utilisés pour l'apprentissage et pour le choix de k. Les résultats présentés sont issus du troisième. Les attributs des étiquettes numériques sont discrétisés en 2, 3 et 4 intervalles.

	k		
	2	3	4
GREC	94.89	95.83	94.69
Letter	85.60	91.33	88.90

TABLE 4.1 – Étude de l'influence de la discrétisation des attributs numériques sur les résultats de classification (en %) des graphes des bases *GREC* et *Letter*.

Le tableau 4.1 montre les résultats de cette expérimentation. Pour la base *GREC*, les meilleurs résultats sont obtenus pour une discrétisation des attributs numériques en 3

intervalles. Pour les sommets, deux attributs numériques sont concernés, donc 9 combinaisons possibles. Il faut également combiner ces 9 classes avec les 4 valeurs de l'attribut symbolique (type), soient 36 classes possibles pour les étiquettes des sommets. L'attribut numérique des arêtes est aussi discrétisé en 3 classes ; la combinaison avec l'attribut symbolique (2 valeurs possibles) donne 6 classes d'étiquettes pour les arêtes. Le nombre total de classes d'étiquettes est donc de 42 auquel il faut rajouter le vecteur de la projection sans étiquettes. La matrice comporte donc $43 \times 11 = 473$ éléments. Pour la base *Letter*, le meilleur résultat est également obtenu en discrétisant les deux attributs des étiquettes des sommets en 3 classes. Nous obtenons ainsi une matrice de taille 11×10. Nous remarquons une certaine stabilité dans les résultats de la base *GREC* ; l'écart entre les taux obtenus avec différentes valeurs de k est plutôt faible. Il est, en revanche, plus important pour la base *Letter*, environ 6 %.

L'utilisation de l'algorithme des K-moyennes nécessite comme entrée le nombre de partitions voulues. Nous avons donc fait varier ce paramètre de 2 à 50 aussi bien pour les étiquettes des sommets (K_1) que des arêtes (K_2). Le tableau 4.2 présente les résultats obtenus sur une tâche de classification par plus proches voisins par validation croisée. Le choix des centroïdes de départs pouvant influencer le partitionnement, nous avons effectué la moyenne sur 5 itérations pour chaque paramètre afin de nous assurer d'une certaine stabilité. Le nombre de résultats étant important, nous présentons uniquement le meilleur taux, le plus faible ainsi que la moyenne des taux obtenus. Pour chaque résultat, nous indiquons le nombre de partitions correspondantes (nombre d'étiquettes des sommets et des arêtes pour *GREC* et des sommets pour *Letter*)

	Meilleur taux (en %)	K_1	K_2	Taille Matrice
GREC	99.17	17	7	11×25
Letter	97.63	22	N/A	11×23
	Taux le plus faible (en %)	K_1	K_2	Taille Matrice
GREC	78.45	2	49	11×52
Letter	80.43	3	N/A	11×4
	Taux moyens (en %)	σ		
GREC	97.20	2.55		
Letter	95.85	2.60		

TABLE 4.2 – Étude de l'influence du partitionnement des étiquettes sur la classification des graphes des bases *GREC* et *Letter*.

Les résultats de la méthode de partitionnement sont reportés dans le tableau 4.2. Pour chaque ensemble de graphes testé, nous avons choisi de mettre en avant quelques résultats significatifs plutôt que l'ensemble des expérimentations, trop nombreuses dont la présentation complète est reportée en Annexe B. Le meilleur taux de classification obtenu est répertorié dans la première partie du tableau, le taux le plus faible dans la seconde. Afin de mesurer l'influence du choix du nombre de classes d'étiquettes K_1 et K_2, nous montrons

également le taux moyen obtenu pour K_1 et K_2 variant dans [2..50]. Il apparaît clairement que le choix du nombre de classes influe beaucoup sur les résultats de classification. Nous pouvons donc en déduire que ce paramètre est déterminant dans la représentativité de notre représentation matricielle puisqu'un mauvais classement peut faire chuter les résultats d'environ 18 % pour *Letter* et d'environ 11 % pour *GREC*. Nous pouvons également évaluer l'influence du partitionnement grâce aux indicateurs de moyenne et de l'écart-type. La moyenne est plutôt élevée par rapport aux maximaux obtenus et que la dispersion autour de cette moyenne est plutôt faible. Nous pouvons noter que même si le partitionnement d'étiquettes est un élément clef dans le résultat de notre méthode de projection (au vu des plus faibles taux obtenus), des résultats corrects et proches des maximums sont obtenus dans la majorité des cas.

Nous constatons que les résultats obtenus par un partitionnement global des étiquettes, opposé à la discrétisation puis la combinaison attribut par attribut, semblent meilleurs. En effet, le gain est d'environ 3.5 % pour la base *GREC* et 6.3 % pour la base *Letter*. De plus, la mise en place du partitionnement est simplifiée dans le sens où seulement deux paramètres sont à régler, le nombre de classes désiré pour les étiquettes de sommets et le nombre de classes d'étiquettes désiré pour les d'arêtes (K_1 et K_2). Dans la méthode de discrétisation et de combinaison, l'optimisation de la discrétisation doit être envisagée attribut par attribut. Suivant les étiquettes, ce nombre peut être important et donc la phase d'apprentissage/validation d'autant plus lourde. De ce point de vue, l'utilisation d'un partitionnement des espaces des étiquettes est à nouveau préférable.

4.3.2 Mesure de l'apport de l'information étiquetée

Nous proposons une deuxième expérimentation visant à quantifier l'enrichissement de notre représentation matricielle par l'ajout de l'information portée par les étiquettes. Il est tout d'abord nécessaire de vérifier si la méthode que nous proposons pour étendre notre projection de graphes ne détériore pas l'information portée par les étiquettes. En effet, nous savons que le partitionnement peut engendrer des confusions aux alentours des frontières entre les classes d'étiquettes. Si la distance entre les classes est faible, alors la classification des étiquettes proches des frontières peut être sujette à ambiguité. Ces confusions sont alors répercutées lors de la projection des graphes et induisent des erreurs lors de leur classification. Nous avons utilisé la méthode du partitionnement pour les graphes comportant des attributs numériques (*GREC* et *Letter*). Nous avons utilisé pour les deux méthodes un classificateur des plus proches voisins dont le k est optimisé sur une base de validation.

Le tableau 4.3 présente un comparatif entre la classification des graphes retenant uniquement l'information topologique, *i.e.* sans étiquettes, et les mêmes graphes étiquetés. Dans un premier temps, la distance d'édition des graphes est utilisée sur les trois bases. Une différence des taux de classification entre les bases avec ou sans étiquettes est effectuée pour obtenir une référence sur l'apport de l'information porté par les étiquettes. Ensuite, nous calculons les différences des résultats obtenus par classification en utilisant notre encapsulation sur les trois bases étiquetées et non étiquetées. Nous remarquons que les différences obtenues sont au moins équivalentes sinon supérieures pour l'encapsulation que nous proposons. Nous pouvons donc en déduire que l'information liée aux étiquettes des graphes est conservée lors de son encapsulation dans la représentation matricielle. L'in-

		GREC	*Letter*	*Mutagenicity*
Distance d'édition de graphes	Sans étiquettes	88.00	58.67	65.80
	Avec étiquettes	95.50	99.60	71.50
	Différence	7.50	40.93	5.70
Représentation matricielle	Sans étiquettes	92.09	43.78	68.39
	Avec étiquettes	99.17	97.63	77.20
	Différence	7.08	53.85	8.81

TABLE 4.3 – Tableau de comparaison de résultats de classification (en %) sur les graphes étiquetés et non étiquetés des bases *GREC*, *Letter* et *Mutagenicity* par l'utilisation de la distance d'édition de graphe et de la représentation matricielle proposée.

dication de ces étiquettes sur les différentes occurrences des motifs topologiques du lexique apporte bien une information supplémentaire contribuant à l'amélioration des résultats de classification.

4.3.3 Comparaison avec la distance d'édition de graphes

Nous voulons maintenant opposer notre approche à une méthode non issue du domaine de l'encapsulation de graphes, à savoir la distance d'édition de graphes proposée dans [Neuhaus *et al.*, 2006]. Nous avons vu que cette distance était considérée comme une référence. Nous voulons donc lui confronter notre approche afin de caractériser sa performance en classification. Les paramètres du classificateur et de la distance d'édition de graphes sont optimisés sur une base de validation, indépendante de la base de test.

	GREC	Letter	Mutagenicity
Distance d'édition de graphe	95.50	99.60	71.50
Notre représentation matricielle	99.17	97.63	77.20

TABLE 4.4 – Tableau de comparaison de résultats de classification (en %) sur les bases *GREC*, *Letter* et *Mutagenicity* par l'utilisation de la distance d'édition de graphe et de la représentation matricielle proposée.

Le tableau 4.4 illustre les résultats obtenus. Dans un premier temps, nous pouvons remarquer l'amélioration notable sur les bases *GREC*, +3.6 %, et *Mutagenicity*, +5.7 %. Nous notons également le taux de classification inférieur de 2 % de sur la base *Letter*. Les résultats peuvent être opposés à ceux mis en avant lors de l'évaluation de notre projection de la topologie des graphes (*cf.* tableau 3.6). Les conclusions peuvent donc être comparables. Il est en effet logique que les imprécisions dues à l'encapsulation de la topologie ne soient pas compensées par l'intégration de l'information étiquetée dans notre méthode. En revanche, les bons taux obtenus sur *GREC* et *Mutagenicity* révèlent la bonne robustesse aux différents bruits de notre représentation matricielle.

4.3.4 Comparaison avec une méthode de projection de graphes

Dans cette quatrième expérimentation, nous avons voulu confronter notre technique de projection avec une méthode issue de la littérature. Nous avons choisi de prendre pour comparaison la méthode de projection de graphes utilisant les dissimilarités. Nous rappelons que cette approche, développée par Riesen, utilise une distance entre différents graphes prototypes de la base pour construire un vecteur de distance. Plusieurs méthodes sont utilisées pour choisir ces graphes prototypes. Les résultats que nous présentons ici sont les meilleurs issus de la série d'expérimentations faites dans [Riesen, 2009]. Les deux méthodes sont alors évaluées sur une tâche de classification de type k-ppv.

	GREC	Letter	Mutagenicity
Projection par dissimilarités	92.40	99.3	72.4
Notre représentation matricielle	99.17	97.63	77.2

TABLE 4.5 – Tableau de comparaison de résultats de classification (en %) sur les bases *GREC*, *Letter* et *Mutagenicity* par l'utilisation de la méthode d'encapsulation de graphe et de la représentation matricielle proposée.

Nous observons que ces résultats sont très probants. Deux résultats sont supérieurs à ceux obtenus en utilisant une encapsulation par dissimilarités. L'amélioration est de 6.77 % pour *GREC* et de 4.8 % pour *Mutagenicity*. Pour *Letter*, une légère diminution de 1.67 % est à noter. En effet, les faibles différences de topologie entre ces graphes provoquent une sensibilité accrue aux bruits structurels. Nous avions déjà notifié en 3.3.4.1 les difficultés sur ce type de graphe, les erreurs dues à la projection de la topologie sont donc logiquement répercutées.

4.4 Conclusion

Dans ce chapitre, nous avons proposé une méthode pour étendre notre projection de la topologie d'un graphe suivant un lexique générique en proposant l'ajout des informations portées par les étiquettes. Nous avons conservé l'idée directrice du dénombrement des occurrences d'un motif en proposant un lexique de motifs étiquetés en supplément des motifs uniquement topologiques. Ce simple ajout permet de conserver la simplicité de notre approche sans augmenter la complexité de la projection. Nous avons cependant vu que dans certains cas, un prétraitement était nécessaire afin de constituer un lexique d'étiquettes. Nous proposons alors deux méthodes pour résoudre cette problématique : une discrétisation des attributs numériques suivi d'une combinaison des valeurs d'attributs possibles ou un double partitionnement de l'ensemble des étiquettes pour les sommets et les arêtes. Pour les deux méthodes, ce prétraitement nécessite une base d'apprentissage.

Des expérimentations ont été faites pour comparer ces différentes approches, évaluer la conservation de l'information portée par les étiquettes et confronter notre méthode aux méthodes de référence de la littérature. Les résultats montrent donc que notre méthode permet de bien exploiter l'information portée par les étiquettes lors de la projection des

graphes vers une représentation matricielle. Il apparaît en effet que les écarts des taux de classification entre des graphes non-étiquetés et les mêmes graphes avec des étiquettes sont stables entre l'utilisation d'une distance de graphe et notre méthode. De plus, dans la majorité des cas, les résultats obtenus avec la projection de graphes que nous proposons sont meilleurs que ceux issus de l'utilisation des méthodes de référence de la littérature. Pour finir, nous montrons aussi une comparaison avec une méthode de projection de graphe et constatons une amélioration des résultats sur deux des trois bases expérimentées. Ces bons résultats sont dus à la prise en considération, lors de la projection dans la représentation matricielle, des informations portées par les étiquettes des sommets et des arêtes combinées aux informations structurelles portées par la topologie du graphe. Notre technique permet ainsi de conserver le pouvoir de description des graphes dans son ensemble.

Si notre méthode s'avère performante en classification, le choix des paramètres reste néanmoins un point délicat. Ce choix s'articule principalement autour du nombre de classes d'étiquettes à fixer pour la représentation matricielle. Nous avons vu que les résultats de classification variaient en fonction de ce paramètre et que cet aspect reste à optimiser avec une base de validation par exemple.

Conclusion générale et perspectives

> On ne fait jamais attention à ce qui a été fait ; on ne voit que ce qui reste à faire.
>
> Marie Curie

Nous dressons maintenant une synthèse des travaux que nous avons réalisés au cours de cette thèse portant sur une contribution aux méthodes de projection de graphe utilisée dans un contexte de reconnaissance structurelle des formes. Aux termes de ces conclusions, nous évoquons quelques perspectives que nous souhaiterions mettre en place dans la continuité de ces travaux.

Dans le chapitre 1, nous avons, dans un premier temps, introduit la thématique de cette thèse : l'application des graphes au domaine de la reconnaissance structurelle des formes. Les graphes sont des outils offrant une grande capacité de représentation en permettant, au sein de la même structure, de décrire une forme en caractérisant ses composantes (information étiquetée) et les relations qui peuvent exister entre elles (information topologique). Les graphes véhiculent donc une information plus riche que celle portée par les vecteurs (dans les approches dites statistiques). De plus, les graphes montrent une plus grande flexibilité en s'adaptant aux données contrairement aux vecteurs, qui présentent une certaine rigidité. Au cours de ce chapitre, nous avons également isolé une problématique. La théorie des graphes est associée à la difficulté de définir un formalisme pour mesurer une similarité entre graphes, notion largement exploitée en reconnaissance des formes.

Dans le chapitre 2, nous avons présenté un état de l'art sur les méthodes de comparaison de graphes qui peuvent être distinguées en plusieurs catégories :

- Les appariements sont des méthodes capables de vérifier un isomorphisme entre deux graphes. Elles renvoient une fonction qui associe les sommets et les arêtes d'un graphe à un autre. Si les appariements exacts sont peu applicables en raison de leur rigidité (aucune erreur de topologie ou d'étiquetage n'est permise), des approches plus souples, dites inexactes ou à correction d'erreurs, permettent une exploitation sur des données réelles et imparfaites. Pour toutes ces approches, la mesure de similarité est néanmoins restreinte à une réponse booléenne, soit deux graphes sont égaux, soit ils ne le sont pas.
- D'autres approches, réunies sous la famille des distances de graphes, proposent de mesurer une similarité en évaluant une distance entre deux graphes. Ces mesures permettent d'estimer plus finement la ressemblance entre deux graphes, en opposition

aux réponses binaires des appariements. En revanche, ces méthodes peuvent perdre les associations nœuds à nœuds (respectivement arcs à arcs) entre deux graphes. Nous notons que pour certaines applications, comme la localisation de motifs, les distances de graphes ne pourront être exploitées que sous certaines contraintes. Remarquons également que le calcul de ces distances est obtenu par des algorithmes dont la complexité peut être un frein à leurs utilisations.

– Les techniques de projection de graphes ont été développées pour répondre à la problématique des temps de calcul. L'idée transversale de ces approches est de projeter un graphe dans un espace vectoriel afin d'en obtenir une représentation vectorielle afin de tirer parti des avantages des outils de reconnaissance statistique des formes, notamment les calculs de distance en temps linéaires.
– Nous avons également abordé le problème de la classification des graphes, *i.e.* identifier la classe d'un graphe requête parmi un ensemble de graphes. Basiquement, la classification se fait au moyen d'une fonction, induite à partir d'un ensemble d'apprentissage (un ensemble de graphes dont la classe est connue *a priori*), qui attribue une classe à un graphe. Ces notions d'apprentissage et de classification font toutefois référence aux mesures de similarités et sont, par conséquent, assez complexes. Afin de réduire les temps de traitement, plusieurs travaux ont été proposés ; notamment ceux utilisant les projections explicites de graphes, qui nous semblent particulièrement pertinents.

Le chapitre 3 présente notre contribution d'une nouvelle projection de graphes en prenant uniquement en compte, dans un premier temps, l'information topologique. À la croisée des méthodes de sondage de graphes et les concepts de sacs de graphes, nous proposons de projeter les graphes en dénombrant les occurrences de sous-graphes (appelés motifs) issus d'un lexique. Ce lexique est généré à partir du réseau des graphes non-isomorphes, ce qui le rend générique (indépendant des graphes à traiter) et surtout porteur d'informations topologiques (les motifs sont de taille et d'ordre élevés permettant ainsi une encapsulation forte de la topologie). Les résultats montrent que cette prise en considération de l'information portée par la topologie du graphe augmente la qualité de la projection par rapport aux sondages de graphes existants.

Le chapitre 4 expose l'enrichissement de notre projection par l'adjonction des informations portées par l'étiquetage des sommets et/ou des arêtes des graphes. Cette extension est rendu possible en passant du dénombrement des occurrences de motifs non-étiquetés au dénombrement des occurrences de motifs étiquetés. La nature (symbolique ou numérique) ainsi que le nombre des attributs composant une étiquette sont les deux contraintes majeures à l'intégration de l'information étiquetée dans notre lexique. Nous attaquons ces problématiques en proposant deux méthodes : une basée sur une discrétisation puis une combinaison des attributs des étiquettes, l'autre fondée sur un partitionnement global des étiquettes. Des expérimentations mettent en avant certains résultats intéressants comparés à des méthodes de référence.

Nous n'avons évidemment pas la prétention d'affirmer que les méthodes proposées apportent les réponses parfaites et indiscutables à la problématique complexe de l'utilisation des graphes dans un système de reconnaissance structurelle de formes. Les travaux produits durant cette thèse apportent un point de vue que nous pensons pertinent et ouvrent également quelques perspectives, parmi tant d'autres, qu'il serait intéressant de développer :

- La recherche des occurrences des motifs est exhaustive et optimale ; par conséquent elle est également de complexité exponentielle. Pour abaisser les coûts en temps de calcul, nous pensons que la mise en place de méthodes de recherche approximatives, ou heuristiques, pourrait présenter un bon compromis entre la réduction de la complexité (qui deviendrait linéaire) et la qualité de la représentation vectorielle utilisée en classification. Dans cette idée, nous avons d'ores et déjà initié un rapprochement avec l'équipe *Ordonnancement et Conduite* du Laboratoire d'Informatique de Tours, spécialisée dans le domaine de l'optimisation et de la recherche opérationnelle et susceptible de nous proposer des algorithmes en adéquation avec nos objectifs.
- Nous envisageons également d'utiliser notre projection dans un système d'indexation de graphes. Dans le domaine de la recherche d'information, les index permettent d'accélérer l'accès aux données. Si les données sont des graphes, il est alors tout a fait possible d'utiliser notre représentation vectorielle comme index pour améliorer la recherche de graphes similaires à un graphe requête. Une optimisation peut également être suggérée en réduisant le nombre de graphes à indexer par l'emploi de graphes médians ou de graphes prototypes. La projection tirerait avantage de la représentation qui annonce de bonnes performances et l'inconvénient lié au temps d'extraction serait gommé (puisqu'il ne serait fait qu'une seule fois).
- Une autre perspective serait d'améliorer l'intégration de l'information portée par les étiquettes au sein de notre projection. Dans cette optique, nous souhaitons mettre en place une méthode de partitionnement appliquée non plus aux seules étiquettes, comme nous l'avons présentée, mais sur les motifs étiquetés directement. Nous pensons que cette idée permettrait un lien encore plus fort dans la relation entre les informations topologiques et étiquetées.
- Nous pouvons également évoquer une dernière idée qui viserait à augmenter la représentativité du lexique. Une étape d'analyse de l'ensemble d'apprentissage permettrait un apprentissage visant à une validation des motifs les plus pertinents et un rejet des motifs les plus perturbateurs. Cet affinage du lexique conduirait donc à adapter celui-ci aux données et à l'augmentation des performances de la projection. Cependant, la spécialisation du lexique va à l'encontre d'une méthode de représentation générique. De plus, retirer des motifs a priori perturberait l'algorithme d'extraction, qui lui est exhaustif dans sa version actuelle. Une réduction de caractéristiques pourrait également être effectuée *a posteriori*.

Ces perspectives laissent percevoir le nombre et la diversité des défis qu'il reste à relever dans la problématique de cette thèse. Le domaine de la reconnaissance structurelle des formes est vaste et les verrous scientifiques sont nombreux. Avec ces travaux, nous espérons avoir apporté une pierre à cet édifice.

Annexes

Annexe A

Optimisation des paramètres de classificateurs

Dans cette annexe, nous présentons les résultats produits lors de la recherche des paramètres optimaux des classificateurs utilisés. Ces taux sont obtenus sur un ensemble de validation indépendant des ensembles d'apprentissage et de test.

Afin d'optimiser la pertinence de ces classificateurs, nous avons fait varier leurs paramètres les plus importants.

Machines à vecteurs de supports

- **C** : paramètre de régularisation.
- **G** : paramètre du noyau gaussien

Perceptron multicouche

- **L** : taux (ou force) apprentissage.
- **H** : nombre de neurones de la couche cachée.
  . i : nombre de neurones de la couche d'entrée (taille de la représentation matricielle).
  . o : nombre de neurones de la couche de sortie (nombre de classes).
  . a : moyenne du nombre de neurones d'entrée et de neurones de sortie.

Forêts Aléatoires

- **I** : nombre d'arbres.
- **K** : profondeur des arbres.

Machines à vecteurs de supports

GREC

		C				
		1	10	100	1000	10000
G	0.0001	45.45	66.55	82.09	92.45	93.27
	0.001	64.91	82.82	92.82	92.55	92.73
	0.01	83.18	92.73	92.91	92.91	92.91
	0.1	91.82	92.82	92.82	92.82	92.82
	0.3	90.91	91.27	91.27	91.27	91.27
	0.5	89.73	90.00	90.00	90.00	90.00
	0.7	89.55	89.55	89.55	89.55	89.55
	1.0	89.45	89.45	89.45	89.45	89.45

Letter

		C				
		1	10	100	1000	10000
G	0.0001	26.49	29.96	49.02	49.91	50.00
	0.001	29.60	49.11	49.82	50.40	50.13
	0.01	48.67	49.73	50.62	50.13	50.13
	0.1	49.60	49.91	50.00	50.00	50.00
	0.3	49.33	49.82	49.82	49.82	49.82
	0.5	49.38	49.60	49.60	49.60	49.60
	0.7	49.51	49.56	49.56	49.56	49.56
	1.0	49.73	49.73	49.73	49.73	49.73

Mutagenicity

		C				
		1	10	100	1000	10000
G	0.0001	66.87	68.55	69.66	70.88	71.50
	0.001	67.79	69.73	70.49	70.12	69.91
	0.01	70.23	70.30	69.47	69.24	69.22
	0.1	68.69	68.64	68.60	68.60	68.60
	0.3	64.70	64.72	64.72	64.72	64.72
	0.5	62.49	62.55	62.55	62.55	62.55
	0.7	62.39	62.30	62.30	62.30	62.30
	1.0	62.12	62.28	62.28	62.28	62.28

Perceptron multicouche

GREC

		L						
		0.001	0.01	0.1	0.3	0.5	0.7	0.9
H	1	4.55	4.45	12.09	15.64	17.64	17.64	18.36
	a	4.91	35.18	87.27	89.36	87.82	88.82	87.27
	i	4.55	27.64	86.82	87.45	88.55	88.27	85.36
	o	5.09	48.27	88.73	88.73	88.00	88.55	89.45

Letter

		L						
		0.001	0.01	0.1	0.3	0.5	0.7	0.9
H	1	6.67	14.36	25.56	26.49	25.78	26.76	25.51
	a	13.29	49.51	50.00	49.64	49.42	49.29	49.47
	i	8.62	49.69	50.09	49.33	49.29	49.38	49.38
	o	13.07	49.33	49.96	49.82	49.38	49.33	49.29

Mutagenicity

		L						
		0.001	0.01	0.1	0.3	0.5	0.7	0.9
H	1	66.31	67.79	67.51	66.73	66.89	62.46	60.53
	a	66.24	68.76	68.18	67.37	66.48	65.46	65.76
	i	66.18	68.55	68.27	67.90	66.43	66.78	66.24
	o	66.13	68.64	67.81	66.08	65.94	66.22	66.18

Forêts aléatoires

GREC

		K					
		0	1	2	3	4	5
I	100	92.91	92.73	93.00	93.00	92.91	93.18
	200	92.73	92.55	93.09	92.91	92.73	92.91
	300	92.82	92.73	92.91	92.82	92.82	93.00
	400	92.82	92.64	92.91	92.91	92.82	93.09
	500	92.91	92.82	92.82	92.82	92.91	92.91

Letter

		K					
		0	1	2	3	4	5
I	100	49.82	49.96	49.91	49.91	49.82	49.82
	200	49.64	49.73	49.64	49.73	49.64	49.56
	300	49.69	49.78	49.78	49.73	49.69	49.64
	400	49.69	49.82	49.82	49.73	49.69	49.69
	500	49.60	49.69	49.64	49.60	49.60	49.56

Mutagenicity

		K					
		0	1	2	3	4	5
I	100	71.27	71.45	71.39	71.48	71.27	71.27
	200	71.34	71.55	71.36	71.48	71.34	71.29
	300	71.43	71.64	71.69	71.73	71.43	71.50
	400	71.55	71.80	71.82	71.66	71.55	71.41
	500	71.52	71.52	71.71	71.55	71.52	71.41

Annexe B

Optimisation du partitionnement d'étiquettes

Dans cette annexe, nous fournissons les résultats de classification obtenus lors de l'optimisation du nombre de partitions d'étiquettes des graphes utilisés lors de nos expérimentations. Deux ensembles de graphes nécessitent un prétraitement. En effet, nous avons noté que les graphes de *GREC* possèdent des attributs numériques sur les sommets et les arêtes, ceux de *Letter* seulement sur les sommets. Les graphes de *Mutagenicity* ne comportent pas de telles étiquettes. Le nombre de classes varient entre 2 et 50.

GREC

Nombre de partitions pour les étiquettes des sommets	Nombre de partitions pour les étiquettes des arêtes																		
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	96.88	97.06	96.05	97.67	97.76	98.28	98.25	98.62	98.47	98.43	98.43	98.25	98.49	98.78	98.92	99.08	98.81	98.73	99.01
3	97.50	97.34	96.51	98.01	98.01	98.43	98.45	98.76	98.73	98.69	98.55	98.35	98.58	98.80	98.92	99.08	98.88	98.81	99.01
4	97.50	97.34	96.51	98.01	98.01	98.43	98.45	98.76	98.73	98.69	98.55	98.35	98.58	98.80	98.92	99.08	98.88	98.81	99.01
5	97.12	97.10	96.48	98.06	98.15	98.47	98.30	98.79	98.82	98.73	98.55	98.38	98.69	98.95	98.75	99.14	98.85	98.81	98.98
6	97.12	97.10	96.48	98.06	98.15	98.47	98.30	98.79	98.82	98.73	98.55	98.38	98.69	98.95	98.75	99.14	98.85	98.81	98.98
7	96.75	96.63	96.20	97.98	98.43	98.75	98.53	98.51	98.82	98.68	98.23	98.57	98.63	99.03	99.10	99.17	98.83	98.80	98.92
8	96.01	96.09	96.77	97.57	98.12	98.19	98.16	98.76	98.68	98.44	98.46	98.53	98.71	98.98	98.65	99.02	98.75	98.57	98.75
9	95.75	96.02	96.32	97.61	97.95	97.86	97.92	98.47	98.46	97.84	98.13	98.29	98.41	98.58	98.32	98.77	98.75	98.37	98.35
10	95.76	95.99	96.35	97.58	97.92	97.84	97.89	98.45	98.45	97.81	98.10	98.26	98.38	98.55	98.29	98.75	98.74	98.35	98.33
11	95.76	95.99	96.35	97.58	97.92	97.84	97.89	98.45	98.45	97.81	98.10	98.26	98.38	98.55	98.29	98.75	98.74	98.35	98.33
12	95.39	95.66	95.66	97.32	98.03	98.08	98.11	98.42	98.11	98.10	98.24	98.60	98.52	98.49	98.55	98.86	98.55	98.31	98.61
13	95.39	95.66	95.66	97.32	98.03	98.08	98.11	98.42	98.11	98.10	98.24	98.60	98.52	98.49	98.55	98.86	98.55	98.31	98.61
14	95.85	96.30	95.97	97.22	97.84	98.06	98.25	98.59	98.22	98.12	98.02	98.36	98.55	98.60	98.52	98.88	98.65	98.27	98.55
15	94.98	95.41	95.13	96.71	97.70	97.72	98.00	98.36	98.14	97.82	97.97	98.22	98.39	98.78	98.55	98.81	98.70	98.34	98.36
16	94.73	96.01	95.32	97.29	97.91	97.85	97.95	98.45	98.15	98.19	97.98	98.26	98.44	98.75	98.64	98.75	98.69	98.37	98.69
17	93.59	95.41	94.00	96.71	97.47	97.89	97.91	98.25	97.99	97.89	98.03	98.02	98.15	98.41	98.65	98.86	98.73	98.38	98.53
18	93.60	94.89	94.13	96.83	97.60	98.00	97.88	98.35	98.23	98.05	98.04	98.05	98.13	98.22	98.28	98.81	98.73	98.17	98.43
19	93.60	94.89	94.13	96.83	97.60	98.00	97.88	98.35	98.23	98.05	98.04	98.05	98.13	98.22	98.28	98.81	98.73	98.17	98.43
20	93.52	95.37	94.66	96.98	97.71	98.13	97.65	98.25	97.79	97.67	97.73	97.64	97.79	98.12	98.11	98.81	98.64	98.24	98.44
21	93.52	95.37	94.66	96.98	97.71	98.13	97.65	98.25	97.79	97.67	97.73	97.64	97.79	98.12	98.11	98.81	98.64	98.24	98.44
22	90.43	93.35	92.93	96.18	96.87	97.42	96.96	98.03	97.62	97.16	96.88	97.30	97.75	98.05	98.03	98.45	98.26	98.00	98.26
23	91.35	93.64	92.34	95.83	97.20	97.45	97.18	97.73	97.95	97.76	97.25	97.46	98.04	98.14	98.29	98.49	98.53	98.25	98.09
24	90.48	93.11	92.66	95.66	96.99	96.84	97.01	97.34	97.56	97.55	96.87	97.34	97.73	98.08	97.93	98.42	98.02	97.94	97.87
25	90.48	93.11	92.66	95.66	96.99	96.84	97.01	97.34	97.56	97.55	96.87	97.34	97.73	98.08	97.93	98.42	98.02	97.94	97.87
26	88.74	91.88	91.29	95.14	96.23	96.50	96.48	97.74	97.19	97.14	97.21	97.24	97.54	97.82	97.97	98.60	98.17	98.01	98.03
27	87.48	90.80	90.12	94.40	95.88	96.28	95.84	97.11	97.01	96.96	96.90	96.91	97.31	97.75	98.04	98.25	98.06	97.78	97.88
28	87.32	91.35	90.89	95.14	95.92	96.43	96.26	97.48	97.19	96.67	97.15	97.26	97.16	97.84	98.00	98.37	97.77	97.59	98.16
29	87.32	91.35	90.89	95.14	95.92	96.43	96.26	97.48	97.19	96.67	97.15	97.26	97.16	97.84	98.00	98.37	97.77	97.59	98.16
30	85.24	89.52	89.51	93.95	95.11	95.75	95.46	96.69	96.60	96.15	96.47	96.53	96.64	96.96	97.65	98.22	97.55	97.20	97.81
31	85.21	89.37	88.25	93.50	94.64	95.77	95.34	96.54	96.64	96.27	96.57	96.56	96.92	96.94	97.73	98.13	97.70	97.24	97.76
32	85.21	89.37	88.25	93.50	94.64	95.77	95.34	96.54	96.64	96.27	96.57	96.56	96.92	96.94	97.73	98.13	97.70	97.24	97.76
33	85.21	89.37	88.25	93.50	94.64	95.77	95.34	96.54	96.64	96.27	96.57	96.56	96.92	96.94	97.73	98.13	97.70	97.24	97.76
34	85.21	89.37	88.25	93.50	94.64	95.77	95.34	96.54	96.64	96.27	96.57	96.56	96.92	96.94	97.73	98.13	97.70	97.24	97.76
35	85.21	89.37	88.25	93.50	94.64	95.77	95.34	96.54	96.64	96.27	96.57	96.56	96.92	96.94	97.73	98.13	97.70	97.24	97.76
36	85.30	88.31	87.88	92.65	93.99	95.28	94.78	95.77	95.96	95.62	96.05	96.01	96.15	96.61	97.35	97.92	97.26	96.87	97.36
37	85.30	88.31	87.88	92.65	93.99	95.28	94.78	95.77	95.96	95.62	96.05	96.01	96.15	96.61	97.35	97.92	97.26	96.87	97.36
38	85.09	88.46	87.93	92.42	93.90	95.32	94.34	95.79	95.90	95.40	95.82	95.77	95.91	96.54	97.17	97.78	97.42	96.92	97.52
39	83.95	88.12	87.35	92.06	93.79	95.28	95.09	96.02	96.04	95.49	95.70	95.76	95.92	96.68	97.30	97.66	97.50	97.08	97.52
40	83.40	87.26	86.35	91.89	93.23	95.09	94.28	94.94	95.49	95.36	95.35	95.38	95.67	96.09	97.00	97.53	97.10	96.92	97.08
41	82.46	86.56	85.35	91.41	93.21	94.78	94.59	94.97	94.99	95.04	95.34	95.11	95.45	96.10	97.01	97.65	96.75	96.43	96.83
42	81.13	85.50	85.33	89.67	91.37	93.77	93.96	94.87	95.08	95.26	95.23	95.22	95.43	96.07	96.95	97.27	96.63	96.56	96.91
43	81.13	85.50	85.33	89.67	91.37	93.77	93.96	94.87	95.08	95.26	95.23	95.22	95.43	96.07	96.95	97.27	96.63	96.56	96.91
44	79.89	84.69	84.14	89.08	90.85	93.65	93.34	94.25	94.38	94.62	94.56	94.56	94.74	95.62	96.52	97.02	96.26	96.21	96.55
45	80.15	83.56	83.79	89.10	90.77	93.02	93.65	93.89	94.46	94.39	94.36	94.50	94.87	95.74	96.50	97.46	96.65	96.65	96.58
46	78.67	82.36	82.20	88.62	90.37	92.77	93.15	93.75	94.27	94.32	94.03	94.05	94.69	95.65	96.35	97.39	96.37	96.45	96.68
47	78.67	82.36	82.20	88.62	90.37	92.77	93.15	93.75	94.27	94.32	94.03	94.05	94.69	95.65	96.35	97.39	96.37	96.45	96.68
48	79.35	82.54	82.45	88.83	90.38	92.56	92.80	93.56	93.67	93.55	93.83	93.96	94.72	95.40	96.13	97.05	96.20	96.43	96.53
49	78.45	82.27	83.09	88.31	90.09	92.37	92.95	93.20	93.62	93.55	93.84	93.94	94.66	95.36	96.03	96.91	96.32	96.41	96.29
50	78.45	82.27	83.09	88.31	90.09	92.37	92.95	93.20	93.62	93.55	93.84	93.94	94.66	95.36	96.03	96.91	96.32	96.41	96.29

Nombre de partitions pour les étiquettes des sommets	Nombre de partitions pour les étiquettes des arêtes																			
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
2	99.08	98.87	98.86	98.64	98.70	98.49	98.93	98.92	98.79	98.62	98.93	98.44	98.45	98.69	98.27	98.14	98.57	98.52	98.52	98.79
3	99.16	98.95	98.95	98.83	98.89	98.57	99.05	99.00	98.95	98.70	99.02	98.54	98.55	98.76	98.35	98.21	98.71	98.70	98.71	98.85
4	99.16	98.95	98.95	98.83	98.89	98.57	99.05	99.00	98.95	98.70	99.02	98.54	98.55	98.76	98.35	98.21	98.71	98.70	98.71	98.85
5	98.89	99.06	98.94	98.80	98.87	98.47	99.09	98.99	98.98	98.71	99.09	98.48	98.44	98.54	98.14	98.07	98.64	98.67	98.72	98.75
6	98.89	99.06	98.94	98.80	98.87	98.47	99.09	98.99	98.98	98.71	99.09	98.48	98.44	98.54	98.14	98.07	98.64	98.67	98.72	98.75
7	98.93	98.82	98.85	98.59	98.69	98.40	98.97	98.94	98.91	98.62	98.99	98.29	98.45	98.23	98.25	98.18	98.70	98.64	98.83	98.93
8	98.90	98.82	98.76	98.63	98.70	98.65	98.99	98.75	98.74	98.60	98.95	98.18	98.20	98.24	98.17	98.12	98.74	98.55	98.77	98.93
9	98.66	98.75	98.44	98.46	98.55	98.39	99.02	98.81	98.79	98.48	98.90	98.36	98.17	98.33	98.14	98.14	98.65	98.49	98.75	98.52
10	98.64	98.72	98.41	98.44	98.53	98.39	99.02	98.81	98.79	98.48	98.89	98.36	98.18	98.31	98.12	98.16	98.67	98.54	98.84	98.59
11	98.64	98.72	98.41	98.44	98.53	98.39	99.02	98.81	98.79	98.48	98.89	98.36	98.18	98.31	98.12	98.16	98.67	98.54	98.84	98.59
12	98.90	98.92	98.67	98.56	98.62	98.70	98.83	98.57	98.55	98.45	98.53	98.22	98.23	98.06	98.20	98.16	98.68	98.53	98.88	98.96
13	98.90	98.92	98.67	98.56	98.62	98.70	98.83	98.57	98.55	98.45	98.53	98.22	98.23	98.06	98.20	98.16	98.68	98.53	98.88	98.96
14	99.05	98.80	98.66	98.37	98.45	98.60	98.58	98.55	98.55	98.48	98.66	98.30	98.30	98.25	98.29	98.26	98.68	98.54	98.68	98.92
15	98.75	98.60	98.50	98.36	98.48	98.55	98.82	98.65	98.64	98.46	98.76	98.23	98.27	98.24	98.26	98.22	98.80	98.59	98.85	98.94
16	98.91	98.79	98.44	98.15	98.24	98.70	98.95	98.64	98.69	98.55	98.76	98.36	98.32	98.13	98.42	98.36	98.85	98.74	98.96	98.92
17	98.80	98.70	98.45	98.30	98.35	98.60	98.74	98.46	98.43	98.29	98.51	98.23	98.21	98.17	98.14	98.10	98.63	98.61	98.87	98.79
18	98.96	98.76	98.41	98.34	98.37	98.52	98.66	98.55	98.58	98.28	98.47	98.20	98.37	98.29	98.20	98.15	98.55	98.62	98.81	98.86
19	98.96	98.76	98.41	98.34	98.37	98.52	98.66	98.55	98.58	98.28	98.47	98.20	98.37	98.29	98.20	98.15	98.55	98.62	98.81	98.86
20	98.75	98.49	98.36	98.06	98.15	98.46	98.55	98.47	98.51	98.13	98.43	98.25	98.33	98.06	98.01	98.00	98.45	98.57	98.90	98.67
21	98.75	98.49	98.36	98.06	98.15	98.46	98.55	98.47	98.51	98.13	98.43	98.25	98.33	98.06	98.01	98.00	98.45	98.57	98.90	98.67
22	98.63	98.25	98.09	97.96	98.03	98.04	98.40	98.39	98.34	98.09	98.39	97.84	97.81	97.73	97.92	98.04	98.11	98.24	98.73	98.45
23	98.75	98.54	98.28	98.44	98.49	98.35	98.57	98.40	98.45	98.17	98.49	98.15	98.19	98.09	97.85	97.85	98.27	98.29	98.65	98.40
24	98.42	98.34	98.15	97.96	98.06	98.05	98.44	98.35	98.40	98.28	98.55	97.98	98.07	97.97	97.93	98.05	98.40	98.41	98.72	98.45
25	98.42	98.34	98.15	97.96	98.06	98.05	98.44	98.35	98.40	98.28	98.55	97.98	98.07	97.97	97.93	98.05	98.40	98.41	98.72	98.45
26	98.51	98.48	98.10	98.00	98.16	98.16	98.36	98.20	98.25	98.16	98.53	97.97	97.89	97.80	98.06	98.21	98.12	98.19	98.76	98.58
27	98.47	98.33	97.83	98.01	98.16	98.07	98.25	98.35	98.40	98.17	98.33	98.00	98.02	97.92	98.05	98.12	98.19	98.26	98.70	98.58
28	98.40	98.15	97.83	97.95	98.05	98.38	98.44	98.27	98.26	98.12	98.35	98.25	98.08	97.85	98.00	98.15	98.33	98.31	98.64	98.46
29	98.40	98.15	97.83	97.95	98.05	98.38	98.44	98.27	98.26	98.12	98.35	98.25	98.08	97.85	98.00	98.15	98.33	98.31	98.64	98.46
30	98.10	97.98	97.60	97.55	97.73	97.89	98.14	97.93	97.97	97.83	97.79	97.90	97.82	97.61	97.86	98.01	97.97	98.10	98.35	98.10
31	98.11	97.86	97.35	97.42	97.61	97.95	98.15	97.96	97.95	97.65	97.66	97.80	97.66	97.45	97.74	97.92	97.96	98.14	98.35	98.22
32	98.11	97.86	97.35	97.42	97.61	97.95	98.15	97.96	97.95	97.65	97.66	97.80	97.66	97.45	97.74	97.92	97.96	98.14	98.35	98.22
33	98.11	97.86	97.35	97.42	97.61	97.95	98.15	97.96	97.95	97.65	97.66	97.80	97.66	97.45	97.74	97.92	97.96	98.14	98.35	98.22
34	98.11	97.86	97.35	97.42	97.61	97.95	98.15	97.96	97.95	97.65	97.66	97.80	97.66	97.45	97.74	97.92	97.96	98.14	98.35	98.22
35	98.11	97.86	97.35	97.42	97.61	97.95	98.15	97.96	97.95	97.65	97.66	97.80	97.66	97.45	97.74	97.92	97.96	98.14	98.35	98.22
36	97.69	97.54	97.10	97.21	97.35	97.74	97.80	97.66	97.65	97.56	97.74	97.66	97.60	97.29	97.51	97.62	97.65	97.91	97.95	97.88
37	97.69	97.54	97.10	97.21	97.35	97.74	97.80	97.66	97.65	97.56	97.74	97.66	97.60	97.29	97.51	97.62	97.65	97.91	97.95	97.88
38	97.75	97.60	97.23	97.25	97.43	97.77	97.68	97.72	97.67	97.42	97.68	97.58	97.46	97.35	97.35	97.44	97.65	97.83	98.04	97.96
39	97.65	97.41	97.20	97.29	97.47	97.69	97.81	97.71	97.74	97.18	97.75	97.49	97.29	97.12	97.18	97.34	97.59	97.83	98.05	97.93
40	97.57	97.40	97.10	97.07	97.32	97.54	97.80	97.68	97.76	97.33	97.37	97.41	97.33	97.11	97.07	97.24	97.68	97.64	97.97	98.04
41	97.20	97.13	96.72	96.87	96.99	97.29	97.61	97.47	97.47	97.30	97.16	97.18	97.21	96.86	97.07	97.34	97.35	97.55	97.82	97.66
42	97.21	97.07	96.87	97.14	97.13	97.41	97.40	97.61	97.54	97.25	97.64	97.50	97.41	97.04	97.23	97.28	97.58	97.46	97.73	97.89
43	97.21	97.07	96.87	97.14	97.13	97.41	97.40	97.61	97.54	97.25	97.64	97.50	97.41	97.04	97.23	97.28	97.58	97.46	97.73	97.89
44	96.83	96.80	96.62	96.99	97.03	97.43	97.28	97.55	97.56	97.05	97.73	97.37	97.25	96.89	97.41	97.51	97.48	97.45	97.65	97.68
45	96.97	97.08	96.88	97.36	97.39	97.44	97.68	97.70	97.75	97.68	97.81	97.47	97.39	97.12	97.25	97.48	97.40	97.41	97.85	97.64
46	96.98	96.86	96.72	97.03	97.01	97.21	97.63	97.55	97.64	97.57	97.58	97.31	97.26	96.90	97.10	97.33	97.37	97.36	97.58	97.54
47	96.98	96.86	96.72	97.03	97.01	97.21	97.63	97.55	97.64	97.57	97.58	97.31	97.26	96.90	97.10	97.33	97.37	97.36	97.58	97.54
48	96.90	96.88	96.77	96.98	97.03	97.20	97.67	97.37	97.36	97.26	97.43	97.15	97.15	96.86	96.95	97.15	97.33	97.49	97.77	97.33
49	96.83	96.64	96.52	96.70	96.80	97.11	97.55	97.17	97.33	97.09	97.35	96.99	96.90	96.75	96.91	96.97	97.17	97.53	97.57	97.29
50	96.83	96.64	96.52	96.70	96.80	97.11	97.55	97.17	97.33	97.09	97.35	96.99	96.90	96.75	96.91	96.97	97.17	97.53	97.57	97.29

Nombre de partitions pour les étiquettes des sommets	Nombre de partitions pour les étiquettes des arêtes									
	41	42	43	44	45	46	47	48	49	50
2	98.46	98.43	98.40	98.48	98.03	98.04	97.89	97.43	98.16	97.73
3	98.56	98.45	98.42	98.49	98.04	98.05	97.90	97.43	98.16	97.74
4	98.56	98.45	98.42	98.49	98.04	98.05	97.90	97.43	98.16	97.74
5	98.57	98.45	98.42	98.47	98.16	98.08	97.81	97.53	98.15	97.58
6	98.57	98.45	98.42	98.47	98.16	98.08	97.81	97.53	98.15	97.58
7	98.75	98.56	98.55	98.63	98.18	98.02	97.92	97.73	98.20	97.74
8	98.65	98.51	98.52	98.58	98.16	98.17	98.22	97.74	98.18	97.80
9	98.53	98.35	98.52	98.49	98.15	98.23	98.11	97.95	98.00	97.59
10	98.57	98.45	98.61	98.49	98.14	98.21	98.14	97.93	98.00	97.56
11	98.57	98.45	98.61	98.49	98.14	98.21	98.14	97.93	98.00	97.56
12	98.60	98.57	98.57	98.60	98.21	98.17	98.13	97.65	98.12	97.89
13	98.60	98.57	98.57	98.60	98.21	98.17	98.13	97.65	98.12	97.89
14	98.65	98.42	98.59	98.69	98.33	98.24	98.36	97.72	98.16	97.73
15	98.75	98.55	98.65	98.69	98.21	98.21	98.24	97.70	98.05	97.69
16	98.68	98.61	98.62	98.71	98.22	98.32	98.29	97.73	98.16	97.96
17	98.46	98.44	98.64	98.55	98.32	98.18	98.17	97.64	98.15	97.91
18	98.67	98.39	98.64	98.57	98.21	98.04	98.31	97.66	98.22	97.75
19	98.67	98.39	98.64	98.57	98.21	98.04	98.31	97.66	98.22	97.75
20	98.64	98.52	98.45	98.47	98.35	98.14	98.24	97.69	98.12	97.82
21	98.64	98.52	98.45	98.47	98.35	98.14	98.24	97.69	98.12	97.82
22	98.46	98.38	98.41	98.34	98.45	98.16	98.06	97.64	97.98	97.65
23	98.55	98.35	98.21	98.31	98.35	98.11	98.21	97.69	98.04	97.72
24	98.40	98.29	98.06	98.26	98.27	98.16	98.30	97.68	97.95	97.63
25	98.40	98.29	98.06	98.26	98.27	98.16	98.30	97.68	97.95	97.63
26	98.41	98.35	98.14	98.28	98.21	98.06	98.35	97.61	97.96	97.84
27	98.45	98.45	98.18	98.25	98.35	98.33	98.15	97.60	97.89	97.85
28	98.39	98.25	98.04	98.16	97.96	97.87	97.85	97.38	97.76	97.76
29	98.39	98.25	98.04	98.16	97.96	97.87	97.85	97.38	97.76	97.76
30	98.26	98.21	98.09	98.07	97.95	98.00	97.75	97.45	97.87	97.59
31	98.19	98.31	98.05	97.97	98.06	98.01	98.05	97.32	97.75	97.55
32	98.19	98.31	98.05	97.97	98.06	98.01	98.05	97.32	97.75	97.55
33	98.19	98.31	98.05	97.97	98.06	98.01	98.05	97.32	97.75	97.55
34	98.19	98.31	98.05	97.97	98.06	98.01	98.05	97.32	97.75	97.55
35	98.19	98.31	98.05	97.97	98.06	98.01	98.05	97.32	97.75	97.55
36	97.96	98.10	97.95	97.83	98.06	98.01	97.77	97.25	97.45	97.41
37	97.96	98.10	97.95	97.83	98.06	98.01	97.77	97.25	97.45	97.41
38	98.10	98.05	97.85	97.66	97.97	97.88	97.66	97.33	97.56	97.32
39	97.93	98.01	97.73	97.70	98.03	97.75	97.64	97.25	97.41	97.53
40	97.97	97.84	97.61	97.63	97.67	98.14	97.81	97.12	97.39	97.12
41	97.75	97.71	97.51	97.75	97.83	97.80	97.76	97.25	97.13	97.05
42	97.91	97.90	97.48	97.70	98.00	97.98	98.06	97.20	97.43	97.23
43	97.91	97.90	97.48	97.70	98.00	97.98	98.06	97.20	97.43	97.23
44	97.69	97.67	97.47	97.43	97.82	97.60	97.97	97.01	97.22	96.91
45	97.86	97.89	97.29	97.37	97.77	97.86	97.84	97.28	97.55	97.36
46	97.93	97.73	97.15	97.30	97.64	97.57	97.73	97.28	97.44	97.23
47	97.93	97.73	97.15	97.30	97.64	97.57	97.73	97.28	97.44	97.23
48	97.72	97.75	97.11	97.18	97.52	97.50	97.36	97.20	97.20	97.25
49	97.56	97.61	97.06	97.05	97.50	97.41	97.55	97.12	97.15	97.17
50	97.56	97.61	97.06	97.05	97.50	97.41	97.55	97.12	97.15	97.17

Letter

Nombre de partitions pour les étiquettes des sommets																		
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
80.43	89.89	93.26	93.59	93.81	94.21	94.36	95.79	96.22	96.41	96.43	96.33	96.33	96.49	96.69	97.23	97.18	96.84	97.1

Nombre de partitions pour les étiquettes des sommets																		
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
97.56	97.63	96.92	96.95	96.93	97.07	96.78	97.09	97.15	96.61	96.48	96.67	96.64	96.59	96.32	96.19	96.15	96.55	96.65	96.61

Nombre de partitions pour les étiquettes des sommets									
41	42	43	44	45	46	47	48	49	50
96.53	96.73	96.57	96.49	96.21	95.85	95.98	96.07	95.84	96.25

Bibliographie

[Aho *et al.*, 1974] Aho, A., Hopcroft, J. et Ullman, J. (1974). *The design and analysis of computer algorithms*. Addison-Wesley series in computer science and information processing. Addison-Wesley Pub. Co.

[Almohamad et Duffuaa, 1993] Almohamad, H. A. et Duffuaa, S. O. (1993). A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Ambauen *et al.*, 2003] Ambauen, R., Fischer, S. et Bunke, H. (2003). Graph edit distance with node splitting and merging, and its application to diatom identification. *In International Workshop on Graph-Based Representations in Pattern Recognition*.

[Apt, 2003] Apt, K. R. (2003). *Principles of constraint programming*. Cambridge University Press.

[Arthur et Vassilvitskii, 2007] Arthur, D. et Vassilvitskii, S. (2007). k-means++ : the advantages of careful seeding. *In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*.

[Babai *et al.*, 1982] Babai, L., Grigoryev, D. Y. et Mount, D. M. (1982). Isomorphism of graphs with bounded eigenvalue multiplicity. *In Proceedings of the fourteenth annual ACM symposium on Theory of computing*.

[Babai et Luks, 1983] Babai, L. et Luks, E. M. (1983). Canonical labeling of graphs. *In Symposium on Theory of Computing*.

[Barbu *et al.*, 2005] Barbu, E., Héroux, P., Adam, S. et Trupin, E. (2005). Clustering document images using a bag of symbols representation. *In International Conference on Document Analysis and Recognition*, pages 1216–1220.

[Belkasim *et al.*, 1991] Belkasim, S. O., Shridhar, M. et Ahmadi, M. (1991). Pattern recognition with moment invariants : A comparative study and new results. *Pattern Recognition*.

[Bellman, 1961] Bellman, R. E. (1961). *Adaptive control processes - A guided tour*. Princeton University Press.

[Berg *et al.*, 1984] Berg, C., Christensen, J. P. R. et Ressel, P. (1984). *Harmonic Analysis on Semigroups*. Springer.

[Berge, 1967] Berge, C. (1967). *Théorie des graphes et ses applications*. Dunod.

[Berge, 1973] Berge, C. (1973). *Graphes et hypergraphes*. Dunod.

[Berretti *et al.*, 2001] Berretti, S., Del Bimbo, A. et Vicario, E. (2001). Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Berztiss, 1973] Berztiss, A. T. (1973). A backtrack procedure for isomorphism of directed graphs. *Journal of the Association for Computing Machinery*.

[Blei *et al.*, 2003] Blei, D. M., Ng, A. Y. et Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*.

[Boeres *et al.*, 2004] Boeres, M. C., Ribeiro, C. C. et Bloch, I. (2004). A randomized heuristic for scene recognition by graph matching. *In* Ribeiro, C. C. et Martins, S. L., éditeurs : *WEA*, volume 3059 de *Lecture Notes in Computer Science*, pages 100–113. Springer.

[Bonev *et al.*, 2007] Bonev, B., Escolano, F., Lozano, M. A., Suau, P., Cazorla, M. A. et Aguilar, W. (2007). Constellations and the unsupervised learning of graphs. *In International Workshop on Graph-Based Representations in Pattern Recognition*.

[Borgelt, 2002] Borgelt, C. (2002). Mining molecular fragments : Finding relevant substructures of molecules. *In International Conference on Data Mining*.

[Brunner et Brunnett, 2004] Brunner, D. et Brunnett, G. (2004). Mesh segmentation using the object skeleton graph. *In Computer Graphics and Imaging*.

[Bunke, 1982] Bunke, H. (1982). Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Bunke, 1997] Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*.

[Bunke, 1999] Bunke, H. (1999). Error correcting graph matching : On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Bunke *et al.*, 2002] Bunke, H., Foggia, P., Guidobaldi, C., Sansone, C. et Vento, M. (2002). A comparison of algorithms for maximum common subgraph on randomly connected graphs. *In International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*.

[Bunke *et al.*, 2003] Bunke, H., Foggia, P., Guidobaldi, C. et Vento, M. (2003). Graph clustering using the weighted minimum common supergraph. *In International Workshop on Graph-Based Representations in Pattern Recognition*.

[Bunke et Riesen, 2007] Bunke, H. et Riesen, K. (2007). A family of novel graph kernels for structural pattern recognition. pages 20–31.

[Bunke et Riesen, 2008] Bunke, H. et Riesen, K. (2008). Graph classification based on dissimilarity space embedding. *In International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*.

[Bunke et Riesen, 2010] Bunke, H. et Riesen, K. (2010). Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition*.

[Bunke et Riesen, 2011] Bunke, H. et Riesen, K. (2011). Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*.

[Bunke et Shearer, 1998] Bunke, H. et Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*.

[Caelli et Kosinov, 2004] Caelli, T. et Kosinov, S. (2004). Inexact graph matching using eigen-subspace projection clustering. *International Journal of Pattern Recognition and Artificial Intelligence*.

[Carcassoni et Hancock, 2001] Carcassoni, M. et Hancock, E. R. (2001). Weighted graph-matching using modal clusters. *In Proceedings of the 9th International Conference on Computer Analysis of Images and Patterns*.

[Caruana et Niculescu-mizil, 2006] Caruana, R. et Niculescu-mizil, A. (2006). An empirical comparison of supervised learning algorithms. *In 23rd International Conference on Machine learning*.

[Celeux et Robert, 1993] Celeux, G. et Robert, C. (1993). Une histoire de discrétisation (avec discussion). *La Revue de Modulad*.

[Cesar jr *et al.*, 2005] Cesar jr, R. M., Bengoetxea, E., Bloch, I. et Larranaga, P. (2005). Inexact graph matching for model-based recognition : Evaluation and comparison of optimization algorithms. *Pattern Recognition*.

[Cesarini *et al.*, 1999] Cesarini, F., Gori, M., Marinai, S. et Soda, G. (1999). Structured document segmentation and representation by the modified X-Y tree. *In International Conference on Document Analysis and Recognition*.

[Champin et Solnon, 2003] Champin, P.-A. et Solnon, C. (2003). Measuring the similarity of labeled graphs. *In International Conference on Case-Based Reasoning*.

[Chung, 1997] Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

[Conte *et al.*, 2004] Conte, D., Foggia, P., Sansone, C. et Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*.

[Conte *et al.*, 2007] Conte, D., Foggia, P., Sansone, C. et Vento, M. (2007). How and why pattern recognition and computer vision applications use graphs. *In Applied Graph Theory in Computer Vision and Pattern Recognition*. Springer.

[Cook et Holder, 2000] Cook, D. J. et Holder, L. B. (2000). Graph-based data mining. *IEEE Intelligent Systems*.

[Cook et Holder, 2006] Cook, D. J. et Holder, L. B. (2006). *Mining Graph Data*. John Wiley & Sons.

[Cordella *et al.*, 1999] Cordella, L. P., Foggia, P., Sansone, C. et Vento, M. (1999). Performance evaluation of the vf graph matching algorithm. *In Proceedings of the 10th International Conference on Image Analysis and Processing*.

[Cordella *et al.*, 2000] Cordella, L. P., Foggia, P., Sansone, C. et Vento, M. (2000). Fast graph matching for detecting cad image components. *In International Conference on Pattern Recognition*.

[Cordella *et al.*, 2001] Cordella, L. P., Foggia, P., Sansone, C. et Vento, M. (2001). An improved algorithm for matching large graphs. *In International Workshop on Graph-Based Representations in Pattern Recognition*.

[Corneil et Gotlieb, 1970] Corneil, D. G. et Gotlieb, C. C. (1970). An efficient algorithm for graph isomorphism. *Journal of the Association for Computing Machinery*, 17(1):51–64.

[Cover et Hart, 1967] Cover, T. et Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*.

[Damiand *et al.*, 2004] Damiand, G., Bertrand, Y. et Fiorio, C. (2004). Topological model for two-dimensional image representation : Definition and optimal extraction algorithm. *Computer Vision and Image Understanding*.

[Dechter, 2003] Dechter, R. (2003). *Constraint processing*. Morgan Kaufmann Publishers.

[Delaunay, 1934] Delaunay, B. (1934). Sur la sphère vide. à la mémoire de georges voronoï. *Izvestia Akademii Nauk SSSR Otdelenie Matematicheskii i Estestvennyka Nauk*.

[Deruyver *et al.*, 2005] Deruyver, A., Hodé, Y., Laemmer, E. et Jolion, J.-M. (2005). Adaptive pyramid and semantic graph : Knowledge driven segmentation. *In International Workshop on Graph-Based Representations in Pattern Recognition*.

[Dickinson *et al.*, 2004] Dickinson, P. J., Kraetzl, M., Bunke, H., Neuhaus, M. et Dadej, A. (2004). Similarity measures for hierarchical representations of graphs with unique node labels. *International Journal of Pattern Recognition and Artificial Intelligence*.

[Dooms *et al.*, 2005] Dooms, G., Deville, Y. et Dupont, P. (2005). Constrained metabolic network analysis : Discovering pathways using cp(graph). *In Workshop on Constraint Based Methods for Bioinformatics*.

[Dosch et Valveny, 2005] Dosch, P. et Valveny, E. (2005). Report on the second symbol recognition contest. *In International Workshop on Graphics Recognition*.

[Dougherty *et al.*, 1995] Dougherty, J., Kohavi, R. et Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *In Proceedings of the Twelfth International Conference on Machine Learning*.

[Duda *et al.*, 2001] Duda, R. O., Hart, P. E. et Stork, D. G. (2001). *Pattern Classification*. Wiley.

[Ehrig *et al.*, 1992] Ehrig, H., Habel, A. et Kreowski, H.-J. (1992). Introduction to graph grammars with applications to semantic networks. *Computers, Mathematics with Applications*.

[Eshera et Fu, 1984] Eshera, M. et Fu, K. (1984). A graph distance measure for image analysis. 14(3):398–408.

[Fayyad et Irani, 1993] Fayyad, U. et Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. pages 1022–1027.

[Fei-Fei et Perona, 2005] Fei-Fei, L. et Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. *In Conference on Computer Vision and Pattern Recognition*.

[Fischer et Meinl, 2004] Fischer, I. et Meinl, T. (2004). Graph based molecular data mining - an overview. *In SMC (5)'04*, pages 4578–4582.

[Fischler et Elschlager, 1973] Fischler, M. et Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*.

[Foggia *et al.*, 2001] Foggia, P., Sansone, C. et Vento, M. (2001). A performance comparison of five algorithms for graph isomorphism. *Internation Workshop on Graph-based Representations in Pattern Recognition*.

[Franco *et al.*, 2003] Franco, P., Ogier, J.-M., Loonis, P. et Mullot, R. (2003). A topological measure for image object recognition. *In International Workshop on Graphics Recognition*.

[Freeman, 1961] Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*.

[Garey et Johnson, 1979] Garey, M. R. et Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman.

[Gärtner, 2003] Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explorations*.

[Gärtner *et al.*, 2003] Gärtner, T., Flach, P. et Wrobel, S. (2003). On graph kernels : Hardness results and efficient alternatives. *In* Schölkopf, B. et Warmuth, M. K., éditeurs : *Proceedings of the 16th Annual Conference on Computational Learning Theory*.

[Godsil et Royle, 2001] Godsil, C. et Royle, G. (2001). *Algebraic Graph Theory*. Springer.

[Gold et Rangarajan, 1996] Gold, S. et Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Gori *et al.*, 2005] Gori, M., Maggini, M. et Sarti, L. (2005). Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27.

[Guyon et Elisseeff, 2003] Guyon, I. et Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*.

[Haj et Aghbari, 2008] Haj, R. et Aghbari, Z. A. (2008). Semantic segmentation tree for image content representation. *International Journal of Virtual Technology and Multimedia*.

[Harchaoui, 2007] Harchaoui, Z. (2007). Image classification with segmentation graph kernels. *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

[Harris et Stephens, 1988] Harris, C. et Stephens, M. (1988). A combined corner and edge detection. *In Proceedings of The Fourth Alvey Vision Conference*.

[Hart *et al.*, 1968] Hart, P., Nilsson, N. et Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems, Man and Cybernetics*.

[Haussler, 1999] Haussler, D. (1999). Convolution kernels on discrete structures. Rapport technique, University of California, Santa Cruz.

[Héroux *et al.*, 2000] Héroux, P., Trupin, E. et Lecourtier, Y. (2000). Structural classification for retrospective conversion of documents. *In International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. *In 22nd international ACM SIGIR conference on Research and development in information retrieval*.

[Hopcroft et Wong, 1974] Hopcroft, J. E. et Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs (preliminary report). *In Proceedings of the sixth annual ACM symposium on Theory of computing*.

[Jain *et al.*, 2000] Jain, A. K., Duin, R. P. et Mao, J. (2000). Statistical pattern recognition : A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Jain *et al.*, 1999] Jain, A. K., Murty, M. N. et Flynn, P. J. (1999). Data clustering : a review. *ACM Computing Surveys*.

[Jaromczyk et Toussaint, 1992] Jaromczyk, J. et Toussaint, G. (1992). Relative neighborhood graphs and their relatives. *In Proceedings of the IEEE*.

[Jia et Abe, 1998] Jia, J. et Abe, K. (1998). Automatic generation of prototypes in 3d structural object recognition. *In International Conference on Pattern Recognition*.

[Jiang et Bunke, 2002] Jiang, X. et Bunke, H. (2002). Optimal lower bound for generalized median problems in metric space. *In International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*.

[Jiang *et al.*, 2001] Jiang, X., Munger, A. et Bunke, H. (2001). On median graphs : Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Jolion, 2003] Jolion, J.-M. (2003). On the deviation of a set of strings. *Pattern Analysis and Applications*.

[Jouili, 2011] Jouili, S. (2011). *Indexation de masses de documents graphiques : approches structurelles*. Thèse de doctorat, Université Nancy II.

[Kashima *et al.*, 2003] Kashima, H., Tsuda, K. et Inokuchi, A. (2003). Marginalized kernels between labeled graphs. *In ICML*, pages 321–328.

[Kazius *et al.*, 2005] Kazius, J., McGuire, R. et Bursi, R. (2005). Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320.

[Köbler *et al.*, 1993] Köbler, J., Schöning, U. et Torán, J. (1993). *The graph isomorphism problem : its structural complexity*. Birkhauser Verlag.

[Kosinov et Caelli, 2002] Kosinov, S. et Caelli, T. (2002). Inexact multisubgraph matching using graph eigenspace and clustering models. *In In Proceedings of SSPR/SPR*, pages 133–142. Springer.

[Kotsiantis, 2007] Kotsiantis, S. B. (2007). Supervised machine learning : A review of classification techniques. *Informatica*.

[Kreher et Stinson, 1999] Kreher, D. L. et Stinson, D. (1999). *Combinatorial algorithms : generation, enumeration, and search*. CRC Press.

[Kuncheva, 2004] Kuncheva, L. I. (2004). *Combining Pattern Classifiers : Methods and Algorithms*. Wiley-Interscience.

[Larrosa et Valiente, 2002] Larrosa, J. et Valiente, G. (2002). Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Sciences*.

[Lazebnik *et al.*, 2006] Lazebnik, S., Schmid, C. et Ponce, J. (2006). Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. *In Conference on Computer Vision and Pattern Recognition*.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*.

[Levi, 1974] Levi, G. (1974). Graph isomorphism : A heuristic edge-partitioning-oriented algorithm. *Computing*.

[Lippmann, 1987] Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE Acoustic, Speech and Signal Processing Magazine*.

[Liu *et al.*, 2004] Liu, C.-L., Jaeger, S. et Nakagawa, M. (2004). Online recognition of chinese characters : The state-of-the-art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Lladós *et al.*, 2001] Lladós, J., Marti, E. et Villanueva, J. J. (2001). Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Lopresti et Wilfong, 2003] Lopresti, D. et Wilfong, G. (2003). A fast technique for comparing graph representations with applications to performance evaluation. *International Journal of Document Analysis and Recognition*.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

[Luks, 1980] Luks, E. M. (1980). Isomorphism of graphs of bounded valence can be tested in polynomial time. *In Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, pages 42–49, Washington, DC, USA. IEEE Computer Society.

[Luo *et al.*, 2003] Luo, B., Wilson, R. et Hancock, E. (2003). Spectral embedding of graphs. *Pattern Recognition*.

[MacQueen, 1967] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *In Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*.

[Maio et Maltoni, 1996] Maio, D. et Maltoni, D. (1996). A structural approach to fingerprint classification. *In International Conference on Pattern Recognition*.

[Marini *et al.*, 2007] Marini, S., Spagnuolo, M. et Falcidieno, B. (2007). Structural shape prototypes for the automatic classification of 3d objects. *IEEE Computer Graphics and Applications*.

[Mathon, 1978] Mathon, R. (1978). Sample graphs for isomorphism testing. *Congressus Numerantium*.

[Matousek et Nesetril, 2004] Matousek, J. et Nesetril, J. (2004). *Introduction aux mathématiques discrètes*. Collection IRIS.

[McGregor, 1982] McGregor, J. J. (1982). Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*.

[McKay, 1981] McKay, B. D. (1981). Practical graph isomorphism. *Congressus Numerantium*.

[McKay, 2004] McKay, B. D. (2004). The nauty page.

[Min, 2009] Min, H. (2009). A global discretization and attribute reduction algorithm based on k-means clustering and rough sets theory. *International Symposium on Knowledge Acquisition and Modeling*, 2:92–95.

[Montes-y Gomez *et al.*, 2000] Montes-y Gomez, M., Lopez-Lopez, A. et Gelbukh, A. F. (2000). Information retrieval with conceptual graph matching. *In International Conference on Database and Expert Systems Applications*.

[Moravec, 1977] Moravec, H. P. (1977). Towards automatic visual obstacle avoidance. *In International Joint Conference on Artificial Intelligence*.

[Morris *et al.*, 1986] Morris, O., Lee, M. J. et Constantinides, A. (1986). Graph theory for image analysis : an approach based on the shortest spanning tree. *Communications, radar and signal processing*.

[Motzkin et Straus, 1965] Motzkin, T. S. et Straus, E. G. (1965). Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*.

[Nadler et Smith, 1993] Nadler, M. et Smith, E. (1993). *Pattern recognition engineering*. Wiley.

[Nagy et Seth, 1984] Nagy, G. et Seth, S. (1984). Hierarchical representation of optically scanned documents. *In International Conference on Pattern Recognition*.

[Neuhaus et Bunke, 2004] Neuhaus, M. et Bunke, H. (2004). A probabilistic approach to learning costs for graph edit distance. *In Proceedings of the 17th International Conference on Pattern Recognition*.

[Neuhaus et Bunke, 2006] Neuhaus, M. et Bunke, H. (2006). Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*.

[Neuhaus et Bunke, 2007] Neuhaus, M. et Bunke, H. (2007). Automatic learning of cost functions for graph edit distance. *Information Sciences*.

[Neuhaus *et al.*, 2006] Neuhaus, M., Riesen, K. et Bunke, H. (2006). Fast suboptimal algorithms for the computation of graph edit distance. *In SSPR/SPR*, volume 4109 de *Lecture Notes in Computer Science*, pages 163–172. Springer.

[Novak et Shafer, 1992] Novak, C. L. et Shafer, S. A. (1992). Anatomy of a color histogram. *In IEEE Conference on Computer Vision and Pattern Recognition*.

[Papadopoulos et Manolopoulos, 1999] Papadopoulos, A. N. et Manolopoulos, Y. (1999). Structure-based similarity search with graph histograms. *In International Workshop on Database and Expert Systems Applications*.

[Pavlidis, 1972] Pavlidis, T. (1972). Representation of figures by labeled graphs. *Pattern Recognition*.

[Pekalska et Duin, 2005] Pekalska, E. et Duin, R. P. W. (2005). *The Dissimilarity Representation for Pattern Recognition : Foundations And Applications*. World Scientific Publishing.

[Pelillo, 1999] Pelillo, M. (1999). Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*.

[Pelillo, 2004] Pelillo, M. (2004). Metrics for attributed graphs based on the maximal similarity common subgraph. *International Journal of Pattern Recognition and Artificial Intelligence*.

[Qureshi, 2008] Qureshi, R. J. (2008). *Reconnaissance de formes et symboles graphiques complexes dans les images de documents*. Thèse de doctorat, Université François Rabelais Tours.

[Rabaséda *et al.*, 1996] Rabaséda, S., Rakotomalala, R. et Sebban, M. (1996). A comparison of some contextual discretization methods. *Information Sciences*.

[Rakotomalala, 1997] Rakotomalala, R. (1997). *Graphes d'induction*. Thèse de doctorat, Université Claude Bernard, Lyon I.

[Ramel, 2009] Ramel, J.-Y. (2009). Graphes en analyse d'images : un problème fortement combinatoire. Séminaire dans le cadre du groupe "Architecture et Images" du GDR ISIS.

[Ramel *et al.*, 2000] Ramel, J.-Y., Vincent, N. et Emptoz, H. (2000). A structural representation for understanding line-drawing images. *International Journal On Document Analysis and Recognition*.

[Read et Corneil, 1977] Read, R. C. et Corneil, D. G. (1977). The graph isomorphism disease. *Journal of Graph Theory*.

[Reingold *et al.*, 1977] Reingold, E., Nievergelt, J. et Deo, N. (1977). *Combinatorial algorithms : theory and practice*. Prentice-Hall.

[Reis *et al.*, 2004] Reis, D. C., Golgher, P. B., Silva, A. S. et Laender, A. F. (2004). Automatic web news extraction using tree edit distance. *In International conference on World Wide Web*.

[Riesen, 2009] Riesen, K. (2009). *Classification and Clustering of Vector Space Embedded Graphs*. Thèse de doctorat, Universität Bern.

[Riesen et Bunke, 2008] Riesen, K. et Bunke, H. (2008). Iam graph database repository for graph based pattern recognition and machine learning. *In International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*.

[Riesen *et al.*, 2007] Riesen, K., Neuhaus, M. et Bunke, H. (2007). Graph embedding in vector spaces by means of prototype selection. *In International Workshop on Graph-Based Representations in Pattern Recognition*, pages 383–393.

[Robles Kelly et Hancock, 2007] Robles Kelly, A. et Hancock, E. (2007). A riemannian approach to graph embedding. *Pattern Recognition*.

[Rocha et Pavlidis, 1994] Rocha, J. et Pavlidis, T. (1994). A shape analysis model with applications to a character recognition system. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 16:393–404.

[Ros *et al.*, 2005] Ros, J., Laurent, C., Jolion, J.-M. et Simand, I. (2005). Comparing string representations and distances in a natural images classification task. *In International Workshop on Graph-Based Representations in Pattern Recognition*.

[Rosenblatt et Laboratory, 1958] Rosenblatt, F. et Laboratory, C. A. (1958). The perceptron : a theory of statistical separability in cognitive systems (project para). Rapport technique, Cornell Aeronautical Laboratory.

[Rosenfeld, 1974] Rosenfeld, A. (1974). Adjacency in digital pictures. *Information and Computation / Information and Control*.

[Rosenstiehl, 2002] Rosenstiehl, P. (2002). Claude berge, ses graphes et hypergraphes. Mathématiques et sciences humaines.

[Rota Bulò *et al.*, 2009] Rota Bulò, S., Torsello, A. et Pelillo, M. (2009). A game-theoretic approach to partial clique enumeration. *Image and Vision Computing*.

[Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1986). Learning internal representations by error propagation. *In Parallel distributed processing : explorations in the microstructure of cognition*, volume 1. MIT Press.

[Salton, 1971] Salton, G. (1971). *The SMART Retrieval System : Experiments in Automatic Document Processing*. Prentice-Hall, Inc.

[Salton et Lesk, 1968] Salton, G. et Lesk, M. E. (1968). Computer evaluation of indexing and text processing. *Journal of the Association for Computing Machinery*.

[Sanfeliu et Fu, 1983] Sanfeliu, A. et Fu, K.-S. (1983). A Distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*.

[Sanromà *et al.*, 2010] Sanromà, G., Alquézar, R. et Serratosa, F. (2010). Attributed graph matching for image-features association using sift descriptors. *In International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*.

[Saunders et Demco, 2007] Saunders, C. et Demco, A. (2007). Kernels for strings and graphs. *In* Hammer, B. et Hitzler, P., éditeurs : *Perspectives of Neural-Symbolic Integration*, volume 77 de *Studies in Computational Intelligence*, chapitre 1, pages 7–22. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Schenker *et al.*, 2005] Schenker, A., Bunke, H., Last, M. et Kandel, A. (2005). *Graph-Theoretic Techniques for Web Content Mining*. World Scientific Publishing.

[Scholkopf et Smola, 2002] Scholkopf, B. et Smola, A. J. (2002). *Learning with kernels : support vector machines, regularization, optimization, and beyond*. The MIT Press.

[Selkow, 1977] Selkow, S. (1977). The tree-to-tree editing problem. *Information Processing Letters*.

[Shawe-Taylor et Cristianini, 2004] Shawe-Taylor, J. et Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

[Shokoufandeh et Dickinson, 2001] Shokoufandeh, A. et Dickinson, S. J. (2001). A unified framework for indexing and matching hierarchical shape structures. *In International Workshop on Visual Form*.

[Sivic *et al.*, 2005] Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A. et Freeman, W. T. (2005). Discovering Object Categories in Image Collections. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Solnon, 2010] Solnon, C. (2010). Alldifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*.

[Sorlin *et al.*, 2007] Sorlin, S., Solnon, C. et Jolion, J.-M. (2007). A generic graph distance measure based on multivalent matchings. *In Applied Graph Theory in Computer Vision and Pattern Recognition*. Springer.

[Stoica, 2011] Stoica, A. R. (2011). Delaunay diagram representations for use in image near-duplicate detection. Bard Digital Commons.

[Suk et Oh, 1986] Suk, M. et Oh, S. (1986). Region adjacency and its application to object detection. *Pattern Recognition*.

[Swain et Ballard, 1991] Swain, M. J. et Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*.

[Tai, 1979] Tai, K.-C. (1979). The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*.

[Tsai et Fu, 1979] Tsai, W. et Fu, K. (1979). Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transaction on Systems, Man and Cybernetics*.

[Tsai et Fu, 1983] Tsai, W. H. et Fu, K. S. (1983). Subgraph Error-Correcting Isomorphism for Syntactic Pattern Recognition. *IEEE Transactions on Systems, Man and Cybernetics*.

[Tupin, 2006] Tupin, F. (2006). Graphes en analyse d'images, vision, reconnaissance. Support de Cours.

[Ullmann, 1976] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42.

[Umeyama, 1988] Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of dependencies based on empirical data*. Springer-Verlag.

[Vapnik, 1995] Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

[Vapnik, 1998] Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

[Vapnik et Chervonenkis, 1971] Vapnik, V. N. et Chervonenkis, A. Y. (1971). On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*.

[Verley, 2002] Verley, J.-L. (2002). Espaces métriques. *In Dictionnaire des mathématiques : algèbre, analyse, géométrie*. Albin Michel.

[Viola et Jones, 2001] Viola, P. et Jones, M. (2001). Robust real-time object detection. *In International Journal of Computer Vision*.

[Watanabe, 1985] Watanabe, S. (1985). *Pattern recognition : human and mechanical*. Wiley.

[Weisfeiler, 1976] Weisfeiler, B. (1976). *On Construction and identification of graphs*. (Lecture Notes in mathematics). Springer-Verlag.

[Weiss et Kulikowski, 1991] Weiss, S. M. et Kulikowski, C. A. (1991). *Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc.

[Wiener, 1947] Wiener, H. (1947). Structural determination of paraffin boiling points. *Journal of the American Chemical Society*.

[Wilson et Hancock, 1997] Wilson, R. C. et Hancock, E. R. (1997). Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:634–648.

[Wilson *et al.*, 2005] Wilson, R. C., Hancock, E. R. et Luo, B. (2005). Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Winston, 1970] Winston, P. (1970). Learning structural descriptions from examples. Rapport technique, Massachusetts Institute of Technology.

[Wong *et al.*, 1990] Wong, A., You, M. et Chan, S. (1990). An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 20.

[Xu et Wunsch, 2005] Xu, R. et Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*.

Résumé :

Les travaux exposés dans cette thèse portent sur une contribution aux techniques de projection de graphes, appliquées à la reconnaissance de formes, visant à tirer parti de la richesse des méthodes structurelles et de l'efficacité des outils statistiques. Nous présentons une nouvelle projection s'inscrivant dans la catégorie des sondages de graphes. La première contribution de cette thèse porte sur l'encapsulation de la topologie du graphe dans une représentation vectorielle, en s'appuyant sur le dénombrement de motifs (sous-graphes) issus d'un lexique généré indépendamment du contexte. Ces motifs permettent de minimiser les pertes de l'information topologique lors de la projection. La deuxième contribution porte sur l'intégration de l'information relative aux étiquettes au sein de notre projection par l'adjonction de leurs dénombrements. Aux problèmes liés à la nature et la variabilité des attributs, nous proposons deux solutions dans le but de constituer des classes d'étiquettes moins nombreuses. La première consiste à discrétiser les attributs numériques puis à les combiner. La deuxième vise à former ces classes par un partitionnement global de l'ensemble des étiquettes. Ces propositions sont ensuite évaluées sur différentes bases de graphes et dans différents contextes.

Abstract :

The work exposed in this thesis focuses on a contribution to techniques of graph embedding, applied to pattern recognition, aiming to take advantages of the richness of structural methods and the efficiency of statistical tools. We present a new embedding, joining the category of graph probing. The first contribution of this thesis deals with the embedding of the graph topology in a vectorial representation, based on the counting of patterns (subgraphs) stemming of a lexicon generated independently of the context. These patterns permit the minimization of losses of the topological information during the embedding. The second contribution focuses on the integration of the information related to labels inside our embedding by adding their counting. To deal with problems linked to the nature and the variability of the attributes, we suggest two solutions to reduce the number of label classes. The first one consists of discretizing numeral attributes and combining them. The second one aims to build these classes by a global clustering on the set of labels. Then, these proposals are evaluated on different datasets of graphs and in different contexts.