Indexation de masses de documents graphiques : approches structurelles

Salim Jouili

Nancy-Université
Université Nancy 2

École doctorale IAEM Lorraine

UFR mathématiques et informatique **Département de formation doctorale en informatique**

Indexation de masses de documents graphiques : approches structurelles

THÈSE

présentée et soutenue publiquement le 30 Mars 2011

pour l'obtention du

Doctorat de l'université Nancy 2

(spécialité informatique)

par

Salim Jouili

Composition du jury

Rapporteurs :	Luc Brun	Professeur, École Nationale Supérieure d'Ingénieurs de Caen
	Christine Solnon	Maître de Conférences HDR, Université Lyon 1
Examinateurs :	Jean-Marc Ogier	Professeur, Université de La Rochelle
	Nicole Vincent	Professeur, Université Paris Descartes
	Ernest Valveny	Associate Professor, Universitat Autònoma de Barcelona
Invité :	Bernard Maigret	Directeur de recherche, CNRS, LORIA
Directeur de thèse :	Salvatore Tabbone	Professeur, Université Nancy 2

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

La première personne que je tiens à remercier est Salvatore-Antoine Tabbone, *mon chef*, sans qui je n'en serais certainement pas là. Je le remercie pour sa patience, sa compétence et sa modestie. Plus qu'un directeur de thèse ou un collègue, je crois avoir trouvé en lui un ami qui m'a aidé aussi bien dans le travail que dans la vie lorsque j'en avais besoin. Ce fut un réel plaisir de travailler ensemble et je souhaite vivement poursuivre cette collaboration.

Je remercie les rapporteurs de cette thèse Christine Solnon et Luc Brun pour la pertinence avec laquelle ils ont lu mon manuscrit et l'intérêt qu'ils ont porté à mon travail. Merci également aux autres membres du jury qui ont accepté de juger ce travail : Ernest Valveny, Nicole Vincent et Jean-Mar Ogier.

Je ne peux également oublier de remercier mes collègues de l'équipe QGAR, en particulier, un grand merci pour Philippe Dosch qui a relu attentivement tout ce manuscrit. Mes remerciements s'adressent également à Anis Mekki, Amine Bel Mabrouk et Mehdi Felhi pour leur bon humeur (et les pauses cafés).

Un grand merci à ma famille et en particulier à mes parents, Saâd Jouili et Hadhom Filali, qui m'ont toujours soutenu dans mes choix. Un grand merci pour mes deux anges gardiens (soeurs) Hayet et Mahbouba et mes frères, Mokhtar, Noureddine et Lazhar.

Un grand merci pour les amis que j'ai eus la chance d'avoir à mes côtés : Ammar Mahdhaoui, Mourad Bougares, Faouzi Hamdi, Fethi Bougares et Zied Lassoued.

Le mot de la fin sera pour Hana, ma femme. Merci à toi d'être et d'avoir toujours été là pour moi. Hana m'a épaulé alors qu'en fin de thèse nous ne parvenions au mieux qu'à nous croiser, ce qui est une magnifique preuve d'amour.

Je dédie cette thèse
à ma mère Hadhom,
à mon père Saâd,
à ma femme Hana.

Table des matières

II Classification et Indexation de graphes

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Introduction générale

Algorithms existed for at least five thousand years, but people did not know that they were algorithmizing. Then came Turing (and Post and Church and Markov and others) and formalized the notion.

Doron Zeilberger

1.1 Reconnaissance de formes

Durant des dizaines de millions d'années d'évolution, les systèmes cognitifs de l'être humain subissaient un développement continu de leurs capacités d'interprétation du flux d'information acquis par les sens [70]. Ces systèmes cruciaux pour la survie permettaient l'identification de visages, la compréhension de la parole, la lecture de l'écriture d'imprimerie ou manuscrite... En intelligence artificielle, la formalisation algorithmique de cette faculté de la perception humaine est à la base des fondements de la reconnaissance de formes. En effet, la reconnaissance de formes se manifeste dans la plupart des activités quotidiennes de chaque individu, comme, par exemple, la reconnaissance des lettres et des mots sur une enseigne publicitaire, la reconnaissance de son véhicule dans un parking, la compréhension d'une parole même en présence du bruit, la reconnaissance de l'odeur du tabac dans une cafétéria, et beaucoup plus. Dans chacun de ces exemples, les organes sensoriels envoient des signaux au cerveau qui les traite très rapidement. Ce traitement consiste à faire correspondre le signal reçu avec un ensemble de modèles déjà stockés en mémoire et décider auquel il correspond. En fait, les modèles stockés en mémoire sont des connaissances que chaque individu a appris. Un individu ne peut en effet identifier que ce qu'il connaît déjà. Il s'agit d'une implémentation neurobiologique de l'appariement de gabarit (*template-matching*) [182].

La discipline scientifique de la reconnaissance de formes vise à élaborer un ensemble de techniques informatiques capables de reproduire, et si possible de dépasser, les capacités humaines de perception, visuelles ou auditives. L'objectif est de concevoir et de développer des algorithmes qui puisse doter les machines de la capacité de la reconnaissance (identification)

des objets à partir de données brutes et l'affectation de chaque objet à une catégorie pertinente. Naturellement, la délégation de la reconnaissance de formes à des machines est particulièrement intéressante et utile pour traiter des tâches complexes dans la recherche scientifique et dans l'industrie. Des machines dotées d'une telle capacité trouveraient des nombreuses applications dans divers domaines comme la médecine, le contrôle de procédés de fabrication, la vision par ordinateur, la lecture optique de documents (e.g. chèques bancaires) et le traitement de données volumineuses d'images (e.g. satellites).

Fruit de plusieurs décennies de recherche, la reconnaissance de formes est, aujourd'hui, une partie intégrante de la plupart des systèmes d'intelligence artificielle conçus pour la prise de décision. En effet, très nombreuses applications de reconnaissance de formes sont maintenant fiables et robustes pour des problèmes considérés autrefois difficiles à traiter. Nous pouvons citer à titre d'exemples le tri automatique du courrier [90, 241], la recherche d'images par le contenu [243, 238, 262], la catégorisation de textes [223, 123, 281], l'identification de visages [287, 155, 46], la reconnaissance de l'écriture manuscrite [189, 144, 104] et la localisation de symboles dans des documents graphiques [179, 289, 210].

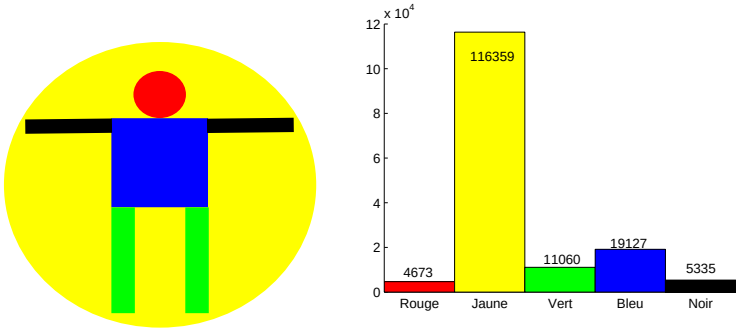
Généralement, un système de reconnaissance de formes procède en trois étapes [119]. La première étape, dite de pré-traitement, consiste à isoler la forme à reconnaître. Il s'agit assez souvent d'une phase de segmentation qui consiste à séparer l'objet concerné de l'arrière plan, précédée par une phase de débruitage qui consiste à supprimer le bruit en vue d'obtenir des objets aussi *propres* que possible. La seconde étape est une étape d'extraction de caractéristiques qui consiste à effectuer un certain nombre de mesures qui vont permettre de caractériser l'objet concerné. La troisième étape consiste à exploiter la description de caractéristiques alors obtenue une décision lors de l'étape de classification. La classification est une tâche essentielle en reconnaissance de formes. L'objectif de la classification est d'identifier les classes auxquelles appartiennent des objets à partir de caractéristiques. Dans la littérature, nous distinguons deux catégories de classificateurs. La première correspond à la classification supervisée où les classes sont connues et l'on dispose d'exemples de chaque classe. Typiquement, une phase, dite d'apprentissage, est effectuée sur ces connaissances *a priori*, appelés aussi base d'apprentissage, afin d'extraire des règles d'affectation. En se basant sur ces règles, l'algorithme de classification devient ainsi capable de déduire la classe d'un nouveau objet (*inconnu*). Alternativement, lorsque nous ne disposons pas de cette connaissance, la classification est dite non supervisée. Dans ce cas, il s'agit, sans aucune connaissance *a priori* ni sur le nombre de classes ni sur leur nature, de répartir les objets en plusieurs catégories au sens d'une meilleure compacité. Une façon de procéder consiste à minimiser la similarité intra-groupe et à maximiser la similarité inter-groupe.

Une notion fondamentale dans la reconnaissance de formes, peu importe la méthode de la classification, c'est la notion de la similarité. La similarité est utilisée en tant que mesure de ressemblance entre deux objets. La quantification de la similarité repose principalement sur la définition et la sélection des caractéristiques des objets impliquées dans l'estimation de la similarité. Les caractéristiques sont des mesures, des attributs ou des primitives extrait de l'objet et jugé pertinents par rapport à sa caractérisation. Ces caractéristiques constituent l'espace des descripteurs dans lequel l'estimation de la similarité est réalisée. En effet, un objet A est d'autant plus semblable à un objet B qu'il en est proche sur les caractéristiques qui décrivent ces deux objets. Du point de vue mathématique, c'est par les différences de distance mathématique entre

les caractéristiques de deux objets qu'on mesure leur degré de similarité. La description des objets joue donc un rôle décisif dans la procédure du calcul des distances entre les objets. Dans la littérature de la reconnaissance de formes, nous distinguons deux grandes familles d'approches de description des caractéristiques : les approches statistiques et les approches structurelles.

1.1.1 Approches statistiques

Dans les approches statistiques, chaque objet est représenté par un vecteur de caractéristiques. Concrètement, un objet est représenté par un vecteur de caractéristiques numérique de taille d. Formellement, chaque objet est considéré comme étant un point dans un espace vectoriel réel de dimension d (d-dimensionnel), i.e. **o** = (x_1,...,x_d) $\in \mathbb{R}^d$. À titre d'exemple, nous considérons l'image de la figure 1.1(a) et choisissons de procéder à l'extraction de caractéristiques relatives aux couleurs. L'histogramme de couleurs [248] est utilisé pour la représentation colorimétrique d'images. Un histogramme de couleurs d'une image est un vecteur à d éléments (x_1,...,x_d) avec d qui est le nombre de classes couleurs considérées et x_i qui dénombre la quantité de pixels de chaque couleur i. Dans la figure 1.1(b), l'histogramme de l'image est présenté avec cinq classes de couleurs (Rouge, Jaune, Vert, Bleu, Noir). Ainsi, le vecteur numérique V=$(4673, 116359, 11060, 19127, 5335)$ décrit l'image dans la figure 1.1(a) en se basant sur les caractéristiques de son information couleur.



(a) Image originale

(b) Histogramme de couleur de l'image originale. Chaque élément i quantifie le nombre de pixels ayant la couleur correspondante à i

FIGURE 1.1 – Exemple de la représentation sous forme d'un vecteur d'une image.

L'utilisation de vecteurs de caractéristiques a de nombreuses propriétés utiles. En effet, les distances entre vecteurs numériques, les notions de moyenne, de médiane et de centre sont intrinsèquement liées aux méthodes mathématiques que l'on peut appliquer dans un espace vectoriel. Toutefois, l'utilisation des vecteurs de caractéristiques a des limitations. En effet, dans une application de reconnaissance de formes, les vecteurs de caractéristiques nécessitent, à notre connaissance, des tailles identiques sans tenir compte de la taille et de la complexité d'objets. De plus, l'utilisation de vecteurs de caractéristiques ne permet pas de décrire les relations qui puissent exister entre les différentes parties d'un objet.

1.1.2 Approches structurelles

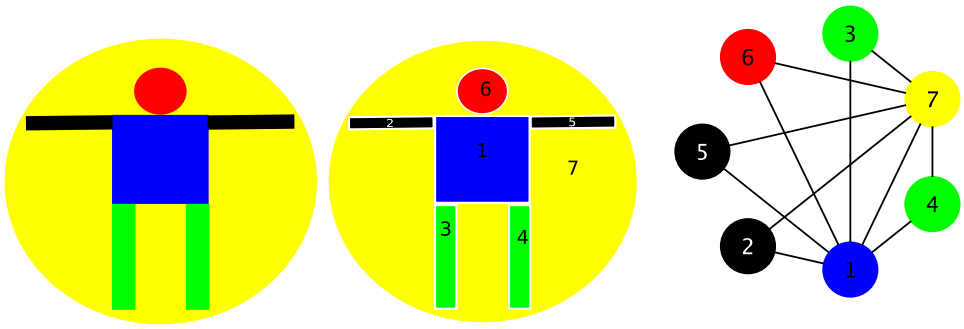
Selon le neuropsychologue canadien Donald Hebb [110] : le système cognitif humain*"n' appréhende pas une forme dans sa globalité, mais il perçoit ses parties comme des entités distinctes"* ayant une organisation spatiale spécifique qui joue un rôle important dans l'apprentissage et la reconnaissance. Partant de cette idée, les approches structurelles se basent sur une décomposition des formes en composants simples.

Dans les approches structurelles, chaque objet est représenté par une structure. Il s'agit ici de définir un modèle structurel qui décrit à la fois les différents composants de l'objet et les relations qui les lient. Cette représentation est caractérisée par une capacité représentative plus expressive que celle via des vecteurs caractéristiques [33]. Parmi les structures de données qui sont largement utilisées dans la reconnaissance de formes structurelle, nous distinguons particulièrement les chaînes, les arbres et les graphes. Du point de vue algorithmique, les graphes généralisent un large ensemble de structures de données [22, 198], en particulier, les chaînes et les arbres. En effet, une chaîne est un graphe dans lequel chaque nœud représente un caractère, et où les caractères consécutifs sont reliés par une arête. Un arbre est aussi un graphe dans lequel chaque paire de nœuds est connectée par un et un seul chemin. En outre, un vecteur peut être aussi représenté par un graphe où les nœuds correspondent aux éléments du vecteur. Par conséquent, les travaux sur les graphes sont applicables sur les autres structures de données.

La représentation d'objets sous forme de graphes offre deux avantages majeurs par rapport aux vecteurs caractéristiques. Premièrement, elle offre une riche représentation bien plus expressive que les vecteurs à travers la description explicite de la structure de l'objet par l'ensemble de ses sous-structures et les interactions entre elles. Deuxièmement, le nombre de nœuds et des arêtes n'est pas limité *a priori* et peut être adapté à la complexité de chaque objet. En outre, la comparaison de deux graphes de différentes tailles est possible. *A contrario*, dans la représentation en vecteur caractéristiques, la comparaison entre les objets requiert généralement des vecteurs de même dimension.

Grâce à ces propriétés, les graphes ont connu une large utilisation dans différents domaines scientifiques [52]. Dans la reconnaissance d'images, la représentation sous forme de graphe a particulièrement prouvé sa flexibilité sur une grande variété de types d'images (les documents anciens, les plans électriques et architecturaux, les images naturelles, les images médicaux ...). Les travaux qui ont eu recours aux graphes ont établi une représentation qui préserve l'information topographique de l'image ainsi que les relations entre les composantes. Actuellement, plusieurs approches dérivées des graphes pour l'analyse d'images ont été proposées, notamment pour les systèmes de recherche par contenu, la segmentation et l'indexation.

Dans la littérature, plusieurs travaux [29, 128, 133] optent pour la représentation d'images sous forme de graphe. À titre d'exemple, la figure 1.2(c) illustre une représentation de l'image originale (figure 1.2(a)) sous forme de graphe en se basant sur une segmentation en régions (figure 1.2(b)). Chaque nœud n_i du graphe représente alors une région de l'image et les arêtes représentent les relations d'adjacence entre les régions. Notons que des informations supplémentaires peuvent être ajoutées aux nœuds et/ou aux arêtes. Ces informations sont appelées des attributs. Dans chaque nœud, les attributs peuvent contenir des valeurs numériques ansi que des valeurs symboliques qui décrivent la région correspondante.



(a) Image originale (b) Image segmentée en régions (c) Représentation sous forme de graphe

FIGURE 1.2 – Exemple de représentation d'une image sous forme d'un graphe.

1.2 Problématique et contributions

1.2.1 Contexte et contributions

Les travaux de cette thèse se situent dans la cadre des approches structurelles de reconnaissance de formes. Précisément, la gestion de bases d'images représentées sous forme structurelle, les graphes. Le choix de la représentation structurelle est justifié par la grande capacité représentative de cette structure des données (i.e. graphes) par rapport à la représentation statistique (i.e. vecteurs). La première étape qui intervient dans l'étude de l'application des graphes dans le domaine des images est de définir une stratégie d'extraction de graphes représentatifs. Ensuite, il faut définir des fonctions nécessaires à la manipulation des bases de graphes. L'une des fonctions cruciales pour manipuler les graphes est la fonction de calcul des distances entre graphes. En effet, le calcul de distances entre graphes est un problème ouvert dans la littérature. De plus, il est NP-difficile dans le cas général. La plupart des solutions proposées dans la littérature présentent différentes limites d'utilisation telle que la taille des graphes, la prise en compte d'attributs, le temps de calcul, etc. Dans cette thèse, nous proposons une nouvelle méthode de calcul de distance de graphes. Outre la distance, le domaine des graphes souffre d'un manque d'algorithmes de classification (non-)supervisée appropriés. Cette thèse présente un ensemble de contributions dont l'objectif est de proposer des algorithmes et des stratégies de classification (non-)supervisée appropriés au domaine des graphes. Ensuite, une méthode d'indexation de graphes est proposée. Cette méthode permet aussi bien l'indexation d'une base de graphes que la navigation dans une base d'images représentées sous forme de graphes.

La figure 1.3 illustre nos contributions à travers un système complet de reconnaissance d'images à base de graphes. À partir d'une base d'images, nous utilisons des méthodes de la littérature pour extraire les graphes représentatifs. Ensuite, les contributions de cette thèse sont appliquées sur ces graphes :

– En premier lieu, nous proposons une nouvelle approximation de la distance d'édition de graphes basée sur la notion de signature de nœuds.
– Ensuite, nous introduisons un algorithme de plongement de graphes. Cet algorithme con-

siste à représenter chaque graphe par un vecteur dans un espace euclidien. Ceci nous permet d'appliquer les algorithmes de classification des vecteurs sur les graphes par le biais du plongement.
– Dans le domaine de la classification non-supervisée (clustering), nous proposons un nouvel algorithme basé sur la notion du graphe médian et la notion du *mean-shift*.
– Enfin, nous proposons une nouvelle méthode d'indexation de graphes basée sur la structure d'hypergraphe. Outre l'indexation, nous introduisons une technique de navigation dans les bases d'images à travers la structure d'hypergraphe.

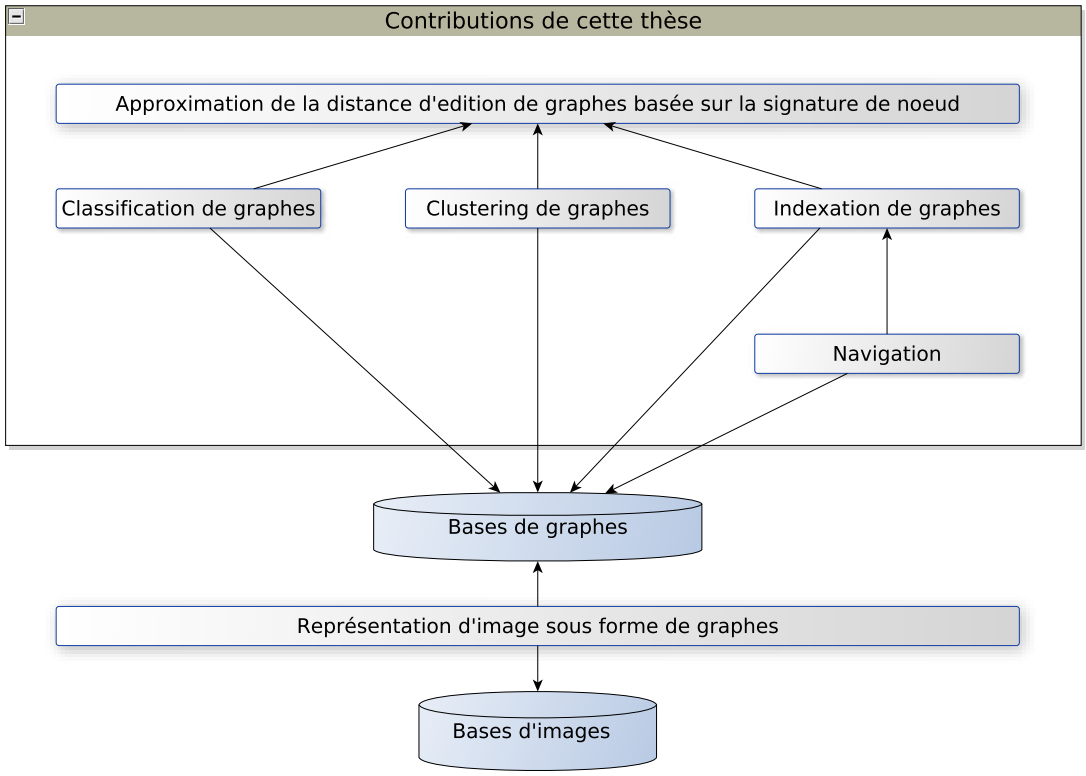


FIGURE 1.3 – Organisation de nos contributions au sein d'un système de reconnaissance de formes à base de graphes

1.2.2 Publications dans le cadre de la thèse

Chapitres d'ouvrage [1]

– S. Jouili, S. Tabbone, Graph Embedding Using Constant Shift Embedding *Recognizing Patterns in Signals, Speech, Images, and Videos*, LNCS 6388, pp. 83-92, Srpinger 2010.

1. Les deux chapitres sont publiés dans deux post-proceedings : le concours GEPR dans le cadre d'ICPR 2010 et le workshop GREC 2009

– S. Jouili, S. Tabbone and Ernest Valveny, Comparing Graph Similarity for Graphical Recognition, *GREC 2009*, LNCS 6020, pp. 37-48, Springer 2010.

Conférences internationales avec comité de lecture

– S. Jouili and S. Tabbone, A hypergraph-based model for graph clustering : application to image indexing, *The 13th International Conference on Computer Analysis of Images and Patterns (CAIP 2009)*, September 2nd-4th, Münster, Germany, LNCS 5702, pp. 360-368, Springer 2009.
– S. Jouili, I. Mili and S. Tabbone, Attributed Graph Matching using Local Descriptions, *Advanced Concepts for Intelligent Vision Systems - ACIVS 2009*, Bordeaux, France, LNCS 5807, pp. 89-99, Springer 2009.
– S. Jouili, M. Coustaty, S. Tabbone and J.M. Ogier, Navidomass : Structural-based approaches towards handling historical documents, *the 20th International Conference on Pattern Recognition*, Istanbul, Turkey, IEEE computer society 2010.
– S. Jouili, S. Tabbone and V. Lacroix, Median graph shift : A new clustering algorithm for graph domain, *the 20th International Conference on Pattern Recognition*, Istanbul, Turkey, IEEE computer society 2010.

Workshops internationals avec comité de lecture

– M. Coustaty, J.M. Ogier, N. Sidère, P. Héroux, J.Y. Ramel, H. Chouaib, N. Vincent, S. Jouili, S. Tabbone, Content-Based Old Documents Indexing, *8th IAPR International Workshop on Graphics Recognition (GREC 2009)*, July 22-23, La Rochelle, France, 2009
– S. Jouili and S. Tabbone, Graph Matching based on Node Signatures, *Proceedings of the 7th IAPR -TC-15 Workshop on Graph-based Representations in Pattern Recognition (GBR 2009)*, Venice, Italy, LNCS 5534, pp. 154-163, Springer 2009.
– S. Jouili, S. Tabbone, and E. Valveny, Evaluation of Graph Matching Measures for Documents Retrieval, *8th IAPR International Workshop on Graphics Recognition (GREC 2009)*, July 22-23, La Rochelle, France, 2009

Conférences nationales avec comité de lecture

– S. Jouili and S. Tabbone, Application des graphes en traitement des images. *International Conference on Relations, Orders and Graphs : Interaction with Computer Science (ROGICS 2008)*, Mahdia, Tunisia, 2008, pp. 434-442.
– S. Jouili and S. Tabbone, Indexation de graphes à partir d'une structure d'hypergraphe, *Colloque International Francophone sur l'Écrit et le Document, CIFED 2010*, Sousse, Tunisie.

1.3 Organisation

Ce mémoire commence par une étude de différentes approches de représentation d'images sous forme de graphes, dans le **chapitre 2**. Dans ce chapitre nous passons en revue les principales méthodes de la littérature et nous étudions l'impact du choix de la stratégie d'extraction de graphes sur les performances de la reconnaissance. La suite de ce mémoire est organisée en deux parties :

- La première partie est dédiée au problème de l'appariement de graphes :
  - Dans le **chapitre 3** un état de l'art des méthodes d'appariement de graphes est présenté.
  - Le **chapitre 4** présente notre méthode de calcul de distance entre graphes.
- La deuxième partie est consacrée aux problèmes de classification et d'indexation des graphes :
  - Une méthode de plongement de graphes dans un espace euclidien est présentée dans le **chapitre 5**. Cette méthode permet l'application des outils de classification classiques sur les graphes.
  - Dans le **chapitre 6** un nouvel algorithme de clustering de graphes est proposé.
  - Le **chapitre 7** présente une nouvelle approche d'indexation de graphes et de navigation dans une base d'images représentées sous forme de graphes.
  - Nous appliquons l'ensemble de nos contributions sur des documents anciens dans le **chapitre 8**. Cette application entre dans le cadre du projet ANR NAVIDOMASS qui a pour objectif le développement d'un ensemble de techniques d'indexation et de navigation dans des documents anciens.

Enfin, le **chapitre 9** présente un bilan de nos contributions et définit nos perspectives de recherche.

Chapitre 2

Représentation d'images de documents sous forme de graphes

Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two facilities, which we may call intuition and ingenuity.

Alan Turing

Sommaire

2.1 Introduction

Les graphes sont des structures de données très flexibles et offrent une grande capacité d'abstraction. Dans le domaine de la reconnaissance de formes, la représentation d'images sous

forme de graphes a été utilisée avec un succès considérable pour plusieurs types d'images tels que les symboles graphiques [126], les formes [253], les documents anciens [136], etc. L'avantage majeur des graphes est la représentation explicite et compacte de la configuration relationnelle entre les différentes primitives d'un objet. Généralement, cette représentation est invariante à plusieurs types de changements (rotation, translation ...). En outre, par le biais de la représentation structurelle utilisant les graphes, nous pouvons transformer un problème de reconnaissance en un problème d'appariement entre graphes. Dans la littérature, nous distinguons plusieurs méthodes d'extraction de graphes à partir d'images. D'une manière générale, dans un graphe qui représente une image, les nœuds correspondent aux caractéristiques jugées saillantes dans l'image et les arêtes décrivent les relations qui peuvent liées à ces caractéristiques. Naturellement, les caractéristiques saillantes et leurs relations dépendent du type d'images considérées. En ce sens, plusieurs type de graphes ont été élaborés pour représenter au mieux les images.

Dans ce chapitre, nous passons en revue les types de graphes les plus utilisés dans la littérature. La première famille contient les graphes à base des points d'intérêt extraits à partir de l'image. La deuxième famille de graphes est basée sur l'agencement des régions constituant une image. Dans la troisième famille, nous décrivons l'extraction des graphes en utilisant les informations spatiales entre les primitives d'une image. Ensuite, l'usage du squelette est étudié en vue de l'analyse d'images. Finalement, nous étudions empiriquement l'impact de la technique de représentation sous forme de graphes sur les résultats de la reconnaissance. Pour ce faire, nous considérons deux bases d'images ; la première base contient des images binaires de silhouettes, la deuxième base consiste en des images de logos.

2.2 Définitions et concepts de base

Dans cette section, nous donnons les définitions des termes se rapportant aux graphes et les principaux concepts liés à la manipulation des graphes dont nous aurons besoin par la suite.

2.2.1 Graphes

Dans la littérature, plusieurs définitions de graphe coexistent avec différents vocabulaires. Néanmoins, elles partagent toutes un ensemble de concepts caractérisant l'aspect structurel et composite du graphe. Ici, nous adoptons celle donnée par Claude Berge dans [19, 20] qui nous paraît la plus générique : *"D'une façon intuitive, un graphe est un schéma constitué par un ensemble de points et par un ensemble de flèches reliant chacune deux de ceux-ci. Les points sont appelés les sommets du graphe, et les flèches les arcs du graphe."*

Formellement, un graphe $G = (X, U)$ est le couple constitué :

1. par un ensemble $X = \{x_1, x_2, ..., x_n\}$, dont les éléments sont appelés sommets ou nœuds,
2. par une famille $U = (u_1, u_2, ..., u_m)$ d'éléments, appelés arcs ou arêtes, du produit cartésien
$$X \times X = \{(x, y)/x \in X, y \in X\}$$

Pour un arc $u = (x_i, x_j)$, x_i est l'extrémité initiale, x_j l'extrémité finale (appelées également origine et destination). Toutefois, lorsqu'on s'intéresse uniquement à l'existence d'arc(s) entre

deux sommets (sans en préciser l'orientation), le graphe est dite non orienté. Un arc dans un graphe $G = (X, U)$ non orienté est appelé arête et U est constitué non pas de couples, mais de paires de sommets non ordonnés.

Dans la théorie des graphes, il existe un ensemble de notions qui servent à décrire et à typer les graphes. Dans ce qui suit, nous mettons l'accent en particulier sur les notions de base pour décrire les graphes ainsi que sur les différents types de graphes que nous utilisons dans ce mémoire.

Taille (Ordre) : La taille (ou l'ordre) d'un graphe G, $\mid G \mid$, est le nombre de nœuds présents dans le graphe G.

Adjacence : Deux nœuds sont adjacents (ou voisins) s'ils sont joints par un arc ou une arête. Pareillement, deux arcs sont adjacents s'ils ont au moins une extrémité commune.

Arête incidente : Pour une arête u=(x_i, x_j), on dit que u est incidente aux sommets x_i et x_j.

Degré : Le degré d'un nœud x_i, noté θ_{x_i}, est le nombre d'arêtes incidentes à x_i.

Graphe complet : Un graphe est complet si pour toute paire de sommets x et y, il existe au moins un arc u=(x, y), i.e. tous les nœuds sont adjacents les uns aux autres.

Graphe biparti : Un graphe est biparti si l'ensemble de ses nœuds peut être partitionné en deux classes de sorte que deux nœuds de la même classe ne soient jamais adjacents.

Graphe étiqueté : Un graphe étiqueté est un graphe dont les arêtes et/ou les nœuds sont affectées d'étiquettes. Ici, nous empruntons à Horst Bunke la définition formelle d'un graphe étiqueté [30]. Soient L_V et L_E deux ensembles d'étiquettes, respectivement, pour les nœuds et les arêtes. Formellement, un graphe étiqueté est définis par un 4-tuple $g = (V, E, \alpha, \beta)$ où

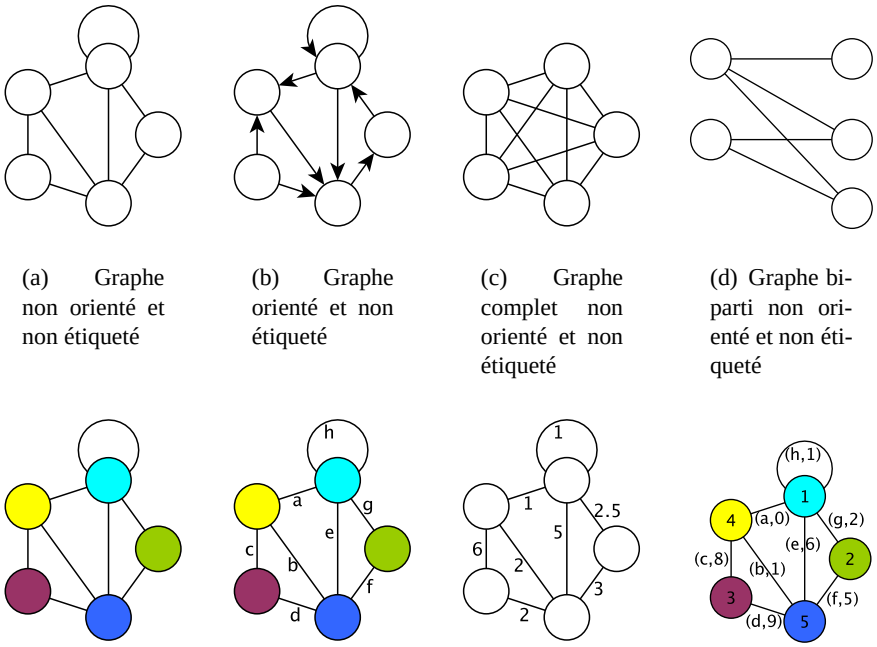
- V un ensemble fini de nœuds,
- E un ensemble d'arêtes ; $E \subseteq V \times V$,
- $\alpha : V \to L_V$ une fonction qui génère une étiquette à chaque nœud, et
- $\beta : E \to L_E$ une fonction qui génère une étiquette à chaque arête.

Notons ici qu'on peut associer à chaque nœud (resp. arête) plusieurs étiquettes en définissant L_V (resp. L_E) comme un ensemble de tuples de valeurs. De plus, les étiquettes peuvent être de différents types (e.g. lettre, mot, symbole, nombre, vecteur numérique ...). Par exemple, le graphe dans la figure 2.1(h) est caractérisé par un ensemble d'étiquettes pour les nœuds donnés par l'ensemble des entiers, i.e. L_V=$\mathbb{N}$ et un ensemble d'étiquettes pour les arêtes données par un ensemble L_E = $\{(x, y)/x \in \{a, b, c...\}, y \in \mathbb{N}\}$.

Graphe pondéré : un graphe pondéré est un graphe étiqueté dont toutes les arêtes sont étiquetées par des nombres positifs. Un graphe $G = (V, E, \alpha, \beta)$ est dit pondéré si et seulement si L_E = $\mathbb{R}^+$ et L_V = $\emptyset$. Dans un graphe pondéré, chaque étiquette est appelée poids de l'arête.

Dans la figure 2.1, quelques exemples de graphes (orienté / non orienté, étiqueté / non étiqueté) sont illustrés.

Sous-graphe partiel : Soient $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ et $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ deux graphes. Le graphe G_1 est un *sous-graphe partiel* de G_2, $G_1 \subseteq G_2$, si on obtient G_1 en enlevant un ou plusieurs nœuds avec leurs arêtes incidentes et/ou des arêtes du graphe G_2. Formellement, $G_1 \subseteq G_2$ si :



(a) Graphe non orienté et non étiqueté

(b) Graphe orienté et non étiqueté

(c) Graphe complet non orienté et non étiqueté

(d) Graphe biparti non orienté et non étiqueté

(e) Graphe non orienté et étiqueté sur les nœuds (les couleurs correspondent aux étiquettes)

(f) Graphe non orienté et étiqueté sur les nœuds et les arêtes

(g) Graphe pondéré non orienté

(h) Graphe non orienté et multi-étiqueté sur les nœuds et les arêtes

FIGURE 2.1 – Différents types de graphes

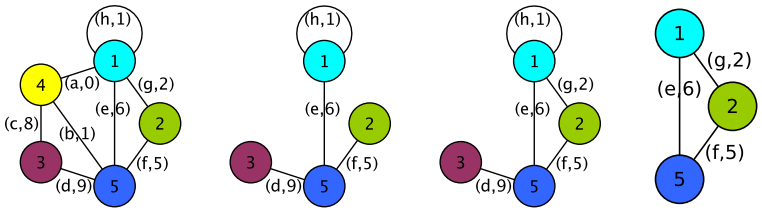


FIGURE 2.2 – (b) : sous-graphe partiel de (a), (c) : sous-graphe induit de (a) et (d) : clique de (a)

- $V_1 \subseteq V_2$,
- $E_1 \subseteq E_2$,
- $\alpha_1(u) = \alpha_2(u), \forall u \in V_1$, et
- $\beta_1(e) = \beta_2(e), \forall e \in E_1$

Sous-graphe induit : Soient $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ et $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ deux graphes. Le graphe G_1 est un *sous-graphe induit* de G_2, $G_1 \subseteq_i G_2$, si G_1 est un *sous-graphe partiel* de G_2. On obtient G_1 en enlevant un ou plusieurs nœuds avec leurs arêtes incidentes au graphe G_2. Le graphe G_1 contient toutes les arêtes de G_2 ayant leurs deux extrémités dans V_1. Formellement, $G_1 \subseteq_i G_2$ si :

- $V_1 \subseteq V_2$,
- $E_1 = E_2 \cap (V_1 \times V_1)$,
- $\alpha_1(u) = \alpha_2(u), \forall u \in V_1$, et
- $\beta_1(e) = \beta_2(e), \forall e \in E_1$

Clique : Une clique d'un graphe G est un sous-graphe complet de G.

Les figures 2.2(b), 2.2(c) et 2.2(d) illustrent, respectivement, un sous-graphe partiel, un sous-graphe induit et une clique de graphe dans la figure 2.2(a).

Matrice d'adjacence : La matrice d'adjacence $A = (A_{ij})_{n \times n}$ d'un graphe $G = (V, E)$ est définie par :

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

avec v_i et $v_j \in V$.

Matrice de degré : La matrice de degré $D = (D_{ij})_{n \times n}$ d'un graphe $G = (V, E)$ est définie par :

$$D_{ij} = \begin{cases} \theta_{v_i} & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

avec $v_i \in V$. Cette matrice de degré est une matrice diagonale carrée dont les coefficients en dehors de la diagonale principale sont nuls. Le *i-ème* coefficient de la diagonale indique le nombre des arêtes incidentes au nœud $v_i \in V$.

Matrice Laplacienne (matrice de Laplace) : La matrice laplacienne $L = (L_{ij})_{n \times n}$ d'un graphe $G = (V, E)$ est définie par :

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

avec v_i et $v_j \in V$. Cette matrice Laplacienne est obtenue par la soustraction de la matrice d'adjacence à la matrice de degré.

Dans cette section, nous avons présenté un ensemble de définitions et de concepts liés aux graphes. Ces notions sont utilisées pour définir les opérations de manipulation de graphes. Parmi ces manipulations, nous distinguons la comparaison de graphes qui est présentée dans la section suivante.

2.2.2 Comparaison de graphes

La structure de graphe est caractérisée principalement par la flexibilité et l'universalité qui permettent l'utilisation de graphes dans des domaines d'applications variés. Quand les graphes sont employés pour la représentation d'objets, trouver des objets similaires à d'autres revient à déterminer la similarité entre les graphes, i.e. la comparaison d'objets revient à comparer les graphes correspondants. D'un point de vue général, pour un graphe de taille n, il existe $n!$ différentes descriptions matricielles possibles (adjacence, degré et Laplacienne) parce qu'il n'existe aucun ordre de tri pour les nœuds et les arêtes. Ainsi, pour comparer deux graphes, nous ne pouvons pas nous contenter de comparer leurs descriptions matricielles. Formellement, le problème de comparaison des graphes [22] peut-être défini comme suit :

Comparaison de graphes [22] *Soient G_1 et G_2 deux graphes de l'espace de graphes ϱ, le problème de comparaison de graphes consiste à définir la fonction :*

$$d : \varrho \times \varrho \rightarrow \mathbb{R}$$

tel que $d(G_1, G_2)$ quantifie la similarité (ou la dissimilarité) entre G_1 et G_2.

Dans la littérature, le problème de comparaison de graphes est généralement abordé comme un problème d'appariement de graphes (*graph matching*). L'appariement de graphes consiste à mettre en correspondance les nœuds et les arêtes de deux graphes. Dans le cas général, un appariement entre deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ est une relation $m \subseteq V_1 \times V_2$ telle que $(u, v) \in m$ signifie que les nœuds u et v sont appariés.

Ainsi, la différence subtile entre la comparaison de graphes et l'appariement des graphes repose sur le fait que l'appariement de graphes consiste à identifier la correspondance entre les composants de deux graphes, tandis que la comparaison de graphes se borne à quantifier la similarité entre deux graphes. Les algorithmes d'appariement de graphes peuvent facilement associer un score aux résultats d'appariement pour définir une quantification de la similarité entre les graphes. En contrepartie, les algorithmes de comparaison de graphes ne peuvent pas être utilisés pour résoudre un problème d'appariement.

(a) Image originale

(b) Points d'intérêt avec l'agorithme de Harris

FIGURE 2.3 – Illustration des points d'intérêt de Harris

L'appariement de graphes est l'une des étapes cruciales dans la reconnaissance d'images à base de graphes. En effet, les distances entre les différentes images sont calculées par le biais des distances entre leurs graphes représentants. Cependant, le choix d'une représentation d'images sous forme de graphes est une étape importante dans le processus de la reconnaissance. Cette étape a un impact important sur les performances d'un système de reconnaissance d'images à base de graphes. Dans la section suivante, nous présentons les principales familles de techniques de représentation d'images de documents en graphes.

2.3 Graphe de points d'intérêt

La notion de point d'intérêt, introduite par Moravec [172], permet de localiser les points où le signal de l'image est riche en information. Dans une image, les points d'intérêt sont généralement les coins, les jonctions ou les points de fortes variations de texture. Ils sont largement utilisés dans la littérature pour la mise en correspondance d'images. Parmi les algorithmes de détection des points d'intérêt, nous distinguons l'algorithme de Harris [108]. Cet algorithme est largement utilisé dans les travaux de représentation d'images en graphes de points d'intérêt [161, 162, 192, 205, 271]. Dans la figure 2.3, nous présentons un exemple de points d'intérêt détectés avec l'algorithme de Harris.

La notion de points d'intérêt est introduite dans le contexte des approches statistiques afin de définir, pour une image, un ensemble de descripteurs locaux au lieu d'un seul descripteur global. Ces descripteurs locaux ont eu un grand succès dans plusieurs applications, telles que le suivi d'objets et la classification d'images [94, 154, 249].

Dans le cadre de la représentation d'images sous forme de graphes, les points d'intérêt fournissent des indications importantes sur la localisation des endroits riches en information dans l'image. Dans la littérature, plusieurs travaux ont utilisés les points d'intérêt pour construire les graphes. Dans [161, 162, 192, 205, 271], les auteurs appliquent la triangulation de Delaunay sur les points d'intérêt d'images pour construire les graphes représentants. Succinctement, si $\mathcal{PI}$ est l'ensemble des points d'intérêt d'une image i, la triangulation de $\mathcal{PI}$ consiste à construire un ensemble de triangles $\mathcal{T}$=$\{T_1, ..., T_n\}$ tels que :

- les sommets des triangles sont des points de $\mathcal{PI}$,
- $\forall p \in \mathcal{PI}, \forall i \in [1, n]$, alors p est un sommet de T_i ou $p \notin T_i$,
- l'ensemble $\mathcal{T}$=$\{T_1, ..., T_n\}$ est une partition de l'image i.

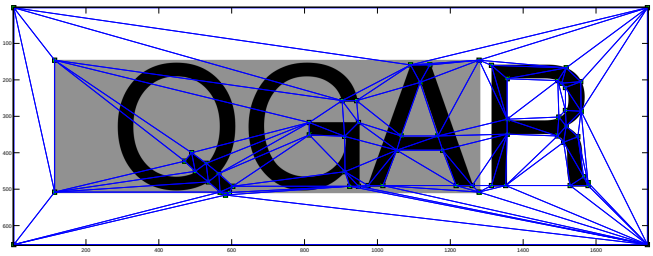


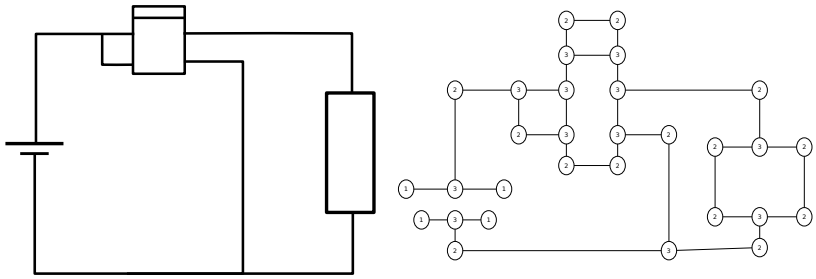
FIGURE 2.4 – Représentation d'un graphe à base des points d'inérêt de Harris

Dans la figure 2.4, nous illustrons une représentation sous forme de graphe basée sur la triangulation des points d'intérêt de l'image de la figure 2.3(b).

Dans un travail récent, Sanromà et al. [214] proposent une méthode de représentation structurelle en utilisant le descripteur SIFT [159]. L'idée est d'utiliser les point clés de SIFT comme les nœuds de graphes et les descripteurs calculés sur ces points comme les étiquettes des nœuds. Les arêtes sont définies à partir de la proximité des points de SIFT dans l'image. En effet, si deux points de SIFT sont proches (selon la distance euclidienne et un seuil donné), alors leurs nœuds correspondants dans le graphe sont adjacents.

Par ailleurs, nous distinguons des méthodes structurelles qui ont leurs propres stratégies de détection des points d'intérêt pour construire des graphes. Généralement, ces méthodes sont dédiées à la représentation sous forme de graphe d'un type spécifique d'images, e.g les documents et les symboles. Par exemple, Bunke propose dans [26] une méthode de représentation sous forme de graphes d'images de circuits électroniques. Cette méthode considère que l'image est donnée en terme de lignes. L'extraction du graphe représentatif de l'image est effectuée comme suit : chaque nœud du graphe représente un sommet dans le circuit électronique. Un sommet dans le circuit correspond soit à un point où plusieurs segments de ligne se croisent soit à un point final d'une ligne. Le nombre de lignes intersectant en un sommet est utilisé comme étiquette de nœud. Cela signifie que l'étiquette d'un nœud correspond à son degré. Les arêtes dans le graphe correspondent aux lignes dans le circuit électronique. Dans la figure 2.5, nous montrons une représentation sous forme de graphe (Fig.2.5(b)) d'un circuit électronique (Fig.2.5(a)).

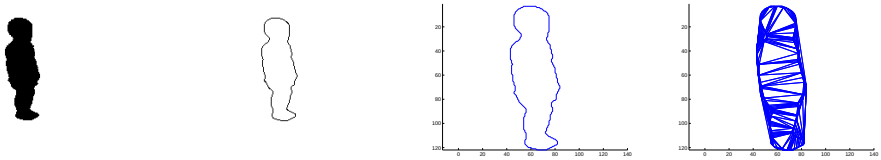
Dans un travail similaire Huet et al. [117] proposent une méthode de représentation d'images sous forme de graphe basée sur la polygonalisation. Dans un premier temps, les auteurs commencent par appliquer d'un détecteur de lignes [158]. Ensuite, un algorithme de polygonalisation est appliqué pour réduire les courbures des lignes extraites de l'image à un polygone. Enfin, les auteurs construisent un graphe dit graphe de N-plus proches voisins où les nœuds sont les centres des segments du polygone. Les arêtes du graphe sont calculées de façon à ce que



(a) Circuit électronique

(b) Représentation sous forme de graphe de (a)

FIGURE 2.5 – Représentation d'un circuit électronique



(a) Image originale

(b) Les contours de l'image détectés par la méthode de Canny

(c) La polygonalisation des contours

(d) La représentation sous forme de graphe

FIGURE 2.6 – Représentation en graphe par les contours et la polygonalisation

chaque nœud soit adjacent aux N plus proches nœuds. La proximité entre les nœuds correspond à la distance euclidienne entre les centres des segments du polygone représentant l'image.

Dans la même optique, nous pouvons envisager l'utilisation des contours d'objets présents dans une image pour représenter l'image par un graphe. Cette représentation sera basée sur l'approximation polygonale des contours de l'image. Ainsi, les nœuds du graphe correspondent aux points de jonctions des segments du polygone, tandis que les arêtes sont calculées par l'application d'une triangulation de Delaunay sur ces points. Un exemple d'extraction de graphe à partir d'une image en utilisant cette méthode est illustré dans la figure 2.6.

2.4 Graphe d'adjacence de régions

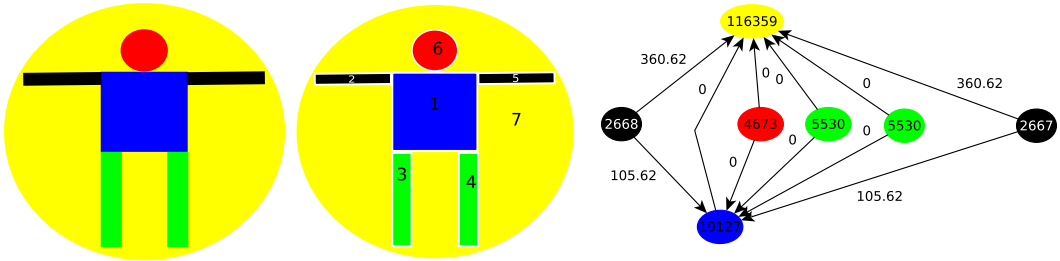
La notion de graphe d'adjacence des régions, noté RAG (*Region Adjacency Graph*), a été introduite par Rosenfeld [207]. Elle consiste à modéliser les relations d'adjacence entre les régions d'une image segmentée. Un RAG est un graphe planaire qui représente les régions de l'image par des nœuds et les relations d'adjacence existant entre ces régions par des arêtes. Ainsi, comme l'illustre la figure 1.2 (page 5), deux nœuds du graphe sont reliés par une arête si les régions qu'ils représentent sont adjacentes.

Graphe d'adjacence des régions Soit $\mathcal{I}$ une image et $\mathcal{R} = \{r_1, ..., r_n\}$ une partition de $\mathcal{I}$. Le

graphe d'adjacence des régions associé à $\mathcal{I}$ est le graphe $G = (V, E, \alpha, \beta)$ tels que
- Chaque région r_i est associée un nœud $v_i \in V$,
- il existe une arête $e = (v_i, v_j) \in E$, si et seulement si les régions r_i et r_j sont adjacentes.

Dans la littérature, plusieurs définitions formelles du RAG coexistent. La différence majeure entre les différentes définitions est liée à l'orientation du graphe. Par exemple, dans un contexte de mesure de similarité entre des images, Baeza et al. [12] définissent un graphe d'adjacence des régions comme suit :

Graphe d'adjacence des régions de Baeza et al. [12] Soit $\mathcal{I}$ une image et $\mathcal{R} = \{r_1, ..., r_n\}$ une partition de $\mathcal{I}$ tels que deux pixels $\mathcal{I}_{ij}$ et $\mathcal{I}_{kl}$ sont dans la même région r_k si et seulement si $\mathcal{I}_{ij}$ et $\mathcal{I}_{kl}$ sont similaires et voisins. Le graphe d'adjacence des régions associé à $\mathcal{I}$ est le graphe orienté $G = (V, E, \alpha, \beta)$ tels que
- Chaque région r_i est associée un nœud $v_i \in V$,
- il existe un arc $e = (v_i, v_j) \in E$ si $moyenne(r_i) \leq moyenne(r_j)$ et r_i et r_j sont adjacentes, avec $moyenne(r)$ est la moyenne des valeurs des pixels dans la région r,
- chaque nœud v_i est étiqueté par $\alpha(v_i)$ qui correspond à la taille de la région r_i,
- chaque arc $e = (v_i, v_j)$ est étiqueté par $\beta(e) = moyenne(r_j) - moyenne(r_i)$.



(a) Image originale

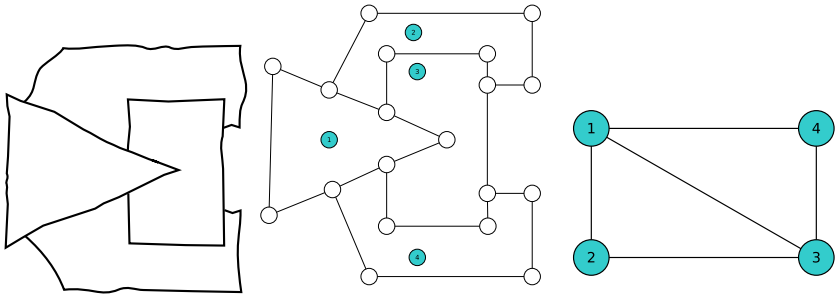
(b) Image segmentée en régions

(c) Représentation sous forme de graphe selon la définition de Baeza et al. [12]

FIGURE 2.7 – Exemple de représentation d'une image sous forme d'un graphe d'adjacence de régions.

Évidement, il faut définir en amont les fonctions de voisinage et de similarité entre les pixels de l'image. Cela revient à définir une stratégie de segmentation d'images. En effet, nous constatons que les méthodes de représentation d'images à base de RAG sont généralement précédées par une phase de segmentation [247]. Dans la figure 2.7, nous illustrons une représentation d'une image couleur en graphe d'adjacence de régions selon la définition de Baeza et al. [12]. Dans cette figure, nous considérons que la valeur d'un pixel correspond à la norme euclidienne de son vecteur RVB.

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. Par exemple, Lladós et al. [126] s'intéressent au problème de la reconnaissance de symboles graphiques. Ils modélisent pour cela les symboles par des



(a) Symbole original (b) Graphe correspondant (c) Graphe d'adjacence des régions

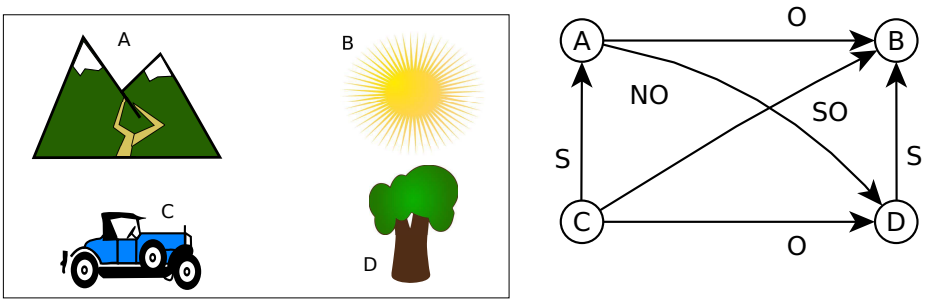
FIGURE 2.8 – Élaboration d'un graphe d'adjacence des régions d'un symbole [126]

graphes d'adjacence des régions en remplaçant la segmentation par une vectorisation d'image. Cette modélisation consiste à générer, en premier lieu, un graphe à partir de la représentation vectorisée des symboles. Ce premier graphe g est généré d'une manière identique à la méthode de Bunke [26], i.e. les nœuds correspondent aux points caractéristiques du symbole (jonctions, point finaux et coins). Ensuite, la construction du graphe d'adjacence des régions g_{rag} consiste à associer chaque région du premier graphe g à un nœud. Une région dans g_{rag} correspond à une boucle minimale fermée dans g. La figure 2.8 illustre un exemple de construction du RAG (Fig.2.8(c)) du symbole de la figure 2.8(a). Le premier graphe (Fig.2.8(b)) construit correspond aux lignes du symbole. Ce graphe contient quatre régions qui sont les nœuds du RAG dont les arcs correspondent aux relations d'adjacence entre ces régions (Fig.2.8(c)).

Dans un autre cas d'utilisation des graphes d'adjacence des régions, Karray et al.[136] manipulent des lettrines issues de documents anciens. Ils procèdent par une segmentation en différentes couches d'informations de ces lettrines afin d'obtenir *" des couches d'informations de zones homogènes"*. Chaque région homogène de la lettrine est associée à un nœud du graphe avec deux attributs, la taille et la forme, et chaque arête entre les nœuds est associée à deux attributs, l'angle et la distance.

2.5 Graphe de relations spatiales

Avec certains types d'images, les relations spatiales [74] entre les différentes parties d'une image s'avèrent importantes pour une meilleure représentation et ainsi une meilleure performance de reconnaissance. En ce sens, les graphes de relations spatiales (en anglais *spatial relation graph*) utilisent les relations spatiales pour la représentation d'images. Concrètement, un graphe de relations spatiales est un graphe, généralement orienté, où les nœuds représentent les composantes primitives de l'image et les arcs représentent les relations spatiales entre ces composantes. Dans [215, 277], les auteurs utilisent les graphes pour représenter des symboles graphiques. Ils commencent par décomposer chaque symbole en différentes familles de primitives (cercles, points d'extrémité...). Ensuite, ils déterminent les relations spatiales entre chaque famille. Les graphes sont alors construits en stockant dans les nœuds les caractéristiques de



(a) Image *symbolique*

(b) Représentation sous forme de graphe de relation spatiale

FIGURE 2.9 – Représentation d'une image

familles de primitives (en utilisant des descripteurs locaux) et dans les arcs les relations spatiales qui décrivent la topologie du symbole considéré.

Dans un contexte de recherche d'image *symbolique*, Hsieh et al. [114] proposent une méthode basée sur la représentation sous forme de graphe. Selon [114], une image symbolique est une image dont tous ses objets sont reconnus et affectés à des classes prédéfinies. Les auteurs utilisent les graphes de relations spatiales pour représenter les images symboliques. Concrètement, pour une image I donnée, une phase de pré-traitement est effectuée pour reconnaître les objets dans I. Cette reconnaissance consiste à déterminer la position et la classe de chaque objet. Ensuite, la construction du graphe de relations spatiales consiste, tout d'abord, à construire un nœud pour chaque objet avec une étiquette déterminant la classe de cet objet. Il faut noter ici que les objets d'une même classe partagent le même nœud dans le graphe. Deuxièmement, les arcs du graphe correspondent aux relations spatiales entre les différents objets dans l'image. En effet, un arc reliant le nœud étiqueté par c_i au nœud étiqueté par c_j correspond à une relation spatiale spécifique entre deux objets dans l'image dont les classes respectives sont c_i et c_j. Les relations spatiales entre les objets sont utilisées comme les étiquettes des arcs dans le graphe. Les auteurs utilisent 4 relations spatiales : S (sud), SO (sud-ouest), O (ouest), NO (nord-ouest), I (identique). Dans la figure 2.9(a), nous présentons une image symbolique avec quatre objets identifiés A, B, C et D. Le graphe de relations spatiales correspondant à cette image est donné dans la figure 2.9(b).

2.6 Graphe du squelette

En analyse de formes, un squelette est une version fine d'une forme. Il est composé par l'ensemble des lignes équidistantes aux contours de la forme et dont l'épaisseur est d'un pixel.

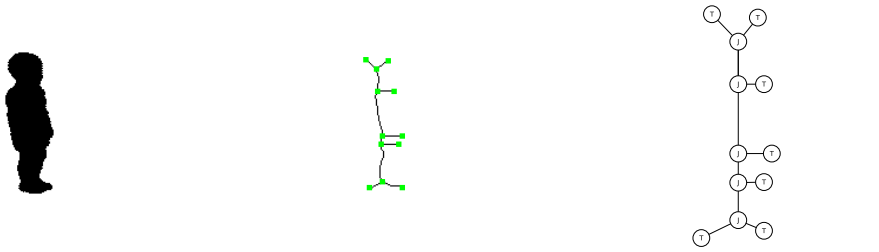
Cette représentation de formes est utilisée dans plusieurs travaux basés sur les graphes. Généralement, les nœuds du graphe du squelette sont calculés à partir de la classification des pixels du squelette en trois classes : points de jonctions, points de branches et point de terminaisons (finaux) [73, 72, 99, 156, 209, 160, 253]. Dans la majorité des travaux sur les graphes

du squelette, ces ensembles de points sont définis comme suit :

- Un point de jonction dans un squelette est un point d'intersection entre au moins trois branches.
- Un point de branche est un point appartenant à une branche.
- Un point de terminaison (ou point final) est un point final d'une branche, qui n'est connecté à aucune autre branche.

Ainsi dans un graphe du squelette $G(V, E)$, les points de jonctions et les points de terminaisons forment l'ensemble V des nœuds du graphe G, et les ensembles de points de branches connectant les points de V forment les arêtes (i.e. E) du graphe. Dans la figure 2.10, le résultat de la squelettisation de la forme de la figure 2.10(a) est illustré dans la figure 2.10(b) et les points caractéristiques sont marqués par des carrés vert. Le graphe du squelette est présenté dans la figure 2.10(c). Dans ce graphe nous avons considéré que chaque nœud est étiqueté par une lettre qui informe sur le type de point représenté par ce nœud (T pour les points de terminaisons et J pour points de jonctions). Le graphe est souvent enrichi par d'autres informations telles que la longueur des branches, la distance morphologique, la variance de la courbure des branches [209], etc.

Cette méthode d'extraction de graphe à partir d'un squelette d'une forme n'est pas unique. Dans la littérature, nous distinguons aussi les graphes de chocs [236, 222] qui considère le squelette d'une forme comme un ensemble de chocs. Succinctement, un graphe de chocs est une abstraction qui décompose une forme en un ensemble de primitives (segments) hiérarchiquement organisés. Ainsi, un graphe de chocs peut être un arbre ou un graphe orienté et acyclique.



(a) Image d'une forme (b) Le squelette de la forme (c) Représentation sous forme de graphe du squelette

FIGURE 2.10 – Représentation d'une image en graphe du squelette

2.7 Discussion

Dans cette section, nous avons présenté les quatre grandes familles de graphes utilisées dans la reconnaissance de formes. Nous constatons que dans chaque famille étudiée il existe plusieurs méthodes d'extraction. Ces méthodes dépendent de la nature des images et des besoins des systèmes de reconnaissance. Par exemple, les méthodes de construction de graphes d'adjacence des régions pour les images de document anciens ne sont pas appropriées pour les symboles graphiques. De même, les définitions d'une région dans une image dépendent des objectifs et des

techniques considérés. Une région est parfois définie par l'ensemble (ou les ensembles disjoints) des pixels (contiguës ou non-contiguës deux à deux) ayant des valeurs similaires. Une région peut être définie aussi par l'ensemble des primitives (composantes) appartenant à une même famille sémantique. Plusieurs autres définitions existent, mais, cette multitude de définitions ne limite pas l'utilisation des graphes. Ceci concerne aussi les autres familles de graphes (points d'intérêt, relations spatiales...). En effet, nous pouvons définir plusieurs stratégies d'extraction de graphes pour chaque type d'images, ce qui illustre la grande capacité de représentation des graphes. Cette capacité est enrichie aussi par la possibilité d'ajout des étiquettes aux nœuds et aux arêtes. Ces étiquettes fournissent un moyen supplémentaire pour mieux représenter les images. Nous avons vu que la spécification des étiquettes permet de définir des relations inter-primitives très appropriées aux type d'images considérés. En effet, dans les graphes de relations spatiales, les étiquettes des arêtes représentent très explicitement les agencements (la topologie) des primitives.

Ces quatre familles de représentation d'images en graphes ne représentent pas les uniques possibilités de passer d'une image à un graphe représentant. En effet, une représentation d'images sous forme de graphes peut être assurée manuellement par un utilisateur (totalement ou partiellement en utilisant des algorithmes assistant les utilisateurs). De plus, nous pouvons envisager la combinaison entre deux ou plusieurs familles de représentation. Par exemple, il est possible de combiner les graphes de points d'intérêt avec les graphes de relations spatiales, en ajoutant des informations spatiales sur les arêtes.

Ce large éventail de possibilités d'extraction de graphes nous amène aux deux questions suivantes :

- *Est-ce que le choix d'une technique de représentation sous forme de graphe dépend de la nature des images considérées ?*
- *Si c'est le cas, comment peut-on choisir la technique adéquate ?*

Intuitivement, la réponse à la première question est *oui*. Pour argumenter notre réponse, nous allons partir de l'hypothèse simple suivante : *Les graphes doivent représenter au mieux les images considérées*. Donc les graphes doivent contenir les caractéristiques saillantes des images. Vu que ces informations importantes dépendent du type d'images alors forcément les graphes doivent dépendre aussi du type d'images. En fait, le choix du type de graphes à utiliser, ainsi que la méthode d'extraction, doit être guidé par le type d'image considéré. Cette réflexion est une réponse partielle à la deuxième question. Cette réponse est partielle parce que plusieurs techniques d'extraction de graphes s'avèrent adéquates pour un cadre applicatif donné. Lorsque nous avons été confrontés à cette situation, nous avons envisagé une étude empirique de la performance des techniques. En ce sens, le choix de la meilleure technique d'extraction de graphes est déduit à partir des résultats expérimentaux.

2.8 Impacts de la représentation en graphes sur les performances

Dans la section précédente, nous avons considéré, d'une manière intuitive, que le choix d'une technique de représentation sous forme de graphe dépend de la nature des images considérées. En conséquence, nous avons affirmé que l'utilisation d'une bonne technique impliquera de bons résultats de reconnaissance. A *contrario*, une technique d'extraction de graphe non-appropriée

impliquera des résultats moins bons.

Dans cette section, nous étudions empiriquement l'impact du choix de la technique d'extraction de graphes sur les résultats de la reconnaissance. De plus, nous évaluons le rang des méthodes d'appariement de graphes en fonction de la représentation choisie. Pour ce faire, nous considérons deux bases d'images : la première base contient des images binaires de silhouettes, la deuxième base contient des images de logos. Dans la suite nous examinons l'impact du type de représentation sous forme de graphes pour chacune des bases.

2.8.1 Impacts de la représentation en graphes sur la base Shape

Dans cette section, trois types d'extraction de graphes sont testés sur la base de silhouettes Shape [229] du laboratoire LEMS de l'université Brown [2]. Cette base contient 216 formes binaires réparties en 18 classes de 12 formes (Fig. 2.11). Chaque classe de cette base contient 12 déformations d'une forme, e.g. occultation, rotation, changement d'échelle.

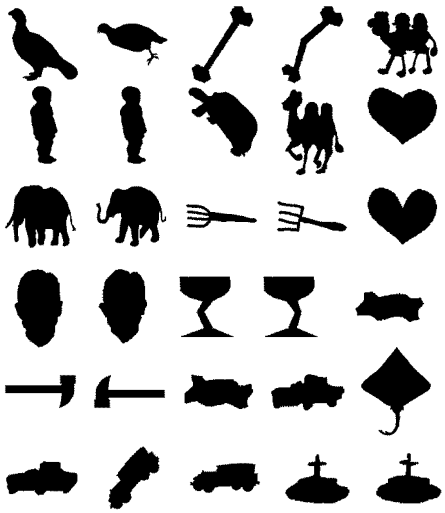
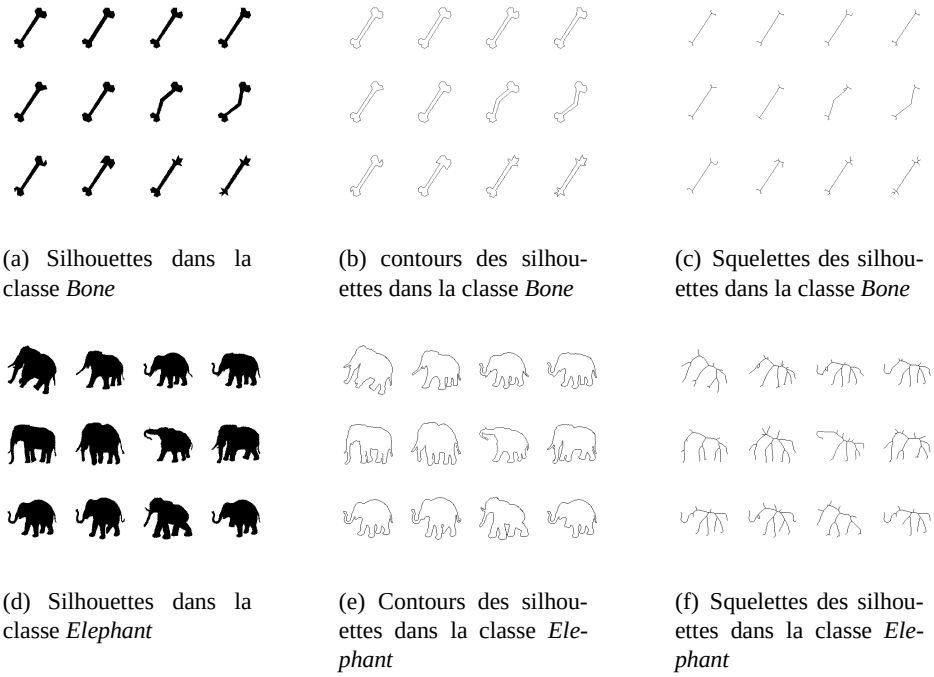


FIGURE 2.11 – Extrait de la base d'images Shape [229]

Pour étudier l'impact de la représentation d'images sous forme de graphes sur les performances de mesures de similarité entre graphes sur cette base, nous commençons par établir la liste des types de graphes adéquats pour ce type d'images (i.e. les silhouettes). Pour ce faire, nous discutons les éventuels avantages et inconvénients de l'utilisation de chaque type de graphes parmi les types évoqués précédemment dans ce chapitre.

Graphe de points d'intérêt : Dans cette base, un bon choix des points d'intérêt peut fournir une bonne description de chaque silhouette. Intuitivement, le choix d'un ensemble des points d'intérêt à partir des contours semblerait intéressant. Cela est justifié par le fait que les contours des silhouettes appartenant à la même classe sont perceptiblement plus similaires qu'aux contours des silhouettes de classes différentes. Cette constatation est

2. `http://www.lems.brown.edu/`



(a) Silhouettes dans la classe *Bone*

(b) contours des silhouettes dans la classe *Bone*

(c) Squelettes des silhouettes dans la classe *Bone*

(d) Silhouettes dans la classe *Elephant*

(e) Contours des silhouettes dans la classe *Elephant*

(f) Squelettes des silhouettes dans la classe *Elephant*

FIGURE 2.12 – Représentation de deux classes de la base Shape en squelette et en contours

validée dans les figures 2.12(b) et 2.12(e) où les contours des silhouettes appartenant à deux classes (*Bone* Fig 2.12(a) et *Elephant* Fig 2.12(d)) sont présentés. En plus des points extraits des contours, nous pouvons aussi envisager l'utilisation des points d'intérêt de Harris [108] qui sont invariants à la rotation.

Graphe d'adjacence de régions : Dans cette base l'adjacence entre les régions n'est pas une caractéristique discriminante entre les différentes classes de la base. En effet, toutes les images sont binaires et composées par deux régions ; la première est le fond blanc de l'image et la deuxième est la silhouette (en noir). Ainsi, une utilisation du graphe d'adjacence de régions pour représenter les images de cette base produira un ensemble de graphes où chaque graphe est composé par deux nœuds (deux régions) connectés par une arêtes. En conséquence, l'utilisation de ce type de graphes pour ce type d'image n'est pas approprié.

Graphe de relations spatiales : Dans cette base, chaque silhouette est considérée comme un tout (une seule composante connexe). Donc, nous ne pouvons pas fournir une décomposition adéquate des silhouettes en primitives. Par ailleurs, les graphes de relations spatiales sont définis principalement sur les relations qui lient les différentes parties (primitives) d'une image. Ainsi une représentation des silhouettes sous forme de graphes de relations spatiales se révèle inappropriée.

Graphe du squelette : Dans la littérature, il est connu que les squelettes sont des représentations très appropriées des formes à partir desquelles ils sont calculés. L'utilisation d'un

squelette pour représenter une forme se résume à abstraire la forme en une nouvelle représentation par des lignes fines (épaisseur d'un pixel) qui décrivent ses caractéristiques géométriques. Donc, l'utilisation des squelettes pour extraire les graphes représentants d'images de la base Shape semble être une piste très intéressante à explorer. Dans les figures 2.12(c) et 2.12(f), nous illustrons les squelettes d'images de deux classes différentes (*Bone* Fig 2.12(a) et *Elephant* Fig 2.12(d)). Nous constatons, perceptiblement, que les squelettes appartenant à la même classe sont plus similaires que les squelettes de classes différentes. Ceci encourage à l'utilisation des squelettes pour extraire les graphes représentants de l'ensemble des silhouettes de la base Shape.

À partir de ces observations, nous constatons que l'utilisation des points d'intérêt et du squelette sont les deux méthodes les plus adéquates. À ce stade d'étude de l'impact du type de représentation, il reste à vérifier si les graphes de points d'intérêt et les graphes du squelette fourniront des performance similaires, et à déterminer lequel de ces deux représentations est la meilleure pour cette base de silhouettes. Pour répondre à cette question, nous avons construit trois bases de graphes à partir de la base Shape. La première base (Contours + Delaunay) consiste à coder les contours de chaque silhouette dans un graphe des points d'intérêt des contours. Concrètement, pour chaque silhouette, nous appliquons l'algorithme de Canny pour détecter les contours, ensuite une polygonalisation est appliquée à ces contours. Ensuite, le graphe est construit à partir de la triangulation de Delaunay des points de jonctions du polygone. La deuxième base (Point d'intérêt de Harris + Delaunay) est similaire à la première base, sauf qu'au lieu d'utiliser les points d'intérêt des contours nous utilisons les points d'intérêt de Harris. Enfin, la troisième base (Squelette) consiste à coder le squelette de chaque silhouette dans un graphe où les nœuds représentent les points de terminaison et les points de jonctions, et les arêtes représentent les branches entre ces points dans le squelette.

Après la construction des bases de graphes, nous évaluons les performances les mesures de similarité entre graphes de chacune d'entre-elles. Nous considérons que les performances associées à chaque représentation sont les taux de classification effectués par l'algorithme de classification des k plus proches voisins. Le fonctionnement de cet algorithme sera détaillé dans le chapitre 5. Notons que la base est divisée en deux ensembles ; le premier constitue 30% de la totalité de la base et constitue l'ensemble d'apprentissage du k-ppv, tandis que le deuxième ensemble (70%) est utilisé comme un ensemble de test. Le choix de ces deux ensembles est totalement aléatoire. Dans la mesure où l'algorithme de k-ppv est sensible à cette configuration, nous avons répété 50 fois l'ensemble de l'expérimentation (sélection aléatoire de deux ensembles et l'application du k-ppv). Les résultats que nous affichons correspondent à la moyenne de ces 50 exécutions du k-ppv.

Pour approfondir notre étude nous utilisons sept mesures de distance de graphes. Ces méthodes de mesure de distance entre graphes sont détaillées dans le chapitre 4. Rappelons que l'objectif de cette expérimentation est d'étudier l'impact de la représentation et non pas les performances des mesures de distance de graphes.

Dans la table 2.1, nous présentons les taux de classification (k-ppv) obtenus sur les trois bases de graphes par chaque méthode de mesure de distance de graphes. Nous nous intéressons à la lecture des lignes de cette table. Chaque ligne présente les résultats de la classification en utilisant une mesure de distance et différentes représentations en graphes de la base Shape. Nous

	Contours + Delaunay	Point d'intérêt de Harris + Delaunay	Squelette
Jouili	51,49%	56,60%	**61.29**%
Robles-Kelly	53,04%	57,89%	**67,52**%
Lopresti	35,50%	**35,99**%	34,72%
Papadopoulos	42.33%	**46,22**%	40,16%
PATH	45.87%	59.35%	**63.40** %
Riesen	29,17%	30,06%	**47,73**%
Neuhaus	30,10%	30,53%	**46,44**%

TABLE 2.1 – Taux de classification de la base Shape [229] avec différentes méthodes d'extraction de graphes. En gras les meilleurs taux par méthode d'appariement.

constatons que, dans toutes les lignes de la table, les résultats varient d'une représentation à une autre. Ceci montre bien que le choix du type de représentation en graphes a un impact sur les performances de la classification. Parmi les trois types utilisés dans cette expérimentation, les graphes du squelette fournissent, en moyenne, les meilleurs résultats. Donc, parmi toutes les représentations en graphes testées, le graphe du squelette est globalement la représentation la plus appropriée pour la base Shape.

Bien que le type de représentation a un impact sur les performances de la classification, l'appariement joue également un rôle important. En effet, en examinant les résultats de chaque méthode d'appariement, nous observons que le rang de ces méthodes varient suivant le type de représentation. Par exemple, la méthode Lopresti [157] est au rang 6 pour la représentation en graphe du squelette et au rang 4 pour les représentations en graphes de contours+Delaunay et graphes de points d'intérêt de Harris+Delaunay (idem pour la méthode de Riesen qui a les rangs : 6,6,3).

2.8.2 Impacts de la représentation en graphes sur la base de logos

Dans cette section, nous étudions l'impact de la représentation sous forme de graphes sur la base de logos[67]. Cette base contient 80 images de logos réparties sur 10 classes de 8 images. La figure 2.13 présente un échantillon des images de cette base. Chaque classe de cette base contient 8 déformations d'un logo, e.g. translation, rotation, changement d'échelle.

Comme précédemment, pour choisir les types de représentation sous forme de graphes, nous commençons par examiner les avantages et les inconvénients de chaque type.

Graphe de points d'intérêt : L'utilisation de la représentation en graphes de points d'intérêt semblerait intéressante pour cette base. En effet, la plupart des logos sont composés par des lettres et un ensemble de formes polygonales. Ainsi, les sommets de polygones et les coins (*corners*) dans les lettres pourraient être un bon support pour construire les graphes.

Graphe d'adjacence de régions : Dans cette base, nous constatons que les images de logos sont composées par différentes régions disjointes. Ces régions forment les lettres et les

FIGURE 2.13 – Extrait de la base d'images de logos

formes constituant les logos. Ainsi, une représentation sous formes de graphes d'adjacence semblerait adéquate pour ce type d'images.

Graphe de relations spatiales : Dans cette base, chaque logo subi des déformations. Parmi ces déformations, nous distinguons la rotation. Lorsque le logo est pivoté les relations spatiales, entre les différentes primitives (régions), changent. Par exemple, si entre deux primitives d'une image il existe une relation spatiales de type "à-droite", la relation change en "à-gauche" en pivotant l'image de 180° . Ainsi une représentation sous formes de graphes de relations spatiales se révélerait inappropriée pour les logos en présence de transformations de type rotation.

Graphe du squelette : Vu que les images de logos contiennent des formes, il est motivant d'utiliser les squelettes pour les représenter. Ceci est dû au succès des squelettes dans la représentation des formes (notamment les silhouettes). Par conséquent, l'utilisation des squelettes pour construire les graphes représentants d'images de la base de logos semblerait être une piste intéressante à explorer. De plus, les résultats accomplis avec les graphes des squelettes dans l'expérimentation précédente encouragent leurs utilisation pour les logos.

À partir de ces observations, nous construisons trois bases de graphes à partir de la base de logos. La première base (Régions) consiste à modéliser chaque logo dans un graphe d'adjacence de régions. Concrètement, pour chaque logo, nous extrayons les régions disjointes, un graphe est ensuite construit tel que les nœuds correspondent aux régions et les arêtes correspondent aux

relations d'adjacence entre les régions. La deuxième base (Point d'intérêt de Harris + Delaunay) consiste à coder chaque logo dans un graphe des points d'intérêt. Pour chaque logo, nous appliquons l'algorithme de Harris pour détecter les points d'intérêt, ensuite le graphe est construit à partir de la triangulation de Delaunay de ces points d'intérêt de Harris. Enfin, la troisième base (Squelette) consiste à coder le squelette de chaque logo dans un graphe où les nœuds représentent les points de terminaison et les points de jonctions, et les arêtes représentent les branches entre ces points dans le squelette.

	Régions	Point d'intérêt de Harris + Delaunay	Squelette
Jouili	**93,40**%	76,13%	75,47%
Robles-Kelly	**82,03**%	69,33%	71,27%
Lopresti	**91,37**%	81,83%	80,93%
Papadopoulos	**94,67**%	84,40%	81,87%
PATH	**81.93**%	42.67%	43.53%
Riesen	**85,57**%	70,49%	67,93%
Neuhaus	**86,70**%	73,93%	69,20%

TABLE 2.2 – Taux de classification de la base de logos avec différentes méthodes d'extraction de graphes

Pour évaluer l'impact du de la représentation en graphes sur la base de logos, nous utilisons le même protocole d'évaluation que pour l'expérimentation précédente. Six méthodes de mesures de similarités entre les graphes sont appliquées dans un contexte de classification avec l'algorithme des k plus proches voisins. Nous rappelons que chaque base est divisée en un ensemble d'apprentissage (30%) et un ensemble de test constitué de 70% restant de la totalité de la base. La sélection de ces deux ensembles est totalement aléatoire. Les résultats considérés ultérieurement sont les moyennes de 50 exécution des k-ppv.

Dans la table 2.2, nous présentons les taux de classification (k-ppv) effectués sur les trois bases de graphes par chaque méthode de mesure de distance de graphes. Comme précédemment, nous nous intéressons à la lecture des lignes de cette table. Chaque ligne présente les résultats de la classification en utilisant une mesure de distance et différentes représentations en graphes de la base de logos.

Nous constatons que les performances de classification des logos varient d'une technique représentation en graphes à une autre. De plus, les méthodes d'appariement jouent également un rôle important dans les performances. Ceci converge vers les conclusions de l'expérimentation précédente (base Shape). Parmi les trois types utilisés dans cette expérimentation, les graphes d'adjacence de régions fournissent les meilleurs résultats. Donc, parmi toutes les représentations en graphes testées, les graphes d'adjacence de régions sont la représentation la plus appropriée pour la base de logos.

Nous pouvons faire la même remarque concernant le rang des méthodes en fonction de la représentation choisie.

2.8.3 Discussion

Les résultats expérimentaux précédents montrent l'impact important du choix du type de représentation en graphes sur le processus de la reconnaissance d'images dans un contexte structurel. De plus, le choix du type de représentation en graphes dépend du type d'images traitées. En effet, pour la base Shape, nous avons constaté que les graphes de squelettes fournissent les meilleurs résultats avec cinq méthodes de mesure de similarité. Par contre, pour la base de logos, les graphes d'adjacence de régions ont fourni les meilleurs résultats avec toutes les mesures de similarité testées. Ainsi, nous pouvons conclure que les graphes de squelettes sont mieux appropriés aux images des silhouettes qu'aux images des logos. Par ailleurs, les graphes d'adjacence de régions sont appropriés aux logos et non-appropriés aux silhouettes.

Outre le type de représentation en graphes, les performances de classification dépendent aussi de la méthode de mesure de similarité entre les graphes. Cet aspect n'a pas été discuté dans ce chapitre parce que notre objectif est d'étudier uniquement l'impact de type de représentation en graphes. L'impact du choix de la mesure de similarité entre les graphes est étudié dans les chapitres 3 et 4.

2.9 Conclusion

Dans ce chapitre, nous avons classé les types de graphes les plus utilisés dans littérature de la reconnaissance de formes en quatre catégories. La première catégorie correspond aux méthodes qui utilisent les points d'intérêt d'une image pour extraire un graphe. La deuxième catégorie contient les représentations où l'agencement des différentes régions d'une image est considéré pour construire un graphe. La troisième catégorie est constituée des techniques d'extraction des graphes où le principal avantage est donné au relations spatiales entre les différentes primitives d'une image. La dernière catégorie contient les méthodes d'extraction de graphes en se basant sur le squelette d'image.

Nous avons pu observer que le choix d'une catégorie pour une base d'images dépend de la nature des images et des besoins des systèmes d'analyse. Nous avons aussi montré empiriquement que le choix de représentation en graphes a un impact important sur les performances de la classification d'images. Finalement, nous avons constaté que le choix de la représentation influe aussi sur le rang de performances de méthodes d'appariement. En effet, en considérant une base d'images, une méthode d'appariement peut être classée parmi les meilleures pour une représentation donnée et parmi les moins bonnes pour une autre représentation. Ainsi, le changement de représentation peut donner un qualificatif de la robustesse des méthodes d'appariement. Ceci nous amène à la conclusion suivante : dans les travaux d'appariement, il faudrait au préalable se poser la question de la représentation lorsqu'on présente des résultats comparatifs et être conscient qu'en fonction de la représentation les performances peuvent être différentes.

Première partie

Appariements de graphes

Chapitre 3

État de l'art

On peut dire, sous forme de boutade, qu'il y a presque autant d'approches différentes des méthodes [...] qu'il y a d'applications différentes et de chercheurs participant à leur mise en œuvre, pour exprimer l'intérêt accru porté à ce domaine scientifique.

Alain Faure

Sommaire

3.1 Introduction

Dans le chapitre précédent, nous avons illustré la grande flexibilité des structures de graphes ainsi que leurs aptitudes à représenter les images pour les problèmes de la représentation de formes. Ces deux qualités sont spécifiques aux graphes. En effet, en utilisant un vecteur caractéristique pour représenter une image, à notre connaissance, il est obligatoire de fixer une taille unique pour tous les vecteurs représentant les images. Par contre, en utilisant les graphes pour représenter des images, la taille de chaque graphe n'est pas fixée *a priori* et dépend uniquement de la complexité de l'image considérée.

Cependant, dans la littérature, la grande majorité [119] des approches utilisent les vecteurs comme modèles de représentation. Ce délaissement des graphes en faveur des vecteurs est principalement dû à la complexité élevée des opérations liées aux graphes. De ce point de vue, la reconnaissance de formes à base de graphes est un vrai challenge. Les défis sont à relever dès les opérations basiques, tel que le calcul de la distance entre deux graphes. En utilisant les vecteurs,

plusieurs standards existent pour définir une distance avec une complexité très faible [3]. A *contrario*, aucun standard n'est défini pour le calcul des distances entre les graphes. Toutefois, dans la littérature, plusieurs approches ont été proposées pour la recherche d'une solution optimale de distance entre deux graphes. Ces méthodes d'appariement de graphes sont réparties en deux classes d'approches : les approches exactes et les approches approximatives. Dans la première classe, l'objectif est de déterminer un isomorphisme exact entre deux graphes. Ces méthodes sont très rigides et faiblement utilisées dans les applications du monde réel de la reconnaissance de formes. Ceci est principalement dû à la sensibilité des techniques d'extraction de graphes aux bruits présents dans les images. En effet, la présence de bruits dans les images entraîne des changements dans les topologies et les étiquettes des graphes représentatifs. La deuxième classe de méthodes d'appariement de graphes contient les approches dites approximatives où l'objectif est de déterminer une distance entre deux graphes. L'appariement approximatif de graphes est introduit pour prendre en considération les changements de structures et d'étiquettes, et rendre ainsi l'appariement de graphes utilisable en pratique. L'objectif est de chercher une distance entre deux graphes même s'ils ne sont pas rigoureusement identiques. Dans ce chapitre, nous passerons en revue quelques méthodes représentatives de la littérature de calcul de distance et de similarité entre les graphes. Pour une revue complète de ces méthodes et leurs applications, nous conseillons la lecture de Conte et al. [52].

3.2 Synthèse des approches d'appariement de graphes

Entre les premiers travaux des années 70 [16, 56, 86] et les recherches plus contemporaines [127, 198], le problème d'appariement de graphes a connu un fort essor dans plusieurs domaines scientifiques. Ces travaux ont concerné non seulement les applications d'appariement de graphes, mais aussi le développement de nouvelles techniques de calcul d'appariement. Particulièrement, ce problème a beaucoup attiré l'attention de la communauté de reconnaissance de formes où plusieurs travaux ont été proposés pour résoudre l'appariement de graphes [28, 29, 52]. Ces travaux se répartissent en deux catégories : *appariement exact* et *appariement approximatif*. Dans cette section, nous passerons en revue les différentes techniques d'appariement de graphes de chaque catégorie.

3.2.1 Appariement exact de graphes

L'appariement exact de graphes nécessite plusieurs conditions. D'abord, il s'agit de trouver une fonction d'association entre les nœuds de deux graphes tout en préservant la structure, i.e, chaque paire de nœuds adjacents du premier graphe correspond à une paire de nœuds adjacents du deuxième graphe. Outre la préservation de la structure, un appariement exacte entre deux graphes préserve aussi les étiquettes, i.e, chaque nœud d'un graphe correspond à un nœud ayant la même étiquette dans le deuxième graphe.

Dans la littérature, plusieurs approches d'appariement exact ont été proposées. Dans la suite, nous donnons une taxonomie de ces approches selon la satisfaction des conditions évoquées ci-

3. Par exemple, il faut que le i-ème élément du premier vecteur soit comparé uniquement avec le i-ème élément du deuxième vecteur.

dessus. L'approche la plus stricte est l'*isomorphisme de graphes* où deux graphes isomorphes correspondent à deux graphes ayant exactement la même structure et les mêmes étiquettes.

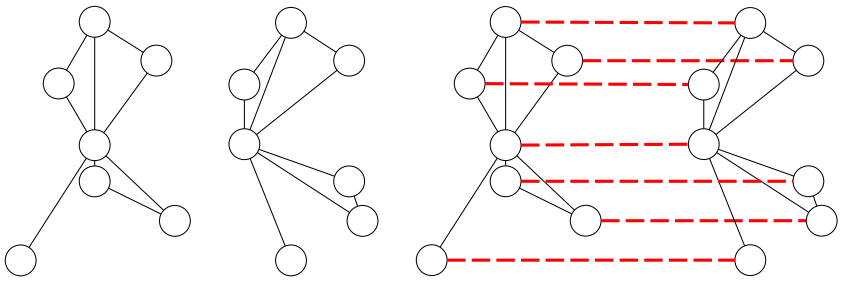
Isomorphisme de graphes *Soient* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *et* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *deux graphes. L'isomorphisme de graphes est une fonction bijective :*

$$f : V_1 \rightarrow V_2$$

Cette fonction doit satisfaire les conditions suivantes :

1. $\forall\, u \in V_1$, $\alpha_1(u) = \alpha_2(f(u))$,
2. $\forall\, e_1 = (u, v) \in E_1$, $\exists\, e_2 = (f(u), f(v)) \in E_2$, *tel que* $\beta_1(e_1) = \beta_2(e_2)$, *et*
3. $\forall\, e_2 = (u, v) \in E_2$, $\exists\, e_1 = (f^{-1}(u), f^{-1}(v)) \in E_1$, *tel que* $\beta_1(e_1) = \beta_2(e_2)$

Pour vérifier si deux graphes sont isomorphes, il faut trouver une fonction bijective f qui met en correspondance les nœuds, un-à-un, de deux graphes tout en préservant la connectivité et les étiquettes des nœuds. Formellement, un nœud u de graphe G_1 est apparié à un nœud $f(u)$ du graphe G_2 si et seulement si leurs étiquettes sont identiques, soit $\alpha_1(u) = \alpha_2(f(u))$. Également, une arête $e_1 = (u, v)$ du graphe G_1 est apparié à une arête $e_2 = f(e_1)$ du graphe G_2 si et seulement si leurs étiquettes sont identiques, soit $\beta_1(e_1) = \beta_2(e_2)$. De plus, deux nœuds u et v adjacents dans G_1 sont appariés respectivement à $f(u)$ et à $f(v)$ du G_2 si et seulement si $f(u)$ et $f(v)$ sont adjacents. La figure 3.1(c) illustre l'isomorphisme entre le graphe de la figure 3.1(a) et le graphe de la figure 3.1(b), les traits interrompus rouges indiquent la correspondance entre les nœuds.



(a) (b) (c)

FIGURE 3.1 – Deux graphes isomorphes

Pour calculer la fonction d'isomorphisme f d'une manière naïve, il faut générer toutes les fonctions possibles, soit $n!$ fonctions (avec $n = \mid V_1 \mid = \mid V_2 \mid$), et tester chaque fonction pour trouver une fonction d'isomorphisme f. La complexité d'un tel algorithme est au plus $n!$, qui n'est certainement pas efficace pour des applications de reconnaissance de formes. Actuellement, il n'existe aucun algorithme avec une complexité polynomiale pour la résolution du problème d'isomorphisme de graphes. Bien que la complexité de ce problème soit considérée dans la classe NP sans connaître si elle est dans P ou si elle est dans NP-complet [142, 237], elle n'est pas encore démontrée [91].

Toutefois, quelques méthodes ont été proposées dans la littérature pour résoudre efficacement l'isomorphisme de graphes. Nous distinguons deux catégories d'approches qui visent à optimiser la recherche d'isomorphisme de graphes.

Une première idée consiste à imposer des restrictions sur les types de graphes à traiter. En fait, pour quelques types spécifiques de graphes, il existe des algorithmes polynomiaux pour le calcul d'isomorphisme tel que les arbres [3] et les graphes planaires [4] [113]. Une deuxième famille d'approches, qui est la plus utilisée dans la littérature, consiste à concevoir une nouvelle représentation de l'espace de recherche d'isomorphisme de graphes et à développer des algorithmes qui réduisent la taille de l'espace de recherche en éliminant les chemins de recherche jugés non adéquats. Ces méthodes sont plus générales dans le sens où elles n'imposent aucune restriction sur les types de graphes. Dans cette famille d'algorithmes, l'un des premiers travaux est l'algorithme de Corneil et Gotlieb [56] dont l'idée est de transformer chaque graphe en une nouvelle représentation canonique. Cette représentation est basée sur une conjecture pour laquelle le calcul d'isomorphisme est plus facile. Mais, il a été démontré dans [165] que la conjecture qui est à la base de l'algorithme de Corneil et Gotlieb [56], n'est pas toujours vraie [87]. Dans la même perspective, Berztiss a proposé dans [21] une méthode basée sur la notion de *K-formule* et le retour arrière (*backtracking*). Cette méthode consiste à représenter chaque graphe, particulièrement les graphes orientés, par une chaîne qui contient toutes les informations pertinentes associées au graphe. Parmi les algorithmes de cette famille de techniques d'appariement, il faut citer l'algorithme *nauty* [169] qui est considéré par plusieurs auteurs [52, 87] comme l'algorithme d'isomorphisme de graphes le plus rapide. Dans cet algorithme, les nœuds de chaque graphe sont ordonnés en se basant sur un étiquetage canonique. En fait, pour chaque nœud n_i d'un graphe l'algorithme calcule une étiquette unique en se basant sur un ensemble de caractéristiques décrivant les relations entre n_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).

Une grande majorité des algorithmes d'isomorphisme de graphes sont basés sur la notion d'arbre de recherche avec retour arrière (*backtracking*) [52]. Ces algorithmes consistent à structurer l'espace de recherche sous forme d'un arbre de recherche. La racine de l'arbre correspond à un appariement vide et les feuilles correspondent à un appariement complet entre les deux graphes à comparer, s'il existe. La construction de chaque niveau de l'arbre correspond à l'ajout d'une paire de nœuds de deux graphes à apparier. Prenons, à titre d'exemple, deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ représentés respectivement dans les figures 3.2(a) et 3.2(b). La figure 3.2(c) illustre un exemple d'un arbre de recherche d'isomorphisme entre les deux graphes G_1 et G_2. À chaque niveau de l'arbre un nœud de $V_2 = \{a, b, c\}$ est apparié à un nœud de $V_1 = \{1, 2, 3\}$. À chaque feuille de l'arbre, la contrainte d'adjacence est vérifiée par l'examen des arêtes dans E_1 et E_2. De la même manière, les étiquettes sont vérifiées si le graphe est étiqueté. Finalement l'isomorphisme est trouvé si ces conditions (adjacence et étiquettes) sont valides dans une feuille de l'arbre de recherche. Pour explorer efficacement l'arbre de recherche, plusieurs méthodes ont recours à la notion du retour arrière (*backtracking*). Parmi ces méthodes, l'algorithme de Ullmann [257], apparu en 1976, reste le plus populaire et l'un des plus efficaces

4. Un graphe est planaire si on peut le dessiner dans le plan sans que les arêtes ne se croisent.

en terme de réduction de la taille de l'arbre de recherche. Ullmann [257] utilise une procédure de raffinement qui consiste à élaguer le maximum de branches de l'arbre de recherche avec un parcours d'abord en profondeur. En fait, à chaque appariement (nœud de l'arbre) l'heuristique de Ullmann filtre les appariements postérieurs afin d'exclure les appariements qui ne sont pas cohérents avec l'appariement en considération. Outre l'algorithme de Ullmann, plusieurs algorithmes ont été proposés. Ils se basent aussi sur des heuristiques pour optimiser le temps de calcul dans l'arbre de recherche. Citons à titre d'exemple, les deux algorithmes VF et VF2 [53, 54, 55], La méthode de Bulò et al. [25] basée sur la théorie de jeux, le soft-assign [100], et la méthode de Messmer et al. [171] basée sur l'arbre de décision.

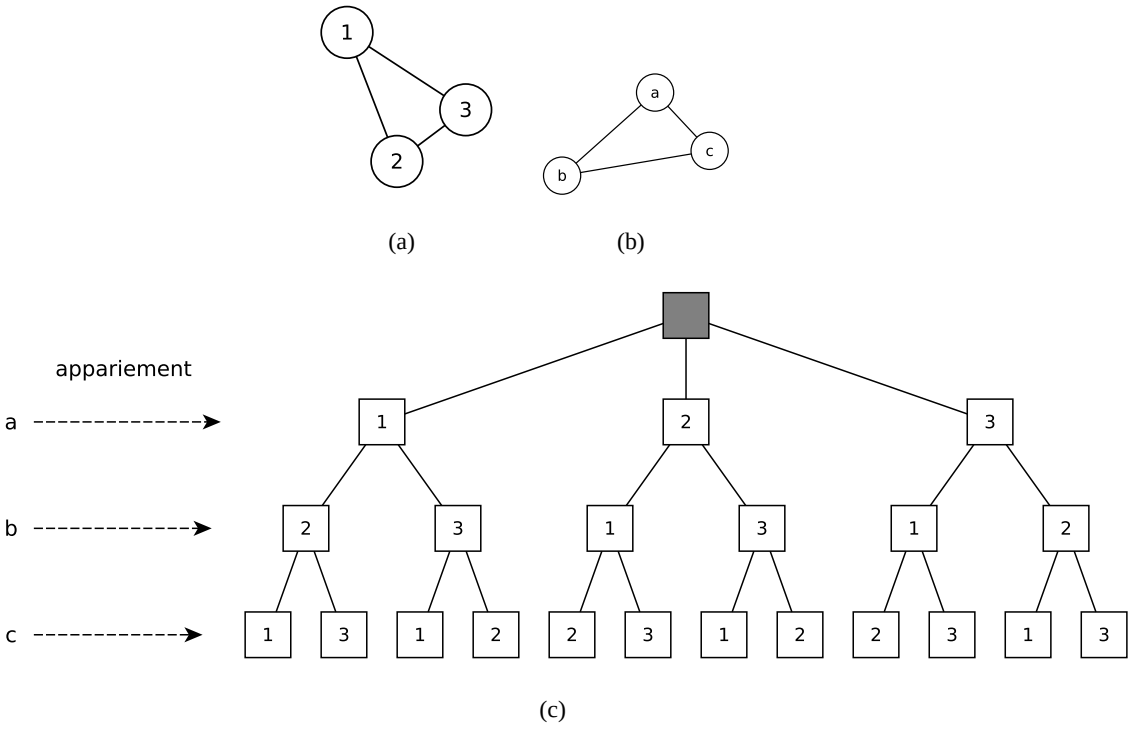
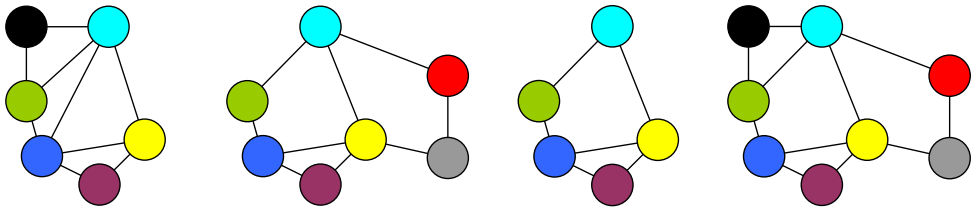


FIGURE 3.2 – Arbre de recherche d'isomorphisme de graphes.

Dans l'appariement exact de graphes, l'isomorphisme de graphes est la manière la plus rigide pour comparer deux graphes. En effet, dès que les deux graphes considérés présentent une infime différence dans la structure ou les étiquettes (e.g. ajout ou suppression d'une arête) ils sont non isomorphes et donc ne pourront pas être appariés. L'isomorphisme de sous-graphes réduit cette rigidité. En effet, pour vérifier l'isomorphisme de sous-graphes entre deux graphes G_1 et G_2, il suffit qu'un isomorphisme existe entre le graphe G_1 et un sous-graphe de G_2.

Isomorphisme de sous-graphes *Soient* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *et* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *deux graphes. L'isomorphisme de sous-graphes est une fonction injective :*

$$f : V_1 \rightarrow V_2$$



(a) (b) (c)

FIGURE 3.3 – (a) deux graphes g_1 et g_2, (b) un sous-graphe maximal commun entre g_1 et g_2, (c) un super-graphe minimal commun entre g_1 et g_2

tel qu'il existe un sous-graphe g de G_2, $g \subseteq G_2$, et f est un isomorphisme entre G_1 et g.

À la différence de l'isomorphisme de graphes, la complexité d'isomorphisme de sous-graphes est démontrée comme NP-Complet [91]. Pour résoudre l'isomorphisme de sous-graphes, plusieurs méthodes d'isomorphisme de graphes peuvent être utilisées. Parmi ces méthodes, nous distinguons particulièrement les méthodes basées sur l'arbre de recherche [53, 54, 55, 257], l'arbre de décision [171] et la satisfaction de contraintes [239, 282].

En se basant sur l'isomorphisme de (sous-)graphes, le problème de la comparaison des graphes se limite ainsi à la recherche d'un isomorphisme. En effet, un algorithme d'isomorphisme de (sous-)graphes retourne, généralement, une valeur booléenne qui indique l'existence ou l'absence d'un isomorphisme entre deux graphes. L'existence d'une fonction d'isomorphisme entre deux graphes nécessite que les deux graphes soient identiques par rapport à la structure et aux étiquettes, ou qu'un graphe soit identique à une partie de l'autre graphe. En revanche, dans les cas pratiques en reconnaissance de formes, les graphes peuvent être très similaires sans être (sous-graphes) isomorphes. Considérons le cas de la figure 3.3(a), il est clair que ces deux graphes sont similaires. Pourtant entre ces deux graphes il n'existe aucune fonction d'isomorphisme de (sous-)graphes.

Les notions de sous-graphe maximal commun (*smc*) et de super-graphe minimal commun (*SMC*) ont été proposées pour contourner les inconvénients d'isomorphisme de (sous-)graphes et établir une mesure de similarité entre les graphes,

Sous-graphe maximal commun (*smc*) *Soient $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ et $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ deux graphes. Un graphe g est dit sous-graphe commun à G_1 et G_2 si et seulement si g est isomorphe à un sous-graphe de G_1 et à un sous-graphe de G_2, i.e. il existe deux isomorphismes de sous-graphes, le premier entre g et G_1 et le deuxième entre g et G_2. Un sous-graphe maximal commun entre G_1 et G_2 correspond au plus grand sous-graphe g commun entre G_1 et G_2.*

Traditionnellement, le calcul du sous-graphe maximal commun entre deux graphes est lié au calcul de la clique maximale dans le graphe d'association de deux graphes considérés [15, 153]. Le graphe d'association $g = (V, E)$ de deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ est un graphe tel que : $V = V_1 \times V_2$ et deux nœuds (u, v) et (x, y) sont adjacents si et seulement si $(u, x) \in E_1$ et $(v, y) \in E_2$ ou $(u, x) \notin E_1$ et $(v, y) \notin E_2$. La figure 3.4, illustre le graphe

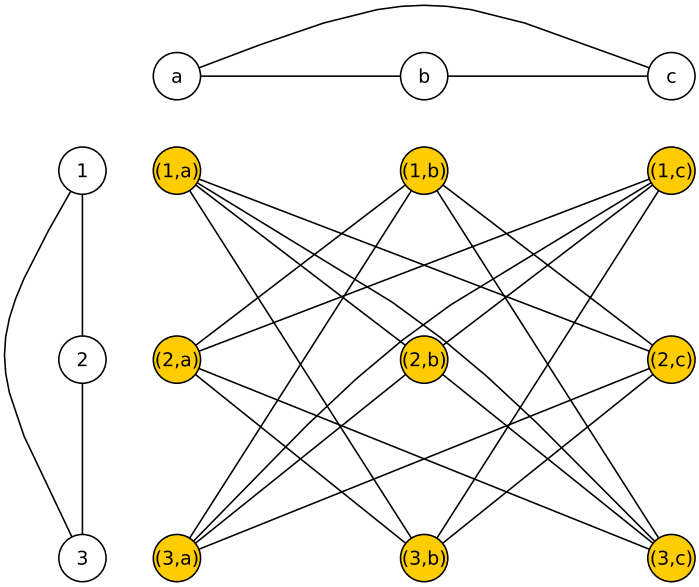


FIGURE 3.4 – Graphe d'association

d'association entre les deux graphes dans les figures 3.2(a) et 3.2(b). Dans la littérature, plusieurs approches ont été proposées pour le calcul de la clique maximale en utilisant aussi l'arbre de recherche [14] avec des heuristiques pour optimiser le temps de calcul [35, 135, 145, 168, 167].

Super-graphe minimal commun (*SMC*) *Soient* $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ *et* $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ *deux graphes. Un graphe* g *est dit super-graphe commun à* G_1 *et* G_2 *si et seulement si* G_1 *est isomorphe à un sous-graphe de* g *et* G_2 *est isomorphe à un sous-graphe de* g*. Un super-graphe minimal commun entre* G_1 *et* G_2 *consiste au plus petit super-graphe* g *commun entre* G_1 *et* G_2

La notion de super-graphe a été introduite par Bunke et al. dans [34]. Cette nouvelle notion consiste à créer le graphe le plus petit possible contenant les deux graphes considérés comme deux sous-graphes. Bunke et al. [34] ont montré que le calcul du supergraphe commun peut se réduire au calcul du sous-graphe maximal commun.

Les notions du sous-graphe maximal commun (*smc*) et du super-graphe minimal commun (*SMC*), présentées ci-dessus, ont été largement utilisées pour définir une mesure de similarité entre les graphes. Intuitivement, la similarité entre deux graphes est calculée à partir de leurs parties communes. En effet, si deux graphes ont une large partie commune, évidemment, leur similarité doit être importante. Inversement, si deux graphes n'ont qu'une petite partie commune, leur similarité doit être faible. Dans ce qui suit, nous présentons quelques mesures de distance qui ont été établies dans littérature pour quantifier la similarité entre deux graphes en se basant sur les notions du sous-graphe maximal commun (*smc*) et du super-graphe minimal commun (*SMC*).

Bunke et al. ont proposé dans [27] une distance entre deux graphes définie comme suit :

$$d_{Bunke97}(g_1, g_2) = \mid g_1 \mid + \mid g_2 \mid -2\times \mid smc(g_1, g_2) \mid$$

À partir de cette mesure, deux graphes g_1 et g_2 sont similaires si la taille de leur sous-graphe maximal commun ($\mid smc(g_1, g_2) \mid$) est proche de la taille de $\mid g_1 \mid$ et $\mid g_2 \mid$, i.e $\mid smc(g_1, g_2) \mid$ est similaire, à la fois, à $\mid g_1 \mid$ et à $\mid g_2 \mid$. Ainsi, la distance sera proche de 0. Par contre, si la taille du sous-graphe maximal commun ($\mid smc(g_1, g_2) \mid$) est proche de 0, la distance sera la plus importante.

Une autre distance entre les graphes est proposée aussi par Bunke et al. dans [37], et est définie par :

$$d_{Bunke98}(g_1, g_2) = 1 - \frac{\mid smc(g_1, g_2) \mid}{max\{\mid g_1 \mid, \mid g_2 \mid\}}$$

Cette distance a la particularité d'être bornée entre 0 et 1, où une valeur proche de 0 désigne une forte similarité entre les graphes comparés, et inversement une valeur de $d_{Bunke98}$ proche de 1 correspond à une faible similarité. En fait, pour que deux graphes soient similaires il faut que la taille de leur sous-graphe maximal commun soit proche de la taille de l'un deux. Ainsi, la fraction sera proche de 1 et la distance entre les deux graphes sera proche de 0.

Une autre distance similaire a été proposée par Wallis et al. dans [265]. Cette distance est basée sur la notion du graphe d'union qui est inspirée de la théorie des groupes. La distance de Wallis entre les graphes est définie comme suit :

$$d_{Wallis}(g_1, g_2) = 1 - \frac{\mid smc(g_1, g_2) \mid}{\mid g_1 \mid + \mid g_2 \mid - \mid smc(g_1, g_2) \mid}$$

Du point de vue de la théorie des groupes, le dénominateur de la fraction correspond à la taille de l'union de deux graphes g_1 et g_2. Il est clair que cette distance est très similaire à la distance $d_{Bunke98}$. Néanmoins, à l'inverse de la distance $d_{Bunke98}$, avec la distance d_{Wallis} le graphe le plus petit des deux graphes g_1 et g_2 a une influence sur la distance.

Outre ces distances uniquement formulées à partir du sous-graphe commun maximal, Fernandez et al. ont proposé dans [81] de combiner le sous-graphe commun maximal et le super-graphe minimal commun pour calculer la distance entre deux graphes. La distance est définie comme suit :

$$d_{Fernandez01}(g_1, g_2) = \mid SMC(g_1, g_2) \mid - \mid smc(g_1, g_2) \mid$$

Selon cette distance, pour que deux graphes g_1 et g_2 soient similaires, il faut que la taille de $SMC(g_1, g_2)$ soit proche de la taille de $smc(g_1, g_2)$, i.e. le sous-graphe commun maximal et le super-graphe minimal commun de g_1 et g_2 sont similaires.

Discussion

Selon la définition de l'appariement exact de graphes, notamment l'isomorphisme de graphes, pour que deux graphes soient similaires il faut qu'ils partagent exactement la même structure et les mêmes étiquettes (s'ils sont étiquetés). Ces conditions très rigides impliquent qu'une détection de graphes isomorphes ne peut avoir lieu qu'à la condition que n'apparaisse aucune

différence entre les graphes, même infime. Pour réduire ce niveau de rigidité de l'appariement, l'isomorphisme de sous-graphes a été introduit. Dans le contexte de l'isomorphisme de sous-graphes, un graphe peut être isomorphe à, seulement, une partie d'un autre graphe.

La similarité, basée sur la notion d'isomorphisme entre deux graphes, est une valeur binaire qui indique uniquement si les graphes sont isomorphes ou non. En revanche, la plupart des applications utilisant les graphes (e.g. en reconnaissance de formes) ont besoin des mesures qui quantifient la similarité entre les graphes. Cette quantification permet d'identifier pour deux graphes donnés leur niveau d'isomorphisme (i.e. à quel point les deux graphes sont isomorphes ou non isomorphes). De ce fait, plusieurs méthodes ont été proposées pour répondre à ces besoins, en se basant sur la structure du sous-graphe commun et du super-graphe commun. Ces mesures ont offert une certaine tolérance aux erreurs [175]. Dans ce contexte, l'appariement ne nécessite pas l'existence d'une fonction d'isomorphisme de (sous-)graphes pour que deux graphes puissent être comparés. Toutefois, même avec ces mesures, les graphes considérés similaires doivent partager des parties isomorphes. Cela signifie que, pour que ces mesures de distance soient faibles (i.e. proche de zéro) et qu'on considère les objets modélisés similaires, les deux graphes comparés doivent partager des parties identiques en terme de structure et d'étiquettes.

3.2.2 Appariement approximatif de graphes

L'appariement exact de graphes est fondé sur des théories mathématiques exactes (e.g la théorie des groupes) qui imposent des contraintes strictes pour la comparaison des graphes. La caractéristique *exacte* de l'appariement de graphes est héritée de ces théories mathématiques et est considérée comme un avantage. En pratique, cet avantage peut devenir un inconvénient. Dans plusieurs applications, deux graphes appartenant à la même classe ne sont pas forcément complètement (ou partiellement) identique en terme de leur structure. Par ailleurs, les étiquettes utilisées en pratique sont fréquemment de nature non discrète en reconnaissance de formes. Par conséquent, deux graphes appartenant à la même classe ont généralement (en plus de la structure) des étiquettes plus ou moins différentes. Ces différences en termes de structures et d'étiquettes sont dues à plusieurs causes telles que la présence de bruit et la variabilité des objets (appartenant à la même classe). En pratique, le bruit est omniprésent et les objets d'une même classe sont soumis à des déformations. Ainsi, il est évident que l'appariement exact de graphes ne répond pas aux attentes d'applications de la vie réelle, d'où sa rare utilisation.

L'appariement approximatif de graphes a été développé pour surmonter l'intolérance aux changements de structures et d'étiquettes et pour rendre l'appariement de graphes utilisable en pratique. L'objectif de l'appariement approximatif est de chercher une distance entre deux graphes même s'ils ne partagent pas une (sous-)structure commune. Ainsi, l'appariement approximatif tolère les différences dans les structures et les étiquettes des graphes considérés. Au lieu d'évaluer si deux graphes sont identiques (ou partiellement identiques), les algorithmes d'appariement approximatif cherchent à évaluer la similarité entre deux graphes. Cette similarité est calculée en affectant à chaque appariement entre une paire de nœuds un coût qui pénalise la différence entre les nœuds en termes de structure et d'étiquettes de graphes et favorise les ressemblances. Ainsi, les algorithmes d'appariement approximatif consiste à trouver un appariement entre deux graphes qui minimise le coût total des appariements nœud-à-nœud. Dans la littérature, plusieurs approches ont été proposées pour calculer un appariement approximatif

entre deux graphes.

Parmi ces méthodes, la distance d'édition de graphes a été largement utilisée comme la mesure de similarité la plus appropriée pour estimer une distance entre deux graphes [28]. L'idée de la distance d'édition de graphes est de définir la similarité de deux graphes par le nombre minimal d'opérations élémentaires d'édition nécessaires pour transformer un graphe à un autre. À l'origine, la distance d'édition de graphes est inspirée de la distance d'édition de chaînes de caractères [152] où les opérations d'édition autorisées sont : l'insertion, la suppression et la substitution de caractères. Ainsi, l'appariement de graphes par minimisation de la distance d'édition évalue la distance entre les graphes en comptant les opérations nécessaires les moins coûteuses pour rendre deux graphes isomorphes. Comme avec les chaînes de caractères, la distance d'édition de graphes est calculée par un ensemble standard d'opérations d'édition, i.e. les insertions, les suppressions et les substitutions. Ces opérations d'édition sont appliquées sur les nœuds et les arêtes. En outre, un certain coût est associé à chacune de ces opérations. Dans ce sens, Bunke [27] a introduit un ensemble de coûts des opérations pour lequel la distance d'édition de graphes est équivalent à la distance de graphes basée sur la taille du sous-graphe maximal commun (cf. $d_{Bunke97}(g_1, g_2)$, page : 40).

De toute évidence, pour chaque paire de graphes A et B, il existe différentes séquences d'opérations d'édition qui transforment A en B. Toutefois, le calcul de la distance d'édition entre deux graphes implique non seulement la recherche d'une séquence d'opérations d'édition pour transformer un graphe à l'autre, mais aussi la recherche d'une séquence qui possède le coût total minimal.

Dans ce qui suit, nous passons en revue les principaux travaux ayant trait à l'appariement approximatif de graphes. Pour une liste exhaustive des méthodes proposées depuis 1973, le lecteur intéressé peut consulter [52, 28].

Arbre de recherche

Comme pour l'appariement exact de graphes, l'arbre de recherche a été largement utilisé dans les algorithmes d'appariement approximatif de graphes. Le travail de Tsai et al. en 1979 [255] a ouvert la voie de l'utilisation d'arbre de recherche et des heuristiques. Toutefois, Tsai et al. ont introduit la notion du coût d'édition de graphes en se limitant aux opérations de substitutions des nœuds et des arêtes. L'heuristique utilisée est basée sur le calcul des futures appariements de nœuds en négligeant l'injectivité de la fonction d'appariement. Cette méthode a été étendue dans [256] pour prendre en compte d'autres opérations d'édition (i.e. insertion et suppression) sur les nœuds et les arêtes. Dans [273], Wong et al ont proposé une amélioration de l'heuristique pour qu'elle prenne en compte les appariements des arêtes.

Dans [30, 212], la notion de distance d'édition de graphes a été définie. Pour le calcul de cette distance, plusieurs techniques ont été élaborées en se basant sur l'algorithme A* [5] avec des heuristiques adaptées. Par ailleurs, l'optimisation du coût d'appariement approximatif entre deux graphes est connu comme un problème NP-difficile. En fait, la taille de l'arbre de recherche est exponentielle par rapport aux tailles des graphes considérés. Pour faire face à ce problème,

5. L'algorithme A* est proposé par Hart et al. dans [109]

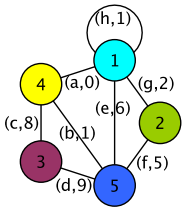
des algorithmes dits sous-optimaux ont été proposés. Le principe d'un algorithme d'appariement sous-optimal est le calcul d'un minimum local du coût d'appariement, au lieu d'un minimum global [96, 228, 226]. Eshera et Fu ont proposé dans [78, 79] une méthode sous-optimale avec une complexité polynomiale basée sur la décomposition de chaque graphe considéré en un ensemble de sous-graphes dits *graphes de base*. Dans cette méthode, pour chaque nœud n_i du graphe considéré, un graphe de base gb_i est construit avec toutes les arêtes et les nœuds adjacents à n_i. La figure 3.5 illustre un exemple de décomposition. Ensuite, l'appariement entre deux graphes est approché par la recherche de l'appariement entre les deux ensembles de graphes de base. Dans [78, 79], la programmation dynamique [6] a été utilisée pour définir la solution d'appariement optimale. Récemment, le couplage d'un graphe biparti (i.e. problème d'affection) [243, 201, 203] et la programmation linéaire [4, 134] étaient appliqués pour résoudre l'appariement approximatif de graphes. Les objectifs de toutes ces méthodes se résument à la réduction de la complexité et de l'espace de recherche d'appariements.

Dans un travail similaire à la méthode d'Eshera et Fu, Neuhaus et al. [178] ont proposé une technique basée sur la décomposition de graphes et l'algorithme A*-*beamsearch*. Cet algorithme explore toutes les possibilités d'appariements entre les nœuds et les arêtes d'un graphe et les nœuds et les arêtes d'un autre graphe grâce à l'utilisation de l'algorithme de recherche A*. Dans Algorithme 1, nous illustrons l'algorithme proposé par Neuhaus et al. dans [178]. L'ensemble OUVERT contient la construction dynamique de l'arbre de recherche A* dont les nœuds représentent les séquences de transformations partielles, et les feuilles, les séquences de transformations complètes. Concrètement, dans cet algorithme, les nœuds du premier graphe G_1 sont traités dans un ordre fixé *a priori*, ici $(u_1, u_2, ...)$. Ensuite, pour chaque nœud, toutes les opérations d'édition possibles sont construites simultanément (ligne 12 pour les suppressions et la ligne 11 pour les substitutions). Ceci produit un nombre de nœuds successeurs dans l'arbre de recherche. Si tous les nœuds du premier graphe sont traités, les nœuds non-traités du deuxième graphe G_2 sont ajoutés à G_1 (ligne 14). Les séquences partielles d'opérations d'édition sont stockées dans l'ensemble OUVERT qui est l'ensemble des nœuds de l'arbre de recherche. Cet ensemble sera considéré dans l'itération suivante de l'algorithme. Un nœud p de l'arbre de recherche (une séquence d'édition) est dit optimal s'il minimise le coût de la recherche de l'algorithme A* $g(p)+h(p)$ (ligne 5). La fonction $g(p)$ mesure les coûts d'édition cumulés jusqu'au nœud p de l'arbre de recherche. La fonction $h(p)$ estime les coûts d'édition à partir du nœud p jusqu'à une feuille de l'arbre de recherche. L'utilisation de *beamsearch* intervient dans la construction de l'arbre de recherche, i.e. l'ensemble OUVERT. En fait, au lieu de construire tous les nœuds possibles de l'arbre de recherche, juste un nombre fixe s de nœuds est maintenu dans OUVERT. Lorsqu'un nouveau nœud (une séquence partielle d'édition) est ajouté dans OUVERT, uniquement les s nœuds qui minimisent $g(p) + h(p)$ sont maintenus.

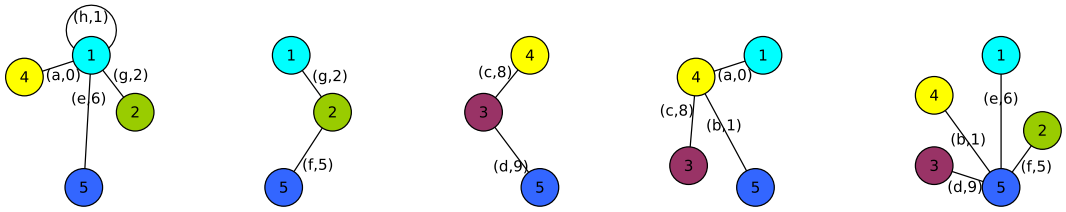
Méthodes spectrales

L'idée principale des méthodes spectrales est l'utilisation des valeurs et vecteurs propres de la matrice d'adjacence et/ou de Laplace. Les valeurs et les vecteurs propres sont invariants par rapport à l'ordre des nœuds d'un graphe. En fait, si deux graphes sont isomorphes, ils ont

6. La notion de la programmation dynamique est introduite dans [18]



(a)



(b) L'ensemble de graphes de base du graphe (a)

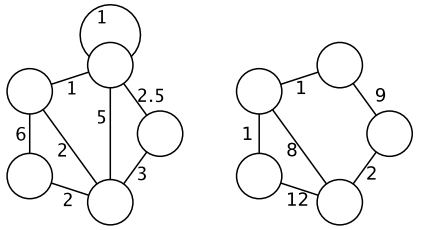
FIGURE 3.5 – Décomposition du graphe (a) en un ensemble de graphes de base (b) selon la méthode de Eshera et Fu [78, 79]

Algorithme 1 Distance d'édition de graphes [178]

ENTRÉES: Deux graphes $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ et $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, avec $V_1 = \{u_1, ..., u_{|V_1|}\}$ et $V_2 = \{v_1, ..., v_{|V_2|}\}$

SORTIES: La séquence d'opérations d'édition à coût minimal entre G_1 et G_2

1: Initialiser OUVERT comme un ensemble vide, i.e. OUVERT = $\emptyset$
2: Pour tout nœud $w \in V_2$, insérer la substitution $\{u_1 \rightarrow w\}$ dans OUVERT
3: Insérer la suppression $\{u_1 \rightarrow \varepsilon\}$ dans OUVERT
4: **Boucler**
5: Supprimer p_{min} = arg min$_{p \in OUVERT}$ $\{g(p) + h(p)\}$ de OUVERT
6: **Si** p_{min} est une séquence d'opérations d'édition complète **Alors**
7: retourner p_{min} comme solution
8: **Sinon**
9: Soit p_{min} =$\{u_1 \rightarrow v_{i_1}, ..., u_k \rightarrow v_{i_k}\}$
10: **Si** $k <$ | V_1 | **Alors**
11: pour tout nœud $w \in V_2 \setminus \{v_{i_1}, ..., v_{i_k}\}$, insérer $p_{min} \cup \{u_{k+1} \rightarrow w\}$ dans OUVERT
12: Insérer $p_{min} \cup \{u_{k+1} \rightarrow \varepsilon\}$ dans OUVERT
13: **Sinon**
14: Insérer $p_{min} \cup \bigcup_{w \in V_2 \setminus \{v_{i_1}, ..., v_{i_k}\}} \{\varepsilon \rightarrow w\}$ dans OUVERT
15: **Fin Si**
16: **Fin Si**
17: **Fin Boucler**



(a) (b)

FIGURE 3.6 – Deux graphes pondérés à apparier avec la méthode de Umeyama [258]

une même décomposition en valeurs propres. En revanche, l'inverse n'est pas toujours vrai, si deux décompositions en valeurs propres sont égales, les deux graphes correspondants ne sont pas forcément isomorphes. Le principal avantage des méthodes spectrales est la rapidité du calcul de décomposition en valeurs propres d'une matrice (i.e polynomiale), tandis que la principale limite est l'incapacité de traiter les graphes étiquetés. Ces méthodes sont dites purement structurelles [52, 160].

Parmi les premiers travaux qui utilisent la théorie spectrale pour l'appariement de graphes, nous détaillons la méthode proposé par Umeyama [258]. Cette méthode utilise la décomposition en valeurs propres des matrices d'adjacences de chaque graphe et définit la notion de matrice d'appariement. Cette matrice décrit l'appariement nœud-à-nœud entre deux graphes. La limitation majeure de l'approche de Umeyama est l'impossibilité à comparer des graphes de tailles différentes. Soient les deux graphes g_1 et g_2 illustrés respectivement dans les figures 3.6(a) et 3.6(b). L'appariement de g_1 et g_2 avec la méthode de Umeyama se déroule comme suit : La première étape consiste à calculer les matrices d'adjacence A_1 et A_2 de g_1 et g_2.

$$A_1 = \begin{pmatrix} 1 & 1 & 0 & 5 & 2.5 \\ 1 & 0 & 6 & 2 & 0 \\ 0 & 6 & 0 & 2 & 0 \\ 5 & 2 & 2 & 0 & 3 \\ 2.5 & 0 & 0 & 3 & 0 \end{pmatrix}, \qquad A_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 9 \\ 1 & 0 & 1 & 8 & 0 \\ 0 & 1 & 0 & 12 & 0 \\ 0 & 8 & 12 & 0 & 2 \\ 9 & 0 & 0 & 2 & 0 \end{pmatrix}$$

Ensuite, nous calculons la décomposition en valeur propres des matrices d'adjacence. Nous obtenons les valeurs propres λ_1 et λ_2 ainsi que les matrices modales U_1 et U_2 de chaque graphe [7].

λ_1 = (-6.1731, -4.7933, -1.9430, 4.7291, 9.1803,)

7. Les colonnes de la matrice modale U_1 (resp. U_2) sont formées par les vecteurs propres de la matrice d'adjacence A_1 (resp. A_2).

$$U_1 = \begin{pmatrix} 0.2324 & -0.5093 & -0.5086 & -0.4368 & -0.4870 \\ -0.6469 & -0.3285 & 0.0001 & 0.5292 & -0.4399 \\ 0.6960 & 0.0936 & 0.1255 & 0.5699 & -0.4078 \\ -0.2074 & 0.7612 & -0.1222 & -0.2400 & -0.5522 \\ 0.0067 & -0.2108 & 0.8430 & -0.3831 & -0.3131 \end{pmatrix}$$

λ_2 = (-14.3077, -8.7600, -0.9040, 8.7908, 15.1808)

$$U_2 = \begin{pmatrix} -0.1443 & 0.6920 & -0.0159 & -0.6962 & -0.1238 \\ 0.3635 & 0.0332 & -0.8361 & 0.0492 & -0.4067 \\ 0.5638 & 0.1986 & 0.5394 & 0.1693 & -0.5684 \\ -0.7025 & -0.1477 & 0.0290 & 0.1199 & -0.6852 \\ 0.1889 & -0.6773 & 0.0945 & -0.6855 & -0.1637 \end{pmatrix}$$

Nous calculons la matrice de similarité $M = \overline{U}_1.\overline{U}_2^T$ entre les vecteurs propres de g_1 et g_2, avec $\overline{U}_i$ qui est la valeur absolue de U_i. Un élément $M(i,j)$ correspond au produit scalaire du i-ème et du j-ème vecteur propre respectivement de g_1 et g_2.

$$M = \overline{U}_1.\overline{U}_2^T = \begin{pmatrix} 0.7585 & 0.7461 & 0.8573 & 0.6393 & 0.8160 \\ 0.7436 & 0.4511 & 0.7696 & 0.8679 & 0.7795 \\ 0.6144 & 0.5549 & 0.8069 & 0.8542 & 0.6641 \\ 0.7942 & 0.4392 & 0.6885 & 0.6688 & 0.8212 \\ 0.4658 & 0.8604 & 0.7432 & 0.3208 & 0.5376 \end{pmatrix}$$

Finalement, un algorithme de résolution du problème d'affection (dans ce cas la méthode hongroise [146]) est appliqué à la matrice M pour obtenir une matrice de permutation P. Cette matrice P décrit la correspondance entre les nœuds de g_1 et les nœuds de g_2. i.e.

$$P_{i,i} = \begin{cases} 1, & \text{si } v_i \text{ est apparié avec } u_j \quad (3.1a) \\ 0, & \text{sinon} \quad (3.1b) \end{cases}$$

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

À partir de la matrice de permutation P, la distance $d(g_1, g_2)$ entre les deux graphes g_1 et g_2 est calculée comme suit :

$$d(g_1, g_2) = d(P) = \| PA_1P^T - A_2 \|^2 = 10.7034$$

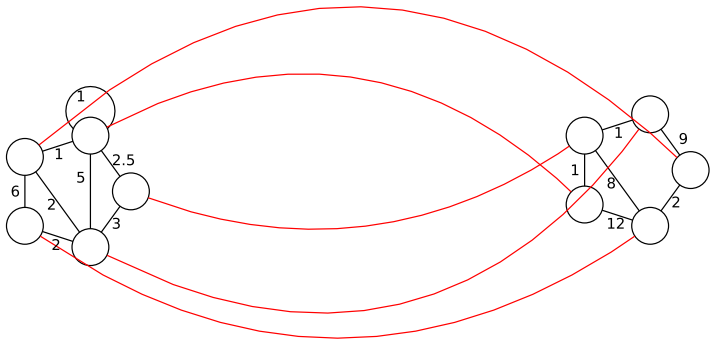


FIGURE 3.7 – L'appariement avec la méthode de Umeyama [258]

avec $\| \cdot \|$ la norme euclidienne d'une matrice($\| A \| = (\sum_{i=1}^{n} \sum_{i=1}^{n} \mid a_{ij} \mid^2)^{1/2}$).

Plus récemment, dans [161, 162], les auteurs proposent une technique spectrale du plongement des graphes dans un espace vectoriel pour exploiter les outils des approches statistiques de la reconnaissance de formes. Dans le même contexte, Wilson et Hancock [270] représentent les graphes sous forme spectrale en utilisant la décomposition en valeurs propres des matrices laplaciennes. La distance entre deux graphes est calculée à partir de la distance d'édition entre les représentations spectrales correspondantes [152]. Dans une autre approche, Robles-Kelly et Hancock [205] proposent une procédure de conversion de graphe en chaîne en utilisant la notion de vecteur propre principal pour ordonner les nœuds. Ensuite, la distance d'édition entre les chaînes correspondantes est calculée par la distance de Levenshtein [152]. Dans une autre perspective, Carcassoni et Hancock [43] utilisent un clustering des nœuds des graphes en se basant sur un ensemble de caractéristiques spectrales. Les appariements sont déduits à partir de ce clustering. Récemment, la marche aléatoire [103] et quantique [75] dans les graphes ont été aussi utilisées dans un contexte spectral pour estimer la distance entre deux graphes.

Les méthodes d'appariement de graphes basées sur la théorie spectrale de graphes fournissent des solutions rapides pour approcher l'appariement entre les graphes. Néanmoins, ces méthodes souffrent d'une sensibilité aux changements structurels dans les graphes [175]. En effet, les changements structurels, notamment sur la suppression de nœuds, engendrent des modifications importantes dans les représentations spectrales utilisées par la plupart des travaux évoquées précédemment. Ceci augmente la sensibilité des méthodes spectrales aux changements structurels dans les graphes et réduit leurs performances dans le processus de reconnaissance (i.e. la classification). Outre cette sensibilité, la grande majorité de méthodes spectrales d'appariement de graphes sont purement structurelles dans le sens où elles sont applicables uniquement aux graphes non-étiquetés, ou aux graphes étiquetés avec un ensemble d'étiquettes soumis à des contraintes (e.g. uniquement une étiquette numérique sur les arêtes).

Autres techniques

Le problème de l'appariement de graphes a été formulé également dans la littérature comme un problème d'étiquetage. Le principe général d'étiquetage par relaxation consiste à associer des étiquettes à l'ensemble des données à traiter avec des coefficients de confiance. Pour l'appariement de graphes, chaque nœud doit être associé à une étiquette qui spécifie son appariement à un nœud dans l'autre graphe. L'étiquetage initial peut être basé sur les attributs des nœuds, la connectivité, etc. Ensuite, à l'aide d'un algorithme itératif les coefficients et les étiquettes sont affinés jusqu'à l'obtention d'un étiquetage acceptable. Il faut noter que les coefficients sont généralement modélisés par des distributions de probabilité gaussienne. La première formulation d'étiquetage par relaxation est introduite par Kittler et Hancock dans [141]. Puis, plusieurs améliorations ont été proposées dans [47, 107, 116, 173, 269], telle que l'utilisation des étiquettes des arêtes et des nœuds dans le processus de relaxation et l'intégration d'une mesure bayésienne pour calculer la distance entre les graphes.

À côté de ces approches, il existe encore de nombreuses autres techniques. On peut citer les méthodes basées sur les réseaux de neurones artificiels [121, 245, 246] où l'appariement de graphes est formulé sous forme de minimisation d'énergie et résolu avec les réseaux de Hopfield. De plus, les algorithmes génétiques sont également utilisés pour apparier les graphes [244, 62, 9] où le problème est défini sous forme d'un ensemble d'états (i.e les appariements) avec des degrés d'adaptation (*fitness*). Parmi les nouvelles méthodes d'appariement de graphes, nous distinguons les méthodes à noyau [22, 105, 175, 177, 220, 230]. L'idée originale des méthodes à noyau est de transférer le problème de reconnaissance de formes vers un espace vectoriel au lieu d'utiliser l'espace des objets d'origine. Pour les graphes, au lieu de définir des outils mathématiques dans l'espace des graphes, tous les graphes sont projetés dans un espace vectoriel où plusieurs outils de calcul de distance performants sont disponibles.

Toutes les méthodes présentées précédemment sont des méthodes dites univoques, dans le sens où chaque nœud d'un graphe est apparié à au plus un nœud de l'autre graphe. Au contraire de ces méthodes, il existe certaines approches dites multivoques [6, 45] qui permettent de mettre en correspondance un nœud d'un graphe à plusieurs nœuds d'un autre graphe. Dans ce sens, la méthode de Ambauen et al. [6] définit une distance d'édition de graphes *étendue* en se basant sur un appariement multivoque et en introduisant les opérations d'édition de fusion et d'éclatement de nœuds.

Discussion

À l'encontre des méthodes d'appariement exacts de graphes, les méthodes approximatives sont tolérantes aux erreurs, dans le sens où il est possible d'apparier deux graphes même s'il ne possèdent pas la même structure et les mêmes étiquettes. Ces différences dans les structures et les étiquettes sont utilisées pour évaluer, généralement, la différence entre deux graphes par une distance. Dans la littérature, l'appariement approximatif de graphes est souvent traité comme un problème de distance d'édition. L'idée de la distance d'édition de graphes est de définir la similarité de deux graphes par le nombre minimal d'opérations d'édition nécessaires pour transformer un graphe à un autre. Ainsi, la distance d'édition de graphes peut être considérée comme une généralisation de la distance d'édition de chaînes de caractères. En effet, les opérations

d'édition considérées pour les graphes sont celles utilisées dans la distance d'édition de chaînes de caractères (i.e. ajout, suppression et substitution). Cependant, dans le domaine des graphes ces opérations sont appliquées aux nœuds et aux arêtes, tandis que dans le domaine des chaînes de caractères les opérations d'édition sont appliquées uniquement sur les caractères.

Bien que la distance d'édition de chaînes de caractères ait une complexité $O(nm)$ [8] [264], la distance d'édition de graphes est caractérisée par une complexité exponentielle en temps et en espace [28]. Pour surpasser cette complexité élevée, quelques méthodes sont proposées dans la littérature pour approximer une solution sous-optimale au lieu d'une solution optimale. Dans ce chapitre, nous avons passé en revue un ensemble de ces méthodes. Ces techniques d'approximation de distance d'édition de graphes se différencient par la manière de modéliser le problème d'approximation. Une première partie de méthodes modélise le problème d'approximation de distance d'édition en un arbre de recherche où les nœuds correspondent aux différentes séquences d'édition possibles entre deux graphes. Dans ces méthodes chaque séquence d'édition est évaluée par la somme de coûts des opérations d'édition qu'elle contient. Néanmoins, la définition des coûts des opérations d'édition est un problème en soi. Dans les travaux proposés par Bunke et al. [30, 171, 177, 178, 203, 201], les coûts des opérations d'édition sont prédéfinis *a priori*. En effet, la définition de ces coûts dépend de l'application et du type de graphes considérés. Dans le cas d'absence d'un apprentissage préalable, la définition de ces coûts n'est possible que d'une manière aléatoire ce qui implique éventuellement une dégradation des résultats. D'autres méthodes contournent ce problème en se basant sur la théorie spectrale de graphes. Ces techniques se caractérisent par l'utilisation de la matrice d'adjacence (ou laplacienne) pour extraire une signature (spectrale) d'un graphe. Ces signatures spectrales sont utilisées pour estimer la distance d'édition entre les graphes. Les travaux proposés par Hancock et al. [116, 160, 43, 162, 205, 192, 76] fournissent plusieurs utilisations de la théorie spectrale pour la distance d'édition de graphes. Mais, l'inconvénient majeur de ces méthodes est que les graphes sont contraints de ne contenir que des étiquettes numériques simples. En effet, les méthodes spectrales ne gèrent pas les graphes étiquetés, notamment lorsqu'il s'agit d'étiquettes symboliques et/ou d'étiquettes composées.

3.3 Conclusion

Dans ce chapitre, nous avons établi un état de l'art des méthodes d'appariement de graphes. Ces méthodes se répartissent en deux catégories : l'appariement exact et l'appariement approximatif. Dans la première catégorie, l'objectif est de tester si deux graphes sont isomorphes en vérifiant leur identité en terme de structures et d'étiquettes. Par contre, dans la catégorie d'appariement approximatif, l'objectif est de déterminer une distance entre deux graphes qui peuvent être structurellement différents. Le premier constat de cette étude bibliographique est la complexité élevée de l'appariement de graphes. Ensuite, nous avons constaté que l'appariement approximatif de graphes est plus adapté que l'exacte dans le domaine de la reconnaissance de formes. Néanmoins, les méthodes d'appariement approximatif existantes ont des limites différentes. Principalement, nous en avons distingué deux : la première limite concerne la nécessité

8. Avec n et m les tailles respectives de deux chaînes de caractères à comparer

de prédéfinir les coûts des opérations d'édition pour certaines méthodes. La deuxième limite concerne les méthodes basées sur la théorie spectrale où les graphes sont contraints de ne contenir que des étiquettes numériques simples.

Dans le chapitre suivant, nous proposons une nouvelle approximation de la distance d'édition de graphes non-paramétrique et sans contraintes sur les étiquettes. Notre méthode calcule automatiquement les coûts d'édition (suppression, ajout et substitution) des nœuds et des arêtes qui peuvent avoir des étiquettes numériques et/ou symboliques. Pour ce faire, nous proposons une nouvelle signature de nœuds qui caractérise chaque nœud dans un graphe en utilisant les caractéristiques de son voisinage.

Chapitre 4

Une distance d'édition basée sur la signature des nœuds

Evidently, for this approach to be advantageous, the simplest subpatterns selected [...] should be much easier to recognize than the patterns themselves.

King Sun Fu

Sommaire

4.1 Introduction

Dans le chapitre 2, nous avons constaté la grande capacité des graphes à représenter différents type d'images. En utilisant les graphes comme un modèle de représentation, nous pouvons ainsi transformer un problème de reconnaissance en un problème d'appariement entre les graphes. Dans le chapitre précédent, nous avons passé en revue plusieurs méthodes d'appariement de graphes. Ces travaux peuvent être classés dans deux familles d'appariement de graphes. Dans la première famille des méthodes, dite appariement exacte, pour que deux graphes

soient similaires il faut qu'ils partagent exactement la même structure et les mêmes étiquettes (s'ils sont étiquetés). Ces conditions très rigides ont limité l'utilisation de ces méthodes dans les applications du monde réel. A *contrario*, les méthodes d'appariement approximatif de graphes sont tolérantes aux erreurs, dans le sens où il est possible d'apparier deux graphes même s'il ne possèdent pas la même structure et les mêmes étiquettes. Ces différences dans les structures et les étiquettes sont utilisées pour évaluer, généralement, la différence entre deux graphes par une distance. Parmi ces méthodes, la distance d'édition de graphes a été largement utilisée comme la mesure de similarité la plus appropriée pour estimer une distance entre deux graphes [28]. L'idée de la distance d'édition de graphes est de définir la similarité de deux graphes par le nombre minimal d'opérations d'édition nécessaires pour transformer un graphe à un autre. Néanmoins, le calcul de la séquence d'opérations d'édition de coût minimal est un problème NP-difficile dans le cas général. Pour surpasser cette complexité élevée, quelques méthodes sont proposées dans la littérature pour approximer une solution sous-optimale au lieu d'une solution optimale. Dans ces méthodes nous avons relevé, dans le chapitre précédent, deux inconvénients majeurs. Le premier inconvénient concerne les coûts des opérations d'édition qui sont prédéfinis *a priori*. En effet, la définition de ces coûts dépend de l'application et du type de graphes considérés. Le deuxième inconvénient concerne les travaux basés sur la théorie spectrale où les graphes sont contraints de ne contenir que des étiquettes numériques simples.

Dans ce chapitre, nous proposons une nouvelle approximation de la distance d'édition de graphes non-paramétrique et sans contrainte sur les étiquettes. Plus précisément, notre méthode calcule directement les coûts d'édition sur les nœuds et les arêtes qui peuvent avoir des étiquettes numériques et/ou symboliques. Nous introduisons dans ce chapitre la signature de nœud qui est un ensemble de vecteurs qui caractérisent chaque nœud dans le graphe. Ensuite les signatures sont utilisées pour transformer le problème d'appariement à un problème d'affection qui sera est via la méthode hongroise.

4.2 Formulation du problème

Les premiers travaux sur la distance d'édition de graphes [212] ont été inspirés de la distance d'édition de chaînes de caractères [152, 264]. Ainsi, avant d'entamer la distance d'édition de graphes, nous commençons par une succincte introduction de la distance d'édition de chaînes de caractères.

Soit s une chaîne de caractères dans l'alphabet Σ_S. On note $s[i]$ le i-ème caractère de s, $s[i...j]$ la sous-chaîne composée par $s[i]...s[j]$, $|s|$ la taille de s (nombre de caractères), et ε le caractère nul ($\varepsilon \notin \Sigma_S$). Une *opération d'édition* de chaînes de caractères est une paire $(a, b) \neq (\varepsilon, \varepsilon)$ notée $a \rightarrow b$ qui est une transformation élémentaire appliquée au caractères a pour obtenir les caractères b. Une transformation $a \rightarrow b$ peut être, soit une substitution si $a \neq \varepsilon$ et $b \neq \varepsilon$, soit une suppression si $b = \varepsilon$, soit une insertion si $a = \varepsilon$. Soient A et B deux chaînes de caractères et $S = s_1, s_2, ..., s_m$ une séquence d'opérations d'édition qui transforme A en B. Soit $c(.)$ une fonction qui affecte à chaque opération d'édition $(a \rightarrow b)$ un coût, i.e. $c(a \rightarrow b) = x$ avec $x \in \mathbb{R}^+$. Le coût total de la transformation de A en B est la somme des coûts d'opérations d'édition appliquées, soit $c(S) = \sum_{i=1}^{m} c(s_i)$. Ainsi, la distance d'édition entre A et B, $d(A, B)$, est définie comme le coût minimal de toutes les

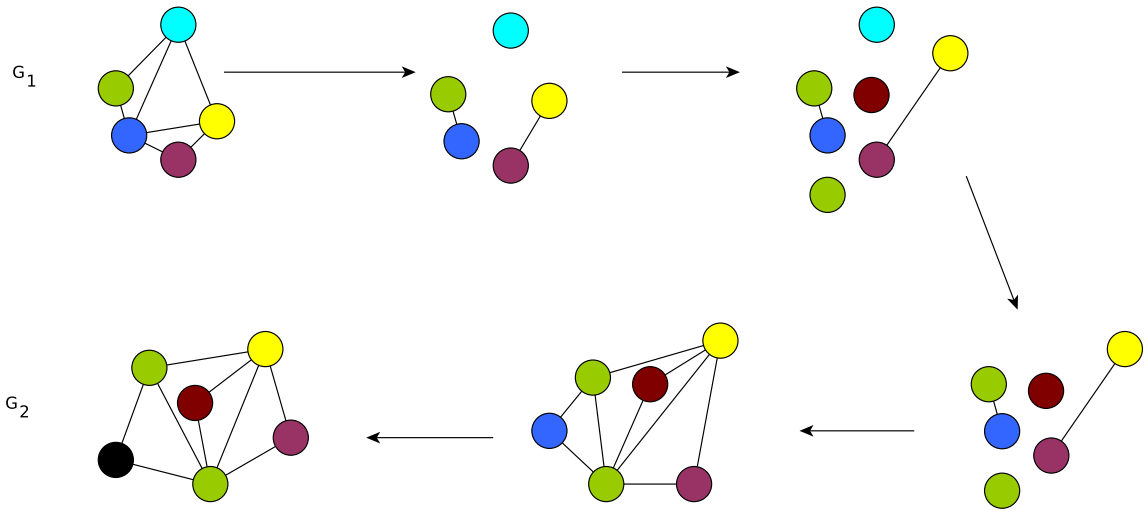


FIGURE 4.1 – Un exemple d'une séquence d'opérations d'édition pour transformer G_1 à G_2

séquences d'opérations d'édition qui transforme A en B. Formellement, $d(A, B) = min\{c(S) \mid S$ est une séquence d'édition qui transforme A en $B\}$.

La distance d'édition de graphes s'inscrit dans la même philosophie que celle des chaînes de caractères. En effet, la distance d'édition entre deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, est calculée à partir de la séquence d'opérations d'édition la moins coûteuse qui transforme G_1 en un graphe isomorphe G_2. Les opérations d'édition standards sont intégralement inspirées des transformations élémentaires appliquées aux chaînes de caractères, soit l'insertion (ajout), la suppression et la substitution (changement). Quoique, quelques transformations élémentaires ont été développées spécialement pour les graphes, comme la fusion et la division des nœuds [6, 44, 101]. Dans le domaine des graphes, les opérations d'édition sont appliquées aux nœuds et arêtes. Formellement, nous définissons la distance d'édition de graphes comme suit :

La distance d'édition de graphes *Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux graphes, la distance d'édition entre G_1 et G_2 est :*

$$d(G_1, G_2) = \min_{(e_1,..,e_k)\in\gamma(G_1,G_2)} \sum_{i=1}^{k} c(e_i)$$

avec $\gamma(G_1, G_2)$ un ensemble de séquences d'opérations d'édition qui transforme le graphe G_1 en G_2 et $c(e_i)$ le coût de l'opération d'édition e_i.

La figure 4.1 illustre un exemple d'une séquence d'opérations d'édition entre deux graphes. Cette séquence est composée de quatre suppressions d'arêtes, une insertion de deux nouveaux nœuds, une suppression de nœud et une insertion de sept nouvelles arêtes.

De toute évidence, pour chaque paire de graphes G_1 et G_2, il existe différentes séquences d'opérations d'édition pour transformer G_1 en G_2. Comme pour les chaînes de caractères, le

calcul de la distance d'édition entre deux graphes implique non seulement la recherche d'une séquence d'opérations d'édition pour transformer un graphe à un autre, mais aussi la recherche d'une séquence qui possède le coût total minimal. Pour évaluer d'une manière quantitative les séquences d'édition afin de déterminer la moins coûteuse, une fonction de coût est définie. Cette fonction affecte à chaque opération d'édition un coût qui reflète le niveau de distorsion de l'opération concernée. Par exemple, la substitution d'un nœud ayant l'étiquette [0, 5.5] par un nœud ayant l'étiquette [10, 60] doit forcément avoir un coût plus important que la substitution de deux nœuds ayant respectivement les étiquettes [0, 5.5] et [0, 5.8].

Par conséquent, la définition d'une fonction de coût est une composante clé de la distance d'édition de graphes. En effet, les résultats de la fonction de coût influent considérablement sur le choix de la séquence d'édition optimale. Les travaux existants, dans la littérature, sur la distance d'édition de graphes ont rarement focalisé leur intérêt sur la détermination de la fonction de coût. Dans la plupart de ces travaux, la fonction de coût est définie *a priori* et est limitée à une ou quelques applications précises. L'un des rares travaux, qui essayent d'établir un protocole pour définir d'une manière automatique les fonctions de coûts, est le travail de Neuhaus [176] où une méthode basée sur l'apprentissage des coûts d'opérations d'édition est proposée.

En plus de la séquence optimale d'opérations d'édition qui rend deux graphes isomorphes, on peut s'interroger sur la façon dont les nœuds des deux graphes sont appariés. L'appariement consiste ici à identifier la correspondance entre les nœuds des deux graphes. Par ailleurs, nous nous intéressons à définir une bijection qui détermine l'appariement optimal entre deux graphes. Évidemment, les graphes peuvent avoir des tailles différentes et quelques nœuds dans un graphe, généralement le plus grand, ne peuvent être appariés. À partir de cette observation nous définissons l'appariement optimal entre deux graphes comme :

Appariement optimal de graphes *Soient A=(V_a,E_a) et B=(V_b,E_b) deux graphes. Soit $\lambda_a \subseteq V_a$ les nœuds appariés dans A et soit $\lambda_b \subseteq V_b$ les nœuds appariés dans B. Un appariement optimal entre A et B est défini par la fonction bijective $\sigma : \lambda_a \rightarrow \lambda_b$, tel que*

$$\forall\, \upsilon \in \lambda_a,\ \exists\, \nu \in \lambda_b, \text{ tel que } \sigma(\upsilon) = \nu \text{ (}\upsilon \text{ est apparié à } \nu\text{)}$$

Il faut noter ici que la définition de l'appariement optimal se limite à l'identification des correspondances optimales entre les nœuds de deux graphes, sans calculer une distance entre les graphes concernés. À partir de la fonction d'appariement optimal σ, nous pouvons déduire les opérations d'édition effectué entre les deux graphes.

Soient $A = (V_a, E_a)$ et $B = (V_b, E_b)$ deux graphes, et $\sigma : \lambda_a \rightarrow \lambda_b$ une fonction d'appariement optimal entre A et B. Nous associons à chaque appariement, entre les nœuds et les arêtes de A et B, l'un des coût suivants :

- $\zeta_{nr}(\upsilon)$: le coût de substitution du nœud $\upsilon \in \lambda_a$ par le nœud $\sigma(\upsilon) \in \lambda_b$.
- $\zeta_{ni}(\upsilon)$: le coût d'insertion du nœud $\upsilon \in V_b/\lambda_b$ dans A.
- $\zeta_{ns}(\upsilon)$: le coût de suppression du nœud $\upsilon \in V_a/\lambda_a$ de A.
- $\zeta_{er}(e)$: le coût de substitution de l'arête $e = \upsilon\nu \in E_a$ par l'arête $\sigma(\upsilon)\sigma(\nu) \in E_b$. Dans la suite, nous notons par E_r l'ensemble des arêtes substituées dans E_a. Ici nous nous n'intéressons pas au multi-graphe, i.e deux nœuds peuvent être réliés par au plus une arête.

- $\zeta_{ei}(e)$: le coût d'insertion de l'arête $e = v\nu \in E_b$ dans A. Dans la suite, nous notons par E_i l'ensemble des arêtes insérées dans A.
- $\zeta_{es}(e)$: le coût de suppression de l'arête $e = v\nu \in E_a$ de A. Dans la suite, nous notons par E_s l'ensemble des arêtes supprimées de A.

Les ensembles E_r, E_i et E_s sont déduits de la fonction d'appariement optimale. Nous définissons ces ensembles comme suit :

L'ensemble des arêtes substituées E_r : Cet ensemble contient les arêtes dont les deux extrémités (nœuds adjacents) sont substituées, dans le graphe A, par deux nœuds adjacents du graphe B. Formellement,

$E_r = \{(u, v) \in E_a | u, v \in \lambda_a \text{ et } (\sigma(u), \sigma(v)) \in E_b\}$

L'ensemble des arêtes insérées E_i : Cet ensemble contient les arêtes, du graphe B, dont les deux extrémités sont insérées dans A, les arêtes ayant une extrémité insérée et l'autre extrémité substituée, et les arêtes, du graphe B, dont les deux extrémités (nœuds adjacents) sont substituées dans le graphe A. Formellement,

$E_i = \{(u, v) \in E_b | u, v \in \lambda_b \text{ et } (\sigma^{-1}(u), \sigma^{-1}(v)) \notin E_a \text{ ou } ((u \notin \lambda_b \text{ ou } v \notin \lambda_b)\}$

L'ensemble des arêtes supprimées E_s : Cet ensemble correspond à l'ensemble des arêtes du graphe A dont au moins une extrémité est supprimée, et les arêtes dont les deux extrémités (nœuds adjacents) sont substituées, dans le graphe A, par deux nœuds non-adjacents du graphe B. Formellement,

$E_s = \{(u, v) \in E_a | u \notin \lambda_a \text{ ou } v \notin \lambda_a \text{ ou } ((u, v) \in \lambda_a \times \lambda_a \text{ et } (\sigma(u), \sigma(v)) \notin E_b)\}$

Nous pouvons ainsi ré-écrire la définition de la distance entre les graphes à partir de la fonction d'appariement optimale. En fait, le résultat de fonction d'appariement σ nous permet de déduire la séquence des opérations d'édition optimale appliquée à un graphe pour le rendre isomorphe à un autre. Par exemple, si un nœud d'un graphe A n'appartient pas à l'ensemble des nœuds appariés λ_a entre A et un graphe B, forcément il sera supprimé lors de la transformation de A en B.

Distance de graphes *Soient A=(V_a,E_a) et B=(V_b,E_b) deux graphes. Soient $\lambda_a \subseteq V_a$ les nœuds appariés dans A, $\lambda_b \subseteq V_b$ les nœuds appariés dans B et l'appariement optimal entre A et B défini par la bijection $\sigma : \lambda_a \to \lambda_b$. La distance entre A et B est donnée par $d(A, B)$:*

$$\begin{aligned} d(A,B) \quad = \quad & \sum_{v \in \lambda_a} \zeta_{nr}(v) + \sum_{v \in V_b/\lambda_b} \zeta_{ni}(v) + \sum_{v \in V_a/\lambda_a} \zeta_{ns}(v) \\ & + \sum_{e \in E_r} \zeta_{er}(e) + \sum_{e \in E_i} \zeta_{ei}(e) + \sum_{e \in E_s} \zeta_{es}(e) \end{aligned} \tag{4.1}$$

Comme indiqué précédemment, la fonction de coût a une influence déterminante sur la performance de la distance d'édition de graphes. Dans cette thèse, nous proposons une technique permettant de calculer le coût d'édition d'une manière indépendante du domaine de l'application. Toutefois, la fonction d'appariement optimal (σ) doit être connue à l'avance. Nous proposons aussi une technique basée sur les signatures des nœuds pour définir la fonction d'appariement.

4.3 Descriptions locales de graphes

Dans ce travail de thèse, nous présentons un algorithme qui réduit le problème d'appariement approximatif de graphes en un problème de couplage de graphe biparti en utilisant la notion de signature de nœuds. Le couplage d'un graphe biparti a été déjà utilisé auparavant dans la littérature pour résoudre l'appariement de graphes. Cette utilisation [203, 233] a été limitée à modéliser l'appariement de graphes sous forme d'un problème d'affectation qui peut se résoudre avec une complexité polynomiale. Notre principal apport est d'intégrer la notion de signature de nœuds dans le calcul du problème d'affection (i.e. couplage de graphe biparti). Dans cette section, nous introduisons la définition d'une signature d'un nœud ainsi que la technique de calcul de distance entre les différentes signatures.

4.3.1 Signature de nœuds

Dans la littérature, la majeure partie des algorithmes d'appariement traitent les graphes avec une représentation globale. Chaque graphe est géré comme une seule entité qui peut être un vecteur [157, 205], une matrice [232] ou une chaîne de caractères [205]. Toutefois, quelques travaux ont proposé d'utiliser des concepts de représentation de graphes locaux, comme les graphes de base [79] et les voisinages des nœuds [233, 103]. Les différences entre ces travaux résultent dans les méthodes d'extraction des descriptions locales et dans les contraintes qu'elles imposent. Parmi les approches d'extraction, nous distinguons l'approche spectrale, l'approche de décomposition et les marches aléatoires. Mais, leur problème commun est la complexité de l'extraction de ces descriptions locales.

Dans notre méthode, nous représentons chaque graphe par un ensemble de descriptions locales qui sont liées aux caractéristiques des nœuds et utilisées pour calculer la distance de nœud-à-nœud. Dans la suite, nous désignons les descriptions locales par les signatures des nœuds. Contrairement aux travaux précédents, notre signature de nœud est un ensemble de vecteurs simples qui sont extraits par un parcours du graphe.

Pour construire une signature pour un nœud dans un graphe, nous utilisons toutes les informations liées à ce nœud et disponibles dans le graphe. Ces informations sont : les attributs du nœud, le degré du nœud, les attributs des arêtes incidentes et les degrés des nœuds adjacents. La collecte de ces informations devrait être affinée dans une structure adéquate qui sera utilisée pour calculer les distances entre les nœuds. Dans cette perspective, nous définissons la signature d'un nœud comme un tuple d'arité 4 constitué de l'attribut du nœud, le degré du nœud et les attributs de ses arêtes incidentes et les degrés des nœuds adjacents.

Signature d'un nœud *Étant donné un graphe $G = (V, E, \alpha, \beta)$, la signature de nœud $n \in V$ est formulé comme suit :*

$$\gamma(n) = \Big(\alpha_n^G,\ \theta_n^G,\ \Delta_n^G,\ \Omega_n^G \Big)$$

avec

- α_n^G : l'ensemble des attributs de n.
- θ_n^G : le degré de n.

– $\Delta_n^G = \{\theta_m^G \mid m \in V, m \text{ adjacent à } n\}$: le multi-ensemble [9] de degrés des nœuds adjacents à n.

– $\Omega_n^G = \{\beta(e) \mid e \in E, e \text{ incidente à } n\}$: le multi-ensemble d'attributs des arêtes incidentes à n.

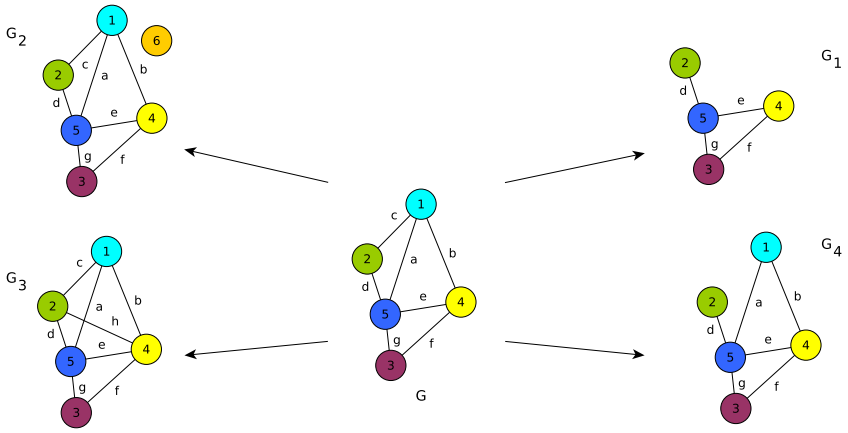


FIGURE 4.2 – Changements structuraux effectués sur un graphe

Afin de calculer la distance d'édition de graphes, il est important que cette notion de signature de nœuds soit particulièrement insensible aux faibles changements structurels, mais sensible aux changements structurels importants. Dans un graphe, les changements structurels qui provoquent une grande déformation de la structure sont généralement liés aux nœuds, notamment l'insertion et la suppression des nœuds. Dans la figure 4.2, nous illustrons deux changements du graphe G liés aux nœuds et deux changements liés aux arêtes. Les deux graphes G_1 et G_2 sont obtenus après, respectivement, une suppression et une insertion d'un nœud de G. Les deux graphes G_3 et G_4 sont obtenus après, respectivement, une insertion et une suppression d'une arête de G. Il est clair que les structures de G_1 et G_2 sont plus déformées que les structures de G_3 et G_4 par rapport au graphe d'origine G.

En effet, la suppression et/ou l'insertion des nœuds dans un graphe impliquent des déformations structurelles plus importantes que les opérations d'édition effectuées sur les arêtes (insertion, suppression, substitution). Dans cette perspective, nous essayons à travers la représentation de graphes sous forme d'ensembles de signatures de nœuds d'accentuer les changements structuraux importants et de minimiser relativement les petits changements. Reprenons l'exemple de la figure 4.2. Dans la table 4.1, nous représentons les trois graphes G, G_1 et G_4 par leurs ensembles de signatures de nœuds. Nous constatons que l'ensemble de signatures de nœuds du graphe G_4 est plus semblable à l'ensemble de signatures de G que l'ensemble de signatures de G_1. Cette similarité s'observe dans le nombre des signatures et le nombre de modifications

9. Un multi-ensemble est un ensemble fini dans lequel chaque élément peut y appartenir plusieurs fois, e.g. M=$\{1, 0, 0, 1, 3\}$

effectuées sur les signatures de G pour retrouver les signatures de G_1 et G_4.

Multi-ensemble de signatures des nœuds d'un graphe *Étant donné un graphe $g = (V, E, \alpha, \beta)$. Le multi-ensemble S_γ de signatures des nœuds de g est défini par :*

$$S_\gamma(\, g\,) = \Big\{\ \gamma(n_i) \mid \forall n_i \in V\ \Big\}$$

S_γ est une collection de descriptions locales. Un changement local dans le graphe n'impliquera des changements que dans les sous-ensembles de S_γ concernés par les modifications, tout en laissant le reste des descriptions (i.e. signatures) inchangées.

4.3.2 La distance entre les signatures des nœuds

Pour évaluer la similarité entre deux entités dans un espace multidimensionnel, une mesure de distance est nécessaire. Bien que plusieurs mesures de distance aient été proposées [268], les plus utilisées sont fonctionnelles uniquement soit pour les vecteurs symboliques soit pour les vecteurs numériques. Il s'agit notamment de la distance Euclidienne et la distance de Manhattan [66] pour les vecteurs avec des attributs numériques, et la distance de Jaccard [118] pour les vecteurs avec des attributs symboliques. Dans notre cas, la signature d'un nœud peut contenir différents types d'attributs. La question qui se pose est : *comment peut-on calculer une distance entre deux signatures qui contiennent des données numériques et symboliques ?*

Dans [268], les auteurs passent en revue une liste de métriques qui gèrent les vecteurs contenant des données numériques et symboliques [1, 2, 97]. Ces métriques sont basées sur la fonction de distance hétérogène. Elles sont classées en deux familles. D'une part, les distances basées sur la différence de valeurs métriques, par exemple la *métrique hétérogène de la différence de valeurs* (*Heterogeneous Value Difference Metric*). D'autre part, les distances basées sur la distance Euclidienne, par exemple la *métrique euclidienne hétérogène de superposition*. Les distances de la première famille ne sont utilisées que dans un contexte de classification. En effet, les distances basées sur la différence de valeurs métriques nécessitent une phase d'apprentissage afin d'utiliser les informations de chaque classe dans le calcul de distances. Cependant, les distances de la deuxième famille n'imposent aucune phase d'apprentissage. Ceci nous a guidé à choisir cette deuxième famille, soit *la métrique euclidienne hétérogène de superposition* (HEOM), pour calculer la distance entre deux signatures de nœuds. La distance HEOM utilise la métrique de superposition pour calculer la distance entre les attributs symboliques, et la distance euclidienne normalisée pour les attributs numériques.

La distance HEOM *Étant donné deux vecteurs hétérogènes I et J, la distance HEOM entre **I** et **J** est :*

$$HEOM(I, J) = \sqrt{\sum_{a=0}^{A} \delta(i_a, j_a)^2} \tag{4.2}$$

avec i_a un attribut de I, j_a un attribut de J et $A = max\{|I|, |J|\}$. La fonction $\delta(i_a, j_a)$ est

TABLE 4.1 – Les signatures des nœuds des graphe G, G_1 et G_4

G		
$\gamma(n_1)$	=	({1,“Cyan”}, 3,{4,2,2},{“a”,“b”,“c”})
$\gamma(n_2)$	=	({2,“Vert”},2,{4,3},{“d”,“c”})
$\gamma(n_3)$	=	({3,“Mauve”},2,{4,3},{“g”,“f”})
$\gamma(n_4)$	=	({4,“Jaune”},3,{4,3,2},{“e”,“b”,“f”})
$\gamma(n_5)$	=	({5,“Bleu”},4,{3,3,2,2},{“a”,“e”,“g”,“d”})
G_1		
$\gamma(n_2)$	=	({2,“Vert”},1,{3},{“d”})
$\gamma(n_3)$	=	({3,“Mauve”},2,{3,2},{“g”,“f”})
$\gamma(n_4)$	=	({4,“Jaune”},2,{3,2},{“e”,“f”})
$\gamma(n_5)$	=	({5,“Bleu”},3,{2,2,1},{“e”,“g”,“d”})
G_4		
$\gamma(n_1)$	=	({1,“Cyan”}, 2,{4,2},{“a”,“b”})
$\gamma(n_2)$	=	({2,“Vert”},1,{3},{“d”})
$\gamma(n_3)$	=	({3,“Mauve”},2,{4,3},{“g”,“f”})
$\gamma(n_4)$	=	({4,“Jaune”},3,{4,2,2},{“e”,“b”,“f”})
$\gamma(n_5)$	=	({5,“Bleu”},4,{3,2,2,1},{“e”,“g”,“a”,“d”})

définie comme suit :

$$\delta(i_a,j_a) = \begin{cases} 1 & \text{si } i_a \text{ ou } j_a \text{ n'existe pas} \\ \textit{Overlap}(i_a,j_a) & \text{si les variables sont symboliques} \\ \textit{rn\_diff}_a(i_a,j_a) & \text{si les variables sont est numériques} \end{cases}$$

avec

$$\textit{Overlap}(i_a,j_a) = \begin{cases} 0 & \text{si } i_a = j_a \\ 1 & \text{sinon} \end{cases}$$

et

$$\textit{rn\_diff}_a(i_a,j_a) = \frac{|i_a - j_a|}{norm_a}$$

La valeur de $norm_a$ est utilisée pour normaliser les attributs.

Cette distance gère les attributs manquants avec le retour d'une valeur maximale égale à 1. De plus, elle gère harmonieusement les attributs symboliques et numériques avec l'utilisation de la normalisation par le biais de $norm_a$. Cette normalisation garantit une valeur globale de la distance HEOM bornée entre 0 et 1.

Considérons maintenant le problème du calcul de distance entre les signatures de nœuds. Pour mesurer une telle distance, nous procédons d'abord par le calcul de la distance entre les sous-ensembles de chaque signature d'une manière indépendante. Soient $A = (V_a, E_a)$ et $B = (V_b, E_b)$ deux graphes, avec $n_a \in V_a$ et $n_b \in V_b$. Soient $\gamma(n_a)$ et $\gamma(n_b)$ les signatures respectives de n_a et n_b, avec :

$$\gamma(n_a) = \left(\alpha^A_{n_a},\ \theta^A_{n_a},\ \Delta^A_{n_a},\ \Omega^A_{n_a} \right)$$

$$\gamma(n_b) = \left(\alpha^B_{n_b},\ \theta^B_{n_b},\ \Delta^B_{n_b},\ \Omega^B_{n_b} \right)$$

La distance entre les signatures des nœuds implique la quantification de la somme des distances entre les différentes composants des signatures. Par conséquent, la distance entre $\gamma(n_a)$ et $\gamma(n_b)$ correspond à la somme des distances entre chaque composant de $\gamma(n_a)$ et son homologue dans $\gamma(n_b)$.

Dans la suite, nous définissons une distance $\mathrm{d}_\alpha(\alpha^A_{n_a}, \alpha^B_{n_b})$ entre les attributs des deux nœuds, $\mathrm{d}_\theta(\theta^A_{n_a}, \theta^B_{n_b})$ la distance entre les degrés de n_a et n_b et $\mathrm{d}_\Delta(\Delta^A_{n_a}, \Delta^B_{n_b})$ la distance entre les ensembles des degrés des nœuds adjacents. Il faut noter que les degrés de nœuds sont ajoutés dans la signature dans un ordre décroissant. Finalement, nous définissons la distance $\mathrm{d}_\Omega(\Omega^A_{n_a}, \Omega^B_{n_b})$ entre les ensembles des attributs des arêtes incidentes. Évidemment, on peut se demander si l'ordre des arêtes de chaque ensemble peut influer sur le résultat de d_Ω. En effet, le changement de l'ordre de l'ajout des attributs des arêtes dans une signature peut changer le résultat final. Comme solution à ce problème, nous adoptons le protocole suivant : les arêtes incidentes sont abordées dans l'ordre des degrés des nœuds adjacents dans $\Delta^A_{n_a}$ ou $\Delta^A_{n_a}$. Autrement dit, le premier élément de l'ensemble des attributs des arêtes incidentes dans la signature $\gamma(n_a)$ correspond aux attributs de l'arête dont l'extrémité (autre que n_a) est le nœud ayant le plus grand degré par rapport aux nœuds adjacents à n_a. Ce problème peut être considéré aussi comme une instance du problème d'affection entre les multi-ensembles d'attributs des arêtes incidentes ou

entre les multi-ensembles de degrés de nœuds adjacents. Nous n'avons pas adopté cette solution parce qu'elle est moins rapide que la nôtre. Dans la suite, nous définissons chaque distance séparément.

Distance d'attributs des nœuds *Soient $\alpha^A_{n_a}$, et $\alpha^B_{n_b}$ les attributs respectifs des deux nœuds $n_a \in A$ et $n_b \in B$, la distance $d_\alpha(\alpha^A_{n_a}, \alpha^B_{n_b})$ est définie par :*

$$d_\alpha(\alpha^A_{n_a}, \alpha^B_{n_b}) \quad = \quad HEOM(\alpha^A_{n_a}, \alpha^B_{n_b})$$

Distance de degré des nœuds *Soient $\theta^A_{n_a}$ et $\theta^B_{n_b}$ les degrés respectifs des deux nœuds $n_a \in A$ et $n_b \in B$, la distance $d_\theta(\theta^A_{n_a}, \theta^B_{n_b})$ est :*

$$d_\theta(\theta^A_{n_a}, \theta^B_{n_b}) \quad = \quad HEOM(\theta^A_{n_a}, \theta^B_{n_b})$$

Distance de degré des nœuds adjacents *Soient $\Delta^A_{n_a}$ et $\Delta^B_{n_b}$ les ensembles des degrés des nœuds adjacents respectivement aux deux nœuds $n_a \in A$ et $n_b \in B$, la distance $d_\Delta(\Delta^A_{n_a}, \Delta^B_{n_b})$ est :*

$$d_\Delta(\Delta^A_{n_a}, \Delta^B_{n_b}) \quad = \quad HEOM(\Delta^A_{n_a}, \Delta^B_{n_b})$$

Évidemment, n_a et n_b peuvent avoir un nombre différent de nœuds adjacents. Pour rendre les deux ensembles $\Delta^A_{n_a}$ et $\Delta^B_{n_b}$ de même taille, nous ajoutons au plus petit ensemble k= $\Big||\Delta^A_{n_a}\mid - \mid\Delta^B_{n_b}|\Big|$ nœuds adjacents virtuels avec des degrés 0.

Distance d'attributs d'arêtes incidentes *Soient $\Omega^A_{n_a}$ et $\Omega^B_{n_b}$ les ensembles des attributs des arêtes incidentes respectivement aux nœuds $n_a \in A$ et $n_b \in B$, la distance $d_\Omega(\Omega^A_{n_a}, \Omega^B_{n_b})$ est :*

$$d_\Omega(\Omega^A_{n_a}, \Omega^B_{n_b}) \quad = \quad \frac{\sum_{i=1}^{|\Omega^A_{n_a}|} HEOM(\Omega^A_{n_a}(i), \Omega^B_{n_b}(i))}{\mid \Omega^A_{n_a} \mid}$$

avec $\Omega^A_{n_a}(i)$ est le i-ième élément de $\Omega^A_{n_a}$

Comme pour les ensembles de degrés des nœuds adjacents, $\Omega^A_{n_a}$ et $\Omega^B_{n_b}$ n'ont pas forcément toujours la même taille. Comme précédemment, nous ajoutons au plus petit ensemble k arêtes incidentes virtuelles avec des attributs ϵ, avec k= $\Big||\Omega^A_{n_a}\mid - \mid\Omega^B_{n_b}|\Big|$.

Distance de signatures des nœuds *Soient A=(V_a,E_a) et B=(V_b,E_b) deux graphes, avec $n_a \in V_a$ et $n_b \in V_b$. Soient $\gamma(n_a)$ et $\gamma(n_b)$ les signatures respectives de n_a et n_b, avec :*

$$\gamma(n_a) = \Big(\ \alpha^A_{n_a},\ \theta^A_{n_a},\ \Delta^A_{n_a},\ \Omega^A_{n_a}\Big)$$

$$\gamma(n_b) = \Big(\ \alpha^B_{n_b},\ \theta^B_{n_b},\ \Delta^B_{n_b},\ \Omega^B_{n_b}\Big)$$

La distance $d_\gamma(\gamma(n_a), \gamma(n_b))$ entre les deux signatures $\gamma(n_a)$ et $\gamma(n_b)$ est :

$$d_\gamma(\gamma(n_a), \gamma(n_b)) = d_\alpha + d_\theta + d_\Delta + d_\Omega \tag{4.3}$$

4.4 Approximation de la distance d'édition de graphes

En se basant sur la signature de nœuds, nous introduisons une approximation de la distance d'édition de graphes. Cette approximation est établie par la réduction du problème de la distance d'édition de graphes en un problème d'affectation. Nous montrons aussi que la définition originale de la distance d'édition de graphes peut être établie en fonction de la solution optimale du problème d'affectation. Bien évidemment, vu que notre approximation calcule uniquement une solution sous-optimale de la distance d'édition de graphes, elle ne satisfait pas toujours les propriétés d'une métrique.

Nous proposons un algorithme qui utilise les signatures de nœuds pour définir un problème d'affection. Nous commençons par définir la distance entre deux ensembles de signatures de nœuds.

Distance d'ensembles de signatures des nœuds *Soient A=(V_a,E_a) et B=(V_b,E_b) deux graphes, étant donné la bijection $\phi : S_\gamma(A) \rightarrow S_\gamma(B)$. La distance φ entre S_γ(A) et S_γ(B) est donnée par :*

$$\varphi(S_\gamma(A), S_\gamma(B)) = \min_{\phi} \sum_{\gamma(n_i) \in S_\gamma(A)} d_\gamma(\gamma(n_i), \phi(\gamma(n_i)))$$

Étant donné que ϕ est une bijection, les deux multi-ensembles $S_\gamma(A)$ et $S_\gamma(B)$ doivent contenir le même nombre d'éléments. Pour contourner ce problème, nous ramenons les deux ensembles des signatures $S_\gamma(A)$ et $S_\gamma(B)$ à une même taille. Pour ce faire, nous ajoutons au plus petit ensemble k signatures vides (i.e. $\gamma(\epsilon) = (\emptyset, 0, \emptyset, \emptyset)$), avec $k = \Big||S_\gamma(A)| - |S_\gamma(B)|\Big|$.

Le calcul de la fonction $\varphi(S_\gamma(A), S_\gamma(B))$ est équivalent à la résolution d'un problème d'affection qui est l'un des problèmes classiques de la recherche opérationnelle. Il consiste à trouver un couplage maximum (ou minimum) dans graphe biparti. Étant donné un graphe biparti complet g = (X $\cup$ Y,X $\times$ Y) où à chaque arête xy un poids w(xy) est associé. Le problème d'affectation consiste à chercher l'appariement M de X vers Y ayant le poids minimal. Prenons un exemple du problème d'affectation. Soient trois ouvriers O_1, O_2 et O_3 et trois tâches T_1, T_2 et T_3. Pour qu'un produit soit prêt il faut que les trois tâches soient accomplies et chaque ouvrier se voit assigner une tâche. Le temps de mise en œuvre est donné par la matrice de coût dans la table 4.2. L'objectif est de minimiser le temps total de mise en œuvre. Ce problème peut être modélisé par un graphe biparti g = (X $\cup$ Y,X $\times$ Y), avec X=$\{O_1, O_2, O_3\}$ et Y=$\{T_1, T_2, T_3\}$ (voir figure 4.3).

TABLE 4.2 – Une matrice de coûts d'un problème d'affectation

	O_1	O_2	O_3
T_1	15	12	8
T_2	24	36	11
T_3	23	20	25

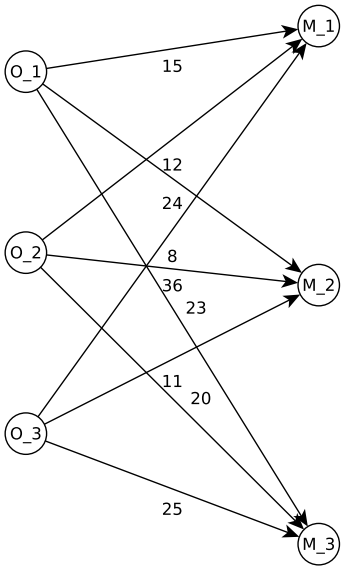


FIGURE 4.3 – Un graphe biparti d'un problème d'affectation

Dans notre cas, étant donné deux graphes A=(V_a,E_a) et B=(V_b,E_b), le problème d'affectation est défini par un graphe biparti g = (X ∪ Y,X × Y), avec X=$S_\gamma(A)$ et Y=$S_\gamma(B)$. Les poids des arêtes xy ∈ $X \times Y$ sont les distances entre les signatures de nœuds. Ainsi, une matrice de coûts est construite où chaque élément ij représente la distance entre la i-ème signature dans $S_\gamma(A)$ et la j-ième signature dans $S_\gamma(B)$. Pour résoudre ce problème d'affection, nous utilisons la méthode hongroise [146] (voir Annexe A) qui est connue comme l'une des méthodes les plus efficaces et les plus rapides dans la littérature [39, 181], avec une complexité en temps de O(n^3).

Une fois la méthode hongroise appliquée sur la matrice de coûts, nous obtenons une matrice de permutation P qui définit l'appariement optimal entre les deux graphes A et B. En se basant sur la matrice P nous définissons la distance φ et nous ré-écrivons la distance entre les multi-ensembles de signatures comme suit :

1. Compléter les multi-ensembles avec des signatures vides afin qu'ils aient la même cardinalité
2. Calculer la matrice carrée d telle que $d[i][j]$ = distance entre les signatures $i \in S_\gamma(A)$ et $j \in S_\gamma(A)$
3. Appliquer la méthode hongroise pour calculer l'appariement bijectif $\phi' : S_\gamma(A) \to S_\gamma(B)$ tel que $\sum_{i \in S_\gamma(A)} d[i][\phi'(i)]$ soit minimal.
4. Retourner ϕ'

Ainsi, nous définissons la distance entre deux multi-ensembles de signatures de nœuds comme suit :

$$\varphi(S_\gamma(A), S_\gamma(B)) = \sum_{\gamma(n_i) \in S_\gamma(A)} d_\gamma(\gamma(n_i), \phi'(\gamma(n_i))) \tag{4.4}$$

Étant donné que nous avons ajouté des signatures vides (i.e. $\gamma(\epsilon)$) au plus petit ensemble de $\mathrm{S}_\gamma(A)$ et $\mathrm{S}_\gamma(B)$, nous pouvons ainsi développer l'équation (4.4) comme suit :

$$\begin{aligned}\varphi(S_\gamma(A), S_\gamma(B)) \quad = \quad & \sum_{\gamma(n_i)\in S_\gamma(A)/\gamma(\epsilon),\phi'(\gamma(n_i))\neq\gamma(\epsilon)} d_\gamma(\gamma(n_i), \phi'(\gamma(n_i))) \\ & + \sum_{\gamma(\epsilon)\in S_\gamma(A)} d_\gamma(\gamma(\epsilon), \phi'(\gamma(\epsilon))) \\ & + \sum_{\gamma(\epsilon)\in S_\gamma(B)} d_\gamma(\phi'^{-1}(\gamma(\epsilon)), \gamma(\epsilon))\end{aligned} \tag{4.5}$$

avec $d_\gamma(\gamma(n_i), \phi'(\gamma(n_i)))$ la distance entre les signatures de deux nœuds de deux graphes A et B ; $d_\gamma(\gamma(\epsilon), \phi'(\gamma(\epsilon)))$ correspond à la distance entre la signature d'un nœud du graphe B appariée à une signature vide dans A ; $d_\gamma(\phi'^{-1}(\gamma(\epsilon)), \gamma(\epsilon))$ correspond à la distance entre la signature d'un nœud du graphe A appariée à une signature vide dans B.

Nous considérons que chacune de ces distances correspond à un coût d'édition d'opération effectuée sur les nœuds pour transformer le graphe A en B. Ainsi, $d_\gamma(\gamma(n_i), \phi'(\gamma(n_i)))$ correspond au coût de substitution du nœud $n_i \in A$ par le nœud du graphe B ayant la signature $\phi'(\gamma(n_i))$. De plus, $d_\gamma(\gamma(\epsilon), \phi'(\gamma(\epsilon)))$ correspond au coût d'insertion d'un nœud du graphe B, ayant la signature $\phi'(\gamma(\epsilon))$, dans A. Enfin, $d_\gamma(\phi'^{-1}(\gamma(\epsilon)), \gamma(\epsilon))$ correspond au coût de suppression du nœud, dont la signature est appariée à une signature vide, de A.

Ces considérations définissent, implicitement, un appariement sur les nœuds en se basant sur l'appariement optimal sur les signatures de nœuds. Il faut noter ici que même si notre appariement sur les signatures est optimal, nous ne pouvons pas affirmer que l'appariement résultant sur les nœuds est toujours optimal.

Finalement, en s'appuyant sur l'équation (4.1) qui calcule une distance d'édition de graphes à partir d'un appariement de nœuds, nous définissons, dans l'équation 4.6, notre approximation de la distance d'édition entre deux graphes A et B ayant, respectivement, les deux multi-ensembles des signatures des nœuds $\mathrm{S}_\gamma(A)$ et $\mathrm{S}_\gamma(B)$.

$$d(A, B) = \varphi(S_\gamma(A), S_\gamma(B)) + \sum_{e\in E_r} \zeta_{er}(e) + \sum_{e\in E_i} \zeta_{ei}(e) + \sum_{e\in E_s} \zeta_{es}(e) \tag{4.6}$$

Cette équation (4.6) est notre approximation de la distance d'édition entre les graphes. À notre connaissance, la grande majorité des algorithmes de distance d'édition de graphes requièrent des fonctions de coûts prédéfinies. Notre approche ne nécessite aucun paramètre et calcule les fonctions de coûts en faisant usage de la signature de nœud. En outre, notre approche réduit la distance d'édition de graphes en un problème d'affectation par le biais de signatures de nœuds.

4.5 Résultats expérimentaux

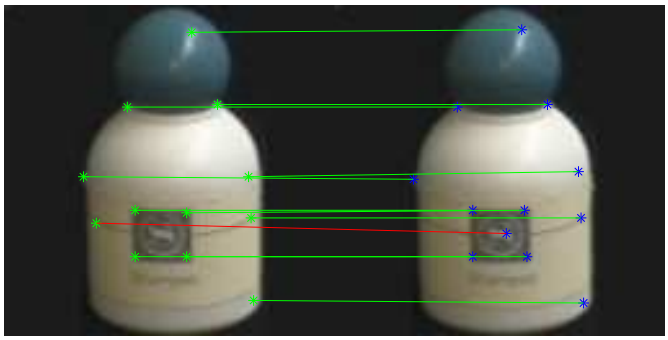
Dans cette section nous fournissons quelques résultats expérimentaux de notre méthode d'appariement de graphes. Cette étude expérimentale est composée de deux parties. La première consiste à évaluer la qualité de notre méthode par rapport à la correspondance nœud-à-nœud. Cette évaluation fournit une comparaison avec deux méthodes connues dans la littérature. Dans la deuxième partie de notre étude expérimentale, nous évaluons notre approximation de la distance d'édition de graphes par la classification de graphes en comparant nos résultats avec ceux de quelques techniques d'appariement de graphes connues.

4.5.1 Appariement nœud-à-nœud

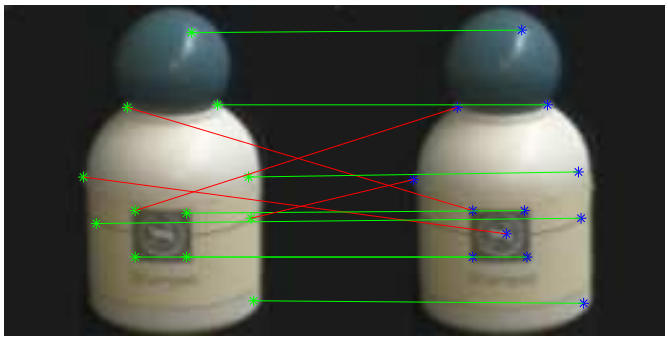
Dans cette partie, nous fournissons une comparaison entre notre algorithme, la méthode de Umeyama [258], la méthode probabiliste de Zass [284] et l'algorithme PATH [283]. Nous avons choisi ces trois méthodes parce qu'elles fournissent une correspondance explicite entre les nœuds de deux graphes, ce qui n'est le cas de toutes les techniques d'appariement de graphes.

- La méthode de Umeyama [258] (voir chapitre 3) utilise la décomposition en valeurs propres des matrices d'adjacences de chaque graphe. Cette méthode introduit la notion de matrice d'appariement qui décrit l'appariement nœud-à-nœud entre deux graphes. Cette approche nécessite que les graphes comparés soient de la même taille.
- La deuxième méthode utilisée dans notre évaluation est la méthode probabiliste de Zass et al. [284]. Dans cette méthode, les auteurs résolvent l'appariement de graphes dans un cadre purement probabiliste. Cette méthode ne gère que les graphes pondérés. Elle est basée sur l'optimisation convexe et les relations algébriques entre les arêtes des graphes. Ces relations sont déduites à partir des matrices d'adjacence de graphes. La solution optimale d'appariement est calculée via un algorithme itératif de projections successives [286].
- La troisième méthode utilisée dans notre évaluation est l'algorithme de PATH [283]. Cet algorithme résout le problème d'appariement de graphes dans un cadre spectral. Il s'appuie sur une technique basée sur l'optimisation convexe et concave pour déterminer la matrice de permutation optimale entre les nœuds de deux graphes. Cette algorithme gère les graphes pondérés de tailles différentes.

Pour établir une comparaison avec l'algorithme de Umeyama, nous sommes dans l'obligation d'utiliser des graphes ayant la même taille. Pour ce faire, nous avons sélectionné un sous-ensemble de la base d'images COIL-100 [174]. La base d'images COIL-100 contient différentes vues d'objets 3D. Les graphes sont extraits par le biais de la triangulation des points d'intérêt de Harris détectés dans les images. Par ailleurs, pour que la correspondance nœud-à-nœud soit significative, les images appariées doivent appartenir à la même classe. Par conséquent, nous avons sélectionné 23 images d'une même classe de COIL-100. Les 23 images sélectionnées ont le même nombre de points d'intérêt, i.e. les graphes ont la même taille. La figure 4.4 (images du $50^{\text{ème}}$ objet avec des rotations de 320^{o} et de 325^{o}) illustre les correspondances (les lignes rouges et vertes) entre les points d'intérêt de chaque image. Les lignes rouges correspondent aux faux appariements et les lignes vertes correspondent aux appariements corrects. Les résultats de la



(a) Les correspondances avec notre algorithme



(b) Les correspondances avec l'algorithme de Umeyama

FIGURE 4.4 – Correspondances entre points d'intérêt de Harris (1) (les lignes vertes correspondent aux correspondances correctes et les lignes rouges aux fausses).

TABLE 4.3 – Comparaison de trois algorithmes d'appariement (TCC : taux de correspondance correctes).

Algorithme	Correspondances correctes	Correspondances fausses	TCC
Umeyama	2120	916	69.83%
Zass	2222	814	73.19%
PATH	2544	492	**83.79%**
Notre méthode	2525	511	83.17%

comparaison sont résumés dans la table 4.3. Ces résultats montrent que notre méthode fournit un taux de correspondances correctes (TCC) très similaire à l'algorithme de PATH et supérieur aux algorithmes de Umeyama et de Zass. Concrètement, le TCC de notre méthode a une différence de +13.34% par rapport au TCC de Umeyama, de $\approx$ +10% par rapport au TCC de Zass, et de -0.62% par rapport au TCC de PATH. Outre ces résultats, notre méthode est plus flexible que celle de Umeyama, au sens où elle gère des graphes de différentes tailles. En effet, une taille fixe de graphes pose un problème dans les applications réelles de reconnaissance de formes. Dans notre expérimentation, la contrainte de taille nous a obligé à considérer uniquement un sous-ensemble réduit d'une classe de la base COIL-100.

Étant donné que les méthodes de Zass et PATH gèrent aussi les graphes de différentes tailles, nous fournissons, dans ce qui suit, une nouvelle évaluation plus poussée et approfondie. Nous comparons les performances de notre méthode et celles des méthodes de Zass et PATH dans le cadre de l'appariement nœud-à-nœud. Cette évaluation consiste à utiliser deux bases de données avec des graphes de tailles différentes. La première base de données est la base d'images CMU/-VASC (*model-house sequence*) qui contient 9 images d'une maison en trois dimensions avec des différents angles de vue de la caméra. La deuxième base de données est la totalité de la base d'images COIL-100, qui contient 7200 images (100 classes et 72 images par classe). Rappelons que ces bases d'images sont utilisées, dans la littérature, pour l'évaluation de plusieurs approximations de distance d'édition de graphes, telle que la méthode proposée dans [205]. Dans les deux bases d'images, les graphes sont obtenus par les triangulations de Delaunay des points d'intérêt de Harris [108], comme dans [205].

Pour COIL-100, l'expérimentation consiste à appliquer notre méthode et les méthodes de Zass et PATH pour faire correspondre les nœuds d'une paire de graphes représentant une paire d'images. Chaque paire d'images appariées correspond à deux objets de même classe avec une différence de rotation de $\pm 5°$, par exemple l'image d'un objet ayant une rotation de 20° est appariée aux images du même objet avec des rotations de 15° et de 25°. Ce choix d'images vient du fait que les différentes vues d'un objet n'ont pas toujours des apparences similaires. Ceci s'accentue notamment lorsque les vues ont une grande différence de rotation. Dans la figure 4.5, nous illustrons un exemple de deux différentes vues d'un même objet. Les deux images de cette figure n'ont pas une apparence similaire et donc l'appariement entre ces deux images n'est pas significatif. C'est pour cela que nous ne considérons que les paires d'images où l'appariement est significatif. L'évaluation des méthodes d'appariement sur cette base implique le calcul du taux

(a) Rotation de 0° (b) Rotation de 240°

FIGURE 4.5 – Deux différentes vues d'un même objet de la base COIL-100

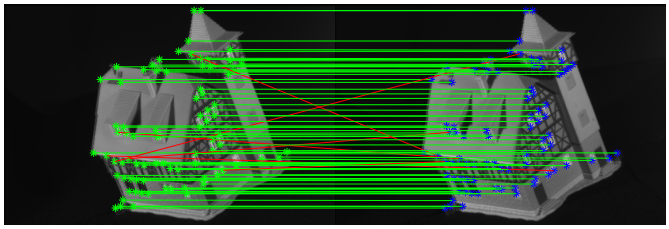
TABLE 4.4 – Comparaison de deux algorithmes d'appariement (TCC : taux de correspondances correctes). -base d'images de COIL-100 -

Algorithme	Correspondances correctes	Correspondances fausses	TCC
Zass	29732	46772	38.86%
PATH	36673	39831	**47.94%**
Notre méthode	33953	42551	44.38%

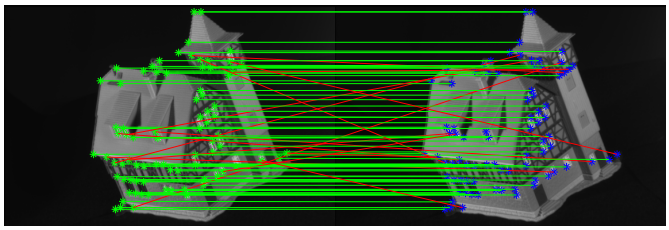
de correspondances correctes globales qui correspond à la moyenne de tous les TCC calculés pour chaque paire d'images appariées.

Les images de la base CMU/VASC sont toutes semblables et composent une classe unique. Par conséquent, le TCC de cette base est la moyenne de tous les TCC obtenus pour chaque paire de graphes en considérant toutes les combinaisons possibles. La figure 4.6 illustre un exemple de résultats obtenus en appariant deux images de la base CMU/VASC à l'aide de notre méthode et les méthodes de Zass et PATH.

Les résultats de cette expérimentation sont présentés dans les tables 4.4 et 4.5. Pour les deux bases d'images, les performances de notre méthode se positionnent en deuxième position suivies par celles de la méthode de Zass. Plus généralement, nous constatons que nos résultats sont très compétitifs par rapport aux méthodes testées dans cette évaluation. Ceci montre que les signatures des nœuds, qui nous utilisons, sont appropriées aussi bien pour les graphes de même tailles que pour les graphes de tailles différentes. De plus, notre méthode gère les graphes étiquetés (i.e. étiquettes symboliques), ce qui n'est pas, à notre connaissance, possible avec les méthodes de Zass et PATH.



(a) Les correspondances avec notre algorithme



(b) Les correspondances avec l'algorithme de Zass

FIGURE 4.6 – Correspondances entre points d'intérêt de Harris (2) (les lignes vertes correspondent aux correspondances correctes et les lignes rouges aux fausses).

TABLE 4.5 – Comparaison de deux algorithmes d'appariement (TCC : taux de correspondances correctes). -base d'images de CMU/VASC-

Algorithme	Correspondances correctes	Correspondances fausses	TCC
Zass	1474	1162	55.92%
PATH	1783	853	**67.64**%
Notre méthode	1658	978	62.90%

4.5.2 Classification de graphes

Protocole d'évaluation

Dans la littérature, certaines méthodes d'appariement de graphes ne gèrent pas les graphes étiquetés (voir Chapitre 3). Souvent, ces méthodes sont appelées des méthodes *purement structurelles* [160, 205]. Cette appellation symbolise le fait que ces méthodes s'intéressent uniquement à l'aspect structurel et topologique pour calculer les distances entre les graphes. Ainsi, les étiquettes des nœuds et des arêtes ne sont nullement utilisées pour calculer la distance de graphes. Outre ces méthodes, il existe des méthodes qui intègrent les étiquettes dans le calcul des distances entre les graphes. Dans la littérature, ces deux types de méthodes ont été utilisés en reconnaissance de formes.

Ainsi, dans un contexte de classification de graphes, nous comparons d'abord notre méthode avec des méthodes purement structurelles. Ensuite, nous fournissons une comparaison entre notre méthode et un ensemble de méthodes qui gèrent les graphes étiquetés. Néanmoins, les méthodes d'appariement, qui gèrent des graphes étiquetés, gèrent également les graphes non-étiquetés par le fait que ces graphes sont des cas particuliers de graphes étiquetés. Dans cette perspective, Riesen et al. [201] proposent d'attribuer la même étiquette *l* à tous les nœuds et à toutes les arêtes pour produire un graphe étiqueté à partir d'un graphe non étiqueté. Ainsi, nous ne pouvons appliquer sur les graphes étiquetés que les méthodes qui gèrent les étiquettes. Par contre, toutes les méthodes peuvent être appliquées aux graphes non-étiquetés.

Avant de procéder aux expérimentations, nous présentons brièvement les méthodes sélectionnées pour la comparaison. Le choix de ces méthodes est principalement basé sur leurs bons résultats obtenus avec des bases de données bien connues dans la littérature.

– L'ensemble des méthodes purement structurelles, utilisées dans notre évaluation, est composé de :

GEDSS : distance d'édition de graphes par sérialisation spectrale. Robles-Kelly et al. [205] proposent une méthode spectrale pour représenter les graphes par des chaînes de caractères, puis la similarité de graphes est mesurée en fonction de la distance d'édition de chaînes de caractères dans un cadre probabiliste. La distance d'édition de graphes est le coût du plus court chemin dans une matrice d'édition qui transforme un graphe en un autre. Les lignes et les colonnes de la matrice d'édition sont indexées par deux chaînes : Y={y_1,y_2,...,$y_{|V_D|}$} pour le premier graphe G_D=(V_D,E_D) et X={x_1,x_2,...,$x_{|V_M|}$} pour le deuxième G_M=(V_M,E_M). Le calcul de la distance d'édition de graphes consiste à trouver le chemin le moins coûteux $\Gamma^* = \langle \gamma_1, \gamma_2, ..., \gamma_k, ..., \gamma_L \rangle$ de (y_1,x_1) à ($y_{|V_D|}$,$x_{|V_M|}$) à travers la matrice d'édition en se basant sur la distance de Levenshtein. Chaque état $\gamma_k \in (V_D \cup \varepsilon) \times (V_M \cup \varepsilon)$ du chemin d'édition est une paire cartésienne. Le coût d'édition du chemin d'édition le moins coûteux est calculé avec l'équation suivante :

$$d(X,Y) = C(\Gamma^*) = \sum_{\gamma_k \in \Gamma} \eta(\gamma_k \rightarrow \gamma_{k+1})$$

où $\eta(\gamma_k \rightarrow \gamma_{k+1}) = -(lnP(\gamma_k|\phi_X^*(x_i), \phi_Y^*(y_j)) + lnP(\gamma_{k+1}|\phi_X^*(x_{i+1}), \phi_Y^*(y_{j+1})) + lnR_{k,k+1})$, et le coefficient de la compatibilité des arêtes $R_{k,k+1}$ est

$\mathrm{R}_{k,k+1} = \frac{P(\gamma_k|\gamma_{k+1})}{P(\gamma_k)P(\gamma_{k+1})} =$

$$\begin{cases} \rho_M\rho_D \\ \quad \text{si } \gamma_k \to \gamma_{k+1} \text{ est une transition diagonale} \\ \rho_M \\ \quad \text{si } \gamma_k \to \gamma_{k+1} \text{ est une transition verticale} \\ \rho_D \\ \quad \text{si } \gamma_k \to \gamma_{k+1} \text{ est une transition horizontal} \\ 1 \\ \quad \text{si } y_j = \varepsilon \text{ ou } x_i = \varepsilon \text{ et } y_{j+1} = \varepsilon \text{ ou } x_{i+1} = \varepsilon \end{cases}$$

$P(\gamma_k|\phi_X^*(x_i), \phi_Y^*(y_j)) =$

$$\begin{cases} \frac{1}{\sqrt{2\pi}\sigma} exp\{-\frac{1}{2\sigma^2}(\phi_X^*(x_i) - \phi_Y^*(y_j))^2\} \\ \quad \text{si } y_j \neq \varepsilon \text{ et } x_i \neq \varepsilon \\ \alpha \\ \quad \text{si } y_j = \varepsilon \text{ et } x_i \neq \varepsilon \end{cases}$$

avec ρ_M et ρ_D sont les densités des arêtes respectives des graphes G_M et G_D ($\rho_M = \frac{|V_M|^2}{E_M}$) et ϕ_X^* et ϕ_Y^* sont, respectivement, le vecteur propre principal des matrices d'adjacences de G_M et G_D.

GH : distance d'édition de graphes par l'histogramme de graphe. Papadopoulos et Manolopoulos [183] présentent une mesure de similarité pour les graphes qui est aussi basée sur le concept d'opérations d'édition. Ils proposent trois opérations primitives différentes : l'insertion de nœuds, la suppression de nœuds et la mise à jour de nœuds. Alors que les opérations de suppression et d'insertion ont des significations évidentes, l'opération de mise à jour est nécessaire pour insérer ou supprimer les arcs incidents pour un nœud. De plus, Papadopoulos et Manolopoulos introduisent la séquence de degrés d'un graphe qui est la donnée en ordre décroissante des degrés de chaque nœud du graphe. La distance de similarité entre deux graphes est définie comme le nombre minimum d'opérations primitives nécessaires pour que les deux graphes possèdent la même séquence de degrés. Pour calculer la mesure de similarité, un histogramme trié de graphe est introduit dans [183]. Soit un graphe $G = (V, E)$, son histogramme est construit par les degrés des nœuds de G de sorte que chaque nœud correspond à une case différente de l'histogramme. En triant ces cases, on obtient un histogramme trié de graphe G (*sorted graph histogram*). Les auteurs utilisent comme mesure de similarité de graphes, la distance de Manhattan entre les histogrammes triés de graphes correspondants. Évidement, si les graphes dans une base de données n'ont pas la même taille, les histogrammes triés de ces graphes ont des dimensions différentes. Pour permettre l'utilisation des structures d'indexation des espaces de vecteurs, une technique de pliage pour les histogrammes a été introduite pour atteindre une dimension constante des histogrammes.

GP : sondage de graphes. Lopresti et al. introduisent dans [157] le paradigme de sondage de graphes. Cette technique consiste à déterminer certaines informations particulières sous forme de vecteurs à l'aide d'une sonde dans les graphes. La mesure de similarité entre deux graphes correspond à la distance de Manhattan entre les deux vecteurs correspondants. La construction de vecteurs à partir de graphes consiste à répondre à la question suivante : *"Combien de nœuds avec un degré n sont présents dans le graphe G = (V, E) ?"*. Concrètement, soit G = (V, E) un graphe non orienté, le vecteur associé à G est le suivant : PR(G)≡(n_0,n_1,n_2, ...) avec n_i=|{ *v* in V | deg(*v*)=*i*}|. Soient A et B deux graphes, et PR_A et PR_B les vecteurs de sondage respectifs de A et B. Selon l'approche de Lopresti et al., la distance d'édition entre A et B correspond à la distance de Manhattan entre PR_A et PR_B, i.e. $L_1(PR_A,PR_B)$.

– Les méthodes qui gèrent les graphes étiquetés, utilisées dans notre évaluation sont :

Astar : distance d'édition de graphes par l'algorithme A$*$. Neuhaus et al. ont proposé [178] une technique basée sur la décomposition de graphes et sur l'algorithme A*-*beamsearch*. Cette méthode a été détaillée dans le chapitre précédent.

BGMEDG : distance d'édition de graphe par le couplage du graphe biparti Riesen et al. [201] considèrent la distance d'édition de graphes comme une instance d'un problème d'affectation. Cette méthode est une approximation sous-optimale de la distance d'édition. La distance entre deux graphes est calculée à l'aide de la méthode hongroise [146] appliquée à une modélisation du problème d'affection. Soient $A = (V_a, E_a)$ et $B = (V_b, E_b)$ deux graphes, les auteurs formulent le problème d'affectation par une matrice de taille $\mid V_a \mid + \mid V_b \mid \times \mid V_a \mid + \mid V_b \mid$. Dans cette matrice, on peut observer quatre parties : la première représente les coûts de toutes les substitutions possibles de nœuds ($\mid V_b \mid \times \mid V_a \mid$). Ces coûts correspondent à la somme de deux valeurs : la première valeur est la distance euclidienne ou la distance d'édition de chaînes de caractères (en fonction du type de l'étiquette en présence) entre les étiquettes de deux nœuds, la deuxième valeur est le coût d'affectation minimal entre les ensemble des arêtes incidentes aux nœuds considérés. La deuxième partie représente une matrice diagonale de coûts de toutes les suppressions possible de nœuds ($\mid V_b \mid \times \mid V_a \mid$). Ces coûts sont considérés comme des valeurs constantes qui sont des paramètres de l'algorithme. La troisième partie de la matrice diagonale représente les coûts de toutes les insertions possibles de nœuds($\mid V_b \mid \times \mid V_a \mid$) et ces coûts sont également des paramètres de l'algorithme et considérés comme des valeurs constantes. Enfin, la dernière partie correspond à des zéros ($\epsilon \rightarrow \epsilon$). Finalement, les auteurs appliquent la méthode hongroise à cette matrice pour définir la séquence d'édition optimale entre A et B.

Notons que, pour les deux méthodes Astar et BGMEDG, le calcul des fonctions de coût d'édition optimales est un problème en soi qui dépend des bases de graphes utilisées. Pour calculer ces coûts d'édition, nous avons procédé de la même manière que dans [175, 201]. Nous extrayons un sous-ensemble, appelé ensemble de validation, de chaque base de graphes. Puis, nous déterminons les coûts d'édition qui sont optimaux dans les jeux de validation. Nous utilisons les mêmes fonctions de coût d'édition pour les deux algorithmes. Dans la table 4.6, nous fournissons les coûts calculés pour chaque base de données utilisée dans cette évaluation.

Vu qu'une partie de cette expérimentation consiste à comparer notre méthode avec des méthodes purement structurelles, nous utilisons, pour cette partie, des bases de graphes non-étiquetés. Par ailleurs, les comparaisons avec les méthodes qui gèrent les graphes étiquetés sont étudiés aussi bien sur des bases de graphes étiquetés que sur les bases de graphes non-étiquetés. Dans ce qui suit, nous décrivons les bases de graphes utilisées dans notre expérimentation.

TABLE 4.6 – Les coûts d'opérations d'édition des méthodes Astar et BGMEDG

	GREC1	Lettrine	Shape	Logo	Mutagenicity	Letter	GREC2
Coût de nœud	1.6	0.3	0.9	0.3	1.1	1	13
Coût d'arête	0.7	0.1	0.5	0.1	0.1	0.5	11

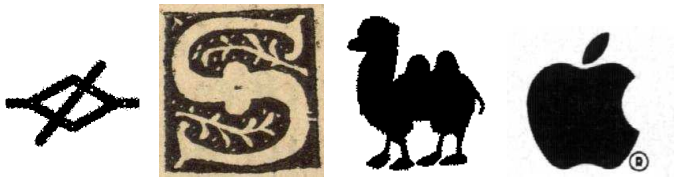
Bases de graphes non-étiquetés

GREC1 : La base GREC [68] (voir figure 4.7(a)) est constituée de graphes qui représentent des symboles extraits des plans architecturaux et électroniques. Les graphes sont définis comme suit : les point finaux (i.e. coins, intersections et cercles) sont les nœuds du graphe, les arêtes correspondent aux lignes qui connectent les points dans les symboles. L'ensemble utilisé dans notre expérimentation est constitué de 528 graphes, 24 classes et 22 graphes par classe.

Lettrines : La base de lettrines (voir figure 4.7(b)) contient des lettrines extraites de documents anciens numérisés [10]. Chaque lettrine contient une texture, un arrière plan décoré et une initiale. Tous ces composants sont sémantiquement intéressants, pour cela l'utilisation des points d'intérêt nous semble non adéquate pour extraire les graphes à partir des lettrines. Nous avons alors choisi d'utiliser les graphes d'adjacence de régions pour représenter les lettrines. Pour ce faire, chaque lettrine est segmentée, tout d'abord, par un algorithme de segmentation en régions [80]. Ensuite, chaque lettrine est représentée par un graphe dont les nœuds correspondent aux régions et les arêtes aux relations d'adjacence de régions. L'ensemble utilisé dans notre expérimentation est constitué de 280 graphes, 4 classes et 70 graphes par classe.

Shape : Cette base consiste à un ensemble d'images de silhouettes. Elle est fournie par le laboratoire LEMS de l'université Brown [229] (voir figure 4.7(c)). Pour la représentation sous forme de graphes de cette base, nous avons utilisé pour chaque méthode d'appariement, la représentation qui a fourni les meilleurs résultats dans le chapitre 2. Ainsi, les graphes de squelettes sont utilisés pour notre méthode et les méthodes suivantes : Robles-Kelly, Riesen et Neuhaus. Les graphes de points d'intérêt (Harris et triangulation de Delaunay) sont utilisés pour les méthodes suivantes : Lopresti et Papadopoulos. La base de graphes utilisée dispose de 216 graphes, 18 classes et 12 graphes par classe.

10. Fournit par le CESR - Université de Tours dans le contexte du projet ANR Navidomass http ://l3iexp.univ-lr.fr/navidomass/



(a) GREC1 (b) Lettrine (c) Shape (d) Logo

FIGURE 4.7 – Échantillons de chaque base d'images

Logo : Cette base [67] (voir figure 4.7(d)) est constituée de graphes qui représentent des images binaires d'un ensemble de logos de marques déposées. Les graphes d'adjacence de régions ont fourni les meilleurs résulats pour cette base dans le chapitre 2. Ainsi, nous utilisons cette représentation (graphes d'adjacence de régions) sous forme de graphes pour cette base. L'ensemble utilisé dans notre expérimentation est constitué de 80 graphes, 10 classes et 8 graphes par classe.

La table 4.7 résume les méthodes de représentation sous forme de graphe utilisées dans nos expérimentations.

TABLE 4.7 – Synthèse des méthodes de représentations sous forme de graphe pour chaque base

	descriptions
GREC1	
Noeud	Point finaux
Arête	Point finaux connexion
Lettrine	
Noeud	Régions
Arête	Relation d'adjacence
Shape	
Noeud	Point d'intérêt de Harris, et Points de terminaison et de jonctions du squelette
Arête	Triangulation de Delaunay, et Branches du squelette
Logo	
Noeud	Régions
Arête	Relation d'adjacence

Bases de graphes étiquetés : Nous utilisons les trois bases de graphes étiquetés suivantes fournit par l'IAM [11] [199] :

Mutagenicity : Cette base se compose de 4337 graphes (2 classes) qui représentent des composés moléculaires, les nœuds représentent les atomes étiquetés du symbole chimique correspondant et les arêtes étiquetées par les valences d'éléments chimiques.

11. *Institute of Computer Science and Applied Mathematics*

Letter : La base Letter comporte des graphes (3000 graphes, 15 classes) qui représentent des dessins déformés de quinze lettres latines. Chaque lettre déformée correspond à un graphe où les arêtes sont les lignes et les nœuds sont les point finaux de ces lignes. Chaque nœud est étiqueté par un attribut à deux dimensions correspondant à sa position (coordonnées).

GREC2 : Cette base contient 1100 graphes divisés en 24 classes. Comme GREC1, les graphes sont les représentations des symboles extraits des plans architecturaux et électroniques. Toutefois, les graphes de GREC2 sont étiquetés : les nœuds par deux attributs numériques pour les coordonnées du point final et un attribut symbolique pour le type de l'intersection. Les arêtes sont étiquetées par trois attributs : deux numériques pour l'angle d'inclinaison de la ligne et la fréquence, et un attribut symbolique pour le type (ligne, cercle ...).

Dans la table 4.8, nous résumons les types d'étiquettes (symboliques ou numériques) associés aux arêtes et aux nœuds de graphes dans les bases utilisées. Par exemple, les graphes de la base GREC2 sont composés par des arêtes et des nœuds avec des étiquettes symboliques et numériques.

TABLE 4.8 – Synthèse de type des étiquettes dans les bases utilisées (A : Arête, N : Noeud)

	Mutagenicity	Letter	GREC2
Symbolique	N		A,N
Numerique	A	N	A,N

Résultats

L'objectif de cette expérimentation est d'approfondir l'évaluation de notre approximation de la distance d'édition de graphes. Pour ce faire, nous procédons à une tâche de classification avec l'algorithme des k plus proches voisins (k-ppv). Ce choix vient du fait que cet algorithme peut être appliqué directement dans le domaine des graphes en utilisant une distance de graphes sans aucune éventuelle adaptation. Dans notre expérimentation, le nombre des plus proches voisins est fixé à 3.

La première partie de cette expérimentation consiste à appliquer l'algorithme des k-ppv sur toutes les bases (GREC1, Lettrine, Shape, Logo, Mutagenicity, Letter, GREC2). Pour chacune de ces bases, 30% des graphes sont utilisés comme un ensemble d'apprentissage et le reste (70%) comme un ensemble de test. La sélection de ces ensembles est faite d'une manière totalement aléatoire. Vu que l'algorithme des k-ppv est très sensible au choix de l'ensemble d'apprentissage, nous avons répété l'exécution de chaque l'expérimentation 50 fois avec des ensembles d'apprentissage (et de test) différents. La table 4.9 présente les taux de classification sur chaque base en utilisant six méthodes de calcul de distance de graphes. Ces taux sont la moyenne des taux obtenus dans les 50 exécutions. À partir de ces résultats, nous pouvons tirer les remarques suivantes :

– En considérant la moyenne des taux effectués sur les bases de graphes non-étiquetés, notre méthode fournit les meilleurs résultats par rapport aux autres méthodes. Toutefois, nous

avons remarqué que sur ce type de bases, les résultats des méthodes GEDSS, GP et GH sont assez proches des résultats de notre méthode. Les méthodes Astar et BGMEDG donnent des résultats moins bons, notamment sur la base GREC1. Une explication possible de ces résultats pourrait être que ces méthodes (Astar et BGMEDG) n'exploitent pas assez les informations structurelles de graphes. En effet, il n'y a pas assez d'informations qui permettent à ces méthodes de ne pas faire la confusion entre les graphes d'une même classe et ceux de classes différentes. Par contre, notre méthode exploite mieux la structure de chaque graphe via la signature de nœuds (notamment les degrés des nœuds adjacents). Dans le cas de GREC1, les graphes sont structurellement similaires entre eux par rapport aux graphes d'autres bases. Ceci montre que notre méthode basée sur la signature de nœuds est bien appropriée pour les graphes purement structurels.

– En considérant la moyenne des taux effectués sur les bases de graphes étiquetés, les trois méthodes fournissent des résultats similaires de l'ordre de 90%. Ces résultats confirment notre explication précédente. En fait, les méthodes Astar et BGMEDG traitent mieux les graphes étiquetés. Toutefois, la moyenne de taux de classification de notre méthode est légèrement meilleure que les moyennes obtenues par les méthodes Astar et BGMEDG. Cette différence de $\approx$ 2% n'est pas statistiquement signifiante. Ainsi, nous considérons que ces résultats effectuées sur les bases de graphes étiquetés sont similaires. Toutefois, notre méthode est non-paramétrique, tandis que les méthodes Astar et BGMEDG requièrent des coûts d'édition. Ceci peut être un avantage dans plusieurs applications de la reconnaissance de formes.

Par ailleurs, la moyenne globale des taux de classification sur toutes les bases montre que notre méthode effectue une performance de $\approx$ +10% par rapport aux méthodes Astar et BGMEDG. Ceci montre bien la flexibilité de notre méthode par rapport aux types de graphes gérés. En effet, nos résultats sont satisfaisants et de bonne qualité aussi bien pour les graphes non-étiquetés que pour les graphes étiquetés.

La deuxième partie de cette expérimentation consiste à appliquer l'algorithme des k-ppv sur la combinaison des quatre bases de graphes non-étiquetés (GREC1, Lettrine, Shape, Logo). Nous utilisons la configuration utilisée dans la première partie de cette expérimentation. Par cette expérimentation, nous examinons comment les distances d'édition de graphes se comportent avec un ensemble de graphes représentant différents types de données (i.e. symboles graphiques, lettrines, silhouettes et logos). De plus, ce test nous permettra d'évaluer la stabilité et les limites des paramètres (les coûts d'édition) des méthodes BGMEDG et Astar. La table 4.10 présente les taux de classification obtenus par l'algorithme k-ppv avec les différentes méthodes de calcul de distance entre les graphes. Le taux de classification obtenu par notre méthode et la méthode GH sont meilleurs que celui des méthodes alternatives. Il est intéressant de noter que dans ce cas de combinaison des bases, il est difficile de déterminer les paramètres optimaux des méthodes Astar et BGMEDG. En fait, une fonction de coût peut être optimale pour un ensemble de graphes, mais pas pour les autres. Ce qui explique les résultats moins bons obtenus par ces méthodes.

Résumons les propriétés de notre méthode par rapport aux méthodes alternatives testées. La table 4.11 illustre un résumé des propriétés de chaque méthode utilisée. Notre méthode est une approche d'approximation de la distance d'édition de graphes non-paramétrique qui ne nécessite

aucune connaissance *a priori* à propos des fonctions de coûts d'édition, et qui ne se limite pas à des types de graphes. Outre le calcul de la distance d'édition de graphes, notre approche fournit une solution explicite de l'appariement nœud-à-nœud qui peut être utile dans différents domaines [5, 185].

4.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approximation de la distance d'édition de graphes basée sur les signatures des nœuds. Une signature d'un nœud consiste en un ensemble de caractéristiques qui décrivent chaque nœud localement dans le graphe. Ces caractéristiques sont : les attributs du nœud, le degré du nœud, les attributs des arêtes incidentes et les degrés des nœuds adjacents. Nous avons réduit le problème d'appariement de graphes en un problème d'affectation en utilisant une distance appropriée entre les signatures. Ensuite, nous avons utilisé la méthode hongroise pour résoudre ce problème et pour déterminer les correspondances entre les nœuds de différents graphes. Finalement, nous avons réécrit la définition de la distance d'édition de graphes en fonctions des distances entre les signatures appariées.

Dans la partie expérimentale de ce chapitre, nous avons établi un ensemble de tests en comparant notre approche à des méthodes connues dans la littérature. Les résultats ont montré que notre méthode est aussi bien appropriée pour les graphes non-étiquetés que les graphes étiquetés. En se basant sur ces résultats, nous utiliserons notre méthode comme la distance de graphes dans la suite de cette thèse. Plus particulièrement, la deuxième partie de ce manuscrit est consacré à la classification et l'indexation de graphes.

TABLE 4.9 – Résultats de la classification $K - ppv$ (k =3)

	Graphes non-étiquetés					Graphes étiquetés				Moyenne combinée
	GREC1	Lettrine	Shape	Logo	Moyenne	Mutagenicity	Letter	GREC2	Moyenne	
Notre méthode	**81.04**%	**84.18**%	61.29%	93.40%	**79.97**%	89.06%	88.64%	**97.90**%	**91.86**%	**85.91**%
GEDSS - Robles-Kelly [205]	72.01%	82.54%	**67.52**%	82.03%	76.02%	-	-	-	-	-
GP - Lopresti [157]	79.82%	75.58%	35.99%	91.37%	70.69%	-	-	-	-	-
GH - Papadopoulos [183]	80.00%	81.57%	46.22%	**94.67**%	75.61%	-	-	-	-	-
PATH - Zaslavskiy [283]	73.45 %	80.23%	63.40%	81.93%	74.75%	-	-	-	-	-
BGMEDG - Riesen [201]	41.87%	76.88%	47.73%	85.57%	63.01%	88.49%	**93.32**%	89.51%	90.44%	74.76%
Astar - Neuhaus [175]	42.29%	78.36%	46.44%	86.70%	63.44%	**89.96**%	90.80%	89.22%	89.99%	74.82%

TABLE 4.10 – Taux de classification k-ppv sur la combinaison des quatre bases de graphes non-étiquetés (k=3)

	GREC1 + Lettrine + Shape + Logo
Notre méthode	**49.96**%
GEDSS - Robles-Kelly [205]	49.24%
GP - Lopresti [157]	47.86%
GH - Papadopoulos [183]	**49.96**%
BGMEDG - Riesen [201]	30.96%
Astar - Neuhaus [175]	32.05%

TABLE 4.11 – Résumé des méthodes d'appariement de graphes testées

	Gestion des graphes étiquetés	Gestion des graphes non-étiquetés	Appariement nœud-à-nœud explicite	Restrictions
Notre méthode	oui	oui	oui	-
Umeyama [258]	non	oui	oui	Graphes de même taille
Zass [284]	non	oui	oui	-
PATH [283]	non	oui	oui	-
Robles-Kelly [205]	non	oui	non	-
Lopresti [157]	oui	oui	non	-
Papadopoulos [183]	non	oui	non	-
Riesen [201]	oui	oui	non	Nécessite les fonctions de coût
Neuhaus [175]	oui	oui	non	Nécessite les fonctions de coût

Deuxième partie

Classification et Indexation de graphes

Chapitre 5

Classification de graphes

If this is coffee, then please - bring me some tea. But if this is tea, please bring me some coffee.

Abraham Lincoln

Sommaire

5.1 Introduction

La classification est une tâche importante dans la reconnaissance de formes. Elle a pour but d'identifier les classes auxquelles appartiennent des objets à partir de leurs traits descriptifs. Autrement dit, la classification est une procédure dans laquelle les objets similaires sont identifiés et regroupés dans une même classe. Dans la littérature, nous distinguons deux catégories de classificateurs. La première correspond à la classification supervisée qui se démarque par la connaissance *a priori* d'un ensemble d'échantillons d'objets dont on connaît leur appartenance à un groupe. Typiquement, une phase, dite d'apprentissage, est effectuée sur ces connaissances

a priori, appelées aussi base d'apprentissage, afin d'extraire des règles d'affectation. En se basant sur ces règles, l'algorithme de classification devient ainsi capable de prédire la classe d'un nouvel objet (*inconnu*). Alternativement, lorsque nous ne disposons pas de cette connaissance, la classification est dite non-supervisée. Dans ce cas, il s'agit, sans aucune connaissance *a priori*, ni sur le nombre de classes ni sur leur nature, de répartir les objets en plusieurs catégories au sens d'une meilleure compacité. Une façon de procéder consiste à minimiser la similarité intra-groupe et à maximiser la similarité inter-groupes.

Dans ce chapitre, nous nous intéressons à la classification de graphes. Nous proposons deux méthodes de classification de graphes. La première consiste en une classification supervisée de graphes en se basant sur une technique de plongement de graphes dans un espace vectoriel. La deuxième méthode proposée correspond à un algorithme de classification non-supervisée (clustering) de graphes. Notre technique de clustering est une adaptation de la méthode de *mean-shift* au domaine des graphes.

5.2 Classification supervisée de graphes

La classification supervisée consiste à affecter un objet à une classe, qui est choisie parmi un ensemble de classes connues. La classification supervisée s'applique lorsque toutes les classes sont connues et que l'on dispose d'exemples de chaque classe. Généralement, une phase, dite d'apprentissage, est effectuée sur ces connaissances *a priori* (les exemples de chaque classe), appelées aussi base d'apprentissage, afin d'extraire des règles d'affectation. En se basant sur ces règles, l'algorithme de classification devient ainsi capable de prédire la classe d'un nouveau objet (*inconnu*). Dans la littérature, plusieurs méthodes de classification supervisée ont été proposées [119]. La quasi-totalité de ces méthodes ont été créées dans un contexte de représentation d'objets sous forme de vecteurs caractéristiques. Ceci est dû au manque de structures mathématiques pour la classification de graphes. Par exemple, ce manque peut se révéler dans les difficultés du calcul d'une somme (pondérée) ou du produit d'une paire de graphes. Par contre, dans un espace vectoriel, le calcul de ces opérations peut se faire via des solutions efficaces et rapides. Ainsi, dans un contexte de classification, l'inconvénient de la représentation sous forme des graphes par rapport aux vecteurs caractéristiques est la complexité élevée des opérations nécessaires pour classifier les objets. Cet inconvénient a longtemps influencé les approches de la reconnaissance de formes de telle sorte que la flexibilité et l'universalité des graphes recule en faveur de l'efficacité des vecteurs caractéristiques, même dans des domaines où les graphes seraient la méthode la plus adéquate [22]. Récemment, plusieurs approches ont visé à combiner les avantages de chaque représentation (structurelle et statistique). Cette combinaison consiste à projeter les graphes dans un espace plus souple avec des structures mathématiques classiques. Les premiers travaux dans ce sens consistaient à projeter les graphes dans un espace de caractéristiques à l'aide de l'extension des méthodes à noyaux sur les graphes [22, 216, 93, 175].

Dans cette partie, nous proposons une technique similaire aux méthodes à noyaux, i.e. cette technique utilise les graphes comme les structures de représentation d'objets et les vecteurs numériques comme les représentations de ces graphes dans un espace vectoriel euclidien. Nous utilisons la notion de *constant shift embedding* proposée par Roth et al. dans [208]. À l'origine, cette idée a été proposée pour projeter des séquences de protéines dans un espace vectoriel

euclidien à n dimensions. Dans cette partie de cette thèse, nous généralisons cette méthode de *constant shift embedding* au domaine des graphes. Notre méthode de reconnaissance de formes est dotée de la représentation et de la flexibilité des structures de données et de l'efficacité et la rapidité des vecteurs.

5.2.1 État de l'art

Noyaux sur graphes

Dans les méthodes à noyaux, les données sont représentées par les comparaisons par paires seulement, au lieu de définir des représentations pour chaque modèle objet. En effet, Dans les approches à noyaux, la représentation explicite des données est d'un intérêt secondaire. 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 $\kappa_{ij} = \kappa(x_i, x_j)$.

Prenons l'exemple simple présenté par Kashima et al. [137]. Dans cet exemple, les auteurs commencent par définir une représentation vectorielle des graphes. L'une des méthodes les plus simple, dans ce sens, est celle qui définie chaque élément du vecteur en fonction du nombre d'apparition d'une étiquette particulière dans le graphe. Soit un graphe $G = (V, E)$ avec $V = \{v_1, v_2, ..., v_{|V|}\}$, le vecteur τ_G représentant G est défini comme suit :

$$\tau_G = \Big(\frac{\eta(\alpha_{v_1}, G)}{|V|}, \frac{\eta(\alpha_{v_2}, G)}{|V|}, ..., \frac{\eta(\alpha_{v_{|V|}}, G)}{|V|}\Big)$$

avec $\eta(\alpha_{v_i}, G)$ est le nombre d'apparitions de l'étiquette α_{v_i} dans G. Ainsi, nous pouvons définir le noyau κ dans le domaine des graphes comme suit :

$$\kappa(G_i, G_j) = \tau_{G_1} \tau_{G_2}^T = \frac{1}{|V_1||V_2|} \sum_{v_i \in V_1} \sum_{v_j \in V_2} k(\alpha_{v_i}, \alpha_{v_j})$$

$$k(\alpha_{v_i}, \alpha_{v_j}) = I(\alpha_{v_i}, \alpha_{v_j})$$

avec I une fonction qui retourne 1 si les deux arguments sont identiques et 0 s'ils sont différents. Le noyau κ peut être considéré comme une composition des sous-noyaux $k(\alpha_{v_i}, \alpha_{v_j})$ appliqués sur les paires des noeuds. Le noyau $k(\alpha_{v_i}, \alpha_{v_j})$ vérifie si les étiquettes des noeuds v_i et v_j sont identiques. Ce choix d'utiliser une décomposition pour définir un noyau sur graphes est le moyen fréquent dans la littérature des approches des noyaux sur les graphes [274]. En fait, ce problème est basé sur des décompositions de graphes en sous-graphes de types particuliers qui sont comparés par des sous-noyaux. Le recours aux décompositions en sous-graphes est dû à l'incapacité de concevoir des algorithmes exacts de définition des noyaux sur graphes. En effet, il a été prouvé dans [92, 195] que les noyaux sur les graphes, en tenant compte de la totalité de leur structure, ne peuvent être ni calculés, ni même approchés de manière efficace. Parmi les sous-structures utilisées pour décomposer les graphes et définir un noyau, on trouve les marches aléatoires [24, 92, 138, 164, 194, 263]. Toutefois, des travaux ont expérimenté d'autres types de sous-structures comme les plus courts chemins [23], sous-arbre [195, 89] et les sous-graphes [170].

Plongement de graphes (*graph embedding*)

Une autre famille de plongement de graphes consiste à trouver une représentation vectorielle qui englobe toutes les caractéristiques de chaque graphe. L'objectif de cette famille d'approches est de construire des vecteurs qui représentent des graphes de telle sorte que la comparaison de ces vecteurs donne une indication précise à propos de la similarité entre les graphes correspondants.

Invariant de graphes

Dans la littérature, la fonction qui assure la projection des graphes dans un espace numérique est généralement appelée *invariant de graphes* [191].

Invariant de graphes *Soit $\sigma : \zeta \rightarrow \mathbb{R}^d$ une fonction de l'espace de graphes ζ dans $\mathbb{R}^d$, avec $d \geq 1$. Soit g_i et $g_j \in \zeta$, si l'existence d'un isomorphisme entre g_i et g_j implique $\sigma(g_i) = \sigma(g_j)$, alors σ est appelé invariant de graphes*

Les racines de cette famille d'approches se trouvent dans le domaine de la bio-informatique. Un des défis majeurs dans ce domaine est la gestion de grandes bases de graphes de molécules et de réactions. Plusieurs méthodes ont été développées pour définir des représentations vectorielles des molécules (graphes) [13, 65, 139, 197, 250] dites *descripteurs topologiques*. Une des méthodes pionnières est celle de Wiener [267], qui date des années 40. Dans cette méthode, chaque graphe est représenté par un indice dit *indice de Wiener*. Cet indice de Wiener est un descripteur topologique défini par la somme de tous les plus courts chemins dans le graphe, formellement :

Indice de Wiener *Soit $G = (V, E)$ un graphe. 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 $l(v_i, v_j)$ une fonction qui définit la longueur du plus court chemin entre le noeud v_i et le noeud v_j dans le graphe G.

Il est démontré [267] que l'indice de Wiener est un invariant de graphes. En effet, un invariant de graphes, e.g. indice de Wiener, est identique pour deux graphes isomorphes. En revanche, deux invariants identiques de deux graphes g_1 et g_2 n'impliquent pas toujours l'isomorphisme entre g_1 et g_2. Dans le cas où l'identité des invariant $I(g_1)$ et $I(g_2)$ implique l'isomorphisme entre g_1 et g_2, l'invariant de graphes $I(.)$ est appelé *invariant complet de graphes* [143].

Le grand nombre de descripteurs topologiques dans la littérature [65] révèle que ces invariants de graphes fournissent une bonne approximation de la mesure de similarité de graphes, notamment dans le domaine de la bio-informatique. En effet, les descripteurs topologiques de graphes sont une solution facile et attractive pour la comparaison de graphes en utilisant des scalaires pour calculer la similarité entre les graphes.

Méthodes spectrales

Dans le domaine de plongement de graphes, les méthodes spectrales [48] ont été largement utilisées dans les travaux récents [41, 40, 76, 162, 206, 254, 270, 271]. Ces approches sont

généralement basées sur les caractéristiques spectrales des matrices d'adjacence de graphes. Dans ce sens, la méthode de Luo et al. [161, 162] convertit les graphes en vecteurs en extrayant quelques caractéristiques spectrales des matrices d'adjacence de graphes. Ensuite, ces vecteurs font l'objet d'une projection dans un espace de caractéristiques en utilisant l'analyse en composantes principales [125] et la représentation affine (*Multidimensional scaling*) [61, 251]. Wilson et al. ont présenté un travail similaire dans [271]. Dans cette approche, les vecteurs qui représentent les graphes sont calculés en utilisant les coefficients des polynômes construits à partir des matrices de Laplace de graphes. Une autre méthode de plongement de graphes basée sur les méthodes spectrales a été introduite par Robles-Kelly et al. dans [206]. L'idée originale de cette méthode est de plonger les noeuds d'un graphe dans un espace métrique et de considérer les arêtes commes des géodisiques [12] entre les paires de points dans une variété Riemannienne.

Sélection de prototypes : représentation par dissimilarités

L'idée principale de cette approche est d'utiliser les distances, entre le graphe considéré et un ensemble présélectionné de graphes, comme la représentation vectorielle du graphe. L'ensemble présélectionné est constitué par des graphes dits prototypes. Cette méthode a initialement été présentée, par Pekalska et al. dans [187, 188], pour le problème du plongement de vecteurs caractéristiques dans un espace de dissimilarité. Dans cette approche, les auteurs montrent que la notion de proximité est plus fondamentale que celle d'une caractéristique ou d'une classe. Autrement dit, l'information de la proximité est plus importante pour la discrimination entre les classes que la composition et les caractéristiques de chaque objet de façon indépendante. En outre, il s'est avéré que cette approche a le potentiel d'unifier les approches statistiques et les approches structurelles [36], parce que les dissimilarités peuvent être calculées à partir d'une représentation structurelle.

L'espace de dissimilarités proposé par Pekalska et al. [187, 188] est défini comme un espace Euclidien. Soit $\chi = \{x_1, x_2, ..., x_{|X|}\}$ l'ensemble des objets à plonger. La construction d'un tel espace nécessite la définition d'un ensemble de prototypes $P = \{p_1, p_2, ..., p_n\} \subseteq X$ où les objets (prototypes) $p_i \in P$ sont sélectionnés adéquatement en se basant sur quelques critères. Les critères de sélection des prototypes peuvent dépendre du type de problème étudié ou de la nature des objets manipulés. En plus de l'ensemble de prototypes, l'espace de dissimilarités est défini par une mesure de distance $d : \chi \times \chi \rightarrow \mathbb{R}^+$. Chaque nouvel objet x est représenté par un vecteur $\sigma(x)$ où ses éléments sont les dissimilarités entre x et les objets dans P.

$$\sigma(r) = [d(x, p_1), d(x, p_2), ..., d(x, p_n)]$$

L'adaptation de cette approche pour le domaine des graphes a d'abord été présentée par Riesen et al. dans [204]. Ensuite, elle a été étudiée davantage dans [36, 85, 202]. Cette adaptation se résume à l'utilisation d'une distance d'édition de graphes et d'un ensemble de prototypes sélectionnés à partir des graphes à traiter pour définir l'espace de dissimilarités. Ainsi, le plongement de graphes est défini formellement comme suit :

12. En géométrie, une géodésique désigne le chemin le plus court, ou l'un des plus courts chemins s'il en existe plusieurs, entre deux points d'un espace pourvu d'une métrique (un moyen de mesurer les distances). [source : *Wikipédia*]

Plongement de graphes dans l'espace de dissimilarités *Soit $\Gamma = \{g_1, g_2, ..., g_n\}$ un ensemble de graphes et $P = \{p_1, p_2, ..., p_m\} \subseteq \Gamma$ un sous-ensemble de prototypes sélectionnés de Γ. Le plongement de graphes :*

$$\sigma : \Gamma \rightarrow \mathbb{R}^m$$

est défini par la fonction

$$\sigma(g) = [d(g, p_1), d(g, p_2), ..., d(g, p_m)]$$

avec $d(g, p_i)$ une mesure de similarité entre le graphe g et le i-ème graphe prototype dans P.

Soient les deux conditions suivantes :
- Si un graphe g appartient à la même classe que p_i alors la distance $d(g, p_i)$ est petite,
- Si g n'appartient pas à la même classe que p_i alors la distance $d(g, p_i)$ est grande.

Si ces deux conditions sont vérifiées, les vecteurs de plongements seront fort probablement constitués par des caractéristiques avec une grande performance de discrimination. En effet, si un graphe g doit être classé dans la même classe que p_i, forcément il est préférable que la caractéristique $d(g, p_i)$ soit plus discriminante qu'une caractéristique $d(g, p_j)$ avec p_i et que p_j n'appartient pas à la même classe. Ainsi, même si le plongement de graphes avec une représentation de dissimilarités semble très attractif et efficace, les résultats de cette méthode sont sensibles aux choix des prototypes. Dans la littérature, plusieurs méthodes de sélection automatique de prototypes ont été proposés [198]. Dans la partie expérimentale, nous détaillerons un ensemble de ces techniques.

5.2.2 Classification par plongement

Le plongement [13] (*embedding*) de graphes consiste à représenter les graphes par des vecteurs projetés dans un espace vectoriel. Les distances entre ces vecteurs doivent respecter au mieux les distances entre les graphes correspondants. Soient G=$\{g_1, ..., g_n\}$ un ensemble de graphes et $d : G \times G \rightarrow \mathbb{R}$ une fonction de distance entre les graphes, soit $D = D_{ij} = d(g_i, g_j) \in \mathbb{R}^{n \times n}$ la matrice de dissimilarité entre les graphes de G. L'objectif de l'embedding de graphes $g_i \in G$ est de fournir n vecteurs x_i dans un espace vectoriel à p dimensions de sorte que la distance entre x_i et x_j soit aussi proche que possible de la dissimilarité $D_{ij} = d(g_i, g_j)$.

Le *Constant shift embedding*

Avant de présenter notre méthode d'embedding [131], nous rappellons la définition de la *matrice centralisée* [208].

Matrice centralisée *Soient P une matrice $n \times n$, I_n une matrice identité d'ordre n et e_n=(1,...,1)$^\intercal$. Soit Q_n =I_n-$\frac{1}{n}e_n e_n^\intercal$. Q_n est la matrice de projection dans le complément orthogonal de e_n. La matrice centralisée P^c est donnée par :*

$$P^c = Q_n P Q_n$$

13. Dans ce mémoire les termes " *plongement*" et "*embedding*" sont utilisés d'une manière interchangeable pour décrire la même tâche du plongement.

Roth et al. [208] propose une méthode de transition d'une matrice de dissimilarité D d'un espace non-euclidien vers une matrice $\tilde{\mathrm{D}}$ de distances euclidiennes. Cette méthode a été définie dans un contexte de classification des séquences de protéines. Son objectif est de représenter les clusters de protéines dans un espace euclidien tout en préservant la distribution originale dans l'espace de protéines. Pour ce faire, cette méthode est basée sur une opération dite *off-diagonal shift*. Cette opération permet la transition sans influencer la distribution des données initiales. Dans notre cas, les données initiales seront les graphes et la matrice D présente les dissimilarités entre ces graphes dans le domaine initial (graphes). Selon Roth et al. le passage de D à $\tilde{\mathrm{D}}$ est donné par l'équation suivante :

$$\tilde{\mathrm{D}} = D + d_0(e_n e_n^{\intercal} - I_n), \text{ ou également } (\tilde{\mathrm{D}}_{ij} = D_{ij} + d_0, \forall i \neq j)$$

avec d_0 une constante.

Ainsi, la nouvelle matrice $\tilde{\mathrm{D}}$ est construite en ajoutant la constante d_0 à chaque élément D_{ij} de la matrice de distances initiales (i.e, chaque élément $\tilde{\mathrm{D}}_{ij} = D_{ij} + d_0$).

La matrice de dissimilarité D est par définition une matrice symétrique et tous les éléments sur la diagonale sont des zéros. En effet, dans notre cas, chaque élément D_{ij} correspond à la distance d'édition entre le i-*ème* graphe et le j-*ème* graphe de la base considérée. Vu que l'approximation de la distance d'édition n'est pas toujours métrique, nous faisons en sorte que la matrice de distance D soit symétrique avec des zéros sur sa diagonale [14].

Donc, d'après Laub et al. [150], la matrice D peut être décomposée en fonction d'une matrice S de la manière suivante :

$$D_{ij} = S_{ii} + S_{jj} - 2S_{ij}$$

Évidement, il n'y a pas une unique matrice S pour décomposer D. En effet, toutes les matrices S + $\alpha \mathrm{e}_n \mathrm{e}_n^{\intercal}$ fournissent la même matrice D, $\forall \alpha \in \mathbb{R}$. Toutefois, il est démontré (Lemme 1 dans [208]) que la version centralisée de la matrice S est unique pour une matrice D donnée. La matrice centralisée de S, S^c, est donnée par :

$$S^c = -\frac{1}{2} D^c, \text{ avec } D^c = Q_n D Q_n$$

Théorème de Torgerson *La matrice D est une matrice de distances euclidiennes si et seulement S^c est semi-définie positive.*

À partir du théorème de Torgerson [252], nous déduisons l'importance de la matrice S^c dans la procédure du plongement. En fait, les similarités données par la matrice D peuvent être projetées dans un espace vectoriel Euclidien si et seulement si la matrice associée S^c est semi-définie positive. Nous rappelons qu'une matrice est dite semi-définie positive si elle est symétrique et que toutes ses valeurs propres sont positives. Dans le domaine des graphes, la matrice de distances D d'un ensemble de graphes est généralement indéfinie, i.e S^c est indéfinie. En effet, D peut posséder des valeurs propres positives et d'autres négatives. Pour surmonter ce problème nous utilisons la notion de *constant shift embedding* introduite dans [208]. En effet,

14. Pratiquement, nous faisons en sorte que la matrice D ne comporte que des 0 sur la diagonale et nous calculons uniquement le triangle inférieur de D ensuite nous le recopions dans le triangle supérieur de D

si nous déplaçons les éléments de la diagonale de la matrice S^c, elle sera transformée en une matrice $\tilde{\mathrm{S}}$ semi-définie positive (voir Lemme 2. dans [208]) :

$$\tilde{\mathrm{S}} = S^c - \lambda_n(S^c) I_n$$

avec $\lambda_n(S^c)$ la valeur propre minimale de la matrice S^c. Le déplacement de la diagonale de la matrice S^c transforme la matrice de distance D d'un ensemble de graphe à une matrice de distances euclidiennes. La matrice résultante du plongement de D est définie par :

$$\tilde{\mathrm{D}}_{ij} = \tilde{\mathrm{S}}_{ii} + \tilde{\mathrm{S}}_{jj} - 2\tilde{\mathrm{S}}_{ij} \Longleftrightarrow \tilde{\mathrm{D}} = D - 2\lambda_n(S^c)(e_n e_n^{\intercal} - I_n)$$

Il est connu que chaque matrice semi-définie positive peut être représentée par une matrice de produits scalaires. Donc, il existe une matrice X tel que $\tilde{\mathrm{S}}^c = \mathrm{XX}^{\intercal}$, où les lignes de X sont les vecteurs x_i. Ainsi, chaque graphe g_i a été plongé dans un espace vectoriel euclidien et est représenté dans cet espace par le vecteur x_i. Il est évident que la matrice $\tilde{\mathrm{D}}$ contient les distances euclidiennes entre ces vecteurs x_i.

Calcul de vecteurs de plongement

Dans cette section, nous présentons un algorithme de construction des vecteurs plongés dans un espace euclidien à partir des graphes. Notre algorithme est inspiré de l'analyse en composantes principales (ACP) [186, 220]. L'algorithme 2 illustre un pseudo-code de notre algorithme. Soient G=$\{g_1, ..., g_n\}$ un ensemble de graphes et D une matrice de dissimilarités entre les graphes $g_i \in G$. En prenant D en entrée l'algorithme retourne en sortie l'ensemble des vecteurs plongés X=$\{x_1, \cdots, x_n\}$ tel que x_i encapsule le graphe g_i. Premièrement, l'algorithme calcule la matrice de distances euclidiennes $\tilde{\mathrm{D}}$ en utilisant la notion de *constant shift embedding* (ligne 1). Ensuite, la matrice centralisée $\tilde{\mathrm{S}}^c = -\frac{1}{2}\tilde{\mathrm{D}}^c$ est calculée (ligne 2). La matrice $\tilde{\mathrm{S}}^c$ est semi-définie positive donc, il existe une matrice X telle que $\tilde{\mathrm{S}}^c = \mathrm{XX}^{\intercal}$, où les lignes de X sont les vecteurs x_i. Les lignes de X sont calculées avec une décomposition en valeurs propres (ligne 3). Il faut noter ici que grâce à la procédure de centralisation, il existe au moins une valeur propre λ_i=0. Donc, la dimension p de l'espace euclidien de plongement est forcément inférieure au nombre total des graphes dans G, i.e. $p \leq n-1$ (ligne 3-4). Finalement, les lignes de la matrice X_p=$V_p(\Lambda_p)^{1/2}$, d'ordre $n \times p$, sont les vecteurs plongés dans l'espace vectoriel à p-dimension.

Dans l'ACP, il est connu que les petites valeurs propres contiennent du bruit. Alors la dimensionalité p peut être réduite en choisissant $t \leq p$ dans la ligne 4 de l'algorithme. En conséquent, la matrice X_p sera remplacée par une matrice X_t=$V_t(\Lambda_t)^{1/2}$ d'ordre $n \times t$, où V_t est la matrice des colonnes des vecteurs propres sélectionnés (les t premiers vecteurs colonnes de V) et Λ_t la matrice diagonale des vecteurs propres correspondant (la sous-matrice de taille $t \times t$ de Λ). On peut se demander comment trouver la valeur de t optimale qui donne la meilleure performance d'un classificateur dans un espace vectoriel. En effet, la dimensionnalité t a une grande influence sur les performances du classificateur. Dans cette thèse, le t optimal est choisi d'une manière empirique. Cela signifie que le t optimal est celui qui offre le meilleur taux de classification avec $2 \leq t \leq p$.

Algorithme 2 Construction de vecteurs

ENTRÉES: La matrice de dissimilarités D de l'ensemble de graphes G=$\{g_1, ..., g_n\}$

SORTIES: Un ensemble de vecteurs X=$\{x_1, \cdots, x_n\}$ où x_i encapsule g_i

1: Calcul de la matrice des distances euclidiennes $\tilde{D}$

2: Calcul de la matrice centralisée $\tilde{S}^c = -\frac{1}{2}\tilde{D}^c$, avec $\tilde{D}^c$ = Q$\tilde{D}$Q.

3: Décomposition en valeurs propres de la matrice $\tilde{S}^c$, $\tilde{S}^c = V\Lambda V^\intercal$
   - Λ=diag($\lambda_1, ...\lambda_n$) est la diagonale de la matrice de valeurs propres.
   - V=$\{\upsilon_1, ...\upsilon_n\}$ est la matrice orthonormée des vecteurs propres υ_i.

   ▷ $\lambda_1 \geq ... \lambda_p \geq \lambda_{p+1} = 0 = ... = \lambda_n$

4: Calcul de la matrice X_p=$V_p(\Lambda_p)^{1/2}$, avec V_p=$\{\upsilon_1, ...\upsilon_p\}$ et Λ_p= diag($\lambda_1, ...\lambda_p$)

5: **Sortie** : les lignes de X_p contenant les vecteurs encapsulés dans un espace à p-dimension.

5.2.3 Résultats du concours GEPR - ICPR

Dans cette section, nous décrivons les résultats du premier concours du plongement de graphes pour la reconnaissance (Graph embedding for Pattern recognition). Ce concours [88] a été organisé par P. Foggia et M. Vento dans le contexte de la conférence internationale ICPR 2010 [15]. Dans ce concours quatre algorithmes de plongement de graphes, dont notre méthode, ont été évalués. Dans la table 5.1, nous présentons les quatre équipes participantes. Dans ce concours, deux types de plongement de graphes sont considérés : implicite et explicite. Le plongement explicite concerne les méthodes de plongement de graphes, dont notre méthode fait partie, qui fournissent explicitement pour chaque graphe un vecteur représentatif. Par contre, le plongement implicite correspond aux méthodes qui produisent un plongement implicite de graphes dans le sens où ils fournissent uniquement le produit scalaire entre les vecteurs associés aux paires de graphes. Les organisateurs ont testé les algorithmes sur les trois bases de graphes suivantes :

aloi-2 Cette base consiste en une sélection de la base ALOI (*The Amsterdam library of object images*) [95] qui est une collection d'images de 1000 objets. Chaque objet est acquis plusieurs fois en changeant son orientation et sa luminosité. La sélection consiste en 25 objets et 72 vues de chaque objet, qui correspond à 1800 images.

coil-2 C'est la même base utilisée que dans le chapitre 4. La sélection utilisée dans ce concours consiste en 25 objets et 72 vues de chaque objet, qui correspond à 1800 images.

odbk-2 [16] Cette base consiste en une collection de 209 objets 3D. Chaque objet est acquis 14 fois sous différentes orientations. La sélection utilisée par les organisateurs consiste en 104 objets et 12 vues de chaque objet, qui correspond à 1248 images.

Les images dans chaque base sont représentées par des graphes d'adjacence de régions. En effet, chaque image est, d'abord, lissée par un filtre gaussien. Ensuite, elle est segmentée par un algorithme de segmentation pyramidale. À partir de cette segmentation, un graphe d'adjacence de régions est extrait. Chaque noeud du graphe est étiqueté par la taille et la moyenne des couleurs

15. 20^{th} International Conference on Pattern Recognition, Istanbul `http://www.icpr2010.org/`
16. `http://www.cnbc.cmu.edu/tarrlab/stimuli/objects/index.html`

TABLE 5.1 – Les participants dans le concours GEPR [88]

Équipe	Établissement	Type	Langage d'implémentation
Y. Osmanlioglu, F. Yilmaz, M. F. Demirci	Drexel University, TOBB University of Economics and Technology	implicit	C/C++
S. Jouili, S. Tabbone	Laboratoire Lorrain de Recherche en Informatique et ses Applications	explicit	Java
K. Riesen, H. Bunke	University of Bern	explicit	Java
M. M. Luqman, J. Lladós, J.-Y. Ramel, T. Brouard	Université François Rabelais de Tours, Universitat Autónoma de Barcelona	explicit	Matlab

de la région correspondante. Par contre, les arêtes ne sont pas étiquetées. Outre ces bases de test, les organisateurs ont fourni trois bases d'apprentissage dont chacune est similaire (une autre sélection de la base initiale) à une base de test.

Pour mesurer les performances de chaque algorithme, les organisateurs ont considéré la vérité terrain de chaque base comme un résultat de clustering. Ensuite, ils ont évalué la séparation entre les clusters de chaque base en utilisant l'indice de C (voir section 5.3.2). La distance euclidienne est utilisée pour définir les distances entre les vecteurs obtenus de chaque algorithme de plongement de graphes explicite. En ce qui concerne la méthode implicite, les organisateurs ont utilisé la distance suivante ;

$$d_{ij} = \sqrt{p_{ii} + p_{jj} - 2p_{ij}}$$

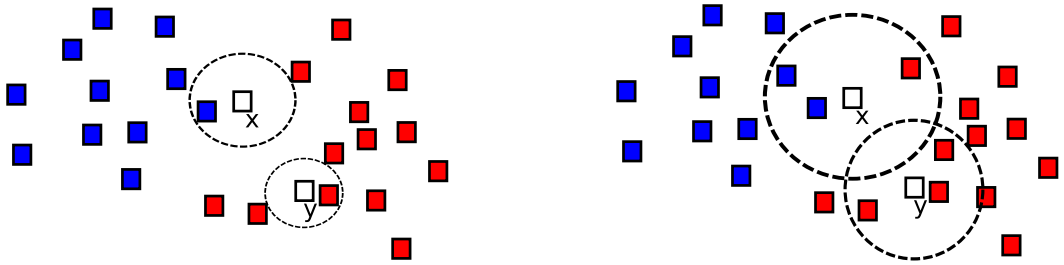
avec d_{ij} la distance entre les graphes g_i et g_j, et p_{ij} le produit scalaire entre g_i et g_j.

La table 5.2 montre les résultats du concours. À partir de ces résultats, nous constatons que l'algorithme de Osmanlioglu et al., qui est une méthode implicite, fournit les meilleurs résultats. Notre méthode est classée troisième après la méthode de Riesen et Bunke avec une petite différence de 0.062. Par ailleurs, notre méthode est loin devant la méthode de Luqman et al. (classée quatrième) avec une différence remarquable qui s'élève à 0.215.

En considérant uniquement les méthodes de plongement explicites, notre méthode est classée deuxième avec des résultats comparables à ceux de la méthode de Riesen et Bunke (classée première). Nos résultats peuvent être améliorés si nous établissons une étude de la valeur optimale de t (voir section 5.2.2). Dans le concours, cette valeur a été fixée à 10% de la taille initiale des vecteurs car avec cette valeur nous avions obtenu les meilleurs résultats sur l'ensemble des bases d'apprentissage. Mais en changeant les bases, la valeur optimale de t devrait potentiellement changer.

TABLE 5.2 – Les résultats du concours GEPR [88]

Équipe	Bases de données			Moyenne géometrique
	aloi-2	coil-2	odbk-2	
Osmanlioglu et al.	0.088	**0.067**	**0.105**	**0.085**
Jouili et Tabbone	0.136	0.199	0.138	0.155
Riesen et Bunke	**0.048**	0.128	0.132	0.093
Luqman et al.	0.379	0.377	0.355	0.370



(a) k=1 (b) k=3

FIGURE 5.1 – Illustration de $k - ppv$

5.2.4 Résultats expérimentaux : classification supervisée

Algorithme des k plus proches voisins ($k - ppv$)

L'algorithme des k plus proches voisins (k-PPV) [60] est certainement un des algorithmes locaux non paramétriques les plus simples de classification supervisée. Il consiste à déterminer pour chaque nouvel objet x que l'on veut classer, l'ensemble des k plus proches voisins parmi l'ensemble d'apprentissage (i.e. objets déjà classés). L'objet x est affecté à la classe qui contient le plus d'objets parmi ces k plus proches voisins. Dans l'algorithme 3, nous présentons un pseudo-code de l'algorithme $k - ppv$. Cet algorithme est défini par trois paramètres principaux : le nombre de voisins k, la mesure de distance et l'ensemble d'apprentissage.

Algorithme 3 Algorithme de classification $k - ppv$

ENTRÉES: Un objet x à classer et un ensemble d'apprentissage A
SORTIES: Affectation de x dans une classe c

1: Déterminer $N_k(x) \subseteq A$, l'ensemble des k plus proches voisins de x, en utilisant une mesure de distance.
2: Choisir la classe c de x sur la base d'un vote majoritaire dans $N_k(x)$. En cas d'égalité, nous choisissons au hasard la classe parmi les classes ambiguës.

Dans la figure 5.1, nous nous intéressons à classer les deux objets x et y. L'ensemble d'apprentissage est constitué par un ensemble d'objets répartis sur deux classes (rouge et bleu). Dans

un premier temps, dans la figure 5.1(a), nous fixons $k = 1$. Les deux cercles noirs contiennent, respectivement, l'objet le plus proche de x et l'objet le plus proche de y. Le voisin le plus proche de x est un objet bleu, donc x est affecté dans la classe bleue. Pareillement, l'objet y est classé avec les rouges. Dans un deuxième temps, dans la figure 5.1(b), nous fixons $k = 3$. Dans ce cas, nous recherchons les trois objets les plus proches à chaque objets à classer (i.e. x et y). Le voisinage de x est composé de deux objets bleus et un objet rouge. En effectuant un vote majoritaire, x sera classé dans la classe bleue. Par ailleurs, le voisinage de y n'est composé que d'objets rouges, donc y sera affecté dans la classe rouge.

L'un des avantages majeurs de l'algorithme de $k - ppv$ est sa simplicité. Cette simplicité a été approuvée par l'applicabilité de cet algorithme dans le domaine des graphes sans aucune adaptation nécessaire. En effet, le voisinage de chaque graphe est défini en utilisant une mesure de similarité de graphes comme mesure de distance.

Les séparateurs à vastes marges (SVM)

Les séparateurs à vaste marge (en anglais : *Support Vector Machines*) sont les algorithmes de classification les plus connus des méthodes à noyaux. Ils sont basés sur la théorie statistique de l'apprentissage de Vapnik [57, 261, 260]. L'idée principale de ces méthodes est d'utiliser une fonction à noyau pour projeter les données de l'espace de représentation initiale dans un espace à dimensionalité plus grande appelé espace support (*feature space*). Cette projection vise à produire dans le nouvel espace une nouvelle distribution de façon à ce que les données deviennent linéairement séparables. Les SVM forment une méthode de classification binaire qui repose sur l'idée que deux classes peuvent être linéairement séparées dans un espace de grande dimension. La séparation entre les deux classes est modélisée par un hyperplan particulier. Néanmoins, s'il existe un hyperplan séparateur entre les points d'apprentissage, il en existe une infinité. La méthode SVM cherche alors parmi ces hyperplans séparateurs celui qui est le meilleur : le plus éloigné de tous les points de l'ensemble d'apprentissage. Cet hyperplan séparateur optimal est appelé l'*hyperplan séparateur de marge maximale*.

La classification avec les séparateurs à vaste marge a été largement étudiée dans la littérature de la reconnaissance [38]. Elle offre l'avantage de gérer à la fois des données continues et discrètes et ne nécessite pas un grand ensemble d'apprentissage. En outre, Les SVM sont connus par la rapidité de traitement qui est due à l'utilisation des vecteurs supports pour la classification d'un nouveau objet. Ces méthodes ont montré de bonnes performances dans différents problèmes de reconnaissance de formes [140, 180, 190, 235].

Cependant, ces méthodes ne sont applicables que sur des données vectorielles. Dans la littérature, il n'existe, à notre connaissance, aucune approche d'adaptation des SVM dans le domaine de graphes. Ainsi, l'utilisation du plongement de graphes est considérée comme le passage incontournable pour profiter de la classification des graphes avec des SVM.

Configuration

Nous utilisons les trois bases de graphes suivantes fournit par l'IAM [17] [199] :

17. *Institute of Computer Science and Applied Mathematics*

Mutagenicity Dans cette base les graphes représentent des composés moléculaires, les noeuds représentent les atomes étiquetés du symbole chimique correspondant et les arêtes sont étiquetés par les valences d'éléments chimiques. L'ensemble de graphes utilisé dans nos expérimentations contient 1500 graphes, 2 classes et 750 graphes par classe.

Letter Dans la base Letter, les graphes représentent des dessins déformés de quinze lettres latines (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z). Chaque lettre déformée correspond à un graphe. Les arêtes sont les lignes et les noeuds sont les points finaux de lignes. Les noeuds sont étiquetés par un attribut à deux dimensions pour donner sa position (coordonnées). L'ensemble des graphes utilisé dans nos expérimentations contient 1500 graphes, 15 classes et 100 graphes par classe.

GREC2 Dans le base GREC2, les graphes sont la représentation des symboles extraits des plans architecturaux et éléctroniques. Mais ici les graphes sont étiquetés : les noeuds par deux attributs numériques qui indiquent la position du point final et un attribut symbolique qui indique le type d'intersection. Les arêtes sont étiquetées par trois attributs : deux numériques pour l'angle d'inclinaison de la ligne et la fréquence, et un attribut symbolique pour le type (ligne, cercle ...). L'ensemble des graphes utilisé dans nos expérimentations contient 814 graphes, 24 classes et 37 graphes par classe.

Les expérimentations consistent à appliquer notre méthode de plongement de graphes sur chaque base de données. Notre intention est de montrer, empiriquement, que la méthode proposée est en mesure de fournir des vecteurs qui peuvent améliorer les résultats de classification obtenus avec la représentation d'origine.

Les expérimentations comportent deux parties. La première partie consiste à comparer les résultats effectués dans le domaine des graphes aux résultats obtenus avec les vecteurs de plongement. Une telle comparaison ne peut aboutir qu'en utilisant un classificateur capable de classifier des graphes ainsi que des vecteurs numériques. Par conséquent, le classificateur $k-ppv$ est utilisé comme algorithme de classification dans la première partie. Dans la deuxième partie, nous utilisons les SVM [18] pour classifier les vecteurs de plongement. L'objectif, dans cette partie, est de vérifier si les classificateurs SVM améliorent les résultats obtenus dans le domaine des graphes.

Dans les deux parties des expérimentations, nous avons comparé les taux de classification obtenus par notre méthode de plongement avec ceux obtenus avec la méthode de plongement de graphes de Bunke et al. [202, 36, 85, 204, 198] basée sur la représentation par dissimilarités (détaillée précédemment). Comme il a été noté dans le paragraphe dédié à la présentation de cette méthode, les performances de cette dernière dépendent du choix de la sélection des prototypes appropriés. Dans cette thèse, nous avons utilisé quatre sélecteurs de prototypes [188]. Le choix de ces quatre techniques est justifié par leurs performances par rapport aux autres sélecteurs de prototypes [19]. Les algorithmes de sélection de prototypes utilisés sont :

k-centers prototype selector (KCPS) Cette méthode sélectionne les prototypes de la même façon que l'algorithme de clustering $K-means$ [163] sélectionne les centres des clus-

18. Nous utilisons le classificateur C-SVM avec une fonction à noyau linéaire disponible dans le logiciel libre *weka* `http://www.cs.waikato.ac.nz/ml/weka/`

19. Plusieurs méthodes existent dans la littérature. Dans la thèse de Kaspar Riesen [198], six techniques de sélection de prototypes ont été présentées.

	Domaine des graphes	Espace vectoriel				
			Bunke avec			
Base de graphes		Notre embedding	KCPS	SPS	TPS	RandPS
GREC2	98.11%	**99.50**%	97.17%	96.19%	97.78%	97.54%
Letter	79.33%	91.33%	92.4%	**92.66**%	92.46%	92.4%
Mutagenicity	63.70%	63.8%	60.93%	63.73%	64.13%	**64.73**%

TABLE 5.3 – Résultats avec le classificateur $k - ppv$

ters. Les n prototypes sélectionnés par cette méthode sont répartis uniformément dans l'ensemble de graphes.

the spanning prototype selector (SPS) Cette méthode commence par le graphe médian comme premier prototype. Ensuite, chaque nouveau prototype est sélectionné de façon à ce qu'il ait la distance la plus grande à tous les prototypes déjà sélectionnés.

the target-sphere prototype selector (TPS) Les n prototypes sélectionnés avec cette méthode sont uniformément répartis entre le centre et le bord. Cette méthode commence par sélectionner le graphe médian c de l'ensemble initial. Ensuite, le graphe f ayant la plus grande distance à c est sélectionné. Les $n-2$ prochains prototypes sont sélectionnés de telle façon que leur distance à c divise l'intervalle [0,$d(c, f)$] en $n - 1$ sous-intervalles équidistants de largeur $\frac{d(c,f)}{n-1}$, avec $d(c, f)$ la distance entre le graphe médian c et le graphe f.

the random prototype selector (RandPS) Cette méthode sélectionne aléatoirement n prototypes sans redondance à partir d'un ensemble de graphes.

Un second paramètre important de cette méthode est le choix du nombre p de prototypes qui doivent être sélectionnés pour obtenir les meilleures performances de classification. Pour surmonter ce problème, nous définissons le p optimal selon la même procédure déjà appliquée pour déterminer le t optimal de notre algorithme (cf. section 5.2.2 page 88).

Dans cette partie expérimentale, le calcul de la distance entre les graphes est assuré par notre distance d'édition de graphes [127, 129] proposée dans le chapitre 4 (page 51).

Résultats

La première expérimentation consiste à appliquer le classificateur $k - ppv$ dans le domaine des graphes et dans l'espace vectoriel de plongement. Dans le tableau 5.3, les taux de classification avec le $k - ppv$ sont donnés pour toutes les bases de données. On peut remarquer que les résultats obtenus dans l'espace vectoriel sont meilleurs que les résultats obtenus dans le domaine des graphes pour tous les ensembles de données. Toutefois, cette amélioration n'est pas importante, car elle ne dépasse pas 2% pour presque tous les ensembles de données, sauf pour la base Letter. Dans l'espace vectoriel de plongement, notre méthode réalise les meilleurs taux de classification sur la base GREC2. Pour les bases Letter et Mutagenicity, la méthode de Bunke réalise des résultats meilleurs que notre méthode. Toutefois, nous constatons que les résultats de la méthode de Bunke varient selon l'algorithme de sélection de prototypes utilisé. De plus, la

	Domaine des graphes	Espace vectoriel				
			Bunke avec			
Base de graphes		Notre embedding	KCPS	SPS	TPS	RandPS
GREC2	-	**99.87**%	**99.87**%	99.50%	99.75%	99.75%
Letter	-	**94.87**%	91.60%	91.40%	91.53%	91.53%
Mutagenicity	-	**71.20**%	57.53%	57.31%	57.48%	57.45%

TABLE 5.4 – Résultats avec le classificateur SVM

différence entre les taux de classification obtenus par notre méthode et ceux obtenus par la méthode de Bunke (avec le meilleur sélecteur de prototypes) n'est pas sensible, i.e. elle ne dépasse pas 2%.

Cette première expérience vise essentiellement à montrer l'intérêt des techniques de plongement pour la classification de graphes. En effet, avec l'usage du même classificateur $k-ppv$, on n'obtient les meilleurs taux de classification qu'avec les vecteurs de plongement. Toutefois, dans l'espace vectoriel nous ne sommes pas limités à un classificateur simple tel que l'algorithme des plus proches voisins. Ainsi, dans le tableau 5.4 les taux de classification effectués avec le classificateur SVM dans l'espace vectoriel de plongement sont donnés pour tous les ensembles de données. On peut remarquer que le plongement de graphes améliore nettement la performance obtenue dans le domaine des graphes. En ce qui concerne la base GREC2, le meilleur taux de classification est réalisé par notre méthode et la méthode de Bunke avec le sélecteur de KCPS. Les SVM fournissent un taux de classification meilleur que celui obtenue par le $k-ppv$ dans le domaine des graphes (une amélioration de 1,76%). Cette amélioration n'a pas de signification statistique. Cela est dû au fait que la classification dans le domaine des graphes de la base des données GREC2 fournit déjà une très bonne performance de 98,11%. Pour la base de données Letter, toutes les méthodes de plongement de graphes améliorent nettement la performance obtenue dans le domaine des graphes avec le classificateur $k-ppv$.

Enfin, les résultats concernant la base Mutagenicity (Table. 5.4) montrent que toutes les variantes de la méthode de Bunke ne parviennent pas à améliorer la performance obtenue dans le domaine des graphes et fournissent les résultats les moins bons. Alors que notre plongement de graphes utilisant le *constant shift* améliore la performance de classification obtenue par le classificateur $k-ppv$ dans la représentation en forme de graphe d'origine (+ 7,5 %). Par conséquent, notre méthode fournit des vecteurs de plongement plus signifiants pour les graphes de la base Mutagenicity.

Pour résumer, en combinant les vecteurs de plongement fournis par notre méthode et le classificateur SVM, le taux de classification s'améliore par rapport aux taux obtenus dans le domaine des graphes pour toutes les bases de données utilisées dans les expérimentations. Ceci est en accord avec notre intention d'améliorer les résultats de classification obtenus dans le domaine des graphes à l'aide d'une technique de plongement de graphes. En outre, la comparaison avec les quatre variantes de la méthode de Bunke a montré que la technique proposée surpasse ces alternatives pour presque toutes les bases de graphes utilisées.

5.3 Classification non-supervisée de graphes

La classification non-supervisée (ou clustering) est un processus qui vise à trouver des partitions d'objets similaires. Il s'agit d'une procédure de reconnaissance non-supervisée car il n'y a pas de classes prédéfinies qui indiquent le regroupement des propriétés dans l'ensemble des données. Le clustering a été largement étudié grâce à son utilité dans de nombreux domaines d'application autre que l'informatique, les sciences sociales et la biologie. Succinctement, le clustering vise à synthétiser une grande quantité de données par un nombre réduit de groupes homogènes et disjoints. Les objets appartenant à un même groupe sont semblables les uns aux autres et les objets les plus dissemblables appartiennent à différents groupes. Un nombre conséquent d'algorithmes de clustering ont été proposés. La plupart de ces algorithmes traitent les données représentées par des vecteurs caractéristiques. Nous renvoyons le lecteur à l'état de l'art des algorithmes de clustering de Xu dans [279] (voir aussi [120]). Néanmoins, seuls quelques travaux traitent le clustering des structures de données, en particulier le clustering de graphes.

Dans la littérature, la signification du terme "*clustering de graphes*" n'est pas unique. Dans [217], le clustering de graphes est défini comme la procédure d'identification des groupes de noeuds similaires dans le même graphe. Tandis que dans [31, 77, 83, 106, 130, 193, 198, 213] le terme est utilisé pour désigner les procédures de clustering de graphes plutôt que de vecteurs caractéristiques. Dans cette thèse, nous adaptons la deuxième définition pour définir le clustering de graphes.

Les algorithmes de clustering de graphes se répartissent grosso modo en deux catégories. La première catégorie comprend les méthodes basées sur le plongement de graphes (cf. chapitre précédent) où une projection du domaine des graphes vers un espace vectoriel est proposée. Ces méthodes utilisent la notion des noyaux sur graphes [200, 161] pour établir la projection vers l'espace vectoriel. Ensuite, des techniques de clustering classiques sont appliquées aux vecteurs de plongement. La deuxième catégorie concerne les méthodes qui effectuent le clustering directement dans le domaine des graphes. Ces approches utilisent des outils appropriés notamment le calcul de représentants de groupes [234] et la notion de graphe médian [276] afin d'adapter le clustering classique dans le domaine des graphes [83, 130].

5.3.1 État de l'art

La quasi-totalité des méthodes de clustering sont exclusivement prédestinées au clustering d'objets décrits sous forme de vecteurs caractéristiques. Seuls quelques travaux sont publiés sur le clustering de graphes, i.e. clustering d'objets représentés sous forme de graphes. Ces travaux proposent des méthodes applicables au domaine des graphes en restant sur les mêmes principes du clustering dans l'espace vectoriel. En effet, pour un ensemble de graphes $G = \{g_1, ..., g_n\}$, le clustering de graphes a pour objectif de produire une partition $\zeta = \{c_1, ..., c_k\}$, tel que $c_i \subseteq G$, $c_i \neq \emptyset$, $\forall i \neq j \; c_i \cap c_j = \emptyset$, et $\bigcup_{i=1}^{k} c_i = G$.

Les algorithmes de clustering de vecteurs utilisent une fonction de distance. Cette fonction est souvent implémentée par la distance Euclidienne. Pareillement, l'utilisation d'une fonction de distance de graphes se révèle cruciale pour aboutir à une adaptation d'une méthode de clustering dans le domaine des graphes. L'existence des approches de calcul de distance entre les graphes (cf chapitre 3) permet ainsi l'adaptation facile d'un ensemble d'algorithme de cluster-

ing, i.e. les algorithmes de clustering hiérarchique. En fait, les algorithmes de clustering hiérarchique peuvent être appliqués directement aux graphes juste en utilisant une distance de graphes. Cette distance est utilisée pour calculer la distance entre les clusters.

Néanmoins, une distance de graphes n'est pas suffisante pour adapter la plupart des algorithmes populaires de clustering au domaine des graphes. L'autre ingrédient nécessaire est le calcul d'un représentant d'un ensemble de graphes. Par exemple, dans l'algorithme des k-moyennes, à chaque itération, un représentant (centre) pour chaque cluster est calculé. De la même manière, dans le clustering flou, les centres de clusters sont utilisés pour actualiser les degrés d'appartenance de chaque objet. Le calcul d'un tel représentant dans l'espace vectoriel est relativement simple (i.e. le vecteur moyen ou le vecteur médian). Par contre dans le domaine des graphes, il n'y pas une technique intuitive pour calculer un représentant d'un ensemble de graphes.

Dans la littérature, quelques méthodes ont été proposées pour calculer le représentant optimal d'un ensemble de graphes. Dans ces travaux, le représentant d'un ensemble de graphes est généralement appelé "graphe médian".

Graphe médian

Étant donné un ensemble d'éléments, le médian peut être un concept très utile pour avoir une représentation qui accumule une information globale de l'ensemble. Dans le domaine des graphes, le graphe médian, introduit par Jiang et Bunke dans [276, 122, 29], a pour objectif l'extraction de l'information essentielle à partir d'un ensemble de graphes en un seul prototype. 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 [276]. Dans la littérature, nous distinguons deux types de graphe médian : le graphe médian et le graphe médian généralisé. La différence principale entre ces deux types est l'ensemble des graphes où le graphe médian est recherché. Si le graphe médian appartient à l'ensemble de graphes de départ, alors on parle du graphe médian. Par contre, si le graphe médian n'appartient pas à l'ensemble de départ (i.e un nouveau graphe), alors on parle du graphe médian généralisé.

Soient $G = \{g_1, ..., g_n\}$ un ensemble de graphes et $\mathcal{U}$ l'ensemble contenant tous les graphes qui peuvent être construits à partir de l'ensemble G (i.e. tous les graphes pouvant être construits en utilisant les ensembles d'étiquettes de noeuds et d'arêtes dans G, $G \subset \mathcal{U}$). Le graphe médian généralisé et le graphe médian sont respectivement définis par :

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

Avec $d(\cdot,\cdot)$ une certaine fonction de distance qui mesure la distance entre deux graphes.

Le concept de graphe médian généralisé est plus intéressant parce qu'il a un plus grand potentiel de capter l'information essentielle de l'ensemble des graphes. Mais aussi plus complexe à calculer, car sa complexité grandit exponentiellement en fonction du nombre de graphes et de

leur taille. Par contre, le graphe médian peut être calculé en $O(n^2)$ étapes (pour chaque étape, nous n'aurons besoin de calculer que la distance entre deux graphes).

Même si le problème du graphe médian généralisé est quelque peu récent, il existe quelques travaux qui proposent différentes méthodes pour calculer le graphe médian généralisé. Par exemple, Jiang [275] utilise un algorithme génétique avec représentation chromosomique. Hlaoui [111] propose une solution basée sur la décomposition du problème de la minimisation de la somme des distances en deux parties : une pour la somme des distances dédiée aux nœuds et l'autre pour les arcs. Encore, la théorie spectrale des graphes [82] ainsi que la programmation linéaire [147] ont été utilisées pour cette problématique.

Outre le graphe médian, nous distinguons la notion de graphe moyen pondéré qui est très proche du graphe médian généralisé. Elle est introduite par Bunke et al. dans [32]. Nous pouvons définir le graphe moyen pondéré comme un graphe médian généralisé de deux graphes en utilisant un poids qui définit la position de ce dernier entre les deux graphes considérés. Formellement, étant donnés deux graphes g_1 et g_2 et un nombre $0 < \alpha < d(g_1, g_2)$, avec d une distance d'édition de graphes, le graphe moyen pondéré par α est le graphe g_α tel que :

- $d(g_1, g_p) = \alpha$ et,
- $d(g_1, g_2) = \alpha + d(g_p, g_2)$

k-moyennes dans le domaine des graphes

Dans [218], les auteurs introduisent une nouvelle version de l'algorithme des k-moyennes qui peut regrouper des objets représentés sous forme de graphes. L'extension des k-moyennes au domaine des graphes s'est avérée simple. Tout d'abord, la mesure de distance entre les objets est réalisée avec une distance d'édition de graphes. Deuxièmement, étant donné qu'il est nécessaire de calculer la distance entre les objets et le centre de chaque cluster, il s'ensuit que les centres des clusters (représentants) doivent également être des graphes. Par conséquent, le représentant d'un cluster est considéré comme le graphe médian de l'ensemble des graphes de ce cluster [219].

Algorithme 4 Algorithme de k-moyennes adapté au domaine des graphes

ENTRÉES: un ensemble de graphes $G = \{g_1, ..., g_n\}$, k : nombre de clusters
SORTIES: un ensemble de clusters $c_1, c_2, ..., c_k$

1: Choisir aléatoirement les k graphes médians initiaux des clusters $g_{m1}, ..., g_{mk}$
2: **Répéter**
3: Affecter chaque graphe g_i au cluster ayant le graphe médian m_j le plus proche de g_i, en utilisant une distance de graphes.
4: Recalculer les graphes médian des clusters $g_{m1}, ..., g_{mk}$
5: **Jusqu'à** Condition d'arrêt satisfaite

La version originale de l'algorithme des k-moyennes [163] est une méthode très simple et applicable exclusivement aux objets représentés par des vecteurs caractéristiques. Chaque objet correspond à un vecteur réel de dimension d projeté dans un espace $\mathbb{R}^d$. Évidemment, dans le domaine de la reconnaissance de formes chaque élément d'un vecteur caractéristique correspond à une caractéristique calculée dans l'objet correspondant.

Pour un ensemble d'objets $\chi = \{x_1, ..., x_n\}$, l'algorithme des k-moyennes a pour objectif de minimiser la fonction de compacité suivante :

$$f_c = \sum_{j=1}^{k} \sum_{x_i \in c_j} d(x_i, m_j) \tag{5.1}$$

avec k le nombre de clusters, $c_1, c_2, ..., c_k$ une partition de χ, $d(.,.)$ une fonction de distance entre les objets de χ, et m_j le centre de cluster c_j.

Le clustering avec les k-moyennes nécessite le nombre de cluster k à produire comme paramètre. Cet algorithme commence par choisir k objets au hasard parmi l'ensemble d'objets. Ensuite, chaque objet est affecté à l'un des clusters en fonction du représentant (centres) le plus proches. Par la suite, les nouveaux centres des clusters sont calculés. L'algorithme itère sur les deux dernières étapes jusqu'à la satisfaction d'une condition d'arrêt. Il existe plusieurs conditions d'arrêt utilisées dans la littérature. Parmi ces conditions, nous trouvons l'arrêt après un nombre prédéfini d'itérations, le changement minimal (ou pas de changement) des centres de clusters. Notons que l'algorithme des k-moyennes utilise, généralement, la distance euclidienne pour calculer la distance entre les objets, et que le centre d'un cluster est considéré comme la moyenne des vecteurs de ce cluster. Dans son adaptation aux graphes, l'algorithme de k-moyennes utilise la distance d'édition de graphes pour calculer les distances, et la notion du graphe médian est utilisée pour déterminer les centres des clusters. Dans l'algorithme 4, nous présentons un pseudo-code de l'adaptation de k-moyennes au domaine des graphes.

5.3.2 Indices de validation de clustering

Les indices de validation de clustering sont des méthodes qui mesurent la qualité d'un clustering donné. Ces méthodes sont largement utilisées pour définir les paramètres optimaux des algorithmes de clustering. Dans le cas des k-moyennes, le choix de k influence énormément le résultat de l'algorithme. Ainsi, on peut exécuter k-moyennes plusieurs fois en spécifiant une valeur différente de k à chaque exécution, et finalement sélectionner la valeur de k qui a fourni la valeur optimale d'un indice de validation de clustering. Dans la suite de cette section, nous passons en revue quelques indices de validation de clustering [71, 64, 102, 196]. Mais dans la littérature, il existe d'autres méthodes pour le calcul des indices de validation de clustering tels que, par exemple, l'indice de *Xie* [278], l'indice de *Calinski* [42] et l'indice $\mathcal{I}$ [166].

Indice de Dunn

Soit $d(c_i, c_j)$ la distance minimale entre deux objets appartenant à deux clusters différents c_i et c_j, cette distance est appelée distance inter-cluster. Soit $\Delta(c_i)$ la distance maximale entre deux objets appartenant à c_i, cette distance est appelée distance intra-cluster,

$$\Delta(c_i) = max\{d(x, y) \mid x, y \in c_i\}$$

Soit Δ_{max} la distance intra-cluster maximale dans les k clusters,

$$\Delta_{max} = max\{\Delta(c_i) \mid i = 1, 2, ..., k\}$$

Soit Δ_{min} la distance inter-cluster minimale entre les k clusters,

$$\Delta_{min} = min\{d(c_i, c_j) \mid i, j = 1, 2, ..., k\}$$

Ainsi l'indice de Dunn [71] est défini par :

$$D = \frac{\Delta_{min}}{\Delta_{max}}$$

D est le quotient de la distance entre les clusters plus proches par la distance intra-cluster du cluster ayant le plus grand diamètre. Il est clair que, plus la valeur de D est large, plus le clustering est considéré meilleur. En fait, une grande valeur de D signifie que soit les clusters sont très compacts (une faible valeur Δ_{max}), soit les clusters sont très éloignés (une grande valeur Δ_{min}), et dans les deux cas le clustering est de bonne qualité.

Indice de Rand

L'indice de Rand [196] mesure à quel point les clusters créés par l'algorithme de clustering correspondent à la vérité terrain. Pour calculer l'*indice de Rand*, nous considérons toutes les paires d'objets (x_i, x_j) avec $x_i \neq x_j$. Soit N_{11} le nombre de paires (g_i, g_j) appartenant à la même classe (dans la vérité terrain) et au même cluster (dans le clustering), soit N_{00} le nombre de paires (g_i, g_j) n'appartenant ni à la même classe (dans la vérité terrain) ni au même cluster (dans le clustering), soit N_{10} le nombre de paires (g_i, g_j) appartenant à la même classe (dans la vérité terrain) et pas au même cluster (dans le clustering). Inversement, N_{01} le nombre de paires (g_i, g_j) appartenant au même cluster (dans le clustering) mais pas à la même classe (dans la vérité terrain). Nous pouvons donc formuler l'indice de Rand par :

$$Rand = \frac{N_{11} + N_{00}}{N_{11} + N_{00} + N_{10} + N_{01}}$$

où l'indice de Rand mesure la fraction du nombre total de paires qui sont soit dans le même groupe et dans la même partition, ou dans des groupes différents et dans des partitions différentes. Les valeurs possibles de l'indice de Rand sont dans l'intervalle [0,1]. Un bon clustering correspond à des valeurs d'indice de Rand proches de 1. Cet indice nécessite l'existence d'une vérité terrain concernant l'appartenance d'objets aux classes.

Indice de Davies-Bouldin

Soit m_i le centre de cluster c_i pour $i = 1, ..., k$. La distance moyenne entre les objets $x_a \in c_i$ et le centre m_i de c_i, est défini par :

$$d_i = \frac{1}{|c_i|} \sum_{x_a \in c_i} d(x_a, m_i)$$

Une mesure de similarité R_{ij} entre deux clusters c_i et c_j est définie comme suit,

$$R_{ij} = R_{ji} = \frac{d_i + d_j}{d(m_i, m_j)}$$

Une petite valeur de R_{ij} correspond à un bon clustering. Considérons maintenant le pire des cas de R_{ij} pour chaque cluster c_i, cette valeur de R_{ij} est noté par R_i. Parmi tous les clusters c_j, nous cherchons la valeur maximale de R_{ij}. Ainsi, R_i est défini par :

$$R_i = max\{R_{ij} \mid j = 1, 2, ..., k; i \neq j\}$$

L'indice de Davies-Bouldin est défini comme la moyenne des R_i calculés pour tout les clusters.

$$DB = \frac{1}{k} \sum_{i=1}^{k} R_i$$

Des petites valeurs de DB indiquent la présence de clusters compacts et bien séparés.

Indice de Goodman-Kruskal

L'indice de Goodman-Kruskal [102] est calculé en considérant tout les quadruplets possibles (q, r, s, t), où q, r, s et t sont des objets. Soit $d(x, y)$ la fonction qui mesure la distance entre les deux objets x et y. Selon l'indice de Goodman-Kruskal, un quadruplet est considéré comme *concordant* si et seulement si les deux conditions suivantes sont satisfaites :

- $d(q, r) < d(s, t) \mid q$ et r appartiennent au même cluster ; s et t appartiennent aux deux clusters différents.
- $d(q, r) > d(s, t) \mid q$ et r appartiennent aux deux clusters différents ; s et t appartiennent au même cluster.

En contrepartie, un quadruplet est *non-concordant* si et seulement si les deux conditions suivantes sont satisfaites :

- $d(q, r) < d(s, t) \mid q$ et r appartiennent aux deux clusters différents ; s et t appartiennent au même cluster.
- $d(q, r) > d(s, t) \mid q$ et r appartiennent au même cluster, et s et t appartiennent aux deux clusters différents.

Intuitivement, les quadruplets concordants correspondent à un bon clustering, tandis que les quadruplets non-concordants impliquent une faible qualité de clustering. Soient Sc le nombre de quadruplets concordants et Sd le nombre de quadruplets non-concordants. Ainsi, l'indice de Goodman-Kruskal est défini comme suit :

$$GK = \frac{Sc - Sd}{Sc + Sd}$$

Il est clair qu'un bon clustering contient plusieurs quadruplets concordants et peu de quadruplets non-concordants. Ainsi, une grande valeur de l'indice de Goodman-Kruskal indique un bon clustering. Il est important de noter que certains quadruplets ne peuvent être ni concordants ni non-concordant.

Indice de *C*

L'indice de C est défini par Hubert et al. [115] comme suit :

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}$$

avec S est la somme des distances entre toutes les paires d'objets appartenant au même cluster, n est le nombre de ces paires. S_{min} est la somme des n plus petites distances entre toutes les paires d'objets. De même S_{max} est la somme des n plus grandes distances entre toutes les paires. Le C-indice est borné entre 0 et 1 ($C \in [0, 1]$). Une valeur de C proche de zéro indique un bon clustering.

Ces indices de validation de clustering ont pour objectif l'analyse de manière quantitative des résultats d'un algorithme de clustering. Ils sont aussi utilisés pour optimiser des paramètres de certains algorithme de clustering (i.e. le nombre de clusters pour l'algorithme des k-moyennes). La majorité de ces indices évalue le clustering en se basant sur la qualité de la séparation et la compacité de clusters. L'objectif est de fournir des clusters compacts et bien séparés.

Par ailleurs, nous distinguons un ensemble d'inconvénients relatifs à certains indices, par exemple, la sensibilité aux objets aberrants (ie. $Dunn$) et la complexité élevée (i.e. $Goodman$-$Kruskal$).

5.3.3 Graphe Median-Shift

Le *Mean-shift* [49] est une méthode d'estimation itérative non paramétrique des modes d'une densité de probabilité associée à une distribution de vecteurs (i.e. points). Elle offre une approche non-paramétrique de clustering sans aucune restriction sur la distribution des objets dans l'espace de caractéristiques. Dans la littérature, plusieurs méthodes ont utilisé le principe de translation (*shifting*) du *mean-shift* pour définir des algorithmes de clustering. Une variante intéressante du *mean-shift* est le *median-shift* [149, 148] où les vecteurs sont translatés vers le médian au lieu de la moyenne. En ce sens, nous proposons une adaptation de cette méthode dans le domaine des graphes.

Dans cette section, nous présentons l'algorithme du graphe *Median-Shift* [132]. Le graphe Median-shift est une adaptation de l'algorithme de $mean - shift$ pour le domaine des graphes. Notre algorithme consiste à translater chaque graphe vers le médian de son voisinage local. Nous utilisons la notion du graphe médian pour calculer le médian d'un ensemble de graphes.

L'adaptation pour le domaine des graphes

Notre contribution consiste à proposer un algorithme de clustering dans le domaine des graphes dans la même veine philosophique que le clustering par mean [49], médian [148, 149, 225] ou medoïd shift [231] dans l'espace vectoriel. Toutefois, dans le domaine des graphes, le calcul d'un représentant (e.g. médian) d'un ensemble de graphes ne peut pas être effectué avec la même facilité que dans l'espace vectoriel. Afin de calculer le médian d'un ensemble de graphes nous utilisons la notion du graphe médian. Dans le domaine des graphes, nous considérons que les régions denses correspondent aux clusters et les régions à faible densité correspondent aux limites des clusters. Ainsi, en utilisant le graphe médian, nous voudrions translater chaque graphe vers la densité maximale de sa plus proche région dense. Ceci est équivalent à déplacer chaque graphe vers le représentant local de son voisinage, dans notre cas le graphe médian. Ainsi, nous définissons le graphe médian-shift pour un graphe g dans un ensemble $\mathcal{E}$ par :

$$g_{medianshift} = median(\{g_i \in \mathcal{E} \mid d(g, g_i) < h\})$$

avec h le rayon qui est un paramètre de l'algorithme et $d(g, g_i)$ notre distance d'édition de graphes. Chaque graphe g_i est translaté vers le graphe médian de son voisinage élaborant ainsi une trajectoire $g_i(t)$ avec $t > 0$. Cette trajectoire commence par $g_i(0) = g_i$ en allant vers la densité maximale du voisinage considéré. Cette trajectoire se termine par un graphe dont la position est quasiment le centre du voisinage de g_i. Ce graphe est appelé le "graphe stationnaire". Le clustering consiste alors à regrouper tous les graphes qui convergent vers un même graphe stationnaire dans un cluster. Dans la figure 5.2, une trajectoire de translations est illustrée, les graphes en bleu correspondent aux graphes médians calculés dans la direction de la densité maximale du voisinage du graphe considéré et le graphe vert correspondant au graphe médian calculé à chaque itération. Enfin, le graphe stationnaire est illustré en rouge.

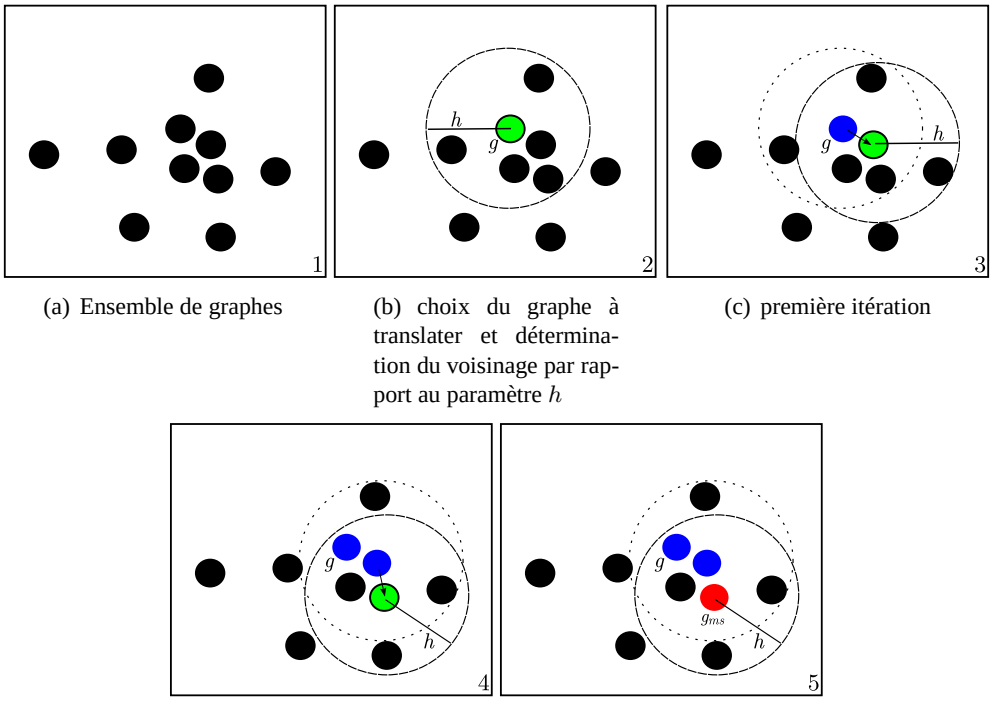
Algorithme

Algorithme 5 Pseudo-code de l'algorithme de graphe Median-Shift

ENTRÉES: Un ensemble de graphes, G=$\{g_1, \cdots, g_n\}$, et un rayon h
SORTIES: Un ensemble de clusters $\{C_j\}_{j=1}^{k}$

1: Affecter à chaque graphe $g_i \in$ G un graphe vide g_{ms_i}
2: **Pour** chaque $g_i \in$ G **Faire**
3: **Répéter**
4: Soit $G_i \subseteq$ G, avec $\forall g_k \in G_i$, d(g_i,g_k)<h
5: $g_m \leftarrow median(G_i)$ ▷ Calcul du graphe médian
6: Translater g_i vers g_m
7: **Jusqu'à** g_i converge vers un graphe median stationnaire
8: $g_{ms_i} \leftarrow g_m$
9: **Fin Pour**
10: Affecter les graphes ayant le même graphe médian stationnaire au même cluster C_j, avec $1 <j< k$ et *k* est le nombre de g_{ms_i} distincts.

Un pseudo-code de l'algorithme proposé est donné dans l'algorithme 5. Pour un ensemble de graphes G donné en entrée, l'algorithme fournit un ensemble de cluster $\{c_1, ..., c_k\}$, tel que $c_i \subseteq G$, $c_i \neq \emptyset$, $\forall i \neq j$ $c_i \cap c_j = \emptyset$, et $\bigcup_{i=1}^{k} c_i = G$. Le rayon h, appelé largeur de bande dans le Mean-Shift classique, est un paramètre de l'algorithme. Le graphe Median-shift procède comme suit. Premièrement, pour chaque graphe $g_i \in G$, un graphe vide g_{msi} est associé (ligne 1). Ensuite, la boucle intérieure (lignes 3-7) est appliquée à chaque graphe $g_i \in G$. Cette boucle consiste à calculer le graphe médian stationnaire g_{msi} pour un graphe g_i. Nous définissons le graphe médian stationnaire par le dernier graphe médian retourné après une série de translations (*shifting*). Concrètement, pour calculer le graphe stationnaire pour un graphe g_i, nous considérons uniquement un sous-ensemble $G_i \subseteq G$ centré sur g_i et avec un rayon h (ligne 4), i.e. $G_i = \{g \mid g \in G; d(g_i, g) \leq h\}$. Ensuite, g_i est déplacé vers le graphe médian du sous-ensemble G_i (ligne 6). Le graphe médian est calculé par la fonction $median()$ (ligne 5). Dans la figure 5.2 nous illustrons une exécution de la boucle "*répéter-jusqu'à*". Dans cet exemple, la convergence du graphe g vers le graphe stationnaire g_{ms} est effectuée en trois itérations. En effet, après chaque itération de la boucle de répétition, le sous-ensemble G_i devient plus compact



(a) Ensemble de graphes

(b) choix du graphe à translater et détermination du voisinage par rapport au paramètre h

(c) première itération

(d) deuxième itération

(e) troisième itération

FIGURE 5.2 – Illustration des itérations de *shifting*.

que l'itération précédente. Cela peut être expliqué par le fait le graphe médian pointe dans la direction de la densité maximale, parce qu'il minimise la somme des distance dans G_i (proche de tous les graphes). Cette propriété justifie ainsi la convergence de l'algorithme. La convergence vers un graphe stationnaire est montrée empiriquement. Ainsi, le graphe stationnaire g_{ms} peut être considéré comme le graphe de convergence d'un cluster. Les graphes dans le voisinage du g_{ms} consistent, exactement, aux graphes qui ont convergé par translation (*shifting*) à g_{ms}. Le résultat final est généré (ligne 10) comme suit : Le nombres de clusters correspond aux graphes stationnaires distincts. Deux graphes g_1 et g_2 sont considérés comme distincts si la distance qui les sépare est non nulle, i.e. $d(g_1, g_2) \neq 0$. Chaque graphe stationnaire g_{msi} est considéré comme le représentant d'un cluster C_i. Ensuite, chaque cluster C_j est composé par les graphes $g_i \in G$ qui convergent vers le graphe stationnaire g_{msj}.

5.3.4 Résultats expérimentaux : Classification non-supervisée

Protocole d'évaluation

Afin d'évaluer notre algorithme de clustering, nous utilisons les trois bases de graphes étiquetés utilisées dans l'expérimentation précédente (i.e Mutagenicity, Letter et GREC2).

Notre évaluation consiste à utiliser deux implémentations de la fonction du calcul du graphe médian $median()$ (Algorithme 5). La première implémentation considérée est le graphe médian

(noté SM) qui est un graphe appartenant à l'ensemble initial de graphes et minimise la somme des distances entre les graphes de cet ensemble. Dans l'algorithme 6, nous illustrons un pseudo-code qui implémente le calcul du graphe médian d'un ensemble de graphes.

Algorithme 6 Pseudo-code de l'algorithme de sélection du graphe médian

ENTRÉES: Un ensemble de graphes, G=$\{g_1, \cdots, g_n\}$
SORTIES: un graphe g_{SM} qui est le médian de l'ensemble G
1: **Pour** chaque $g_i \in$ G **Faire**
2: Calculer la somme des distances $SD_i = \sum_{g_k \neg g_i \in G} d(g_i, g_k)$
3: **Fin Pour**
4: Sélectionner le graphe g_{SM} ayant la somme des distances SD_{SM} minimale.

La deuxième implémentation de $median()$ (Algorithme 5) est celle du graphe médian généralisé (noté GM). Le problème de calcul de ce type de graphe médian est exponentiel par rapport au nombre de noeuds présents dans l'ensemble de graphes. Pour cela nous utilisons une approximation du problème basée sur le plongement de graphes dans un espace vectoriel $\mathbb{R}^n$. Cette méthode est proposé par Ferrer et al. dans [84]. Partant d'un ensemble de graphe G, les auteurs proposent d'utiliser le plongement de graphes $g_i \in G$ via la représentation des dissimilarités afin de produire un ensemble de vecteurs V qui plonge l'ensemble G dans $\mathbb{R}^n$. Ensuite, le vecteur médian v_m de l'ensemble V est calculé. Par la suite, les deux vecteurs $v_i \in V$ et $v_j \in V$ les plus proches de v_m sont sélectionnés et leur vecteur médian v'_m est alors calculé. La distance entre v_i et v'_m est utilisée comme le poids pour calculer le graphe moyen pondéré g_m entre g_i et g_j. Finalement, le graphe g_m est considéré comme l'approximation du graphe médian généralisé de l'ensemble G.

Par ailleurs, pour approfondir notre évaluation, nous évaluons aussi l'impact du rayon h sur les performances de l'algorithme proposé. Pour ce faire nous exécutons plusieurs fois chaque expérimentation avec des valeurs différentes de h qui varie entre la distance minimale et maximale dans chaque base de graphes. Nous adoptons trois indices de validation de clustering pour évaluer les partitions fournies par notre algorithme. Enfin, nous comparons nos résultats avec les résultats de l'algorithme de k-moyennes (k-means). Vu que l'algorithme des k-moyennes peut être appliqué aux graphes et aux vecteurs, nous comparons aussi les résultats des k-moyennes appliqués sur les vecteurs de plongement obtenues dans l'expérimentation précédente.

Résultats

Dans les figures 5.3, 5.4 et 5.5, nous présentons les résultats des indices de validation en fonction du rayon h pour chaque base de graphes. Chaque figure est constituée de six courbes réparties sur deux colonnes : la colonne de droite contient les trois courbes qui représentent les résultats des trois indices de validation en fonction du rayon h en utilisant le graphe médian pour implémenter la fonction $median()$. La colonne de gauche de chaque figure représente les trois courbes qui correspondent aux résultats des trois indices de validation en fonction du rayon h en utilisant le graphe médian généralisé pour implémenter la fonction $median()$.

Les résultats de l'indice de $Rand$ montrent qu'avec la plupart des valeurs testées de h le graphe médian fournit des résultats meilleurs que le graphe médian généralisé. Autrement dit, le

graphe Median-shift avec le graphe médian fournit des clusters plus similaires à la partition dans la vérité terrain que ceux fournis en utilisant le graphe médian généralisé. Ceci est nettement remarquable sur la base de Letter où le graphe médian fournit de très bons résultats. En effet, nous observons deux valeurs proches de 1 et parmi les huit valeurs quatre sont supérieures à 0.5.

Les résultats de l'indice de $Dunn$ montrent que le graphe médian généralisé est mieux que le graphe médian. Ceci implique qu'en utilisant le graphe médian généralisé les clusters fournis sont plus compacts et mieux séparés que les clusters fournis en utilisant le graphe médian.

Les résultats de l'indice de $Goodman\text{-}Kruskal$ montrent que les deux techniques utilisées pour calculer le graphe médian fournissent des résultats similaires, avec un succès du graphe médian généralisé pour quelques valeurs de h.

À partir de ces observations, nous pouvons conclure que notre algorithme de clustering est capable de fournir une partition significative d'un ensemble de graphes. Ceci est justifié par les bons résultats des indices de validation de clustering, notamment, l'indice de $Rand$. Cette observation est valable pour les deux méthodes de calcul de graphe médian utilisées. Par ailleurs les résultats d'indice de $Dunn$ se focalisent sur le fait que les clusters fournis sont compacts et bien séparés. Néanmoins, nous remarquons que les résultats de l'algorithme proposé dépendent du rayon h.

Dans la suite, nous considérons que la meilleure valeur du rayon h est celle qui maximise les trois indices. Formellement, le meilleur rayon h est calculé selon l'équation suivante :

$$meilleur_h = \underset{h}{argmax} \left\{ \frac{Dunn(h)}{max(Dunn)} + \frac{Rand(h)}{max(Rand)} + \frac{GK(h)}{max(GK)} \right\}$$

avec $Dunn(h)$, $Rand(h)$ et $GK(h)$ les valeurs de l'indice de $Rand$, l'indice de $Dunn$ et l'indice de $Goodman\text{-}Kruskal$, respectivement pour une valeur de h donnée. $max(Dunn)$, $max(Rand)$ et $max(GK)$ sont les valeurs maximales des indices pour toutes les valeurs de h testées, nous les utilisons juste pour normaliser nos calculs.

Une fois que nous sélectionnons la meilleure valeur de h pour chaque base, nous procédons à une comparaison de nos résultats avec l'algorithme des k-moyennes. Nous appliquons, dans un premier temps, l'algorithme des k-moyennes sur les graphes en utilisant son adaptation au domaine des graphes. Dans un deuxième temps, nous appliquons la version originale des k-moyennes sur les vecteurs de plongements résultant de l'expérimentation précédente.

Vu que la méthode des k-moyennes n'est pas déterministe (les résultats dépendent du choix des centres initiaux), nous l'exécutons 10 fois et nous prenons la moyenne de ces 10 exécutions de chaque indice de validation. La table 5.5 contient la meilleure valeur de h pour chaque base ainsi que les valeurs d'indice de validation effectuées par les k-moyennes et notre algorithme en utilisant le graphe médian et le graphe médian généralisé. Nous observons que pour l'indice de $Goodman\text{-}Kruskal$ notre algorithme surpasse les performances des k-moyennes. Le graphe Median-Shift fournit les meilleurs résultats des indices de $Rand$ et $Dunn$ sur la base Mutagenicity. L'algorithme des k-moyennes appliqué sur les vecteurs de plongement fournit les meilleurs résultats des indices de $Rand$ et $dunn$ sur la base GREC2. L'adaptation de l'algorithme des k-moyennes au domaine des graphes fournit les meilleurs résultats des indices de $Rand$ et $dunn$ sur la base Letter.

Pour résumer, nous pouvons considérer que l'algorithme de clustering proposé dans ce

chapitre est généralement meilleur que les k-moyennes vis-à-vis de la séparation entre les clusters, la compacité et la similarité avec la vérité terrain. Dans notre expérimentation, nous avons choisi de considérer la valeur de h qui optimise les trois indices de validation utilisés. Mais, on peut choisir une autre valeur de h qui n'optimise qu'un seul indice. Par exemple, si l'objectif est juste de produire une partition des graphes la plus similaire possible à la vérité terrain, h serait alors choisi en optimisant l'indice de $Rand$. De la même manière, si l'objectif de clustering est de minimiser la compacité des clusters et maximiser leur séparation, la valeur de h appropriée serait la valeur qui optimise l'indice de $Rand$. Notons que ce problème est très semblable au problème de choix de k pour l'algorithme des k-moyennes dans le cas où le nombre de classes dans la vérité de terrain est ignoré.

5.4 Conclusion

Dans la littérature plusieurs méthodes de classification (non-)supervisée de données robustes et efficaces existent dans le cadre de la reconnaissance de formes. Presque la totalité de ces méthodes est dédiée à la représentation vectorielle d'objets. Dans le contexte de la représentation sous forme de graphes, il y a un manque considérable de méthodes de classification. En effet, la classification supervisée de graphes est souvent limitée à l'utilisation du classificateur des plus proches voisins en utilisant une mesure de similarité entre les graphes et la classification non-supervisée est limitée à un nombre restreint d'algorithmes.

Dans ce chapitre, nous avons proposé deux contributions au problème de classification de graphes. La première contribution consiste en une nouvelle technique de plongement de graphes basée sur le *constant shift embedding*. La méthode du *constant shift embedding* a été développée originalement pour traiter les données dans un espace vectoriel. Elle ajoute aux dissimilarités dans l'espace d'origine une constante pour produire un ensemble de distances euclidiennes. Notre contribution consiste à généraliser cette méthode au domaine des graphes. Dans la partie expérimentale, nous avons pu remarquer que l'utilisation des SVM sur les vecteurs de plongement a amélioré les résultats obtenus par le classificateur des plus proches voisins dans le domaine des graphes. La deuxième contribution proposée dans ce chapitre consiste en un algorithme de clustering dans le domaine de graphe. Cette contribution est une adaptation de la méthode *mean-shift* au domaine des graphes. Nous avons testé deux méthodes connues pour implémenter l'opération de translation (*shifting*) au lieu de la moyenne utilisée par le *mean-shift* dans l'espace vectoriel. Notre algorithme, le graphe Median-Shift, est non-paramétrique et déterministe. Nous le considérons comme non-paramétrique parce que le nombre de clusters n'est pas défini en paramètre mais il est calculé durant l'exécution de l'algorithme. Dans la partie expérimentale, les résultats ont montré que l'algorithme de graphe Median-Shift est capable de produire une partition significative d'un ensemble de graphes initial.

Dans le chapitre suivant, nous proposons une méthode d'indexation dédiée aux bases d'images représentées sous forme de graphes.

			GREC2	Mutagenicity	Letter
$meilleur_h$	SM		1567.1	14.389	3.059
	GM		1044.7	9.592	0.764
Indices					
$Rand$	SM		0.373	**0.539**	0.544
	GM		0.346	0.5	0.232
	k-means	graphes	0.363	0.512	**0.924**
		vecteurs	**0.960**	0.510	0.900
$Dunn$	SM		0.008	0.047	0.04
	GM		0.008	**2.978**	0.03
	k-means	graphes	0.008	0.013	**0.052**
		vecteurs	**0.620**	1.350	0.028
GK	SM		0.239	0.306	0.189
	GM		**0.278**	**0.621**	**0.578**
	k-means	graphes	0.234	0.164	-0.236
		vecteurs	0.250	0.270	0.400

TABLE 5.5 – Comparaison de nos résultats avec l'algorithme des k-moyennes

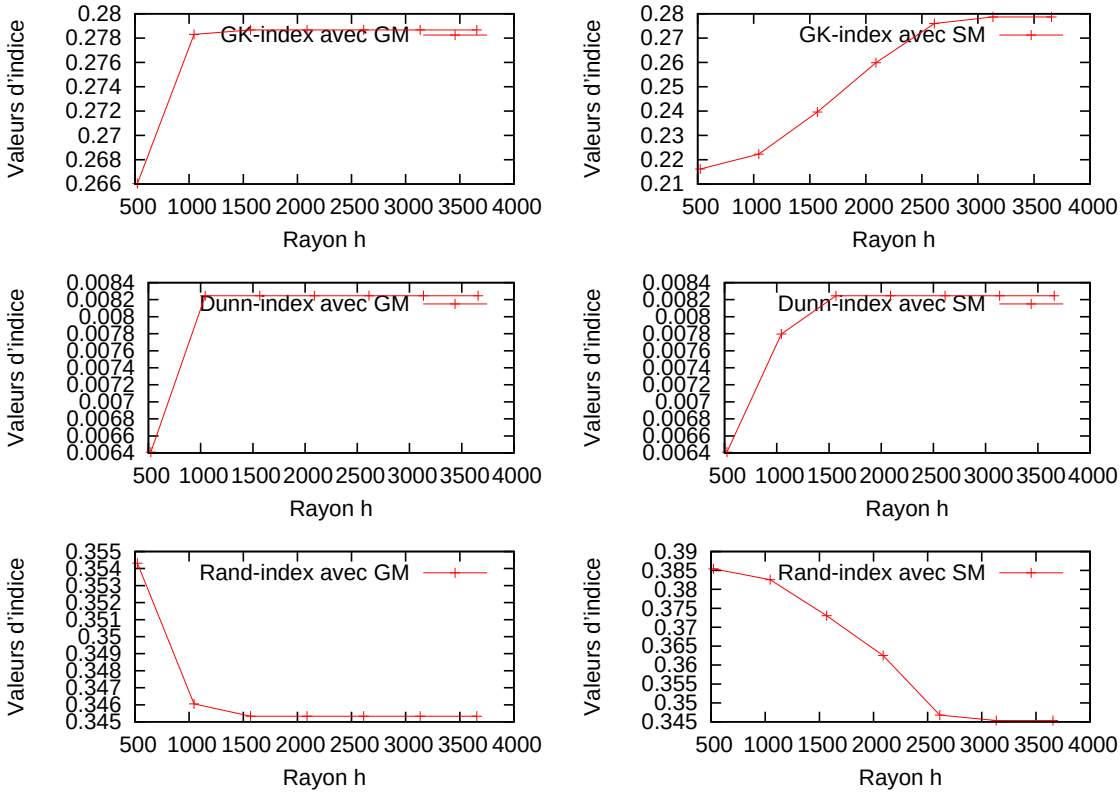


FIGURE 5.3 – Comportement des indices de validation en fonction du rayon h dans la base de Graphe GREC2

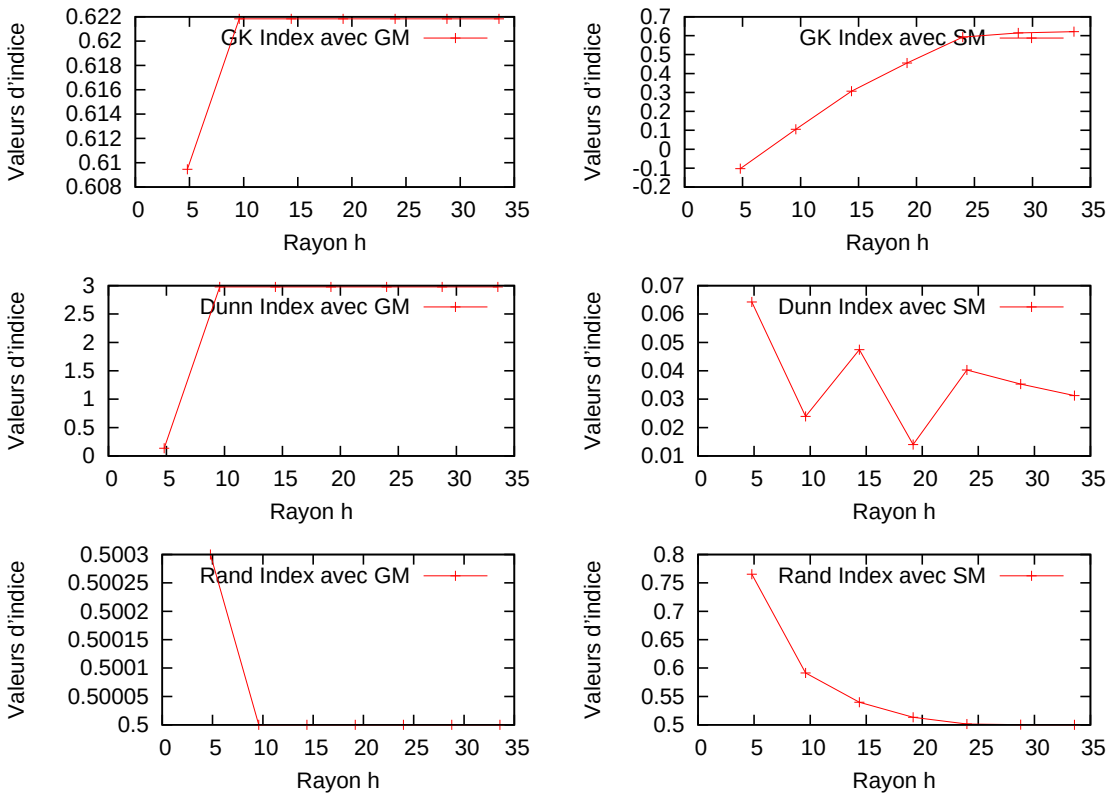


FIGURE 5.4 – Comportement des indices de validation en fonction du rayon h dans la base de Graphe Mutagenicity

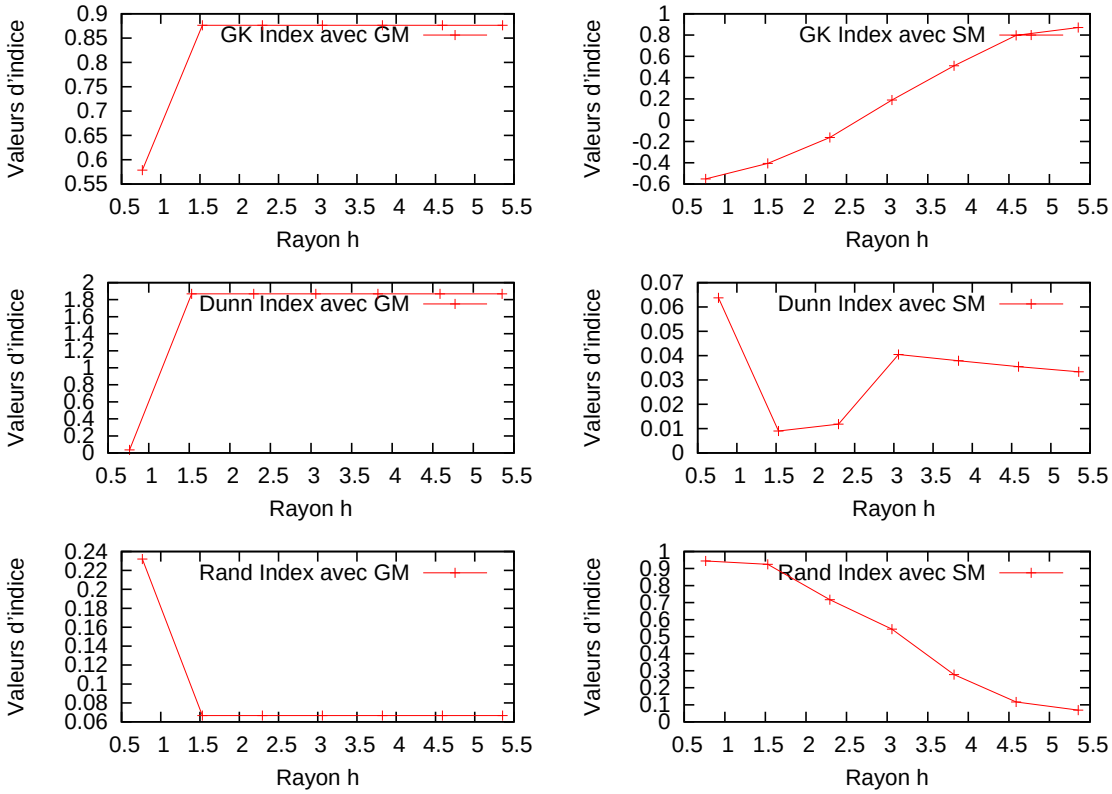


FIGURE 5.5 – Comportement des indices de validation en fonction du rayon h dans la base de Graphe Letter

Chapitre 6

Indexation d'images à base des graphes

A picture is worth a thousand words. An interface is worth a thousand pictures.

Ben Shneiderman

Sommaire

6.1 Introduction

Généralement, un système de recherche d'images (SRI) se compose en deux parties. La première partie consiste à décrire l'image. Cette description peut être textuelle, dans ce cas l'image est associée à un ensemble de mots (annotations) qui la décrivent. Ces techniques sont largement utilisées dans les SRI sur internet (e.g. google images [20]). Dans ces systèmes, les annotations décrivant l'image sont potentiellement extraites automatiquement de la page web qui la contient, tels que le nom du fichier de l'image, le titre de la page, etc. Ces annotations ne reflètent pas toujours le contenu d'une image, d'où le recours à une annotation manuelle des images. Vu l'augmentation vertigineuse de nombre d'images disponibles, une telle annotation

20. `http://images.google.fr/`

s'avère très coûteuse. Pour surmonter ce problème, certains SRI utilisent une description visuelle de l'image. Il s'agit d'utiliser des techniques de reconnaissance de formes pour extraire les caractéristiques importantes qui seront utilisées pour une éventuelle recherche de l'image concernée. Les systèmes qui utilisent la description visuelle sont dits systèmes de recherche d'images par le contenu (en anglais *content based image retrieval* (CBIR)). Durant la dernière décennie, plusieurs systèmes de recherche d'images par le contenu ont été proposés [63, 238].

La deuxième partie d'un SRI est l'indexation des descripteurs. L'indexation consiste à organiser les descripteurs d'images pour garantir l'accès le plus rapidement possible aux images recherchées. Cette partie est cruciale dans tout système de recherche d'information, particulièrement la recherche d'images. En effet, l'indexation évite la recherche séquentielle des informations dans une base d'images par l'accès directe au bloc (un ensemble réduit) contenant les images les plus similaires à l'image requête. Plusieurs méthodes d'indexation sont utilisées d'une manière robuste et performante pour la recherche d'images (e.g. arbre-B, arbre-B+, arbre-kd) [17, 51, 211].

Presque tous les systèmes de recherche d'images utilisent les vecteurs comme forme de modélisation d'images. Le choix des vecteurs est influencé par leur facilité de manipulation (i.e. calcul des distances) et par la possibilité de navigation dans l'espace vectoriel. En effet, certaines méthodes d'indexation utilisent le principe de partition d'espace euclidien pour indexer les vecteurs représentant les images.

Dans les chapitres précédents de cette thèse, nous avons bien montré l'intérêt de l'utilisation des graphes pour représenter les images. En effet, Les graphes sont beaucoup plus puissants en termes de représentation et plus flexibles que les vecteurs caractéristiques qui ne fournissent aucune possibilité directe pour décrire les relations structurelles des objets en considération. En outre, alors que la taille d'un graphe peut être ajustée à la taille de l'objet, un vecteur est limité à une taille prédéfinie, qui doit être préservée pour tous les objets rencontrés dans une application particulière. Donc, il est intéressant de développer un système de recherche d'images représenté sous forme de graphes. Un tel système aura aussi deux parties : la première consiste à extraire les graphes représentant les images, tandis que la deuxième partie consiste à indexer ces graphes. La première partie est déjà bien élaborée dans la littérature (voir chapitre 2). Par contre, très peu de travaux ont porté sur l'indexation des graphes pour les systèmes de recherche d'images. La plupart des travaux sont développés dans un cadre d'indexation des structures des molécules chimiques [242]. Parmi les autres travaux plus générique, nous distinguons le *GraphGrep* [98] et le *gIndex* [280]. Ces deux méthodes sont basées sur l'utilisation d'une sous-structure de chaque graphe comme son entrée d'index (*index features*). Le bon fonctionnement de ces travaux nécessite que les graphes soient étiquetés par des valeurs discrètes. Ceci est rare pour les graphes représentant les images où les étiquettes sont généralement des valeurs continues qui quantifient des caractéristiques locales d'une image.

Dans ce chapitre, nous proposons une méthode originale pour l'indexation des graphes. Outre l'indexation, notre méthode permet aussi la navigation dans une base d'images représentées sous forme des graphes. L'idée principale de cette contribution est d'organiser l'espace (domaine) de graphes en une structure d'hypergraphe. Dans cette structure d'hypergraphe, chaque sommet correspond à un graphe de la base et chaque hyperarête correspond à un ensemble de graphes similaires. Les entrées de notre index sont les graphes centroïdes des hyperarêtes. Ainsi,

nous définissons une technique de recherche d'images à base de cette structure. Ensuite, une technique de navigation dans la base est introduite en traversant l'hypergraphe d'une hyperarête à une autre.

6.2 La notion d'hypergraphe

Les hypergraphes sont des objets mathématiques généralisant la notion de graphes où les arêtes peuvent connecter n'importe quel nombre de sommets. L'hypergraphe a été défini par Berge [19] comme suit, soit H = (ϑ,ξ) un hypergraphe, où :

- ϑ={x_1,x_2, x_3 ... , x_n} : est un ensemble fini de sommets
- ξ={E_1, E_2, E_3 ... , E_m} : est une famille de sous-ensembles de ϑ.
- $E_j \neq \emptyset$,$\bigcup_{j=1,...,m} E_j$=ϑ.

ϑ est appelé l'ensemble des sommets, ξ est l'ensemble des hyperacs et $|\vartheta|$ est la cardinalité de H. Dans la figure 6.1(a), une hyperarête E_i est représentée par une ligne entourant ses sommets si $|E_i| > 2$ (par exemple E_1 dans la figure 6.1(a)), par une boucle sur l'élément si $|E_i|$=1(E_4 dans figure 6.1(a)), et par une ligne reliant les deux éléments si $|E_i|$=2 (E_5 dans figure 6.1(a)). Si $|E_i|$=2 pour tous les *i*, l'hypergraphe devient un graphe non orienté ordinaire.

Dans un hypergraphe, deux sommets x_i et x_j sont dits adjacents s'il existe une hyperarête E_k, qui contient les deux sommets ($x_i \in E_k$, $x_j \in E_k$). Deux hyperarêtes E_i et E_j sont dites adjacentes si leur intersection n'est pas vide. Tout hypergraphe a une matrice d'incidence A_i^j de taille mxn avec m colonnes représentant les hyperarêtes et les n lignes représentant les sommets. Les éléments dans A indiquent l'appartenance de sommets aux hyperarêtes comme suit :

$$A_i^j = \begin{cases} 1 & \text{if} \quad x_i \in E_j \\ 0 & \text{if} \quad x_i \notin E_j \end{cases}$$

Par exemple, considerons l'hypergraphe H=(ϑ, ξ) dans la figure 6.1, ϑ={x_1,x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}} et ξ={ E_1, E_2, E_3, E_4, E_5, E_6}. La cardinalité de H est $|\vartheta|$= 13, et la

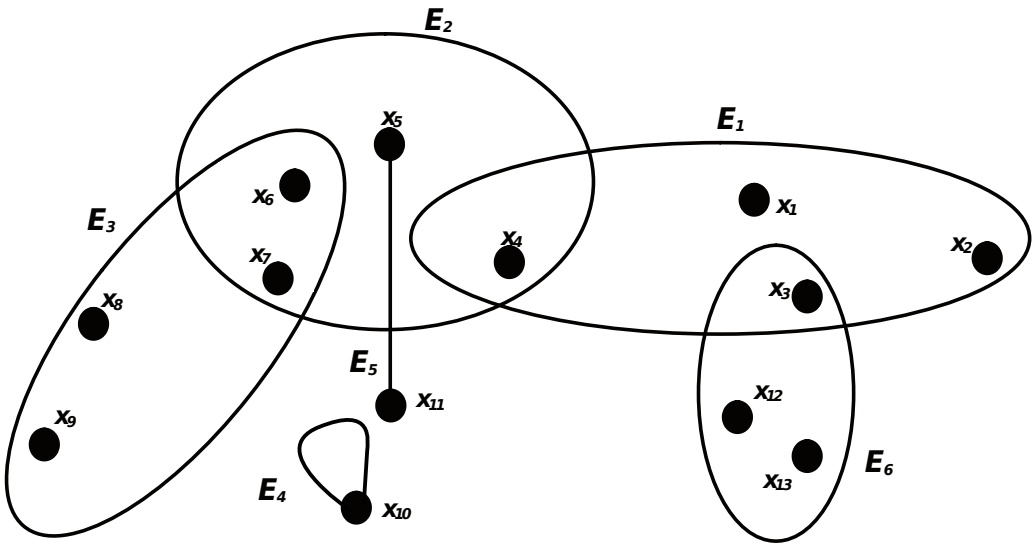


FIGURE 6.1 – Exemple d'un hypergraphe

matrice d'incidence est définie comme suit :

	E_1	E_2	E_3	E_4	E_5	E_6
x_1	1	0	0	0	0	0
x_2	1	0	0	0	0	0
x_3	1	0	0	0	0	1
x_4	1	1	0	0	0	0
x_5	0	1	0	0	1	0
x_6	0	1	1	0	0	0
x_7	0	1	1	0	0	0
x_8	0	0	1	0	0	0
x_9	0	0	1	0	0	0
x_{10}	0	0	0	1	0	0
x_{11}	0	0	0	0	1	0
x_{12}	0	0	0	0	0	1
x_{13}	0	0	0	0	0	1

Le *degré* d'un sommet, noté Δ_ϑ, est le nombre d'hyperarêtes auxquelles il appartient, et le degré d'une hyperarête, noté Δ_ξ (*h*), est le nombre de sommets qu'elle contient.

Nous pouvons noter que la différence entre une arête dans un graphe et une hyperarête dans un hypergraphe est que le premier est toujours un sous-ensemble d'un ou deux sommets, et dans

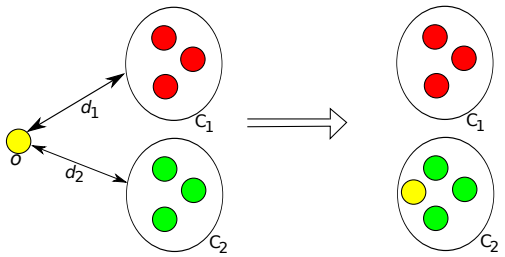


FIGURE 6.2 – Illustration d'une classification d'un objet

le second, le sous-ensemble de sommets peut être de cardinalité arbitraire.

6.3 Modélisation par hypergraphe

Il est important d'organiser les graphes en ensembles cohérents afin de faciliter leur indexation ultérieure. Une telle organisation peut s'effectuer avec une technique de classification non-supervisée. Cependant, l'utilisation de la classification à des fins d'indexation exige des modifications dans la stratégie des classificateurs. En effet, dans les approches classiques de classification (non-)supervisées, un objet o est toujours affecté à **une et une seule** classe c. Cette affection résulte d'un ensemble de mesures qui jugent que l'objet o est plus similaire aux objets de la classe c que les objets des autres classes. Évidemment, cette similarité est basée sur l'ensemble des caractéristiques de chaque objet. D'une manière générale, pour classifier un objet dans une classe parmi k classes, d'abord les k distances entre l'objet et les k classes sont calculées, ensuite l'objet est affecté à la classe ayant la distance minimale. Cette stratégie est conservée même si les différences entre ces distances sont très faibles. La figure 6.2 illustre une classification de l'objet o dans la classe C_2. Cette affectation est établie par le fait que la distance d_2 séparant o de C_2 est inférieure à la distance d_1 qui sépare o de C_1. Dans cette illustration, nous constatons que les deux distances d_1 et d_2 sont très similaires, pourtant il fallait affecter l'objet dans C_2. Dans le cas où les objets sont des graphes, nous considérons que cette stratégie peut contraindre l'indexation. Étant donnée une base d'objets où l'indexation est basée sur l'ensemble de classes $C = \{c1, ..., c_n\}$ résultant d'une classification classique. La recherche d'objets similaires à un objet requête o_r entraine l'accès direct à la classe c_i la plus proche de o_r. Ainsi, la récupération de l'ensemble d'objets similaires à o_r se limite aux objets appartenant à c_i, i.e, toutes les autres classes sont omises. Par contre, il est fort probable que des objets n'appartenant pas à c_i soient similaires à o_r.

Dans la suite de ce chapitre , nous proposons une solution à ce problème en se basant sur la structure de l'hypergraphe. L'idée est de représenter un ensemble de graphes en un hypergraphe où les sommets correspondent aux graphes et les hyperarêtes aux ensembles cohérents (clusters).

Ainsi, un graphe peut appartenir à plusieurs hyperarêtes (clusters) simultanément. Chaque graphe G_i dans la structure proposée est affectée à $\Delta_\vartheta(G_i)$ clusters et chaque cluster C_j contient $\Delta_\xi(C_j)$ graphes. Cette modélisation étend le principe classique d'affectation unique en une stratégie de multi-affectation d'un graphe aux clusters. Cette multi-affectation permet d'attacher un graphe à autant de clusters que nécessaire. Ceci évite alors l'obligation d'établir un choix

strict d'un et un seul cluster auquel le graphe sera affecté, et permet de l'affecter dans tous les clusters similaires.

Cependant, deux problèmes clés se posent dans la structuration d'un ensemble de graphes avec un modèle d'hypergraphe : la détermination du nombre de clusters (hyperarêtes) et la détermination de graphes connexes (graphes similaires) qui peuvent être regroupés dans une hyperarête. Dans cette perspective, nous considérons que le nombre d'hyperarêtes est égal à la taille d'un ensemble représentatif, défini sur une sélection de graphes les plus représentatifs de l'ensemble. On note chaque graphe sélectionné comme le centroïde d'une hyperarête. La sélection de ces graphes est similaire au problème de la sélection de prototypes [11, 204, 240]. Riesen et al. [204] énumèrent quelques techniques de sélection de prototypes à partir d'un ensemble d'apprentissage. Ces techniques nécessitent de préciser le nombre de prototypes et il n'y a pas de règles pour déterminer automatiquement ce nombre. Par conséquent, si nous nous situons dans un contexte non supervisé, où aucune information sur le nombre de graphes représentatifs n'est disponible, ce nombre ne sera déterminé que d'une manière empirique. Dans cette perspective, Spath [240] propose un algorithme utilisant les "leaders" et un seuil où le nombre de prototypes sélectionnés est inversement proportionnel à la valeur du seuil sélectionné. Cependant, l'algorithme de Leader [240] est sensible à la sélection du premier prototype choisi au hasard parmi les données en entrée. Pour surmonter ce problème, nous introduisons une technique de sélection de graphes représentatifs (centroïdes des hyperarêtes) fondée sur une stratégie de pelure d'oignon. Cette méthode peut être considérée comme une amélioration de l'algorithme du Leader et de l'algorithme de *K*-Centres [204]. Après la sélection des centroïdes des hyperarêtes, nous définissons la structure d'hypergraphe en attribuant chaque graphe aux hyperarêtes correspondants.

6.3.1 Sélection de centroïdes des hyperarêtes.

Comme indiqué ci-dessus, la sélection des centroïdes des hyperarêtes est similaire au problème de sélection de prototypes. Par conséquent, notre objectif est de sélectionner un sous-ensemble de graphes qui captent les aspects les plus significatifs d'un ensemble de graphes. Nous introduisons une amélioration de l'algorithme du Leader [240]. L'algorithme proposé est présenté dans l'algorithme 7. Cet algorithme est divisé en quatre étapes :

1. Sélectionner le graphe médian G_m des graphes non affectés de l'ensemble initial des graphes *S*. Puis choisir le graphe le plus distant G_{p_k}, en fonction d'une mesure de distance choisie, (qui n'a pas été préalablement affecté) de G_m, et affecter ce graphe au cluster C_k comme centroïde. Dans la première itération, le graphe G_{p_k} est le prototype sélectionné initialement.
2. Comparer les distances de tous les graphes non affectés $g_i \in S \setminus \{G_{p_k}\}$ à celle du dernier prototype sélectionné G_{p_k}. Si les distances $d(g_i, G_{p_k})$ et $d(g_i, g_{j \in C_k})$ sont inférieures à un seuil prédéfini *T* que nous définissons de manière automatique en fonction de nos données, le graphe g_i est affecté au cluster C_k avec le centroïde G_{p_k} et g_i est étiquetté et affecté.
3. Recalculer le graphe médian G_{m_k} de C_k, si $G_{m_k} \neq G_{p_k}$, remplacer G_{p_k} par G_{m_k}. Si aucun remplacement n'est fait ($G_{m_k} = G_{p_k}$), passer à l'étape suivante, sinon tous $g_j \in C_k$ sont

marqués comme non affectés, puis revenir à l'étape 2.

4. Si *S* contient encore des graphes non-affectés revenir à l'étape 1, sinon arrêter.

Algorithme 7 Algorithme de sélection des graphes centroïdes

ENTRÉES: Un ensemble de graphes *S*=$\{g_1, ..., g_n\}$, un seuil T
SORTIES: Un ensemble de graphes prototypes $\mathcal{G}_p = \{G_{p_1}, ..., G_{p_m}\}$

Répéter
  Étape 1
  $G_m \leftarrow$ *median*(*S*)
  $G_{p_k} = \max_{g \in S} d(g, g_m)$
  Créer le cluster C_k avec le graphe centroïde G_{p_k}
  Ajouter le graphe G_{p_k} à l'ensemble $\mathcal{G}_p$
  Étape 2
  Pour $g_i \in S \setminus \{G_{p_k}\}$ **Faire**
    Pour $\forall g_j \in C_k$ **Faire**
      Si $d(g_i, G_{p_k})$ et $d(g_i, g_j) < T$ **Alors**
        Ajouter le graphe g_i au cluster C_k
        $S \leftarrow S \setminus g_i$
      Fin Si
    Fin Pour
  Fin Pour
  $G_{m_k} \leftarrow$ *median*(C_k)
  Si $G_{m_k} \neq G_{p_k}$ **Alors**
    Remplacer G_{p_k} par G_{m_k}
    $S \leftarrow S \cup C_k$
    Aller à **Étape 2**
  Fin Si
  Aller à **Étape 1**
Jusqu'à $S \neq \emptyset$

Une première amélioration consiste en une adaptation de l'algorithme du Leader dans l'espace de graphes en utilisant la notion de graphe médian. Ensuite, une nouvelle méthode de sélection du premier prototype a été mise au point. Dans l'algorithme du Leader, le choix du prototype initial est fait par hasard ce qui influe sur le résultat final du clustering (i.e. à chaque exécution les résultats changent). Nous avons choisi d'utiliser comme prototype initial de l'algorithme le graphe le plus éloigné du médian de la base. Ce choix garantit un certain déterminisme de l'algorithme. Une deuxième amélioration consiste à ajouter une méthode itérative de sélection de prototype. En fait, comme il est indiqué dans le pseudo-code de l'algorithme, dès que le prototype initial est sélectionné, nous considérons tous les graphes qui ont une distance au prototype inférieure ou égal à un seuil donné et la distance entre chaque couple de graphes de ce sous-ensemble doit être aussi inférieure ou égale au même seuil. En revanche, dans l'algorithme du Leader, on ne considère que les distances entre le prototype et chaque objet (il n'y a

pas de critère sur les distances entre les objets). Une fois le sous-ensemble construit, on recalcule le nouveau graphe médian et on itère jusqu'à convergence vers un graphe médian prototype considéré comme un centroïde.

Compte tenu d'un seuil *T*, l'algorithme ci-dessus regroupe l'ensemble des graphes avec une inertie intra-classe (I_i) inférieure ou égale à *T*. Cette propriété est conservée dans l'étape 2. De plus, cet algorithme garantit : 1) la sélection des prototypes qui sont les centroïdes des clusters ; 2) une certaine séparabilité entre classes d'une partition. En outre, en fixant le prototype initial comme le graphe le plus distant du graphe médian dans l'ensemble des graphes, l'algorithme produit des résultats identiques et est déterministe.

Par ailleurs, les distances entre les prototypes sélectionnés sont strictement supérieures au seuil *T*. En effet, pour tout prototype p sélectionné l'algorithme considère que tous les graphes g_i ayant une distance $d(g_i, p) < T$ comme les éléments du cluster de centre p. Ces graphes g_i ne sont pas considérés pour la sélection de prochains prototypes. Ainsi, les autres prototypes p_j sont sélectionnés à partir des graphes g_j tel que $d(g_j, p) \geq T$.

6.3.2 La représentation sous forme d'hypergraphe

Soit *S* un ensemble de graphes et *P* l'ensemble de prototypes sélectionnés *P* (*P*⊂*S*). Les techniques de clustering classiques cherchent pour chaque graphe *g*∈ $S \setminus P$ son plus proche voisin $p_i \in P$ et ajoutent le graphe dans le cluster C_i correspondant au prototype p_i. En fait, si un graphe *g* présente une distance similaire à deux prototypes p_i et p_j, g est ajouté au cluster avec le prototype le plus proche même si la différence entre les deux distances est très mineure. En outre, les clusters sont disjoints et peuvent être exploités pour une tâche de recherche [205, 221, 224, 227], mais il sera difficile de trouver un algorithme pour naviguer dans l'ensemble de la base de graphes.

Au contraire, nous proposons un modèle basé sur l'hypergraphe qui permet le chevauchement des clusters. Désormais, les clusters seront considérés comme des hyperarêtes d'hypergraphe et les graphes comme les sommets. Tout d'abord, pour chaque prototype p_i une hyperarête h_i est définie avec un centroïde p_i. Deuxièmement, chaque hyperarête est définie comme suit : chaque graphe *g*∈ $S \setminus P$ est ajouté aux hyperarêtes ayant les centroïdes proches de *g* (leur distance à *g* est inférieure au seuil *T* utilisé dans l'algorithme précédent). À partir de cette définition, nous concluons le théorème suivant :

Théorème : *Soient la distance métrique de graphes $d(.,.)$ et l'hypergraphe $\mathcal{H}$ généré avec un seuil T. Si un graphe g est partagé par deux hyperarêtes $h_i \in \mathcal{H}$ et $h_j \in \mathcal{H}$ ayant respectivement les centroïdes p_i et p_j, alors $T < d(p_i, p_j) < 2 * T$.*

Preuve : Soit *P* l'ensemble de prototypes sélectionnés par l'algorithme 7, donc

$$\forall p_i, p_j \in P, d(p_i, p_j) > T \ (1)$$

Vu que la distance entre les graphes est une métrique, alors selon la propriété de l'inégalité triangulaire

$$d(p_i, p_j) \leq d(p_i, g) + d(g, p_j) \ (2)$$

On a $g \in h_i$ et $g \in h_j$, donc

$$d(p_i, g) < T\ (3)\ et\ d(g, p_j) < T\ (4)$$

D'autres parts, il est évident que le seuil T et les distances sont positives. Donc, à partir de (3) et (4),

$$d(p_i, g) + d(g, p_j) < 2 * T$$

Ce qui signifie (selon (2)) que,

$$d(p_i, p_j) < 2 * T\ (5)$$

Finalement, en combinant (1) et (5), nous avons

$$T < d(p_i, p_j) < 2 * T$$

□.

Ce théorème montre que, dans le cas où la distance de graphes utilisée est une métrique, le chevauchement est réalisé uniquement entre les hyperarêtes voisins. Particulièrement, les hyperarêtes, qui partagent des graphes en commun, possèdent des centroïdes proches. Cette propriété montre ainsi l'aspect sémantique que nous cherchons à établir par notre méthode. Cet aspect se résume dans la formule suivante :"*Dès qu'un graphe partage des informations avec deux (ou plusieurs) clusters proches, il doit être affecté à tous ces clusters*". Évidemment, ce théorème n'est pas toujours valide dans le cas où la distance de graphes n'est pas une métrique.

La figure 6.3 illustre notre motivation. Soit d_i = d(p_i, g_1), nous supposons que d_1 et d_2 sont inférieures ou égales à un seuil *T*, alors le graphe g_1 partage quelques informations avec p_1 et p_2 (les informations sont illustrées en couleur). Avec le modèle d'hypergraphe nous serons en mesure d'affecter g_1 à la fois aux hyperarêtes h_1 et h_2. La partie la plus à droite de la figure 6.3 décrit comment deux hyperarêtes (clusters) peuvent se chevaucher avec un graphe en commun. Ici, $\Delta_\vartheta(g_1)$=2 et $\Delta_\xi(h_1)$=$\Delta_\xi(h_2)$=2.

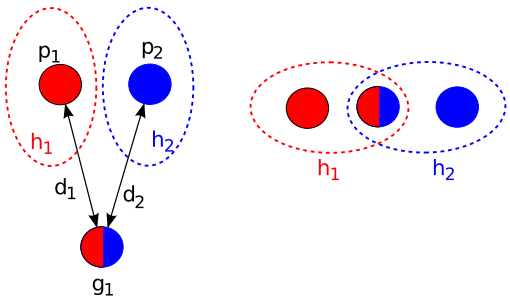


FIGURE 6.3 – Illustration du modèle proposé

Une fois que toutes les hyperarêtes sont définies à partir des graphes, on recalcule, pour chaque hyperarête, le graphe médian qui sera le nouveau centroïde de l'hyperarête. L'objectif de cette étape est de mettre à jour le centroïde de l'hyperarête et de maintenir autant d'informations que possible des graphes dans l'hyperarête correspondant. Nous avons utilisé le graphe médian

pour définir le centroïde d'un cluster car contrairement au graphe médian généralisé et au supergraphe minimum et commun [31] il est moins coûteux en temps de calcul. Nous utilisons aussi cette technique lors de l'ajout de nouveaux graphes dans la base. Concrètement, chaque nouveau graphe g_n est ajouté dans les hyperarêtes ayant des centroïdes proches de g_n (leur distance à g_n est inférieure au seuil *T* utilisé précédemment). Ensuite, nous changeons le centroïde de chaque hyperarête concernée par son nouveau graphe médian. Les hyperarêtes concernées par la mise à jour de centroïde sont celles où le graphe g_n est ajouté.

6.3.3 Recherche d'images

Smeulders et al. [238] distinguent trois paradigmes dans la recherche d'images selon les objectifs de l'utilisateur :

- Recherche *associative* : Les utilisateurs de la recherche associative n'ont pas de but véritablement défini. L'objectif de ce paradigme de recherche d'images consiste à aider les utilisateurs à explorer la base d'images. La recherche associative implique souvent le raffinement, via des interactions, de l'ensemble des images présenté initialement à l'utilisateur.
- Recherche de *cibles* : Ce paradigme concerne les utilisateurs qui ont pour but la recherche d'une image spécifique, par exemple la recherche de l'image d'un objet bien précis dans une collection de musée.
- Recherche de *catégories* : Ce paradigme consiste à rechercher le maximum d'images appartenant à une classe spécifique définie par l'utilisateur. Il peut correspondre au cas où l'utilisateur a un exemple (image) et cherche d'autres images de la même classe. Les catégories peuvent être dérivées à partir d'étiquettes ou à partir de la base de données en utilisant des mesures de similarité adéquates.

En ce qui concerne les graphes, l'interrogation d'un ensemble de graphes consiste, généralement, à rechercher les graphes les plus semblables à une requête donnée. Cette tâche de recherche trie les distances entre la requête et les graphes de la base dans un ordre croissant. Ce type de recherche correspond au second paradigme de Smeulders et al. lorsque l'utilisateur a pour objectif la recherche d'une image précise (représentée en graphe). Dans ce cas, l'image recherchée correspond à l'image ayant le graphe le plus proche de la requête. Par ailleurs, l'interrogation classique d'un ensemble de graphes correspond aussi au troisième paradigme de Smeulders et al.. Ceci est le cas quand l'utilisateur a pour but la recherche des images (représentées en graphes) les plus similaires à sa requête. Néanmoins, l'omniprésence de bruit dans les images ne favorise pas l'utilisation de l'interrogation classique des graphes pour la recherche de catégories. En effet, le bruit dans les images entraîne des transformations dans les graphes correspondants ce qui occasionne souvent des résultats faux positifs. Dans la figure 6.4, nous illustrons un exemple des faux positifs résultant d'une recherche d'images représentées sous forme de graphes. Dans cet exemple l'image requête appartient à la classe *"Glass"* de la base Shape. Parmi les 10 images retrouvées par la recherche classique, six images uniquement appartiennent à la classe *"Glass"*. Les quatre autres images constituent des résultats faux positifs parce qu'elles n'appartiennent pas à la classe *"Glass"*. Ceci est dû au bruit dans les images de la base Shape qui influe sur la similarité entre les graphes des images retrouvées et le graphe de l'image requête. Dans la littérature, certains auteurs [205, 221, 224, 227] considèrent ce problème comme une conséquence

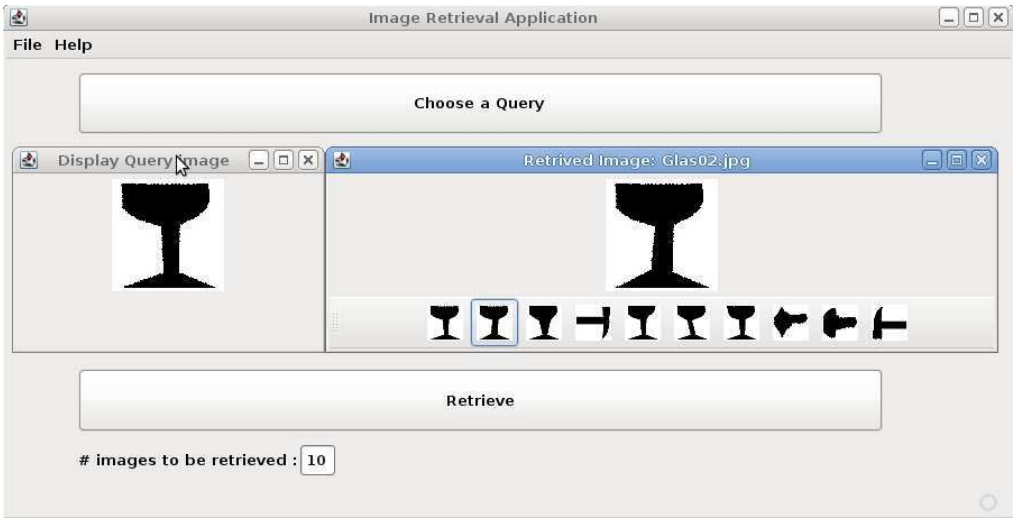


FIGURE 6.4 – Une interface graphique de recherche classique d'images représentées sous forme de graphe

du manque d'une exploitation efficace de la distribution des distances entre les graphes. En effet, afin d'améliorer les performances de la recherche de *catégories*, quelques travaux (e.g. [205]) proposent des techniques de recherche basées sur le clustering de graphes pour bien exploiter la distribution de distances. Dans ces travaux, les images similaires à la requête sont recherchées dans le cluster le plus proche de l'image requête au lieu de les rechercher dans toute la base. Ceci nécessite alors une phase préalable de clustering de la base d'images à interroger.

Dans la même veine, nous introduisons une procédure qui implique le modèle de l'hypergraphe proposé précédemment. L'idée principale est de trouver le centroïde (parmi tous les centroïdes des hyperarêtes) le plus proche d'une requête donnée. Puis, nous recherchons les graphes les plus similaires au sein de l'hyperarête. Nous décrivons la procédure de recherche dans le modèle de l'hypergraphe comme suit :

1. Pour un graphe de requête g_q, calculer l'ensemble des distances entre g_q et le centroïde de chaque hyperarête.
2. Initialiser à vide l'ensemble $\mathcal{R}$ de graphes à retrouver, i.e. $\mathcal{R} = \emptyset$.
3. Déterminer le plus proche centroïde p_i à g_q. Soit h_i l'hyperarête avec le centroïde p_i.
4. Récupérer l'ensemble $\mathcal{R} = \mathcal{R} \cup \{g_j\}$ de graphes les plus similaires à g_q, où $g_j \in h_i$.
5. Si le nombre de graphes retrouvés n'est pas suffisant ou si le graphe recherché n'est pas retrouvé, aller à l'étape suivante. Sinon, fin de l'algorithme.
6. Déterminer le centroïde suivant p_j le plus proche à g_q et h_j l'hyperarête avec le centroïde p_j. Retourner à l'étape 3 en passant à l'hyperarête suivante (h_j), i.e. $h_i \leftarrow h_j$.

La figure 6.5 présente une illustration de cette méthode de recherche. En effet, pour une requête donnée (Figure 6.5(a)), le plus proche centroïde est repéré (Figure 6.5(b) et 6.5(c)). Ensuite, la recherche s'effectue dans l'hyperarête contenant ce centroïde (Figure 6.5(d)). Dans

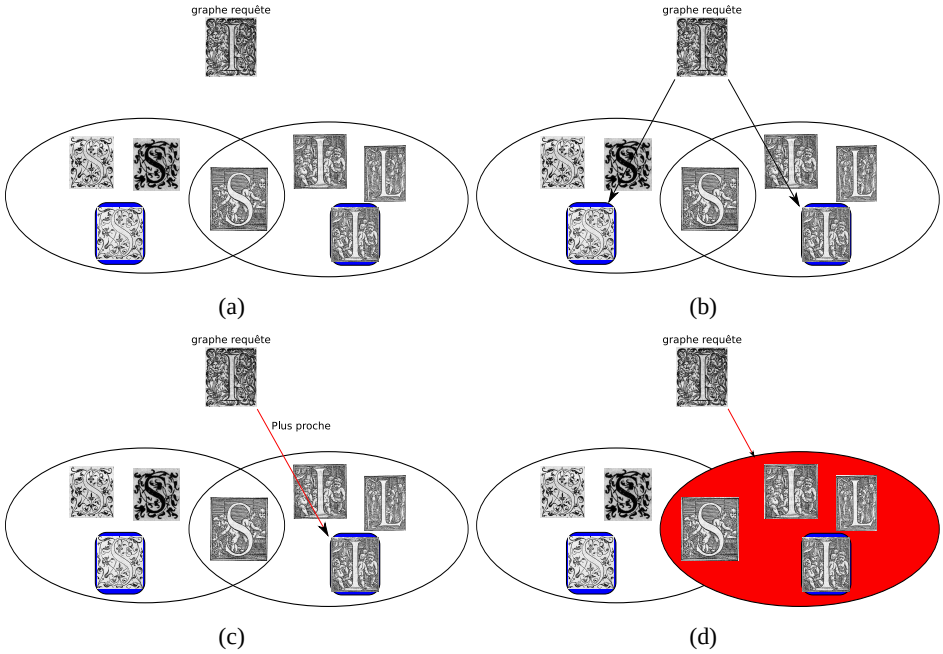


FIGURE 6.5 – Illustration de la recherche de lettrines avec le modèle d'hypergraphe

cet exemple, nous considérons que la réponse à la requête est satisfaite par les images de la première hyperarête (rouge). Par ailleurs, si ce n'était pas le cas, l'algorithme va récupérer aussi les images dans la deuxième hyperarête.

Notre contribution peut être aussi utilisée dans un contexte de recherche *associative* d'images. Nous rappelons que ce paradigme de recherche d'images consiste à aider les utilisateurs à explorer la base d'images. Pour permettre aux utilisateurs de naviguer (explorer) dans la base d'images, nous utilisons une technique de parcours en profondeur modifié de l'hypergraphe. Notre méthode consiste à utiliser les centroïdes des hyperarêtes comme les images de départ de la navigation. L'utilisateur choisit alors l'hyperarête à exploiter en sélectionnant le centroïde correspondant. Ensuite, la navigation correspond à un parcours en profondeur de l'hypergraphe. Ce parcours consiste à visiter les images appartenant à des hyperarêtes connexes. Il faut noter ici que le parcours de l'hypergraphe dépend du choix de l'utilisateur. En effet, à une étape e_i de la navigation, le parcours est défini à partir d'une image im choisie par l'utilisateur. Ainsi, à l'étape suivante e_{i+1}, la partie explorée de l'hypergraphe consiste aux images voisines de im [21]. Concrètement, la navigation de l'hypergraphe est réalisée à travers une interface graphique. Dans cette interface, l'utilisateur fait son parcours de l'hypergraphe en cliquant sur les images qui l'intéressent. Dans la figure 6.13 les clusters (hyperarêtes) sont représentés par des ellipses qui se chevauchent. Dans cette figure, l'utilisateur a sélectionné la lettrine avec la lettre D et, en cliquant sur les lettrines voisines, a pu explorer une partie de la base liée à sa requête. Les images au centre de chaque hyperarête sont les centroïdes.

21. Les images voisines d'une image i correspondent aux images appartenant aux hyperarêtes qui contiennent l'image i

6.4 Expérimentations

6.4.1 Protocole d'évaluation

Dans cette partie expérimentale, nous nous focalisons sur l'étude du comportement de notre méthode vis-à-vis de la valeur du seuil T utilisé (cf. Algorithme 7). Cette étude concerne, pour une valeur donnée de T, les cinq points suivants :

1. Le nombre d'hyperarêtes généré par notre méthode : vet aspect décrit le rapport entre nombre de clusters détectés dans une base de graphes et la valeur du seuil utilisée.
2. La taille moyenne des hyperarêtes : cet aspect décrit l'évolution de la taille moyenne des clusters, fournis par notre méthode, par rapport à la valeur du seuil.
3. Le taux de chevauchement des hyperarêtes : le chevauchement entre deux hyperarêtes h_i et h_j correspond à l'ensemble $\mathcal{G}$ des graphes partagés par h_i et h_j, i.e. $\mathcal{G} = \{g_i \mid g_i \in h_i et g_i \in h_j\}$. Pour calculer le taux de chevauchement de notre structure d'hypergraphe, nous utilisons la mesure de Zhiling et al. [112]. Notons que cette mesure a été développée dans le contexte du clustering flou, qui permet le chevauchement entre les clusters. Concrètement, le taux de chevauchement entre deux hyperarêtes h_i et h_j ($\tau(h_i, h_j)$) correspond au quotient du nombre de graphes partagés par le nombre de graphes de la plus petite hyperarête. Formellement,

   $$\tau(h_i, h_j) = \frac{N_{partagé}}{N_{min}}$$

   avec $N_{partagé}$ le nombre de graphes partagés entre h_i et h_j, et N_{min}=min($|h_i|$,$|h_j|$) la taille (nombre de graphes) de la plus petite hyperarête entre h_i et h_j. Le taux de chevauchement est dans l'intervalle [0, 1] (une valeur proche de 1 indique un fort chevauchement).

   Le taux de chevauchement final d'un hypergraphe $\mathcal{H}$ ($\tau(\mathcal{H})$) correspond à la moyenne des taux de chevauchement de toutes les paires d'hyperarêtes dans l'hypergraphe, soit formellement :

   $$\tau(\mathcal{H}) = \frac{\sum \tau(h_i, h_j)}{\frac{n(n-1)}{2}}$$

   avec n le nombre d'hyperarêtes dans $\mathcal{H}$.
4. Le nombre d'appariements de graphes : cet aspect détermine le nombre d'appariements de graphes (distance d'édition de graphes) effectué dans une procédure de recherche d'images similaires à une requête donnée.
5. La précision des résultats de la recherche : cet aspect évalue la qualité de la recherche d'images effectuée par notre méthode. La précision est le quotient du nombre d'images pertinentes retrouvées par rapport au nombre total d'images proposées par notre méthode pour une requête donnée.

Pour l'évaluation de notre méthode, nous utilisons les sept bases de graphes utilisées dans le chapitre **4** (GREC1, Lettrine, Shape, Logo, Mutagenicity, Letter et GREC2). Pour chaque base de graphes, nous calculons les cinq critères, établis précédemment, pour plusieurs valeurs du seuil T. En ce qui concerne la précision et le nombre d'appariements, nous calculons, pour chaque base, la précision moyenne (respectivement le nombre d'appariements moyen) sur les

dix premières images retenues par notre méthode, en prenant tour à tour chaque image de la base comme image requête. Nous rappelons que dans le cas où le nombre de graphes dans la première hyperarête est inférieur à 10, notre méthode récupère les graphes de l'hyperarête plus proche suivante (cf. section 6.3.3).

Nous comparons la précision et le nombre d'appariements effectués par notre méthode avec ceux de la méthode classique de recherche d'images dans laquelle la requête est comparée à tous les graphes de la base.

6.4.2 Résultats

Les figures 6.6, 6.7, 6.8, 6.9, 6.10, 6.11 et 6.12 présentent, respectivement, les résultats de l'analyse du comportement de notre méthode pour les bases : GREC1, Lettrine, Shape, Logo, Mutagenicity, Letter et GREC2. Chaque figure comporte cinq courbes dont chacune présente le comportement d'un aspect de notre méthode en fonction de la valeur du seuil. À partir de ces résultats, nous pouvons tirer les remarques suivantes :

- En considérant les critères du nombre et de la taille moyenne d'hyperarêtes générées en fonction du seuil, nous observons que les courbes ont la même allure (leurs pentes sont approximativement les mêmes) pour toutes les bases de graphes. Cette allure montre que pour des petites valeurs du seuil le nombre d'hyperarêtes est important et leur taille moyenne est faible. En effet, pour une petite valeur du seuil, chaque hyperarête contient un petit ensemble de graphes très similaires (leurs distances sont inférieures au seuil). Par ailleurs, pour des grandes valeurs du seuil, la méthode génère peu d'hyperarêtes mais elles sont de grandes tailles.
- En considérant le taux de chevauchement des hyperarêtes, nous observons aussi que les courbes de cet aspect ont la même allure pour toutes les bases de graphes. En effet, plus le seuil est grand, plus le taux de chevauchement est important. En fait, pour des grandes valeurs du seuil, les graphes ont plus de chances d'appartenir à plusieurs hyperarêtes simultanément. Par ailleurs, si la valeur du seuil dépasse la distance entre les deux graphes les plus distants dans une base, alors notre méthode génère une seule hyperarête qui contient toute la base. Ceci est expliqué par le fait que toutes les distances entre les graphes sont inférieures au seuil. Dans ce cas, le taux de chevauchement est zéro.
- En considérant le nombre d'appariements de graphes effectué, nous observons qu'à partir d'une certaine valeur du seuil ce nombre devient proportionnellement similaire à la taille moyenne des hyperarêtes. En effet, pour les petites valeurs du seuil la méthode génère un nombre important d'hyperarêtes dont les centroïdes sont tous comparés avec la requête à la recherche des plus proches hyperarêtes. Ceci implique, potentiellement, un nombre élevé d'appariements. Par ailleurs, plus les hyperarêtes visitées, lors de la recherche, sont grandes plus le nombre d'appariements est important. En fait, notre méthode recherche les images similaires à une requête dans les hyperarêtes ayant les centroïdes les plus proches à la requête. Par ailleurs, en combinant ces résultats avec ceux de la précision, nous remarquons que notre méthode permet d'obtenir le même degré de précision que la recherche classique, en dépit du fait que nous explorons uniquement une partie de la base, alors que la méthode de recherche d'images classique compare le graphe de l'image requête avec tous les graphes de la base. Ceci met en avant l'intérêt de notre modèle d'hypergraphe

Base de graphes	Notre méthode		Recherche Classique		Seuil
	Précision	Nombre d'appariement	Précision	Nombre d'appariement	
GREC1	0.88	69.1	0.88	528	1.25
Lettrine	0.58	87.1	0.58	280	484.98
Shape	0.53	55.3	0.53	216	4.60
Logo	0.87	29.7	0.87	80	1.44
Mutagenicity	0.64	72.8	0.64	400	2.59
Letter	0.90	102.6	0.90	750	0.16
GREC2	0.98	66.4	0.98	528	4.77

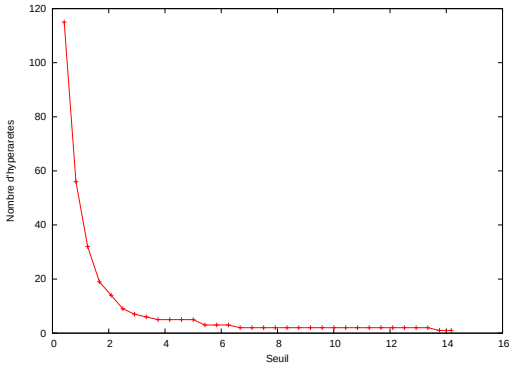
TABLE 6.1 – Résultats de la recherche

pour l'indexation de graphes. Dans la table 6.1, nous illustrons une comparaison entre la méthode de recherche classique et notre modèle en fixant la valeur du seuil pour chaque base. Les valeurs du seuil sélectionnées correspondent aux valeurs qui fournissent les meilleures performances. Ces performances consistent en une valeur de précision maximale et un nombre d'appariements minimal. Ces résultats montrent que notre méthode affiche les mêmes performances en terme de précision que la recherche classique, bien que le nombre moyen d'appariements effectués par notre méthode est nettement inférieur à celui effectué par la méthode de recherche d'images classique. À titre d'exemple, pour la recherche d'images dans la base Letter, notre méthode basée sur l'hypergraphe réalise la même précision (0.90) que la méthode classique. Par contre, notre méthode exécute en moyenne 102.6 appariements de graphes de la base Letter contre 750 appariements exécutés par la méthode classique. Pareillement, notre méthode réalise des performances similaires pour toutes les bases testées, ce qui souligne l'efficacité et la rapidité de notre méthode. L'efficacité est expliquée par le fait que la précision de la recherche par notre méthode est équivalente à celle de la recherche classique. Ainsi, l'organisation d'une base de graphes sous forme d'une structure d'hypergraphe n'engendre pas une altération de la précision de la recherche. De plus, cette structure d'hypergraphe offre une solution rapide de recherche de graphes en n'explorant qu'un sous-hypergraphe [22].

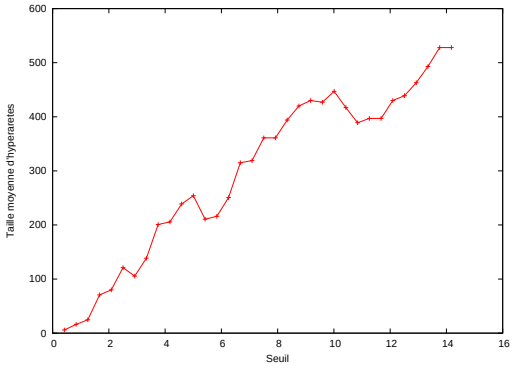
6.5 Conclusion

Dans ce chapitre, nous avons étudié comment la structure d'hypergraphe peut être utilisée à des fins de représentation de base de graphes. Nous avons proposé une méthode basée sur la sélection de prototypes pour indexer les graphes. La sélection des prototypes proposée permet de définir automatiquement le nombre d'hyperarêtes (clusters de graphes). Ce travail permet également la multi-affectation d'un graphe, à savoir qu'un graphe peut être affecté à plusieurs clusters.

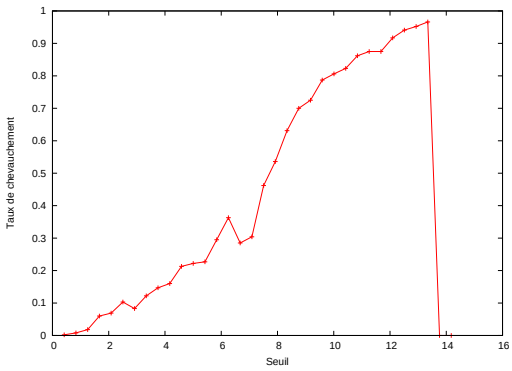
22. Un sous-hypergraphe H' = (V',E') d'un hypergraphe H = (V,E) est tel que :
– V' $\subseteq$ V et
– $\forall E_i \in E'$, $E_i \subseteq V'$ et $E_i \in$ E.



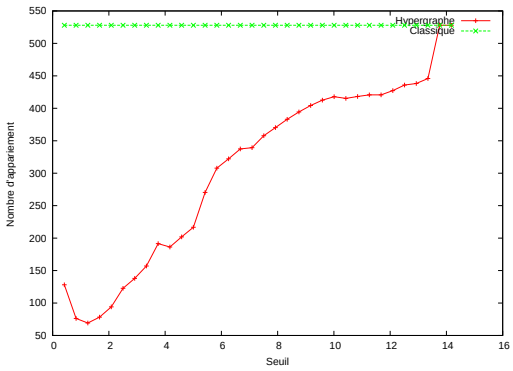
(a) Nombre d'hyperarêtes générées en fonction du seuil



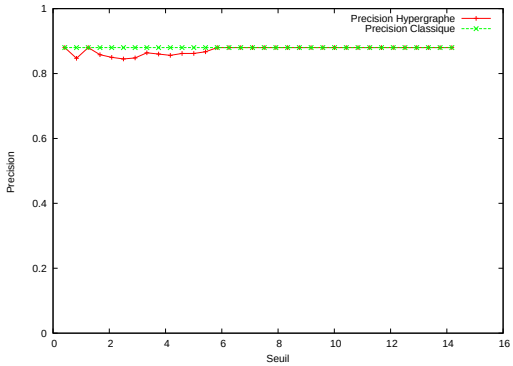
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes

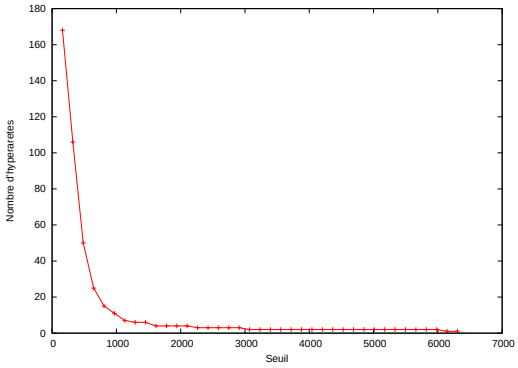


(d) Nombre d'appariements de graphes

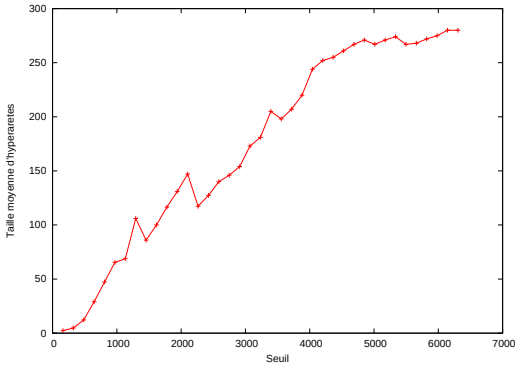


(e) Précision de résultats de la recherche

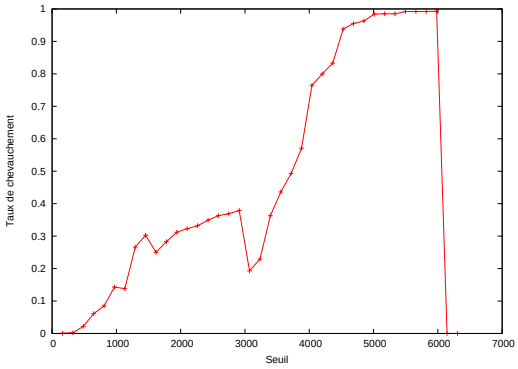
FIGURE 6.6 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base GREC1



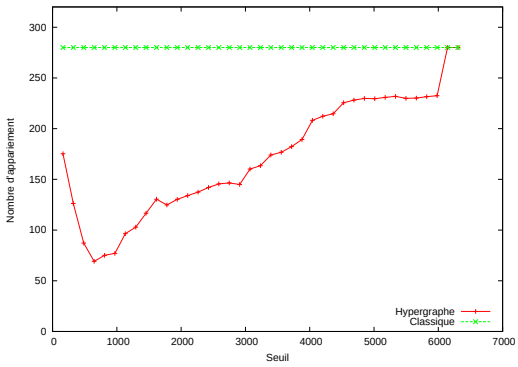
(a) Nombre d'hyperarêtes générées en fonction du seuil



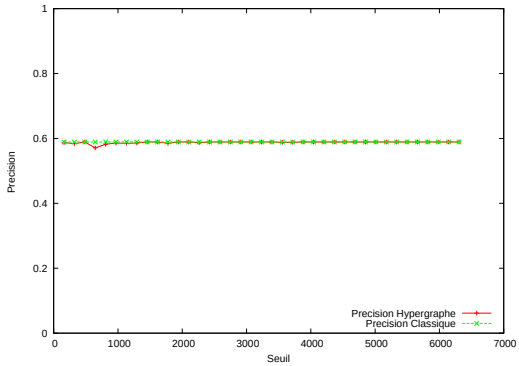
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes

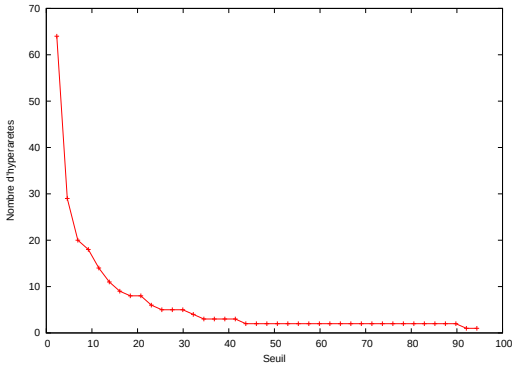


(d) Nombre d'appariements de graphes

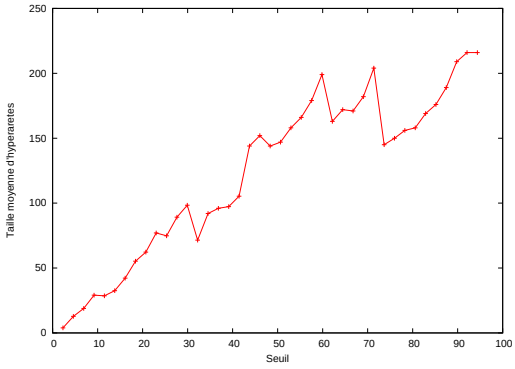


(e) Précision de résultats de la recherche

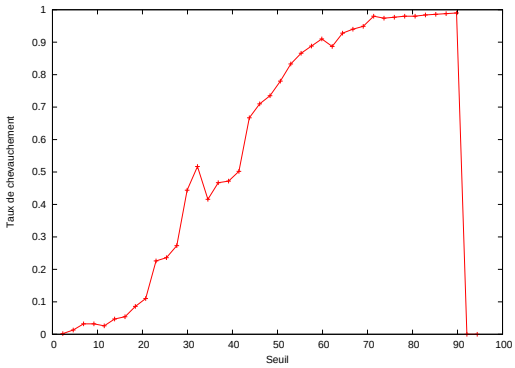
FIGURE 6.7 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base Lettrine



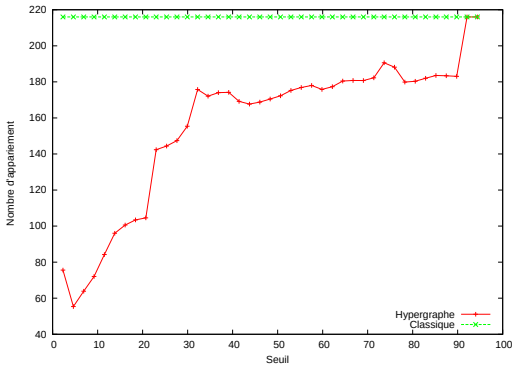
(a) Nombre d'hyperarêtes générées en fonction du seuil



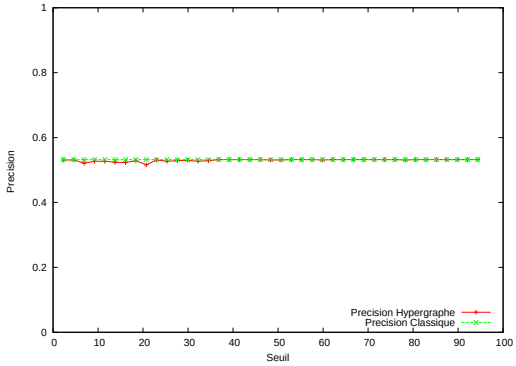
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes

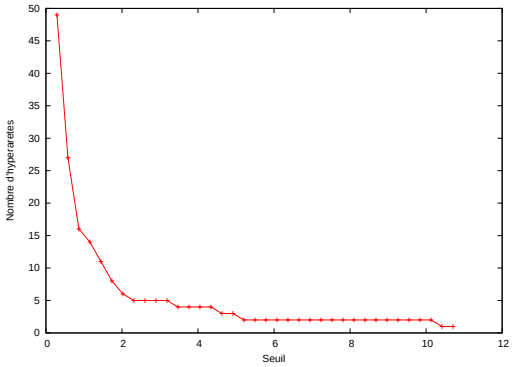


(d) Nombre d'appariements de graphes

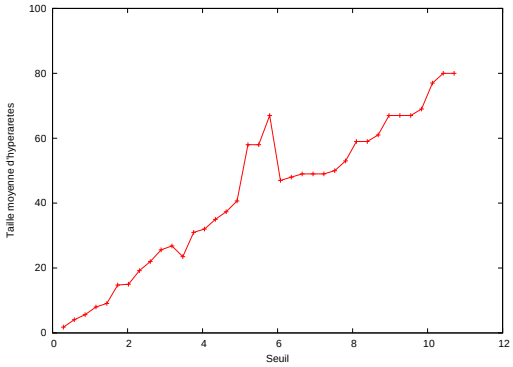


(e) Précision de résultats de la recherche

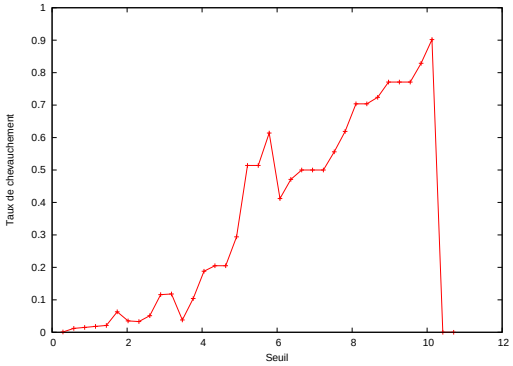
FIGURE 6.8 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base Shape



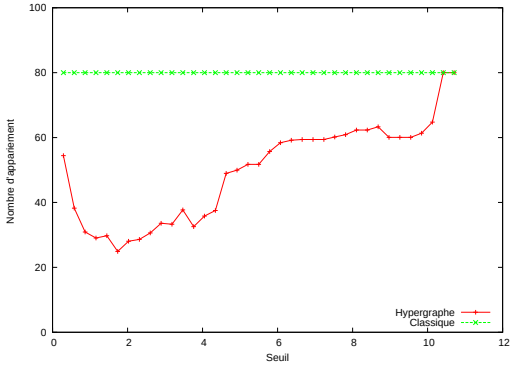
(a) Nombre d'hyperarêtes générées en fonction du seuil



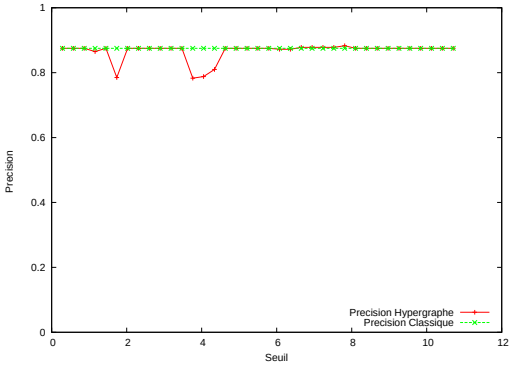
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes

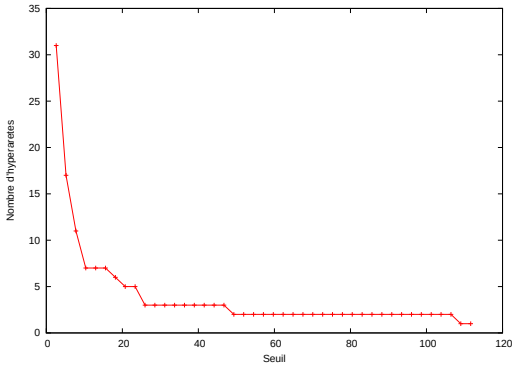


(d) Nombre d'appariements de graphes

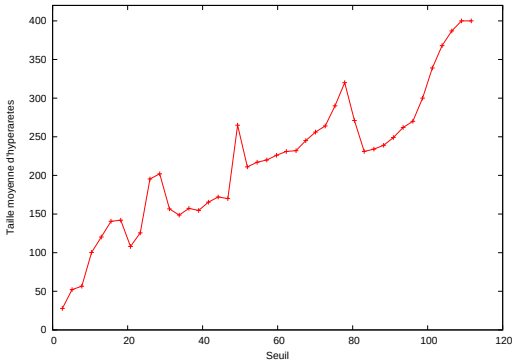


(e) Précision de résultats de la recherche

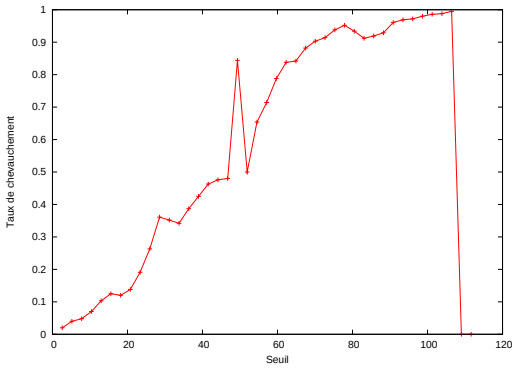
FIGURE 6.9 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base Logo



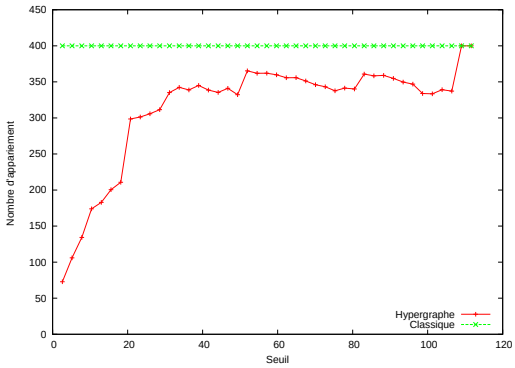
(a) Nombre d'hyperarêtes générées en fonction du seuil



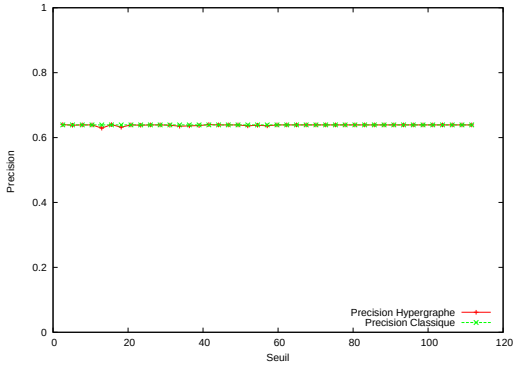
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes

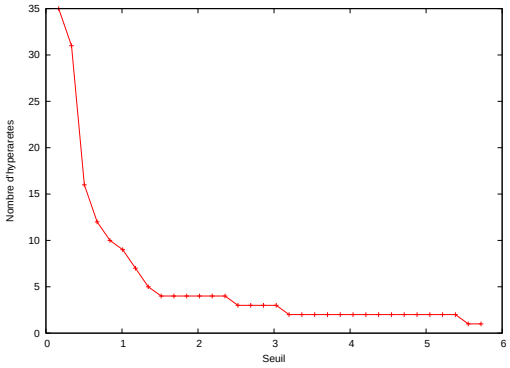


(d) Nombre d'appariements de graphes

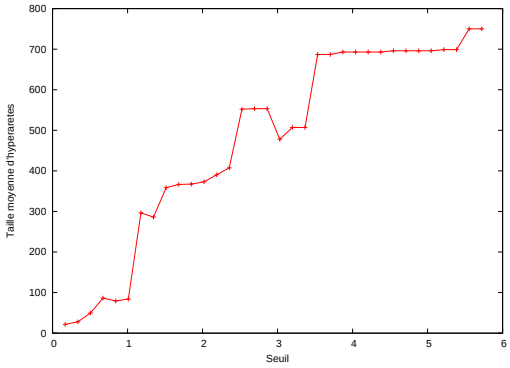


(e) Précision de résultats de la recherche

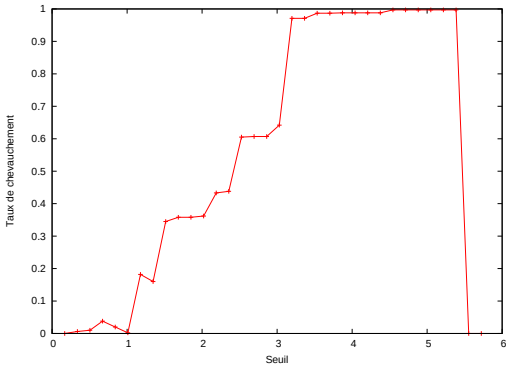
FIGURE 6.10 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base Mutagenicity



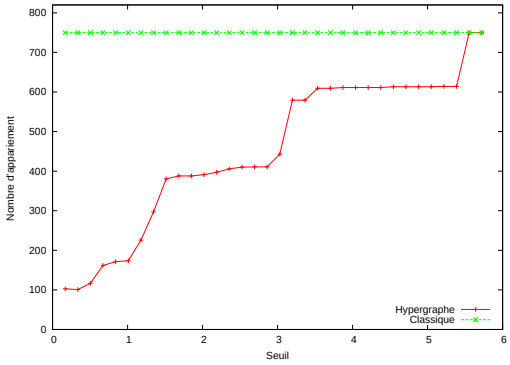
(a) Nombre d'hyperarêtes générées en fonction du seuil



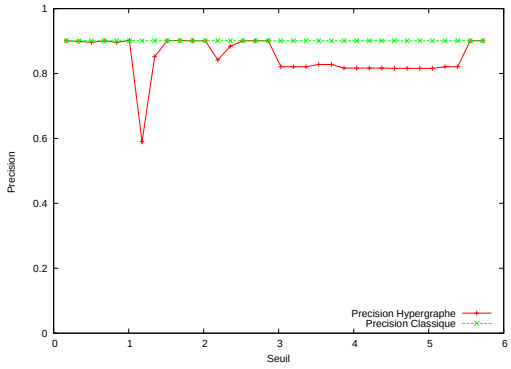
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes

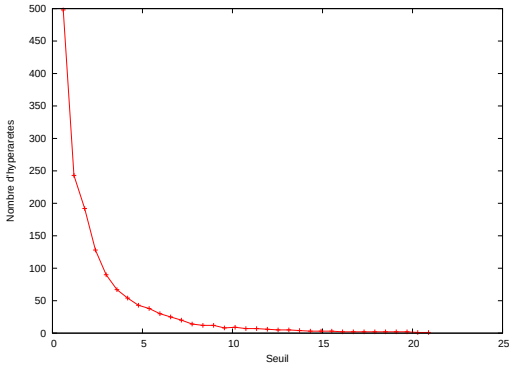


(d) Nombre d'appariements de graphes

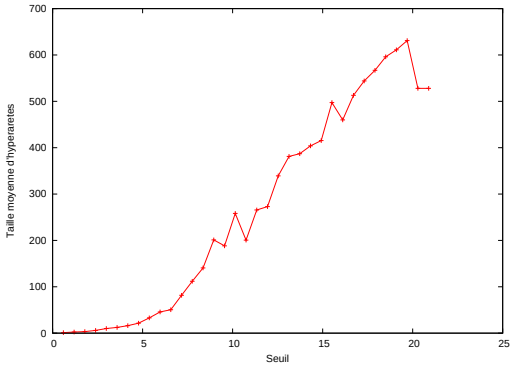


(e) Précision de résultats de la recherche

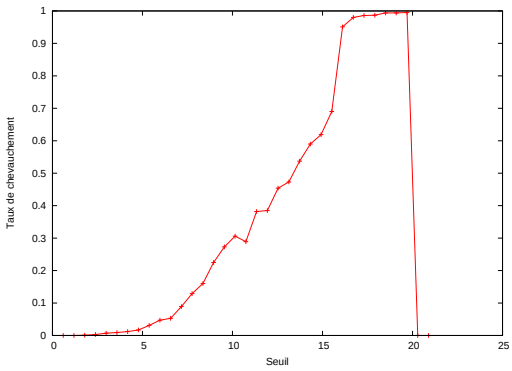
FIGURE 6.11 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base Letter



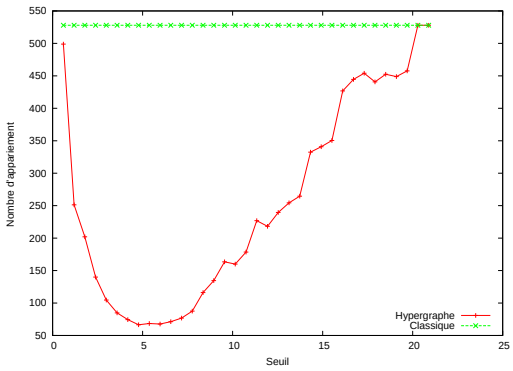
(a) Nombre d'hyperarêtes générées en fonction du seuil



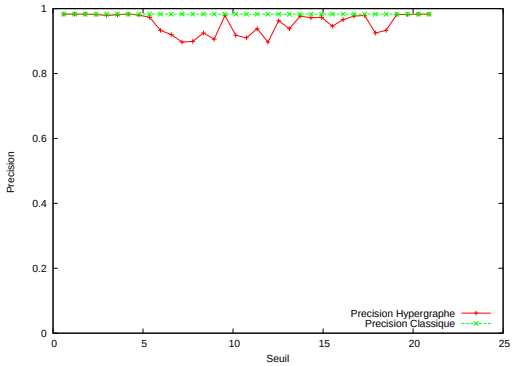
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes



(d) Nombre d'appariements de graphes



(e) Précision de résultats de la recherche

FIGURE 6.12 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base GREC2

Les résultats expérimentaux ont montré que notre méthode réalise les mêmes performances en terme de précision que la recherche classique, bien que le nombre moyen d'appariements effectués par notre méthode est nettement inférieur à celui effectué par la méthode de recherche d'images classique.

Dans le chapitre suivant nous mettons à profit nos contributions dans le cadre du projet ANR NAVIDOMASS dans lequel a évolué cette thèse.

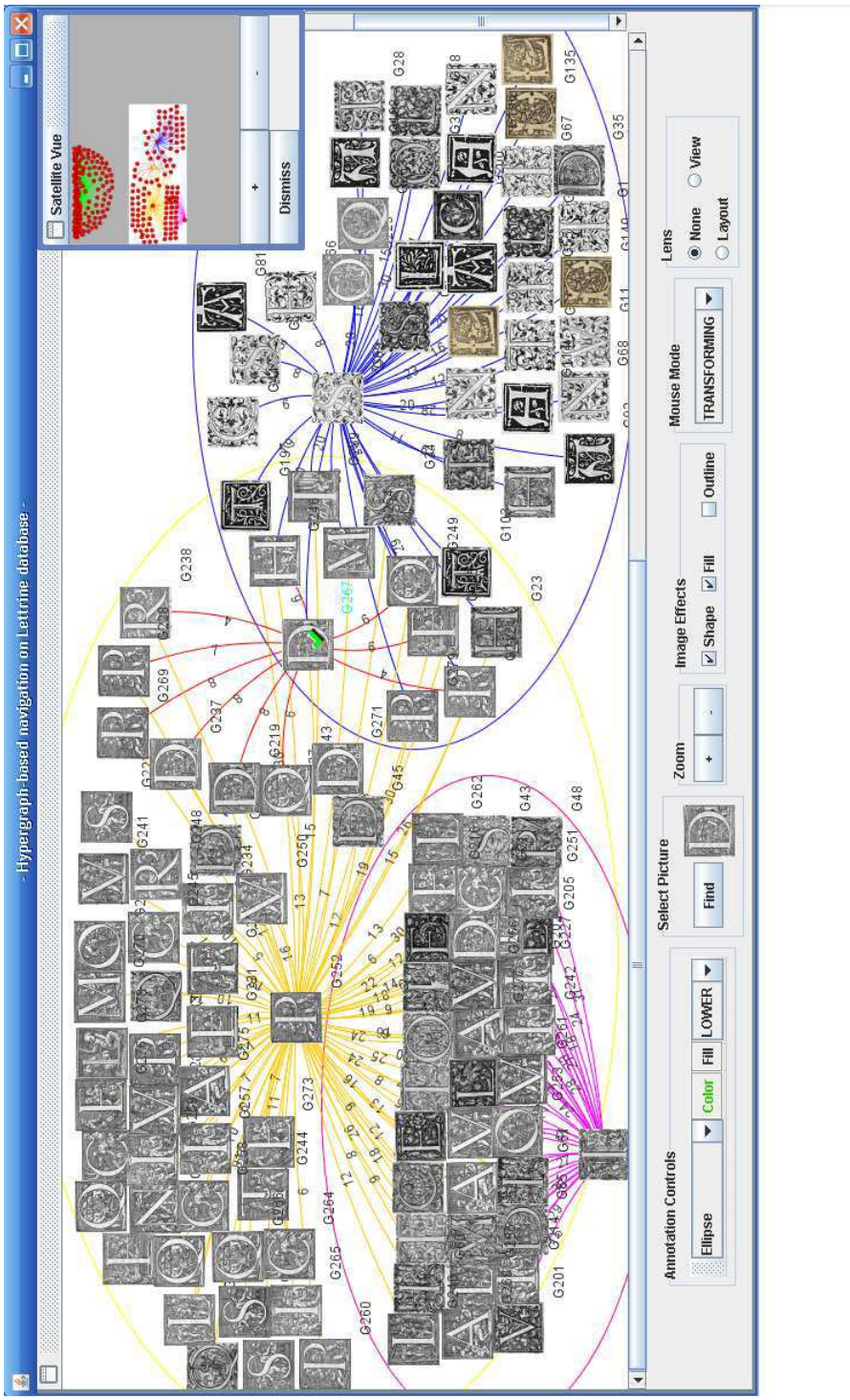


FIGURE 6.13 – Illustration de l'interface développée pour la navigation dans une base de graphes

Chapitre 7

Application au projet NAVIDOMASS

The true method of knowledge is experiment.
William Blake

Sommaire

7.1 Le Projet Navidomass

Durant la dernière décennie, la numérisation de l'héritage historique a fait l'objet de plusieurs projets de recherche et industriels. Cette numérisation a pour objectif de rendre les documents historiques plus accessibles en utilisant les nouvelles technologies (i.e. Internet). Ainsi, il est impératif de recourir aux techniques de reconnaissance de formes pour reconnaître et rechercher ce type de documents automatiquement. Néanmoins, les document historiques sont très riches en information ce qui les rend difficiles à traiter par les techniques classiques de reconnaissance. Parmi ces documents historiques, nous distinguons les lettrines qui sont utilisées par les historiens dans différentes tâches, e.g. déterminer l'époque d'un ouvrage, identifier l'imprimeur, etc. Les lettrines sont caractérisées par une structure particulièrement complexe où chaque composante contient de l'information. Dans la figure 7.1, nous illustrons quelques exemples de lettrines. Nous pouvons remarquer que chaque image de lettrine est un mélange de composantes simples telles que l'arrière plan, la lettre et la texture. Par ailleurs, quelques dégradations, liées

à l'état du document original, peuvent être observées. Afin de traiter et reconnaître ces lettrines, il est nécessaire qu'un système de reconnaissance prenne en compte toutes ces caractéristiques. Cette thèse se situe dans le contexte d'un projet ANR, appelé NAVIDOMASS, où la reconnaissance de lettrines fait partie des axes principaux.

NAVIDOMASS [23] est un projet de recherche financé par l'Agence Nationale de Recherche [24]. Académiquement, NAVIDOMASS est une collaboration entre cinq laboratoires d'informatique français et le centre d'études supérieures de la Renaissance. L'objectif principal est la navigation dans les grandes bases de données des archives historiques numérisées. Ces archives sont numérisées sous forme d'images contenant des textes, images, illustrations et des schémas. Le projet NAVIDOMASS est divisé en cinq axes de travail : (1) les besoins utilisateurs, conception collaborative et vérité terrain, (2) l'analyse des couches des documents et indexation basée sur la structure, (3) la recherche d'informations, (4) la structuration de l'espace d'information (5) l'extraction et le contrôle interactifs.

Cette thèse entre dans le cadre du deuxième axe de travail de NAVIDOMASS qui concerne la partie de la classification, l'indexation et la navigation dans une base de documents anciens, notamment les lettrines. Dans ce chapitre, nous appliquons l'ensemble des contributions précédentes sur les lettrines.

Avant de manipuler ces documents, il faut faire un choix entre les deux grandes familles d'approches de description des caractéristiques d'un document : soit les approches statistiques, soit les approches structurelles (voir chapitre 1). Nous rappelons que dans les approches statistiques, chaque document est représenté par un vecteur numérique de taille d qui est constitué par les valeurs numériques de d caractéristiques mesurées sur le document. Dans les approches structurelles, un document est représenté par un modèle structurel qui décrit les différents composants de l'objet et leurs relations. Cette représentation est caractérisée par une capacité de représentation élevée par rapport à la représentation via les vecteurs caractéristiques [33]. La représentation structurelle nous semble la plus adéquate pour représenter les lettrines vu qu'elles représentent des documents complexes et riches en information.

Dans ce chapitre l'ensemble des contributions précédentes sont appliquées aux Lettrines. Ainsi, nous utilisons notre approximation de la distance d'édition pour calculer la distance entre deux lettrines. Ensuite, nous appliquons nos méthodes de classification supervisée et non-supervisée pour classifier les lettrines.

7.2 La représentation de lettrines sous forme de graphes

Les images utilisées dans cette thèse sont les images des lettrines (Fig. 7.1) extraites à partir des documents du quinzième et seizième siècle. Elles étaient largement utilisées à l'époque de la Renaissance dans les ouvrages pour identifier, généralement, le début d'un chapitre ou d'un paragraphe. La numérisation de ces documents est faite par le Centre d'Étude Supérieur de la Renaissance de l'université de Tours.

Nous commençons par une discussion sur le type de représentation. Nous opposons deux types d'approches : une générique basée sur la méthode de segmentation de Felzenszwalb et al.

23. `http://navidomass.univ-lr.fr`
24. ANR : `http://www.agence-nationale-recherche.fr/`

[80] et une dédiée développée par Coustaty et al. [59, 58] dans le contexte de cette ANR.

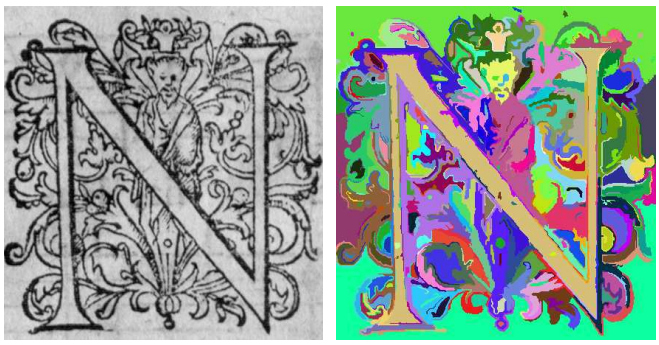
(a) (b) (c) (d)

FIGURE 7.1 – Exemples d'images de Lettrines

7.2.1 Graphe d'adjacence de régions

Chaque lettrine est représentée par un graphe d'adjacence de régions dont les noeuds correspondent aux régions homogènes obtenues par une technique de segmentation. La méthode de segmentation utilisée est celle de Felzenszwalb et al. [80]. Succinctement, cette méthode commence par représenter l'image par un graphe où chaque pixel de l'image correspond à un noeud et où les arêtes correspondent aux paires des nœuds voisins. Ensuite, la segmentation de l'image consiste à partitionner les noeuds du graphe en fixant un seuil et une distance entre ces noeuds.

La figure 7.2 présente une illustration du résultat de la segmentation d'une lettrine. Une fois la lettrine segmentée, nous la représentons par un graphe comme suit : chaque région est représentée par un noeud étiqueté par un vecteur caractéristique qui contient les descriptions suivante de la région : la surface, le périmètre et les coordonnées du centre. Les arêtes entre noeuds décrivent les relations d'adjacence entre les régions associées à ces noeuds.



(a) Image Originale (b) Image segmentée

FIGURE 7.2 – Exemple de la segmentation d'une lettrine avec la méthode de Felzenszwalb [80]

7.2.2 Une représentation sous forme de graphes spécifique aux lettrines

Cette approche de représentation de lettrines sous forme de graphes est développée dans le cadre du projet NAVIDOMASS par Coustaty et al. [59, 58]. L'objectif de cette méthode est d'identifier les composantes géométrique saillantes d'une lettrine et de les stocker sous forme d'un graphe.

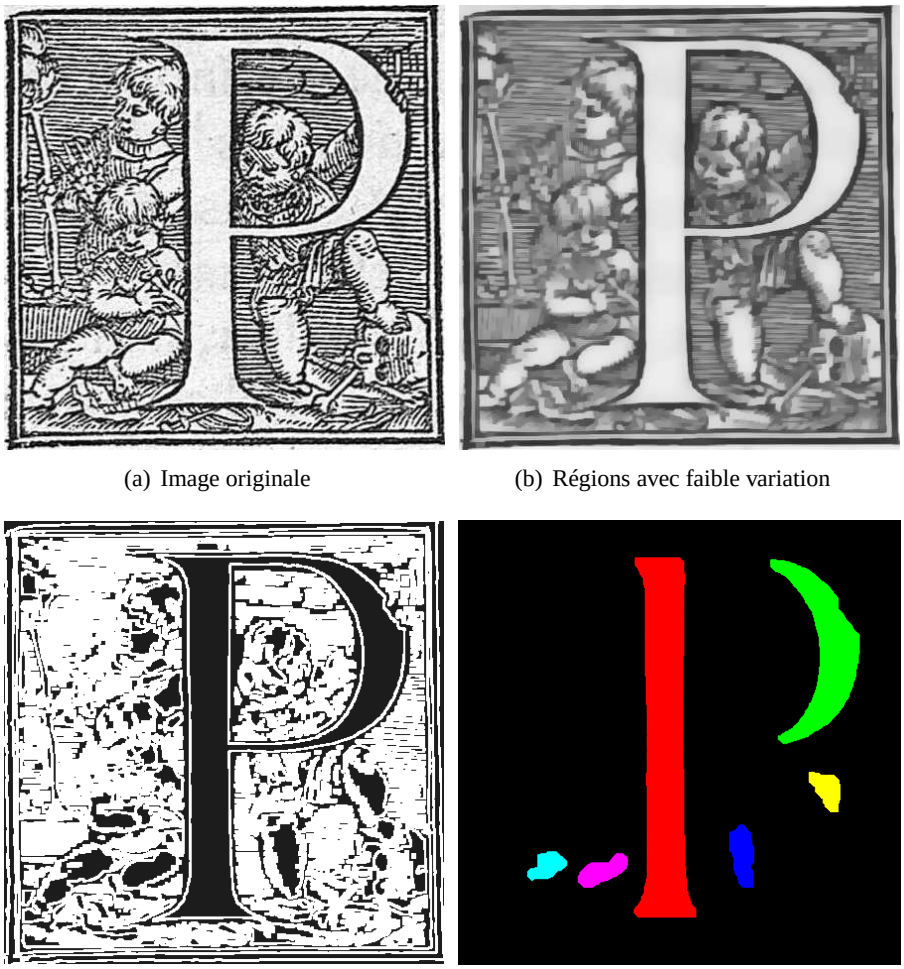
Les auteurs commencent par utiliser un modèle de décomposition en couches [7, 8, 69] qui seront plus simples à caractériser. Ainsi, chaque lettrine est décomposée en trois couches. La première représente la structure (aspects géométriques) de l'image, la deuxième correspond à la texture et la troisième représente le bruit présent dans l'image. Pour mieux définir ces différentes couches et seulement dans un souci d'exhaustivité, nous reproduisons dans la suite les définitions des couches établies par Coustaty et al. [58] :

La couche géométrique est une couche régularisée qui *"correspond aux zones de l'image qui ont une faible variation de niveaux de gris. Cette couche permet de mettre en évidence la composante géométrique qui correspond aux formes de l'images."*

La couche de texture est une couche oscillante qui *"correspond aux zones aux variations rapides de niveaux de gris. Dans notre cas, cette couche permet de mettre en évidence les textures des lettrines, c'est-à-dire les zones composées de traits parallèles."*

La couche de bruit est une couche très oscillante *"qui correspond au bruit dans l'image. En fait, cette couche est composée de tout ce qui n'appartient pas aux deux premières couches. Dans notre cas, on peut retrouver dans cette couche le texte du verso de la page et les problèmes dus au vieillissement du papier"*

Nous remarquons ici que la grande partie d'information concernant la lettrine est localisée dans la première couche. Cette couche contient toute la structure géométrique de la lettrine. Évidemment, la structure géométrique peut être considérée comme l'ensemble des formes composant l'image de la lettrine. Par conséquence, l'utilisation de ces formes pour construire les graphes semble intéressante. Toutefois, l'extraction de ces formes à partir de la première couche ne peut aboutir avec une simple méthode d'extraction de composantes [58]. Pour extraire et sélectionner les formes les plus intéressantes de la lettrine, les auteurs proposent d'utiliser une segmentation basée sur la loi de Zipf [288]. Cette loi consiste à déterminer la relation entre le rang et la fréquence d'un mot dans un texte. Ainsi, si on dispose de l'ensemble des mots d'un texte ordonnés par fréquence décroissante, Zipf observe que la fréquence d'utilisation d'un mot est inversement proportionnel à son rang. L'application de la loi de Zipf pour segmenter les images est inspirée par les travaux de Pareti et al. [184]. L'idée principale est de considérer des imagettes (sous-images) de l'image originale pour le calcul de leurs fréquences et leurs rangs. Ensuite la segmentation de la lettrine est basée sur les observations de la loi de Zipf. Après l'extraction des formes, une sélection est élaborée pour extraire les formes particulièrement intéressantes du point de vue de la taille, de la localisation et du centre. Dans la figure 7.3, nous illustrons un exemple d'extraction des formes intéressantes d'une lettrine en utilisant cette méthode [59]. En se basant sur les formes sélectionnées, un graphe complet représentant la lettrine est construit dont les nœuds représentent ces formes. Chaque noeud est étiqueté par un quadruplet qui contient les coordonnées du centre, la superficie et l'excentricité de chaque forme.



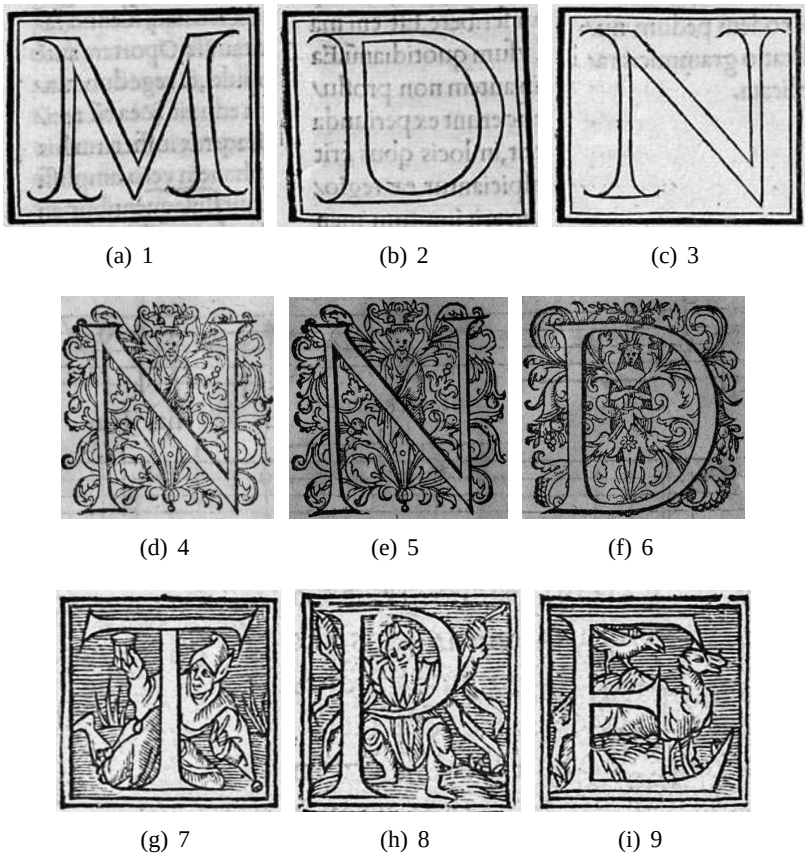
(a) Image originale

(b) Régions avec faible variation

(c) Formes segmentées avec la loi de Zipf (en noir)

(d) Les six composantes connexes les plus large

FIGURE 7.3 – Exemple d'extraction des formes à partir d'une lettrine



(a) 1 (b) 2 (c) 3

(d) 4 (e) 5 (f) 6

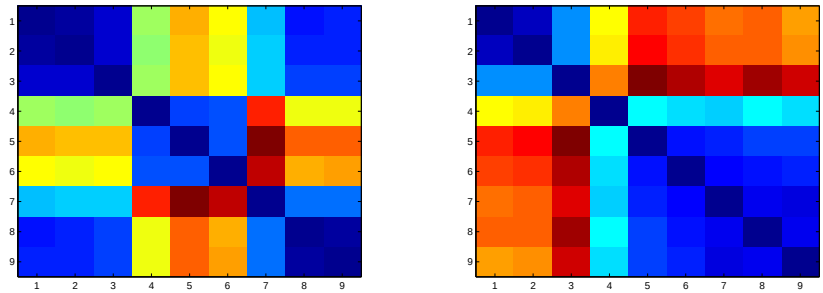
(g) 7 (h) 8 (i) 9

FIGURE 7.4 – Ensemble de lettrines utilisées dans l'évaluation de la distance d'édition de graphes

7.3 Mesure de distance entre les lettrines

Dans cette section, nous étudions l'application de notre méthode d'approximation de distance sur les lettrines. Pour ce faire nous avons sélectionné 9 lettrines de trois types différents (3 lettrines de chaque type). Dans la figure 7.4, nous illustrons les neuf lettrines sélectionnées. Les graphes qui représentent ces lettrines sont construits, en premier lieu, en utilisant la méthode basée sur l'adjacence des régions. Deuxièmement, un autre ensemble de graphes est créé en utilisant la méthode de Coustaty et al. [59] dédiée aux lettrines. L'objectif ici est de visualiser les distances calculées par notre méthode sur les deux ensembles de graphes. Nous estimons que les résultats accomplies avec la représentation dédiée aux lettrines seront mieux que les résultats en utilisant une simple segmentation en régions. Ces attentes peuvent être justifiées par le fait que la représentation de Coustaty semble plus appropriée aux lettrines que la segmentation classique et ceci devrait être reflété par la distance d'édition de graphes.

Pour visualiser les résultats, nous calculons la matrice de distance pour chaque ensemble de graphes. Évidemment, chaque matrice est de taille $9x9$. La figure 7.5 présente les deux matrices



(a) Matrice de distance entre les graphes construite avec la méthode de coustaty

(b) Matrice de distance entre les graphes d'adjacence de régions

FIGURE 7.5 – Matrices de distances entre les lettrines en utilisant deux représentations de graphes différentes

de distances. Chaque classe (trois lettrines similaires) correspond à un bloc dans ces matrices. Les images de 1 à 3 forment la première classe, les images de 4 à 6 forment la deuxième et finalement la troisième classe est formée par les images de 7 à 9. Les lignes et les colonnes de chaque matrice indexent les distances entre les graphes, i.e. l'élément (i, j) d'une matrice de distance correspond à la distance entre le graphe qui représente la i-ième lettrine et le graphe qui représente la j-ième lettrine.

Dans la matrice de distance entre les graphes produits avec la méthode de Coustaty (Fig. 7.5(a)), nous observons clairement trois blocs sur la diagonale avec des faibles intensités. Ceci signifie que la valeur des distances intra-classes sont plus faibles que les distance inter-classes. Autrement dit, les graphes d'une même classe sont plus proches les uns des autres que des graphes de différentes classes. Par contre dans la matrice de distance entre les graphes d'adjacence de régions classiques (Fig. 7.5(b)), nous n'observons que deux blocs sur la diagonale, ce qui signifie que les distances intra-classes sont très proches des distance inter-classes, particulièrement entre la deuxième et la troisième classe. En effet, en utilisant les graphes d'adjacence de régions, notre méthode de distance d'édition de graphes a produit des distances similaires entre les graphes appartenant à la deuxième et à la troisième classe. Afin de comprendre mieux ces résultats, nous illustrons dans la figure 7.6 le résultat de segmentation d'une lettrine de chaque classe. Nous observons, d'une part, que la lettrine segmentée de la première classe est clairement différente des autres lettrines segmentées. Les différences se réfléchissent sur le nombre de régions ainsi que sur leur taille. D'autre part, nous constatons que les deux lettrines segmentées de la deuxième et troisième classe sont proches dans le sens où chacune est composée de plusieurs régions ayant une petite surface. Ceci implique une ressemblance entre les structures des graphes d'adjacence de régions produits. Cette ressemblance entre les structures des graphes entraîne forcément une distance plus faibles entre ces graphes.

À partir de ces observations, nous concluons par les points suivants :

– Notre méthode d'approximation de la distance d'édition de graphes est appropriée pour

(a) Segmentation d'une lettrine de la première classe

(b) Segmentation d'une lettrine de la deuxième classe

(c) Segmentation d'une lettrine de la troisième classe

FIGURE 7.6 – Exemple de segmentation avec la méthode de Felzenszwalb [80] de lettrines appartenant aux différentes classes

mesurer la distance entre les lettrines représentées sous forme de graphes.

– En combinant une technique de représentation appropriée aux lettrines avec notre distance, les résultats s'améliorent.

Ces points convergent vers ce que nous avons envisagé avant l'expérimentation. En effet, les résultats accomplies avec la représentation dédiée aux lettrines sont mieux que les résultats en utilisant une simple segmentation en régions. Ceci est une propriété classique commune à toutes les mesures de distance de graphes (voir chapitre 2).

7.4 Classification de Lettrines

Suite aux observations faites précédemment, nous utilisons les graphes extraits des lettrines avec la méthode de Coustaty et al. [59]. La base de Lettrines est constitué de 824 images. Cette base est délivrée par le centre d'études supérieures de la Renaissance (CESR). La vérité terrain jointe avec cette base n'est pas unique. En effet, chaque lettrine est associée à un ensemble de métadonnées. Ces informations sont élaborées par les historiens du CESR. Parmi les métadonnées intéressantes à notre expérimentation, nous distinguons les deux suivantes [25] :

Fond : cette métadonnée précise le type du fond de la lettrine. L'ensemble des fonds dans la base sont : fond hachuré, fond criblé, fond blanc, fond noir et fond indéfini. La figure 7.7 illustre une lettrine de chaque de type de fond.

Type : cette métadonnée décrit le type de la lettrine. Dans la base il en existe deux types : les lettrines décoratives et les lettrines figuratives. La figure 7.8 illustre une lettrine de chaque type.

Nous considérons que ces informations sont intéressantes pour notre étude parce qu'il s'agit de métadonnées qui décrivent la structure de la lettrine. En effet, deux lettrines de fonds et de

25. Les autres métadonnées concerne l'initiale figurant dans la lettrine, la couleur (blanche ou noire), la police, l'imprimeur, la date et le lieu d'impression et le type d'alphabet de l'initiale.

(a) Une lettrine de la classe *Blanc*

(b) Une lettrine de la classe *Criblé*

(c) Une lettrine de la classe *Hachuré*

(d) Une lettrine de la classe *Noir*

FIGURE 7.7 – Exemple de lettrines de chaque type de fond

(a) Une lettrine décorative

(b) Une lettrine figurative

FIGURE 7.8 – Exemple de lettrines de chaque type

types différents auront, éventuellement, deux structures différentes. Ainsi, il est préférable que les lettrines de différents fonds (respectivement de différents types) appartiennent à des classes différentes. Ces connaissances nous fourniront une bonne référence pour évaluer la représentation des lettrines par les graphes, dans un contexte de classification supervisée. Dans notre expérimentation, nous construisons deux vérités terrain pour la base de Lettrines, une basée sur le fond et l'autre sur le type.

Pour classifier les lettrines, nous appliquons, dans un premier temps, l'algorithme de k plus proches voisins (k-ppv) dans le domaine de graphes. Ensuite, nous utilisons notre méthode de plongement de graphes dans l'espace euclidien pour plonger les graphes représentant les lettrines dans des vecteurs numériques. Cette manipulation nous permet, en premier lieu, d'utiliser le classificateur SVM (Machine à vecteurs de support) qui est connu pour être plus performant que le k-ppv. En deuxième lieu, les vecteurs de plongement nous permettent de comparer nos résultats avec les approches statistiques dans une même configuration d'expérimentation. En effet, dans la littérature, la comparaison entre les approches statistiques et les approches structurelles est très peu abordée. Cependant, nous sommes convaincu que cette comparaison est importante parce qu'elle nous fournit une quantification (empirique) de la différence de performances entre les deux grandes familles d'approches [124, 33, 175]. Nous choisissons d'utiliser le descripteur générique de Fourier (Generic Fourier Descriptor (GFD)), comme approche statistique, qui est basée sur la transformée de Fourier [285]. Le choix de ce descripteur est guidé par ces bonnes

Vérité terrain basée sur le fond

Classe	Blanc	Hachuré	Criblé	Noir	Indéfini
Nombre de lettrines	483	152	138	2	49

Vérité terrain basée sur le type de la lettrine

Classe	Décoratif	Figuratif
Nombre de lettrines	570	254

TABLE 7.1 – Répartition des classes de la base de lettrines

performances [259].

Dans cette expérimentation, 30% des graphes sont utilisés comme un ensemble d'apprentissage et le reste (70%) comme un ensemble de test. La sélection de ces ensembles est faite d'une manière totalement aléatoire. Vu que l'algorithme des k-ppv est très sensible au choix de l'ensemble d'apprentissage, nous avons répété l'exécution de chaque l'expérimentation 50 fois avec des ensembles d'apprentissage (et de test) différents. Les taux finaux sont la moyenne des taux obtenus dans les 50 exécutions.

Les taux de classification sont calculés selon les deux vérités terrain. Dans la table 7.1, nous présentons la taille des classes pour chaque vérité terrain de la base de lettrines. Nous rappelons que la représentation des graphes ne change pas d'une vérité à une autre, ce qui change concrètement ce sont les labels qui indiquent l'appartenance d'une lettrine à une classe.

Dans la classification avec les k-ppv, les distances entre les graphes sont calculées par notre méthode d'approximation de distance d'édition de graphes. Les distances dans l'espace vectoriel sont calculées par la distance euclidienne.

Pour récapituler, l'expérimentation consiste à évaluer les k-ppv dans le domaine de graphes avec les deux vérités terrain. Ensuite, nous évaluons, aussi avec les deux vérités terrain, les taux de classification dans l'espace vectoriel des vecteurs de plongement de graphes et des vecteurs caractéristiques du GFD avec les classificateurs k-ppv et SVM.

Les résultats sont présentés dans la table 7.2. Nous remarquons que les résultats accomplis avec la vérité terrain basée sur le fond de lettrines sont moins bons que ceux avec la vérité terrain basée sur le type. Ceci peut signifier que la représentation de Coustaty est mieux adaptée au type qu'au fond de lettrines. Néanmoins, le nombre réduit et les tailles des classes (seulement deux classes dont une contient $\approx$ 70% de la base) peuvent aussi influer positivement sur la performance de classification avec la vérité terrain basée sur le type. Nous constatons aussi que les performances de l'algorithme des k-ppv dans l'espace de graphes sont meilleures que ses performances dans l'espace vectoriel. Par contre, pour les deux vérités terrain considérées, les taux accomplis par le classificateur SVM avec les vecteurs plongés dépassent ($\simeq$ 5%) tous les autres résultats. Ceci justifie l'intérêt de l'utilisation du plongement de graphes pour améliorer les résultats. D'autre part, il est bien clair que les résultats du GFD sont moins bien que les résultats accomplies en utilisant les graphes avec ou sans le plongement. Par conséquent, nous pouvons conclure que les graphes sont mieux appropriés pour représenter les lettrines que les vecteurs caractéristiques résultant d'une approche statistique. Ceci est dû à la grande capacité

	k-ppv	SVM	k-ppv (Graphe)
Vérité terrain basée sur le fond			
Vecteurs de plongement de graphe %	61.65%	**70.38** %	65.53%
Descripteur GFD	53.14 % %	58.61%	-
Vérité terrain basée sur le type de la lettrine			
Vecteurs de plongement de graphe	65.16 %	**79,84**%	75.48%
Descripteur GFD	59.38%	66.18%	-

TABLE 7.2 – Résultats de la classification de la base de lettrines

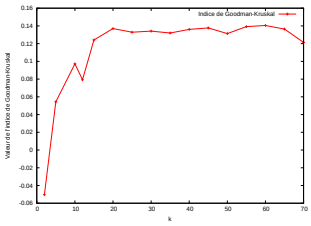
des graphes à représenter des quantités importantes d'informations dans une image, notamment de documents anciens.

7.5 Clustering de lettrines

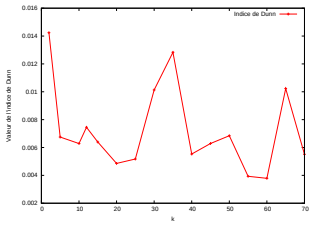
Dans cette section, nous nous intéressons à l'application de l'algorithme de graphe Median-shift pour le clustering des lettrines. Pour cela, nous considérons la base de lettrines où les lettrines sont représentées sous forme de graphes (cf. 7.2.2). En plus, nous comparons les performances de notre algorithme avec celles de l'algorithme de k-means. Les distances entre les graphes sont calculées par notre méthode d'approximation de distance d'édition de graphes. Nous évaluons la qualité du clustering par les indices de validation de $Dunn$ et de $Goodman$-$Kruskal$. Afin de produire les meilleurs résultats pour chaque algorithme, nous commençons par définir, empiriquement, le nombre de clusters k optimal pour le k-means, et le rayon h optimal pour notre algorithme. Pour ce faire, nous testons chaque algorithme plusieurs fois en fixant à chaque fois une nouvelle valeur du paramètre. Ensuite, le paramètre qui optimise la moyenne des deux indices de clustering est considéré comme le paramètre optimal. Nous utilisons la formule définie précédemment pour calculer la moyenne des indices (cf. chapitre 5). Les figures 7.9 et 7.10 présentent les courbes de chaque indice de clustering en fonction du paramètre de chaque algorithme. Dans ces figures, nous présentons aussi les deux courbes (Fig. 7.9(c) et 7.10(c)) de la moyenne de deux indices pour chaque algorithme.

Les résultats présentés dans la table 7.3 correspondent aux valeurs des indices de clustering de chaque algorithme en utilisant les paramètres optimaux déterminés précédemment. Les valeurs de l'indice de $Goodman$-$Kruskal$ indiquent que la partition de l'ensemble de lettrines fournie par notre algorithme est plus cohérente que celle fournie par le k-means. Autrement dit, le nombre de quadruplets de lettrines concordants est supérieur au nombre de quadruplets non-concordants. Rappelant qu'un quadruplet (q, r, s, t) est dit concordant si et seulement si les deux conditions suivantes sont satisfaites :

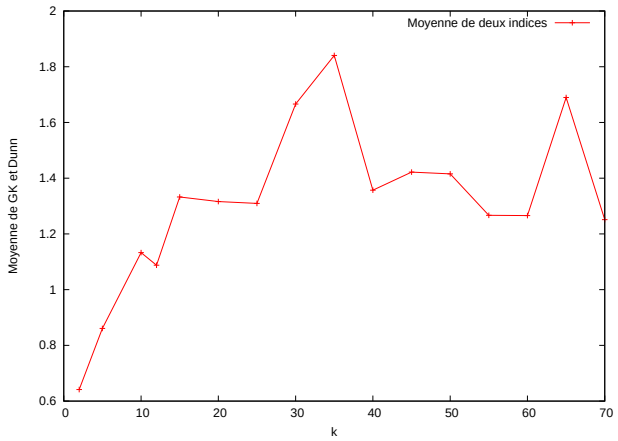
- $d(q, r) < d(s, t) \mid q$ et r appartiennent au même cluster ; s et t appartiennent à deux clusters différents.
- $d(q, r) > d(s, t) \mid q$ et r appartiennent aux deux clusters différents ; s et t appartiennent au même cluster.



(a) k-means : Valeur de l'indice de $Goodman\text{-}Kruskal$ en fonction de k

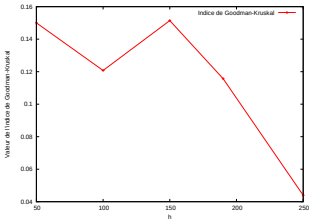


(b) k-means : Valeur de l'indice de $Dunn$ en fonction de k

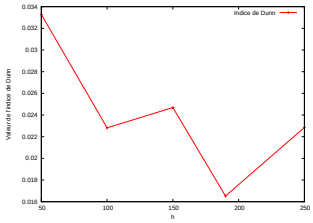


(c) k-means : Moyenne de deux indices en fonction de k

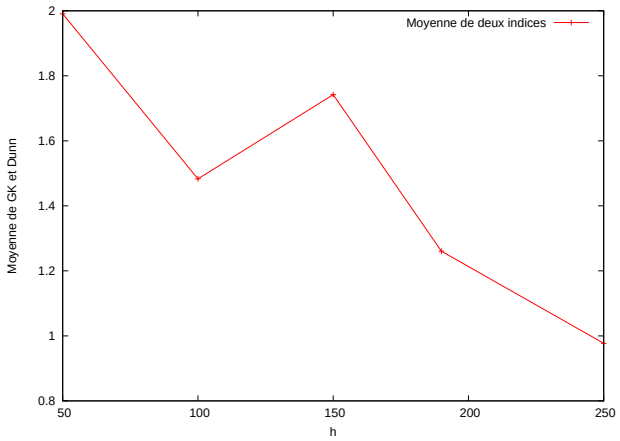
FIGURE 7.9 – Résultats de la validation du meilleur paramètre k pour l'algorithme des k-means



(a) Graphe Median-shift : Valeur de l'indice de $Goodman\text{-}Kruskal$ en fonction de h



(b) Graphe Median-shift : Valeur de l'indice de $Dunn$ en fonction de h



(c) Graphe Median-shift : Moyenne des deux indices en fonction de h

FIGURE 7.10 – Résultats de la validation du meilleur rayon h pour l'algorithme de graphe Median-shift

	GK-Index	Dunn Index
Graphe Median-shift	**0.15**	**0.033**
k-means	0.13	0.005

TABLE 7.3 – Résultats de clustering de la base de lettrines

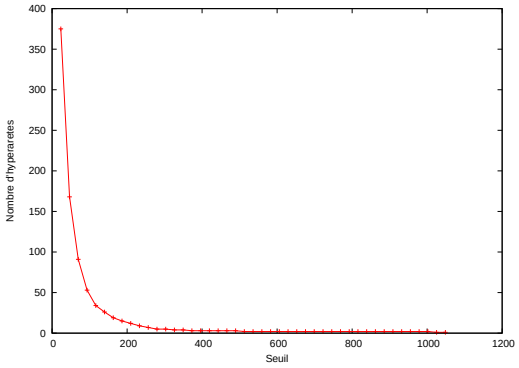
Notre méthode			Recherche classique	
Précision	Nombre d'appariements	Seuil	Précision	Nombre d'appariements
0.65	230.2	69.90	0.65	824

TABLE 7.4 – Résultats de recherche de lettrines

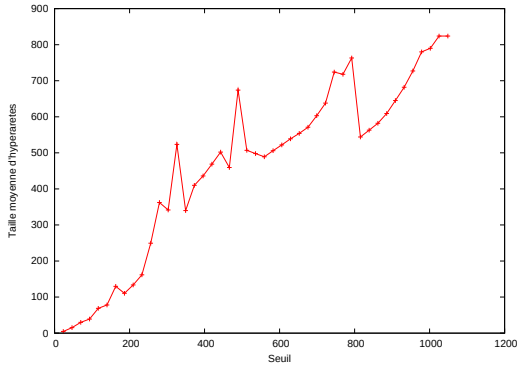
Ainsi, en utilisant notre algorithme, les distances entre les graphes appartenant à un cluster sont plus petites que les distances entre des graphes de différents clusters. Ceci est cohérent avec les résultats de l'indice de $Dunn$ qui sont aussi en faveur de notre algorithme. À partir de ces résultats, nous concluons qu'avec notre algorithme les distances inter-clusters sont plus grandes que les distances intra-clusters, ce qui correspond à un bon clustering (mieux que le partitionnement de k-means).

7.6 Recherche de lettrines

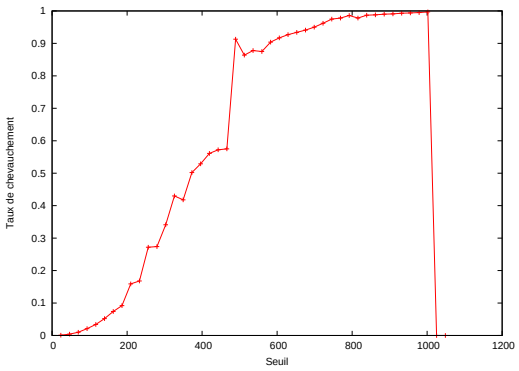
Dans cette section, nous nous focalisons à l'application de notre méthode de recherche d'images, basée sur la structure d'hypergraphe, sur la base de lettrines. Pour cela, nous considérons la base de lettrines où les lettrines sont représentées sous forme de graphes (cf. 7.2.2). Afin de choisir le seuil optimal pour notre méthode, nous effectuons des tests sur un ensemble de valeur du seuil pour choisir le meilleur. Nous rappelons que le seuil optimal correspond à une valeur de précision maximale et un nombre d'appariements minimal. Dans la figure 7.11, nous illustrons l'ensemble d'aspect de notre méthode en fonction de la valeur du seuil. Ces aspects sont : (a) le nombre d'hyperarêtes générées, (b) la taille moyenne des hyperarêtes, (c) le taux de chevauchement des hyperarêtes, (d) le nombre d'appariements de graphes et la (e) la précision des résultats de la recherche. À partir de ces résultats nous choisissons la valeur optimale du seuil qui maximise la précision et minimise le nombre d'appariements. En utilisant cette valeur du seuil, nous donnons, dans la table 7.4, les résultats obtenus par notre méthode en termes de précision et de nombre moyen d'appariements effectués ainsi que les résultats obtenus avec la recherche d'images classique. Notre méthode basée sur l'hypergraphe réalise la même précision (0.65) que la méthode classique. Par contre, notre méthode exécute en moyenne 230.2 appariements de graphes contre 824 appariements exécutés par la méthode classique. Nous concluons ainsi que notre méthode est efficace et rapide pour la recherche de lettrines.



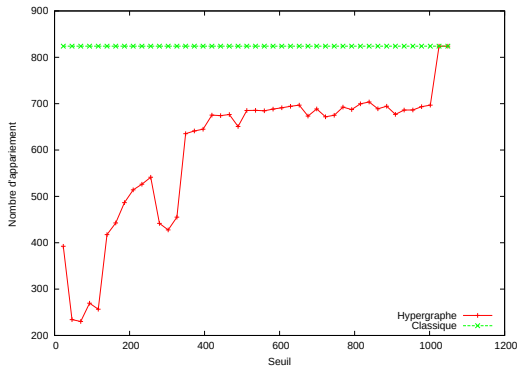
(a) Nombre d'hyperarêtes générées en fonction du seuil



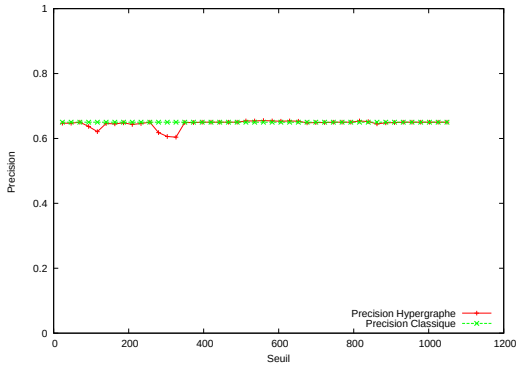
(b) Taille moyenne des hyperarêtes



(c) Taux de chevauchement des hyperarêtes



(d) Nombre d'appariements de graphes



(e) Précision de résultats de la recherche

FIGURE 7.11 – Comportement de la structure d'hypergraphe en fonction du seuil pour la base de Lettrines

7.7 Conclusion

Dans le cadre du projet NAVIDOMASS, nous avons appliqué l'ensemble de nos contributions dans le domaine des graphes sur une base de documents anciens. Cette base est un ensemble d'images de lettrines et est délivrée par le centre d'études supérieures de la Renaissance de l'université de Tours. Nous avons utilisé une technique d'extraction de graphes dédiée aux lettrines. Au cours des expérimentations présentées dans ce chapitre, nous avons montré l'intérêt de la représentation sous forme de graphes dans le domaine de la reconnaissance de documents anciens. Particulièrement, en utilisant notre technique de plongement de graphes, nous avons pu comparer notre approche structurelle avec une approche statistique (le descripteur GFD). Cette comparaison a montré que la description structurelle est plus performante que la description en vecteurs caractéristiques.

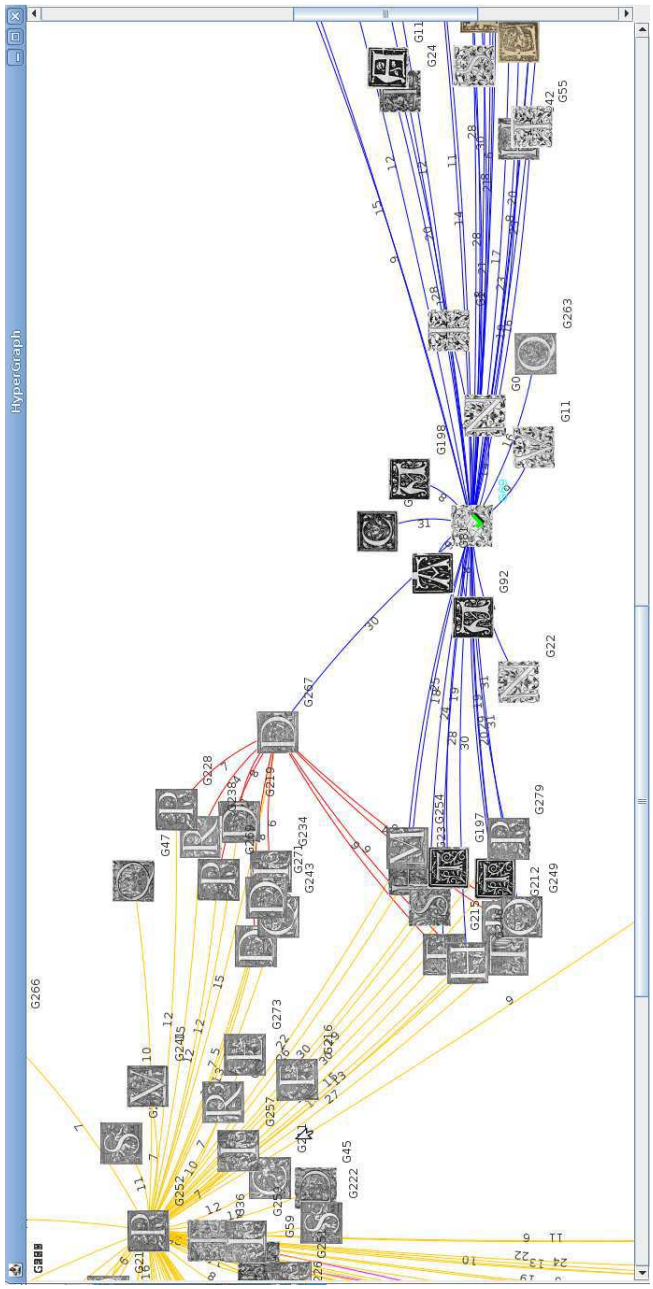


FIGURE 7.12 – Illustration d'une partie explorée de la base des lettrines

Chapitre 8

Conclusions et perspectives

If we knew what it was we were doing, it would not be called research, would it ?

Albert Einstein

Sommaire

Nous concluons cette thèse par un résumé de l'ensemble de nos contributions. Ensuite, nous donnons notre vision sur un ensemble de perspectives potentielles dans le domaine de la reconnaissance de formes à base de graphes.

8.1 Contexte

Cette thèse se situe dans le cadre de la reconnaissance de formes en utilisant les graphes comme modèles de représentation d'images. Un graphe est une structure de donnée composée d'un ensemble de noeuds qui sont éventuellement connectés par des arêtes. L'utilisation de graphes pour la modélisation d'images offre une grande capacité de représentation. Cette capacité de représentation est due à la capacité des graphes de représenter non seulement les caractéristiques mais aussi les relations qui peuvent exister entre les primitives (composantes) d'une image. Dans les approches non-structurelles, dites statistiques, chaque image est représentée par un nombre constant d'attributs numériques qui constituent un vecteur caractéristique de l'image. Par contre, en utilisant les graphes, il n y a aucune limitation sur la taille et aucune restriction sur le type d'attributs. En fait, la taille et le type d'attributs d'un graphe sont adaptés aux besoins

de chaque image. Ces propriétés sollicitent l'utilisation des graphes pour modéliser les images dont la structure joue un rôle dans la reconnaissance.

Néanmoins, dans le domaine de reconnaissance de formes, l'utilisation des graphes nécessite le développement de nouveaux formalismes mathématiques applicables dans le domaine des graphes. En effet, presque toutes les opérations mathématiques classiques de base sont limitées à une utilisation dans un espace vectoriel. Par exemple, le calcul d'une distance entre deux vecteurs est une opération classique et dispose de plusieurs solutions fondées sur des théories solides. Cependant, le calcul d'une distance entre deux graphes est un problème plus complexe, et considéré comme NP-complet. En effet, ces coûts élevés des opérations sur les graphes sont l'inconvénient majeur de la reconnaissance de formes à base de graphes.

8.2 Contributions

Nous avons commencé cette thèse (cf. chapitre 2) en examinant les principaux aspects de la représentation d'images sous forme de graphes. Cette analyse consiste en une catégorisation des méthodes de représentation structurelle d'images, largement utilisées dans la littérature, en quatre grandes familles. Ces familles sont les suivantes : les graphes de point d'intérêt, les graphes d'adjacence de régions, les graphes de relations spatiales et les graphes des squelettes. Nous avons remarqué que le choix d'une méthode dépend de la nature des images à représenter. Pour mieux argumenter nos observations, nous avons procédé à une étude empirique des représentations. Pour une base d'images, nous avons utilisé trois méthodes d'extraction des graphes pour en construire trois bases de graphes pour une même base d'images. Nous avons ensuite testé les performances de chaque représentation dans un contexte de classification (algorithme de k-nn). Nous n'avons pas limité notre expérimentation sur une méthode pour mesurer les distances entre les graphes. En effet, six méthodes de mesure de distance de graphes ont été considérées. Les résultats ont convergé vers nos observations. Nous avons pu aussi remarquer que le type de représentation choisi à un impact important sur les performances d'un système de reconnaissance de formes.

Nous avons commencé la partie suivante de la thèse par une étude bibliographique des mesures de similarité existantes pour les graphes. Ces méthodes d'appariement de graphes sont réparties en deux classes d'approches. La première classe contient les approches dites exactes où l'objectif est de déterminer un isomorphisme exact entre deux graphes. Ces méthodes sont très rigides et faiblement utilisées dans des applications du monde réel de la reconnaissance de formes. La rigidité de ces approches vient du fait que deux graphes sont considérés isomorphes si et seulement s'ils partagent exactement la même structure et les mêmes étiquettes (s'ils sont étiquetés). Pour réduire ce niveau de rigidité d'appariement, l'isomorphisme de sous-graphes a été introduit. Dans le contexte de l'isomorphisme de sous-graphes, un graphe peut être isomorphe à, seulement, une partie d'un autre graphe. En contrepartie, la deuxième classe des méthodes d'appariement de graphes contient les approches dites approximatives où l'objectif est de déterminer une distance entre deux graphes. L'appariement approximatif de graphes a été développé pour surmonter l'intolérance aux changements des structures et d'étiquettes et pour rendre l'appariement de graphes utilisable en pratique. Leur objectif est de chercher une distance entre deux graphes même s'ils ne partagent pas une (sous-)structure. Ainsi, l'appariement approximatif

de graphes tolère les différences dans les structures et les étiquettes des graphes considérés. Ces différences dans les structures et les étiquettes sont utilisées pour évaluer, généralement, la différence entre deux graphes par une distance. Parmi ces méthodes, la distance d'édition de graphes a été largement utilisée comme la mesure de similarité la plus appropriée pour estimer une distance entre deux graphes. L'idée de la distance d'édition de graphes est de définir la similarité de deux graphes par le nombre minimal d'opérations d'édition nécessaires pour transformer un graphe en un autre. Néanmoins, le calcul de la séquence d'opérations d'édition ayant le coût total minimum pour transformer un graphe en un autre implique une complexité caractérisée par un temps et un espace exponentiel.

8.2.1 Appariement de graphes

Dans le chapitre 4, nous avons proposé une nouvelle approche approximative pour rendre la distance d'édition de graphes moins coûteuse du point du vue du temps et de l'espace de calcul. Dans cette proposition, nous avons introduit la notion de signature de noeud, qui correspond à un ensemble de caractéristiques qui décrivent le noeud considéré dans le graphe. Chaque signature de noeud est composée par : les attributs du noeud, le degré du noeud, les attributs des arêtes incidentes et les degrés des noeuds adjacents. Ensuite, nous avons considéré que chaque graphe est un ensemble de signatures des noeuds qui le composent. L'appariement entre deux graphes a été alors transformé en un problème d'affection entre les deux ensembles de signatures représentants les graphes. Un problème d'affection est un des problèmes classiques de la recherche opérationnelle. Il consiste à trouver un couplage maximum (ou minimum) dans un graphe biparti. Autrement dit, il cherche le coût d'affectation minimal (ou maximal) entre les éléments de deux ensembles. Dans notre cas, les éléments de deux ensembles sont les signatures des noeuds et le coût est défini selon une distance entre ces signatures. Vu que les signatures peuvent contenir des attributs symboliques et/ou numériques, nous ne pouvons pas utiliser une distance classique. Pour ce faire, nous avons utilisé une distance, dite hétérogène, qui traite des vecteurs composés d'attributs numériques et symboliques et qui gère aussi l'absence d'attributs. Ainsi, la distance entre deux graphes est le coût d'affection majoré par les coûts d'ajouts des arêtes incidentes à des noeuds non-appariés.

Pour évaluer notre approximation de la distance d'édition de graphes, nous avons procédé à différentes expérimentations où à chaque fois nous avons produit une comparaison avec l'ensemble des méthodes alternatives connues dans la littérature. Le premier constat à l'observation des résultats est que notre méthode est robuste par rapport aux types de graphes (étiquetés ou non-étiquetés). Deuxièmement, nous avons constaté l'avantage de l'aspect non-paramétrique de notre méthode par rapport aux autres méthodes où les coûts d'édition sont des paramètres à apprendre.

8.2.2 Classification et indexation de graphes

Dans le domaine des graphes, la classification supervisée est limitée à l'utilisation de l'algorithme de k plus proches voisins et la classification non-supervisée (clustering) est limitée à l'utilisation de l'algorithme de k-moyennes (k-means). Cependant, ces deux algorithmes ne sont pas les plus performants parmi les algorithmes de classification de reconnaissance de formes. Dans le chapitre 5, nous avons proposé une nouvelle technique pour plonger les graphes dans

un espace euclidien. Ceci consiste à représenter chaque graphe par un vecteur de taille d qui peut être perçu comme un point dans un espace euclidien de dimension d. Cette technique de plongement (graph embedding) est une généralisation au domaine des graphes de la technique du *constant shift embedding* déjà proposée pour le plongement des protéines. Dans le chapitre 5, nous avons aussi proposé une contribution dans le domaine de la classification non-supervisée des graphes. Notre algorithme, appelé graphe Median-shift, est une adaptation du célèbre algorithme de *mean-shift* au domaine de graphes. Cette adaptation a nécessité l'utilisation de la notion du graphe médian qui correspond au graphe le plus représentatif d'un ensemble de graphes. L'algorithme de graphe Median-shift considère que les régions denses correspondent aux clusters et les régions à faible densité correspondent aux limites des clusters. Ainsi, en utilisant le graphe médian, pour chaque graphe, l'algorithme détecte à quelle région dense ce graphe appartient en remontant dans le sens de la pente positive de la densité. Ceci est équivalent à déplacer chaque graphe vers le représentant local de son voisinage, dans notre cas le graphe médian.

Les deux contributions proposées pour la classification ont fait l'objet d'une étude expérimentale qui ont montré de bonnes performances en comparaison avec une approche alternative.

Notre troisième contribution dans cette partie de la thèse correspond à une nouvelle méthode d'indexation de graphes (cf. chapitre 6). Cette méthode est basée sur la notion de l'hypergraphe où chaque sommet correspond à un graphe et les hyperarcs correspondent aux ensembles des graphes similaires. Nous avons proposé une méthode basée sur la sélection de prototypes pour le regroupement des graphes similaires et l'identification des représentants qui sont utilisés comme entrée d'index de la base de graphes. L'algorithme de sélection des prototypes proposé permet de définir automatiquement le nombre de graphes. En modélisant une base de graphes en une structure d'hypergraphe, nous permettons la multi-affectation d'un graphe, à savoir qu'un graphe peut être affecté à plusieurs groupes. Nous avons aussi montré que la structure d'hypergraphe améliore les résultats de la recherche et peut être utilisée pour naviguer dans une base de graphes.

8.2.3 Application des approches structurelles aux documents anciens

Cette thèse entre dans le cadre du projet ANR NAVIDOMASS qui a pour objectif le développement d'un ensemble de techniques qui permettent la navigation dans des documents anciens. Ces types de document sont caractérisés par leur complexité et leur dégradation. Dans le chapitre 7, nous nous sommes focalisés sur notre tâche d'indexation des lettrines définies dans le contexte de NAVIDOMASS. Une image représentant une lettrine est définie par un ensemble de composantes où chaque composante est porteuse d'information. Parmi ces composantes nous avons distingué le fond, l'initiale et les motifs décoratifs. Cette complexité de structure nous a incité à appliquer nos approches à base de graphes pour étudier l'intérêt de la représentation sous forme de graphes pour les documents anciens. L'ensemble des méthodes proposées dans cette thèse est appliqué à une base d'images de lettrines fournie par le centre d'études supérieures de la Renaissance. Les résultats ont montré que les graphes sont appropriés aux documents anciens. L'intérêt de la représentation sous forme de graphes des lettrines est validé par la comparaison des résultats avec les résultats accomplis par un descripteur statistique (GFD). Cette comparaison est faite en utilisant notre technique de plongement de graphes.

8.3 Perspectives

Nos travaux de recherche ne prétendent pas apporter toutes les réponses parfaites et indiscutables à une problématique complexe qui est celle de la reconnaissance d'images à base de graphes. Nous avons essentiellement proposé un ensemble de méthodes qui offrent une utilisation des graphes pour reconnaître les images, notamment les images de documents. Ces travaux pourraient être développés davantage sur certains points.

- Un premier développement consiste à proposer de nouvelles techniques et/ou de critères de sélection de dimension de l'espace euclidien pour l'approche du plongement de graphes. En effet, le choix actuel de la dimension est fait empiriquement. L'une des méthodes envisageables est celle de la sélection de caractéristiques afin d'améliorer les performances de classification [266].
- Le deuxième développement concerne la sélection du rayon optimal dans l'algorithme de graphe Median-Shift. Pour ce faire, nous pouvons nous baser sur les travaux concernant le rayon adaptatif [10, 50] dans l'algorithme de mean-shift.
- Une amélioration de notre méthode d'appariement pourrait être envisagée pour le calcul du sous-graphe commun maximum. Celui-ci serait intéressant dans plusieurs domaines d'applications telle que la localisation d'objets dans une image (*object spotting*).
- Nous pourrions aussi envisager une amélioration de notre méthode d'indexation en ajoutant une structure hiérarchique de centroïdes. Pour ce faire, Nous pouvons utiliser l'algorithme de Lee-Kwang et al. [151] qui consiste en une méthode de réduction d'hypergraphe en structure hiérarchique.
- Dans une optique de travail connexe, l'ensemble de nos contributions pourrait être appliqué dans le domaine de la fouille de données à base de graphes (*graph mining*). En ce sens, nous avons entamé une collaboration avec l'équipe *ORPAILLEUR* du *LORIA* pour traiter de plus grandes bases de molécules chimiques représentées sous forme de graphes.

Dans une perspective à long terme, il serait intéressant de proposer un protocole d'évaluation des performances dédié aux graphes. En effet, tous les travaux actuels utilisent des techniques d'évaluation dédiées aux approches statistiques qui ne sont pas forcément fiables pour les graphes, notamment pour le clustering. De plus, il serait intéressant de proposer un protocole de comparaison de performances des méthodes structurelles et statistiques. Ensuite, il serait aussi intéressant de réaliser un benchmark complet aussi bien pour les méthodes d'appariement de graphes que pour les techniques d'extraction de graphes à partir d'images. L'objectif d'un tel benchmark serait de proposer des recommandations sur les choix des méthodes. Ainsi, pour chaque type d'images, nous déterminerions la technique d'extraction de graphes et la méthode d'appariement les plus appropriées. Ceci est faisable d'une manière empirique en testant différentes approches, recensées comme robustes dans la littérature, avec plusieurs types d'expérimentations.

Annexe A

Exemple d'exécution de notre approximation de distance d'édition de graphes

Dans cette annexe, nous illustrons un exemple d'exécution de notre approximation de distance d'édition de graphes. Pour ce faire, nous considérons les deux graphes g_1 et g_2 de la figure A.1. Ces deux graphes sont étiquetés par des étiquettes numériques sur les arêtes et des étiquettes symboliques sur les noeuds. De plus, ces graphes partagent un sous-graphe commun composé par deux noeuds adjacents (les noeuds étiquetés respectivement par "a" et "b").

(a) (b)

FIGURE A.1 – Deux graphes à apparier : (a) g_1, (b) g_2

La première étape de notre méthode consiste à élaborer un ensemble de signatures de noeuds pour chaque graphe, i.e. $S_\gamma(g_1)$ et $S_\gamma(g_2)$. Nous rappelons que $S_\gamma(g)$ correspond à l'ensemble de signatures de noeuds du graphe g. Formellement,

$$S_\gamma(\, g\,) = \left\{\, \gamma(n_i) \mid \forall n_i \in V \,\right\}$$

$$\gamma(n) = \left\{\, \alpha_n^g,\ \theta_n^g,\ \Delta_n^g,\ \Omega_n^g \right\}$$

avec

- α_n^G : l'ensemble des attributs de n.
- θ_n^G : le degré de n.
- $\Delta_n^G = \{\theta_m^G \mid m \in V, m$ adjacent à $n\}$: l'ensemble des degrés des noeuds adjacents à n.
- $\Omega_n^G = \{\beta(e) \mid e \in E, e$ incidente à $n\}$: l'ensemble des attributs des arêtes incidentes à n.

Dans la table A.1, nous donnons les signatures de noeuds, respectivement, de deux graphes g_1 et g_2.

TABLE A.1 – $S_\gamma(g_1)$ et $S_\gamma(g_2)$: les ensembles des signatures de noeuds des graphes g_1 et g_2

$S_\gamma(g_1)$ = { { {"a"}, {1},{3},{1} }, { {"b"},{3},{2,2,1},{1,1,1} }, { {"c"},{2},{3,2},{1,1} }, { {"d"},{2},{3,2},{1,1} } }

$S_\gamma(g_2)$= { { {"a"}, {2},{3,3},{1,1} } { {"b"},{3},{3,2,2},{1,1,1} } { {"g"},{3},{3,2,2},{1,1,1} } { {"h"},{2},{3,3},{1,1} } }

Une fois les signatures de noeuds sont extraits, nous réduisons le problème d'appariement de graphes en un problème d'affection. Pour les deux graphes g_1 et g_2, le problème d'affectation est défini par un graphe biparti g = (X $\cup$ Y,X $\times$ Y), avec X=$S_\gamma(g_1)$ et Y=$S_\gamma(g_2)$. Les poids des arêtes xy $\in X \times Y$ sont les distances entre les signatures de noeuds. Ces distances sont calculées en utilisant la métrique euclidienne hétérogène de superposition (HEOM). Ainsi, une matrice de coûts est construite où chaque élément ij représente la distance entre la i-ème signature dans $S_\gamma(g_1)$ et la j-ième signature dans $S_\gamma(g_2)$. Cette matrice est donnée dans la table A.2.

Pour résoudre ce problème d'affection, nous utilisons la méthode hongroise [146] qui est

TABLE A.2 – La matrice de coûts entre $S_\gamma(g_1)$ et $S_\gamma(g_2)$

	{“a”}, {2},{3,3},{1,1}	{“b”},{3},{3,2,2},{1,1,1}	{“g”},{3},{3,2,2},{1,1,1}	{“h”},{2},{3,3},{1,1}
{“a”}, {1},{3},{1}	1.6	3.1	3.1	2.6
{“b”},{3},{2,2,1},{1,1,1}	2.1	0.4	1.4	2.1
{“c”},{2},{3,2},{1,1}	1.3	2.2	2.2	1.3
{“d”},{2},{3,2},{1,1}	1.3	2.2	2.2	1.3

connue comme la méthode la plus efficace et la plus rapide dans la littérature [39, 181]. Dans Algorithme 8, nous donnons un pseudo-code de la méthode hongroise.

Nous appliquons l'algorithme hongrois sur les deux ensembles $S_\gamma(g_1)$ et $S_\gamma(g_2)$ pour déterminer l'appariement optimal entre les noeuds de g_1 et g_2. La table A.3 illustre la première étape de l'algorithme où nous construisons une nouvelle matrice en soustrayant de chaque ligne de la matrice de coûts sa valeur minimale. Par exemple, la valeur minimale de la première ligne est “1.6”, ainsi la ligne devient (1.6-1.6=0, 3.1-1.6=1.5, 3.1-1.6=1.5, 2.6-1.6=1). Dans la deuxième étape (table A.4) nous soustrayons de chaque colonne de la nouvelle matrice sa valeur minimale, e.g. la première colonne reste inchangée parce que sa valeur minimale est “0”. Le résultat de cette étape est stocké dans une nouvelle matrice. Cette matrice est considérée dans la troisième étape où nous couvrons tous les zéros par le nombre minimal de lignes. Dans la table A.5, nous couvrons tous les zéros par quatre lignes (deux verticales et deux horizontales). Vu que le nombre total de lignes dans matrice est quatre aussi, donc nous passons à la cinquième étape. Dans cette dernière étape, nous déterminons la solution optimale d'appariement. Il faut choisir la solution parmi les zéros de la matrice de telle manière qu'on choisisse de chaque ligne et de chaque colonne un seul zéro. La table A.6 illustre la solution d'appariement optimale. À partir de ces résultats nous générons, dans la table A.7, la matrice de permutation P qui définie l'appariement optimal entre les deux graphes g_1 et g_2. L'appariement noeud-à-noeud entre les deux graphes est illustré dans la figure A.2.

En se basant sur la matrice P nous définissons la distance φ et nous re-écrivons la distance entre les ensembles de signatures comme suit :

Algorithme 8 Pseudo-code de l'algorithme hongrois

ENTRÉES: Une matrice de coûts

SORTIES: Une matrice de permutation

1: **Étape 1 :** Réduction des lignes : Trouver l'élément minimum dans chaque ligne de la matrice. Construire une nouvelle matrice en soustrayant de chaque coût le minimum dans sa ligne

2: **Étape 2 :** Réduction des colonnes : Trouver l'élément minimum dans chaque colonne de la matrice. Construire une nouvelle matrice en soustrayant de chaque coût le minimum dans sa colonne

3: **Étape 3 :** Tracer le nombre minimum de lignes (horizontales ou verticales) pour couvrir tous les zéros dans cette nouvelle matrice (appelée la matrice des coûts réduits). Si ce nombre est égal au nombre de lignes (ou colonnes), la matrice est réduite ; aller à l'**étape 5**. Si ce nombre est inférieur au nombre de lignes (ou colonnes), aller à l'**étape 4**.

4: **Étape 4 :** Trouver l'élément de valeur minimum non-couvert par une ligne à l'**étape 2**. Soustraire cette valeur de tous les éléments non-couverts. Ajouter cette valeur aux éléments situés à l'intersection de deux lignes. Retourner à l'**étape 3**.

5: **Étape 5 :** Déterminer la solution optimale. Générer la matrice binaire de permutation qui définie l'affectation optimale.

En se basant sur la matrice P, la distance φ entre les ensembles de signatures est donnée par :

$$\varphi(S_\gamma(g_1), S_\gamma(g_2)) = 1.6 + 0.4 + 2.2 + 1.3 = 5.5$$

Outre la distance $\varphi(S_\gamma(g_1), S_\gamma(g_2))$, nous considérons aussi , dans notre distance d'édition, les opérations effectuées sur les arêtes. Dans l'appariement entre g_1 et g_2, il y a quatre substitutions des arêtes et un ajout d'une arête dans g_1. Vu que toutes les arêtes de g_1 et g_2 ont la même étiquette, donc les coûts de substitution des arêtes sont zéro (i.e. HEOM(1,1)=0). Ainsi donc, la distance d'édition est la somme de $\varphi(S_\gamma(g_1), S_\gamma(g_2))$ et le coût de l'ajout d'une arête ((ag) dans g_1).

La distance d'édition entre g_1 et g_2 est :

$$d(g_1, g_2) = \varphi(S_\gamma(g_1), S_\gamma(g_2)) + HEOM(0, 1) = 5.5 + 1 = 6.5$$

TABLE A.3 – Application de la méthode hongroise : étape 1

	{{“a”}, {2},{3,3},{1,1}}	{{“b”},{3},{3,2,2},{1,1,1}}	{{“g”},{3},{3,2,2},{1,1,1}}	{{“h”},{2},{3,3},{1,1}}	Minimum
{{“a”}, {1},{3},{1}}	0	1.5	1.5	1	**1.6**
{{“b”},{3},{2,2,1},{1,1,1}}	1.7	0	1	1.7	**0.4**
{{“c”},{2},{3,2},{1,1}}	0	0.9	0.9	0	**1.3**
{{“d”},{2},{3,2},{1,1}}	0	0.9	0.9	0	**1.3**

TABLE A.4 – Application de la méthode hongroise : étape 2

	{{“a”}, {2},{3,3},{1,1}}	{{“b”},{3},{3,2,2},{1,1,1}}	{{“g”},{3},{3,2,2},{1,1,1}}	{{“h”},{2},{3,3},{1,1}}
{{“a”}, {1},{3},{1}}	0	1.5	0.6	1
{{“b”},{3},{2,2,1},{1,1,1}}	1.7	0	0.1	1.7
{{“c”},{2},{3,2},{1,1}}	0	0.9	0	0
{{“d”},{2},{3,2},{1,1}}	0	0.9	0	0
Minimum	**0**	**0**	**0.9**	**0**

TABLE A.5 – Application de la méthode hongroise : étape 3

	{"a"}, {2},{3,3},{1,1} }	{"b"},{3},{3,2,2},{1,1,1} }	{"g"},{3},{3,2,2},{1,1,1} }	{"h"},{2},{3,3},{1,1} }
{"a"}, {1},{3},{1} }	0	1.5	0.6	1
{"b"},{3},{2,2,1},{1,1,1} }	1.7	0	0.1	1.7
{"c"},{2},{3,2},{1,1} }	0	0.9	0	0
{"d"},{2},{3,2},{1,1} }	0	0.9	0	0

TABLE A.6 – Application de la méthode hongroise : étape 5

	{"a"}, {2},{3,3},{1,1} }	{"b"},{3},{3,2,2},{1,1,1} }	{"g"},{3},{3,2,2},{1,1,1} }	{"h"},{2},{3,3},{1,1} }
{"a"}, {1},{3},{1} }	0	1.5	0.6	1
{"b"},{3},{2,2,1},{1,1,1} }	1.7	0	0.1	1.7
{"c"},{2},{3,2},{1,1} }	0	0.9	0	0
{"d"},{2},{3,2},{1,1} }	0	0.9	0	0

TABLE A.7 – La matrice de permutation entre $S_\gamma(g_1)$ et $S_\gamma(g_2)$

	{“a”}, {2},{3,3},{1,1}	{“b”},{3},{3,2,2},{1,1,1}	{“g”},{3},{3,2,2},{1,1,1}	{“h”},{2},{3,3},{1,1}
{“a”}, {1},{3},{1}	1	0	0	0
{“b”},{3},{2,2,1},{1,1,1}	0	1	0	0
{“c”},{2},{3,2},{1,1}	0	0	1	0
{“d”},{2},{3,2},{1,1}	0	0	0	1

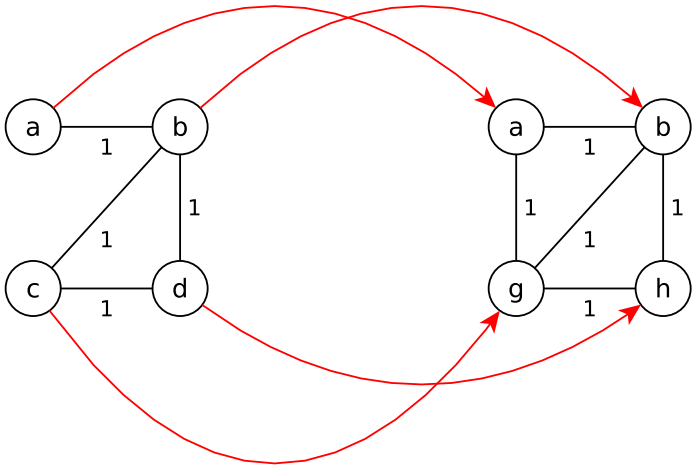


FIGURE A.2 – L’appariement entre g_1 et g_2

Annexe B

GenGraph : un générateur de graphes

GenGraph est un générateur aléatoire de graphes que nous avons mis en oeuvre. Cet outil génère un ensemble de graphes en se basant sur des paramètres fixés par l'utilisateur. Les paramètres considérés sont les suivants :

- Le nombre de graphes à générer.
- La taille maximale t_{max}(nombre de noeuds) des graphes générés. Ce paramètre limite, uniquement, la taille maximale. En effet, les graphes générés ont une taille aléatoire entre 2 et t_{max}.
- Les noms des attributs ainsi que leurs valeurs possibles.

La figure B.1 présente une capture d'écran de l'interface du GenGraph. Les graphes sont générés en format GXL [272] qui est une variante de XML dédiée au graphes.

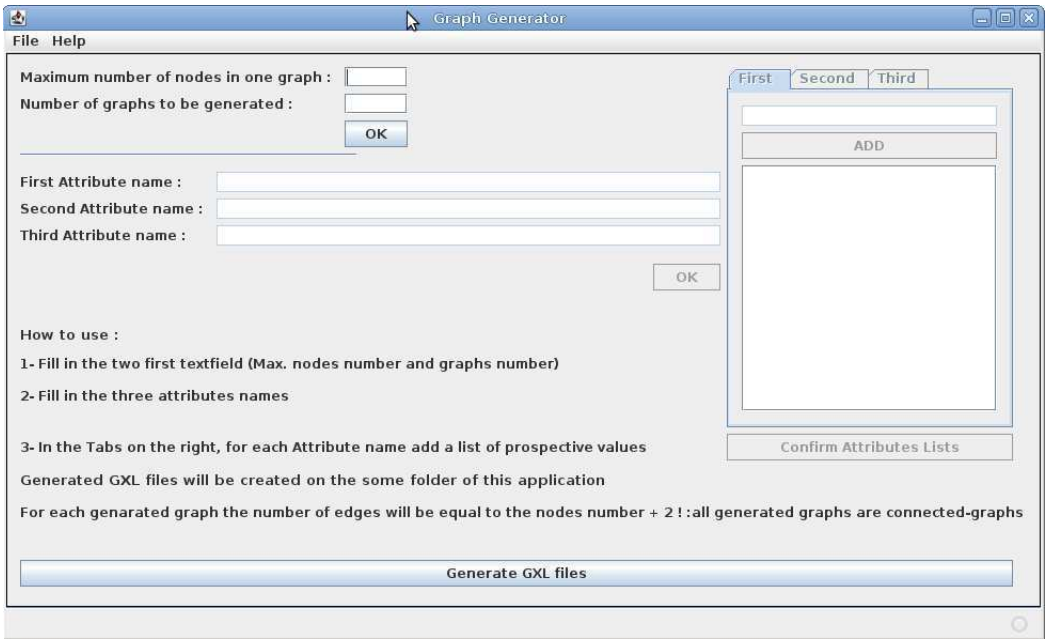


FIGURE B.1 – Une capture d'écran de l'interface du GenGraph

Exemple d'un graphe généré par GenGraph :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl>
        <graph id="id0">
                <node id="noeud0">
                        <attr name="type">
                                <string>Cercle</string>
                        </attr>
                        <attr name="couleur">
                                <string>Marron</string>
                        </attr>
                        <attr name="taille">
                                <string>Moyen</string>
                        </attr>
                </node>
                <node id="noeud1">
                        <attr name="type">
                                <string>Rectangle</string>
                        </attr>
                        <attr name="couleur">
                                <string>Blanc</string>
                        </attr>
                        <attr name="taille">
                                <string>Grand</string>
                        </attr>
                </node>
                <node id="noeud2">
                        <attr name="type">
                                <string>Rectangle</string>
                        </attr>
                        <attr name="couleur">
                                <string>Rouge</string>
                        </attr>
                        <attr name="taille">
                                <string>Grand</string>
                        </attr>
                </node>
                <node id="noeud3">
                        <attr name="type">
                                <string>Losange</string>
                        </attr>
                        <attr name="couleur">
                                <string>Blanc</string>
                        </attr>
                        <attr name="taille">
                                <string>Moyen</string>
```

```
                </attr>
            </node>
            <node id="noeud4">
                <attr name="type">
                    <string>Hexagone</string>
                </attr>
                <attr name="couleur">
                    <string>Marron</string>
                </attr>
                <attr name="taille">
                    <string>Petit</string>
                </attr>
            </node>
            <node id="noeud5">
                <attr name="type">
                    <string>Cercle</string>
                </attr>
                <attr name="couleur">
                    <string>Gris</string>
                </attr>
                <attr name="taille">
                    <string>Petit</string>
                </attr>
            </node>
            <edge from="noeud0" to="noeud3"/>
            <edge from="noeud0" to="noeud4"/>
            <edge from="noeud0" to="noeud1"/>
            <edge from="noeud5" to="noeud3"/>
            <edge from="noeud2" to="noeud0"/>
            <edge from="noeud2" to="noeud5"/>
            <edge from="noeud5" to="noeud4"/>
            <edge from="noeud5" to="noeud0"/>
        </graph>
</gxl>
```

Annexe C

Caractéristiques sur les bases de graphes

TABLE C.1 – Nombre de noeuds dans les bases de graphes utilisées

Base	Nombre Moyen	Min	Max
Lettrine	98.30	36	280
GREC 1 & 2	11.54	4	25
Shape : C+D	343.11	216	709
Shape : H+D	11.51	5	25
Shape : Skeleton	26.60	15	43
Logo : H+D	102.53	28	292
Logo : RAG	9.78	2	81
Logo : Skeleton	49.33	13	183
Letter	4.7	2	9
Mutagenicity	30.38	113	417

Bibliographie

[1] D. W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2) :267–287, 1992.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6 :37–66, 1991.

[3] A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[4] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5) :522–525, 1993.

[5] Alt and Guibas. Discrete geometric shapes : Matching, interpolation, and approximation. In H. of Computational Geometry, J.-R. Sack, and J. Urrutia, editors, *Elsevier, 2000*. 2000.

[6] R. Ambauen, S. Fischer, and H. Bunke. Graph edit distance with node splitting and merging, and its application to diatom idenfication. In E. R. Hancock and M. Vento, editors, *GbRPR*, volume 2726 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2003.

[7] J.-F. Aujol and A. Chambolle. Dual norms and image decomposition models. *International Journal of Computer Vision*, 63(1) :85–104, 2005.

[8] J.-F. Aujol, G. Gilboa, T. Chan, and S. Osher. Structure-texture image decomposition - modeling, algorithms, and parameter selection. *International Journal of Computer Vision*, 67(1) :111–136, 2006.

[9] S. Auwatanamongkol. Inexact graph matching using a genetic algorithm for image recognition. *Pattern Recognition Letters*, 28(12) :1428–1437, 2007.

[10] S. P. Awate and R. T. Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3) :364–376, 2006.

[11] T. R. Babu and M. N. Murty. Comparaison of genetic algorithm based prototype selection schemes. *Pattern Recognition*, 34 :523–525, 2001.

[12] R. Baeza-Yates and G. Valiente. An image similarity measure based on graph matching. *Proc. International Symposium on String Processing Information Retrieval*, pages 28–38, 2000.

[13] A. Balaban. Highly discriminating distance-based topological index. *Chemical Physics Letters*, 89(5) :399–404, 1982.

[14] E. Balas and C. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15 :1054, 1986.

[15] H. G. Barrow and R. M. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Inf. Process. Lett.*, 4(4) :83–84, 1976.

[16] H. G. Barrow and R. Popplestone. Relational descriptions in picture processing. *In Machine Intelligence*, 4 :377–396, 1971.

[17] R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indices. *Acta Informatica*, 1 :173–189, 1972.

[18] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[19] C. Berge. *Graphes et Hypergraphes*. Paris Dunod, 1970.

[20] C. Berge. *Graphs and Hypergraphs*. Elsevier Science Ltd, 1985.

[21] A. T. Berztiss. A backtrack procedure for isomorphism of directed graphs. *Journal of the ACM*, 20(3) :365–377, 1973.

[22] K. M. Borgwardt. *Graph Kernels*. PhD thesis, Ludwig–Maximilians University, Munich, 2007.

[23] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005)*, pages 74–81, Washington, DC, USA, 2005. IEEE Computer Society.

[24] K. M. Borgwardt, C. S. Ong, S. Schãnauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1 :i47–56, June 2005.

[25] S. R. Bulò, A. Torsello, and M. Pelillo. A game-theoretic approach to partial clique enumeration. *Image Vision Comput.*, 27(7) :911–922, 2009.

[26] H. Bunke. Attributed of programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6) :574–582, Novembre 1982.

[27] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8) :689–694, 1997.

[28] H. Bunke. Recent developments in graph matching. In *International Conference on Pattern Recognition*, pages 117–124 vol.2, 2000.

[29] H. Bunke. Recent advances in structural pattern recognition with applications to visual form analysis. In C. Arcelli, L. P. Cordella, and G. S. di Baja, editors, *IWVF*, volume 2059 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2001.

[30] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4) :245–253, 1983.

[31] H. Bunke, P. Foggia, C. Guidobaldi, and M. Vento. Graph clustering using the weighted minimum common supergraph. *IAPR Workshop GbRPR 2003, LNCS 2726*, pages 235–246, 2003.

[32] H. Bunke and S. Günter. Weighted mean of a pair of graphs. *Computing*, 67(3) :209–224, 2001.

[33] H. Bunke, S. Günter, and X. Jiang. Towards bridging the gap between statistical and structural pattern recognition : Two new concepts in graph matching. In *ICAPR, LNCS 2013*, pages 1–11, 2001.

[34] H. Bunke, X. Jiang, and A. Kandel. On the minimum common supergraph of two graphs. *Computing*, 65(1) :13–25, 2000.

[35] H. Bunke and A. Kandel. Mean and maximum common subgraph of two graphs. *Pattern Recognition Letters*, 21, 2000.

[36] H. Bunke and K. Riesen. Graph classification based on dissimilarity space embedding. In *IAPR International Workshop, SSPR & SPR 2008, LNCS 5342*, pages 996–1007, 2008.

[37] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19 :255–259, 1998.

[38] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2) :121–167, 1998.

[39] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. Society for Industrial Mathematics, 2009.

[40] T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4) :515–519, 2004.

[41] T. Caelli and S. Kosinov. Inexact graph matching using eigen-subspace projection clustering. *IJPRAI*, 18(3) :329–354, 2004.

[42] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics Simulation and Computation*, 3(1) :1–27, 1974.

[43] M. Carcassoni and E. R. Hancock. Correspondence matching with modal clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12) :1609–1615, 2003.

[44] R. M. Cesar, E. Bengoetxea, and I. Bloch. Inexact graph matching using stochastic optimization techniques for facial feature recognition. In *International Conference on Pattern Recognition - Volume II*, pages 465–468, 2002.

[45] P.-A. Champin and C. Solnon. Measuring the similarity of labeled graphs. In K. D. Ashley and D. G. Bridge, editors, *ICCBR*, volume 2689 of *Lecture Notes in Computer Science*, pages 80–95. Springer, 2003.

[46] J.-T. Chien and C.-C. Wu. Discriminant waveletfaces and nearest feature classifiers for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12) :1644–1649, 2002.

[47] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8) :749–764, 1995.

[48] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[49] D. Comaniciu, P. Meer, and S. Member. Mean shift : A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 :603–619, 2002.

[50] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *ICCV*, pages 438–445, 2001.

[51] D. Comer. The ubiquitous b-tree. *ACM Computing Surveys*, 11(2) :121–137, 1979.

[52] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3) :265–298, 2004.

[53] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Evaluating Performance of the VF Graph Matching Algorithm. In *Proc. of the 10th International Conference on Image Analysis and Processing, IEEE Computer Society Press*, pages 1172–1177, 1999.

[54] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Fast graph matching for detecting cad image components. In *15th International Conference on Pattern Recognition, 2000. Proceedings*, volume 2, 2000.

[55] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10) :1367–1372, 2004.

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

[57] C. Cortes and V. Vapnik. Support–vector networks. *Machine Learning*, 20, 1995.

[58] M. Coustaty, G. NGuyen, V. Courboulay, and J.-M. Ogier. Approche complexe de l'-analyse de documents anciens. In S. B. Yahia and J.-M. Petit, editors, *EGC*, volume RNTI-E-19 of *Revue des Nouvelles Technologies de l'Information*, pages 597–609, 2010.

[59] M. Coustaty, J.-M. Ogier, R. Pareti, and N. Vincent. Drop caps decomposition for indexing a new letter extraction method. In *ICDAR*, pages 476–480. IEEE Computer Society, 2009.

[60] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1) :21–27, 1967.

[61] T. F. Cox and M. A. Cox. *Multidimensional scaling*. Chapman and Hall, 2001.

[62] A. D. J. Cross, R. C. Wilson, and E. R. Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6) :953–970, 1997.

[63] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval : Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2) :1–60, 2008.

[64] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2) :224–227, 1979.

[65] J. Devillers and A. Balaban. *Topological indices and related descriptors in QSAR and QSPR*. CRC, 2000.

[66] E. Deza and M. Deza. *Dictionary of distances*. Elsevier Science Ltd, 2006.

[67] D. S. Doermann, E. Rivlin, and I. Weiss. Logo Recognition. Technical Report CS-TR-3145, University of Maryland, College Park, College Park, MD, 1993.

[68] P. Dosch and E. Valveny. Report on the second symbol recognition contest. In W. Liu and J. Lladós, editors, *GREC*, volume 3926 of *Lecture Notes in Computer Science*, pages 381–397. Springer, 2005.

[69] S. Dubois, M. Lugiez, R. Péteri, and M. Ménard. Adding a noise component to a color decomposition model for improving color texture extraction. In *4th European Conference on Colour in Graphics, Imaging, and Vision, CGIV2008*, pages 394–398, 2008.

[70] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[71] J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4 :95–104, 1974.

[72] F.-X. Dupé and L. Brun. Edition within a graph kernel framework for shape recognition. In A. Torsello, F. Escolano, and L. Brun, editors, *GbRPR*, volume 5534 of *Lecture Notes in Computer Science*, pages 11–20. Springer, 2009.

[73] F.-X. Dupé and L. Brun. Shape classification using a flexible graph kernel. In X. Jiang and N. Petkov, editors, *CAIP*, volume 5702 of *Lecture Notes in Computer Science*, pages 705–713. Springer, 2009.

[74] M. J. Egenhofer and A. R. B. M. Shariff. Metric details for natural-language spatial relations. *ACM Transactions on Information Systems*, 16(4) :295–321, 1998.

[75] D. Emms, S. Severini, R. C. Wilson, and E. R. Hancock. Coined quantum walks lift the cospectrality of graphs and trees. *Pattern Recognition*, 42(9) :1988–2002, 2009.

[76] D. Emms, R. C. Wilson, and E. R. Hancock. Graph embedding using quantum commute times. In F. Escolano and M. Vento, editors, *GbRPR*, volume 4538 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 2007.

[77] R. Englert and R. Glantz. Towards the clustering of graphs. *IAPR Workshop GbRPR 1999, Austrian Computer Society*, pages 125–133, 1999.

[78] M. Eshera and K. Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 14(3) :398–408, May 1984.

[79] M. Eshera and K. Fu. A similarity measure between attributed relational graphs for image analysis. In *International Conference on Pattern Recognition*, pages 75–77, 1984.

[80] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), Septembre 2004.

[81] M.-L. Fernandez and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6/7) :753–758, 2001.

[82] M. Ferrer, F. Serratosa, and A. Sanfeliu. Synthesis of median spectral graph. *IbPRIA, LNCS 3523*, pages 139–146, 2005.

[83] M. Ferrer, E. Valveny, F. Serratosa, I. BardajÃÂ, and H. Bunke. Graph-based k-means clustering : A comparison of the set median versus the generalized median graph. In X. Jiang and N. Petkov, editors, *CAIP*, volume 5702 of *Lecture Notes in Computer Science*, pages 342–350. Springer, 2009.

[84] M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, and H. Bunke. Generalized median graph computation by means of graph embedding in vector spaces. *Pattern Recognition*, 43(4) :1642–1655, 2010.

[85] A. Fischer, K. Riesen, and H. Bunke. An experimental study of graph classification using prototype selection. In *International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[86] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 100(22) :67–92, 1973.

[87] P. Foggia, C. Sansone, and M. Vento. A performance comparison of five algorithms for graph isomorphism. In *Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pages 188–199. Citeseer, 2001.

[88] P. Foggia and M. Vento. Graph embedding for pattern recognition. In D. Ünay, Z. Çataltepe, and S. Aksoy, editors, *Recognizing Patterns in Signals, Speech, Images, and Videos - ICPR 2010 Contests*, volume 6388 of *Lecture Notes in Computer Science*, pages 75–82. Springer, 2010.

[89] H. Fröhlich, J. K. Wegner, F. Sieker, and A. Zell. Optimal assignment kernels for attributed molecular graphs. In L. D. Raedt and S. Wrobel, editors, *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 225–232. ACM, 2005.

[90] D. Gaceb, V. Eglin, F. Lebourgeois, and H. Emptoz. Improvement of postal mail sorting system. *International Journal on Document Analysis and Recognition*, 11(2) :67–80, 2008.

[91] M. R. Garey and D. S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. W.J. Freeman and Company, San Francisco, CA, 1979.

[92] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels : Hardness results and efficient alternatives. In B. Schölkopf and M. K. Warmuth, editors, *COLT*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2003.

[93] T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3) :205–232, 2004.

[94] C. Geng and X. Jiang. Face recognition using sift features. In *International Conference on Image Processing*, pages 3313–3316. IEEE, 2009.

[95] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1) :103–112, 2005.

[96] D. Ghahraman, A. Wong, and T. Au. Graph optimal monomorphism algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, 10 :181–188, 1980.

[97] C. Giraud-Carrier and T. Martinez. An efficient metric for heterogeneous inductive learning applications in the attribute-value language. *Intelligent Systems*, pages 341–350, 1995.

[98] R. Giugno and D. Shasha. Graphgrep : A fast and universal method for querying graphs. In *International Conference on Pattern Recognition - Volume II*, pages 112–115, 2002.

[99] W.-B. Goh. Strategies for shape matching using skeletons. *Computer Vision and Image Understanding*, 110(3) :326–345, 2008.

[100] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18 :377–388, 1996.

[101] C. Gomila and F. Meyer. Graph-based object tracking. In *International Conference on Image Processing - Volume II*, pages 41–44, 2003.

[102] L. Goodman and W. Kruskal. Measures of association for cross-classifications. *J. American Statististical Association*, 1954.

[103] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7) :1100–1111, 2005.

[104] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5) :855–868, 2009.

[105] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. *Journal of Machine Learning Research*, 1 :1–10, 2008.

[106] S. Günter and H. Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23(4) :405–417, 2002.

[107] E. R. Hancock and J. Kittler. Discrete relaxation. *Pattern Recognition*, 23(7) :711–733, 1990.

[108] C. Harris and M. Stephens. A combined corner and edge detection. *Proc. 4th Alvey Vision Conf.*, pages 189–192, 1988.

[109] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Man and Cybernetics*, 4(2) :100–107, 1968.

[110] D. O. Hebb. *The Organization of Behavior : A Neuropsychological Theory*. John Wiley, New York, 1949.

[111] A. Hlaoui and S. Wang. A new median graph algorithm. *IAPR Workshop on GbRPR, LNCS 2726*, pages 225–234, 2003.

[112] Z. Hong, Q. Jiang, and S. Wang. Measuring overlap-rate in a hierarchical approach for color image segmentation. In *ICICIC '07 : Proceedings of the Second International Conference on Innovative Computing, Informatio and Control*, page 554, Washington, DC, USA, 2007. IEEE Computer Society.

[113] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *STOC '74 : Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 172–184, New York, NY, USA, 1974. ACM.

[114] S.-M. Hsieh and C.-C. Hsu. Graph-based representation for similarity retrieval of symbolic images. *Data Knowledge Engineering*, 65(3) :401–418, 2008.

[115] L. Hubert and J. Schultz. Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29(1) :190–241, 1976.

[116] B. Huet and E. R. Hancock. Shape recognition from large image libraries by inexact graph matching. *Pattern Recognition Letters*, 20(11-13) :1259–1269, 1999.

[117] B. Huet and E. R. Hancock. Relational object recognition from large structural libraries. *Pattern Recognition*, 35(9) :1895–1915, 2002.

[118] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37 :547–579, 1901.

[119] A. K. Jain, R. P. Duin, and J. Mao. Statistical pattern recognition : A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 :4–37, 2000.

[120] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : a review. *ACM Comput. Surv.*, 31(3) :264–323, 1999.

[121] B. J. Jain and F. Wysotzki. Solving inexact graph isomorphism problems using neural networks. *Neurocomputing*, 63 :45–67, 2005.

[122] X. Jiang and H. Bunke. Optimal lower bound for generalized median problems in metric space. *IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR), Windsor, Canada, LNCS 2396*, pages 143–151, 2002.

[123] T. Joachims and F. Sebastiani. Special issue on automated text categorization. *Journal of Intelligent Information Systems*, 18(2-3), 2002.

[124] J.-M. Jolion. Some experiments on clustering a set of strings. In E. R. Hancock and M. Vento, editors, *GbRPR*, volume 2726 of *Lecture Notes in Computer Science*, pages 214–224. Springer, 2003.

[125] I. T. Jolliffe. *Principal component analysis*. Springer Verlag, New York, 1986.

[126] Josep Lladòs, Enric Martí, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10) :1137–1143, 2001.

[127] S. Jouili, I. Mili, and S. Tabbone. Attributed graph matching using local descriptions. In J. Blanc-Talon, W. Philips, D. C. Popescu, and P. Scheunders, editors, *ACIVS*, volume 5807 of *Lecture Notes in Computer Science*, pages 89–99. Springer, 2009.

[128] S. Jouili and S. Tabbone. Applications des graphes en traitement d'images. In *International Conference on Relations, Orders and Graphs : Interaction with Computer Science, ROGICS'08*, pages 434–442, 2008.

[129] S. Jouili and S. Tabbone. Graph matching based on node signatures. In A. Torsello, F. Escolano, and L. Brun, editors, *GbRPR*, volume 5534 of *Lecture Notes in Computer Science*, pages 154–163. Springer, 2009.

[130] S. Jouili and S. Tabbone. A hypergraph-based model for graph clustering : Application to image indexing. In X. Jiang and N. Petkov, editors, *CAIP*, volume 5702 of *Lecture Notes in Computer Science*, pages 360–368. Springer, 2009.

[131] S. Jouili and S. Tabbone. Graph embedding using constant shift embedding. In D. Ünay, Z. Çataltepe, and S. Aksoy, editors, *Recognizing Patterns in Signals, Speech, Images, and Videos - ICPR 2010 Contests*, volume 6388 of *Lecture Notes in Computer Science*, pages 83–92. Springer, 2010.

[132] S. Jouili, S. Tabbone, and V. Lacroix. Median graph shift : A new clustering algorithm for graph domain. In *International Conference on Pattern Recognition (ICPR'10)*, pages 950–953, 2010.

[133] S. Jouili, S. Tabbone, and E. Valveny. Comparing graph similarity measures for graphical recognition. In J.-M. Ogier, W. Liu, and J. Lladós, editors, *GREC*, volume 6020 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2010.

[134] D. Justice and A. O. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8) :1200–1214, 2006.

[135] Kann. On the approximability of the maximum common subgraph problem. In *STACS : Annual Symposium on Theoretical Aspects of Computer Science*, 1992.

[136] A. Karray, J.-M. Ogier, S. Kanoun, and M. A. Alimi. An ancient graphic documents indexing method based on spatial similarity. In W. Liu, J. Lladós, and J.-M. Ogier, editors, *GREC*, volume 5046 of *Lecture Notes in Computer Science*, pages 126–134. Springer, 2007.

[137] H. Kashima and A. Inokuchi. Kernels for graph classification. In *ICDM Workshop on Active Mining*, volume 2002, page 25, 2002.

[138] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 321–328. AAAI Press, Chicago, IL, USA, 2003.

[139] L. Kier and L. Hall. *Molecular connectivity in chemistry and drug research*. Academic Press New York, 1976.

[140] S. Kim, Y. J. Park, K.-A. Toh, and S. Lee. Svm-based feature extraction for face recognition. *Pattern Recognition*, 43(8) :2871–2881, 2010.

[141] J. Kittler and E. R. Hancock. Combining evidence in probabilistic relaxation. *International Journal of Pattern Recognition and Artificial Intelligence*, 3 :29–51, 1989.

[142] J. Kobler, U. Schoning, and J. Toran. *The Graph Isomorphism Problem : Its Structural Complexity*. Birkhauser, Boston, 1993.

[143] J. Köbler and O. Verbitsky. From invariants to canonization in parallel. In E. A. Hirsch, A. A. Razborov, A. L. Semenov, and A. Slissenko, editors, *CSR*, volume 5010 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2008.

[144] A. L. Koerich, R. Sabourin, and C. Y. Suen. Recognition and verification of unconstrained handwritten words. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10) :1509–1522, 2005.

[145] Krissinel and Henrick. Common subgraph isomorphism detection by backtracking search. *SOFTPREX : Software–Practice and Experience*, 34, 2004.

[146] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2 :83–97, 1955.

[147] L. Mukherjee, V. Singh, J. Peng, J. Xu, M.J. Zeitz, and R. Berezney. Generalized median graphs : Theory and applications. *IAPR Workshop on GbRPR, LNCS 2726*, 2007.

[148] V. Lacroix. Automatic palette identification of colored graphics. In J.-M. Ogier, W. Liu, and J. Lladós, editors, *GREC*, volume 6020 of *Lecture Notes in Computer Science*, pages 61–68. Springer, 2009.

[149] V. Lacroix. Raster-to-vector conversion : Problems and tools towards a solution . In *ICAPR*, pages 318–321. IEEE Computer Society, 2009.

[150] J. Laub and K.-R. Muller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning Research*, 5 :801–818, 2004.

[151] H. Lee-Kwang and C. H. Cho. Fuzzy hypergraph and fuzzy partition. *IEEE Transactions on Systems, Man and Cybernetics*, 25(1) :196–201, 1995.

[152] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8) :707–710, 1966.

[153] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9(4) :341–352, 1973.

[154] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5) :840–853, 2007.

[155] C.-C. Liu and D.-Q. Dai. Face recognition using dual-tree complex wavelet features. *IEEE Transactions on Image Processing*, 18(11) :2593–2599, 2009.

[156] K. Liu, Y. S. Huang, and C. Y. Suen. Identification of fork points on the skeletons of handwritten chinese characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10) :1095–1100, 1999.

[157] D. P. Lopresti and G. T. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *International Journal on Document Analysis and Recognition*, 6(4) :219–229, 2003.

[158] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3) :355–395, 1987.

[159] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, Nov. 2004.

[160] B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10) :1120–1136, 2001.

[161] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral feature vectors for graph clustering. *IAPR Workshop on S+SSPR, LNCS 2396*, pages 83–93, 2002.

[162] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36(10) :2213–2230, 2003.

[163] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, pages 281–297. University of California Press, Berkeley, CA, USA, 1967.

[164] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In C. E. Brodley, editor, *ICML*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[165] R. Mathon. Sample graphs for isomorphism testing. *Congressus Numerantium*, 21 :499–517, 1978.

[166] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12) :1650–1654, 2002.

[167] J. McGregor and P. Willett. Use of a maximum common subgraph algorithm in the automatic identification of ostensible bond changes occurring in chemical reactions. *Journal of Chemical Information and Computer Sciences*, 21(3) :137–140, 1981.

[168] J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software, Practice and Experience*, 12(1) :23–34, 1982.

[169] B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30(30) :47–87, 1981.

[170] S. Menchetti, F. Costa, and P. Frasconi. Weighted decomposition kernels. In L. D. Raedt and S. Wrobel, editors, *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 585–592. ACM, 2005.

[171] B. T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12) :1979–1998, 1999.

[172] H. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, August 1977.

[173] R. Myers, R. C. Wilson, and E. R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6) :628–635, 2000.

[174] S. Nene, S. Nayar, and H. Murase. Columbia object image library (coil-100). *technical report, Columbia Univ.*, 1996.

[175] M. Neuhaus. *Bridging the gap between Graph edit distance and Kernel Machines*. PhD thesis, University of Bern, 2006.

[176] M. Neuhaus and H. Bunke. Self-organizing maps for learning the edit costs in graph matching. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(3) :503–514, 2005.

[177] M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10) :1852–1863, 2006.

[178] M. Neuhaus, K. Riesen, and H. Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In D.-Y. Yeung, J. T. Kwok, A. L. N. Fred, F. Roli, and D. de Ridder, editors, *SSPR/SPR*, volume 4109 of *Lecture Notes in Computer Science*, pages 163–172. Springer, 2006.

[179] T.-O. Nguyen, S. Tabbone, and A. Boucher. A symbol spotting approach based on the vector model and a visual vocabulary. In *ICDAR*, pages 708–712. IEEE Computer Society, 2009.

[180] C. Orsenigo and C. Vercellis. Combining discrete svm and fixed cardinality warping distances for multivariate time series classification. *Pattern Recognition*, 43(11) :3787–3794, 2010.

[181] E. S. Page. A note on assignment problems. *The Computer Journal*, 6 :241–243, 1963.

[182] S. K. Pal and A. Pal. *Pattern Recognition : From Classical to Modern Approaches*, chapter Pattern recognition : evolution of methodologies and data mining. World Scientific, 2002.

[183] A. N. Papadopoulos and Y. Manolopoulos. Structure-based similarity search with graph histograms. *Proceedings of International Workshop on Similarity Search (DEXA IWOSS'99)*, pages 174–178, Septembre 1999.

[184] R. Pareti and N. Vincent. Ancient initial letters indexing. In *International Conference on Pattern Recognition - Volume II*, pages 756–759. IEEE Computer Society, 2006.

[185] B. G. Park, K. M. Lee, and S. U. Lee. Face recognition using face-arg matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12) :1982–1988, 2005.

[186] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2 :559–572, 1901.

[187] E. Pekalska and R. P. W. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific Publishing, 2005.

[188] E. Pekalska, R. P. W. Duin, and P. PaclaÂk. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2) :189–208, 2006.

[189] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition : A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :63–84, 2000.

[190] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(6) :637–646, 1998.

[191] M. Pouzet and N. Thiéry. Invariants algébriques de graphes et reconstruction : Algebraic graph invariants and reconstruction. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 333(9) :821–826, 2001.

[192] H. Qiu and E. R. Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1) :22–34, 2006.

[193] N. A. Rahman and E. R. Hancock. Commute-time convolution kernels for graph clustering. In E. R. Hancock, R. C. Wilson, T. Windeatt, I. Ulusoy, and F. Escolano, editors, *SSPR/SPR*, volume 6218 of *Lecture Notes in Computer Science*, pages 316–323. Springer, 2010.

[194] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8) :1093–1110, 2005.

[195] J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74. Citeseer, 2003.

[196] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336) :846–850, 1971.

[197] M. Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23) :6609–6615, 1975.

[198] K. Riesen. *Classification and Clustering of Vector Space Embedded Graphs*. PhD thesis, University of Bern, Bern, Oct. 2009.

[199] K. Riesen and H. Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In N. da Vitoria Lobo, T. Kasparis, F. Roli, J. T.-Y. Kwok, M. Georgiopoulos, G. C. Anagnostopoulos, and M. Loog, editors, *SSPR/SPR*, volume 5342 of *Lecture Notes in Computer Science*, pages 287–297. Springer, 2008.

[200] K. Riesen and H. Bunke. Kernel k-means clustering applied to vector space embeddings of graphs. In *3rd IAPR Workshop on ANNPR, LNCS 5064*, pages 24–35, 2008.

[201] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, 27(7) :950–959, 2009.

[202] K. Riesen and H. Bunke. Graph classification based on vector space embedding. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(6) :1053–1081, 2009.

[203] K. Riesen, M. Neuhaus, and H. Bunke. Bipartite graph matching for computing the edit distance of graphs. In F. Escolano and M. Vento, editors, *GbRPR*, volume 4538 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2007.

[204] K. Riesen, M. Neuhaus, and H. Bunke. Graph embedding in vector spaces by means of prototype selection. In F. Escolano and M. Vento, editors, *GbRPR*, volume 4538 of *Lecture Notes in Computer Science*, pages 383–393. Springer, 2007.

[205] A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3) :365–378, 2005.

[206] A. Robles-Kelly and E. R. Hancock. A riemannian approach to graph embedding. *Pattern Recognition*, 40(3) :1042–1056, 2007.

[207] A. Rosenfeld. Adjacency in digital pictures. *Information and Control*, 26(1) :24–33, September 1974.

[208] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12) :1540–1551, 2003.

[209] C. D. Ruberto. Recognition of shapes by attributed skeletal graphs. *Pattern Recognition*, 37(1) :21–31, 2004.

[210] M. Rusiñol and J. Lladós. A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *International Journal on Document Analysis and Recognition*, 12(2) :83–96, 2009.

[211] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[212] A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3) :353–362, May 1983.

[213] A. Sanfeliu, F. Serratosa, and R. Alquezar. Clustering of attributed graphs and unsupervised synthesis of function-described graphs. In *International Conference on Pattern Recognition*, pages 1022–1025, 2000.

[214] G. Sanromã, R. Alquãzar, and F. Serratosa. Attributed graph matching for image-features association using sift descriptors. In E. R. Hancock, R. C. Wilson, T. Windeatt, I. Ulusoy, and F. Escolano, editors, *SSPR/SPR*, volume 6218 of *Lecture Notes in Computer Science*, pages 254–263. Springer, 2010.

[215] K. Santosh, L. Wendling, and B. Lamiroy. Using spatial relations for graphical symbol description. In *International Conference on Pattern Recognition*, pages 2041–2044. IEEE Computer Society, 2010.

[216] C. Saunders and A. Demco. *Kernels for Strings and Graphs*. Springer, 2007.

[217] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1) :27–64, 2007.

[218] A. Schenker, M. Last, H. Bunke, and A. Kandel. Clustering of web documents using a graph model. In A. Antonacopoulos and J. Hu, editors, *Web Document Analysis : Challenges and Opportunities*, volume Machine Perception an Artificial Intelligence, pages 3–18. World Scientific Publishing, 2003.

[219] A. Schenker, M. Last, H. Bunke, and A. Kandel. Comparison of algorithms for web document clustering using graph representations of data. In A. L. N. Fred, T. Caelli, R. P. W. Duin, A. C. Campilho, and D. de Ridder, editors, *SSPR/SPR*, volume 3138 of *Lecture Notes in Computer Science*, pages 190–197. Springer, 2004.

[220] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5) :1299–1319, 1998.

[221] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Shock-based indexing into large shape databases. In *7th European Conference on Computer Vision, LNCS 2352*, pages 731–746, 2002.

[222] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5) :550–571, 2004.

[223] F. Sebastiani and C. N. D. Ricerche. Machine learning in automated text categorization. *ACM Computing Surveys*, 34 :1–47, 2002.

[224] K. Sengupta and K. Boyer. Organizing large structural modelbases. *IEEE TPAMI*, 17(4) :321–332, Apr 1995.

[225] L. Shapira, S. Avidan, and A. Shamir. Mode-detection via median-shift. In *IEEE 13th Intl. Conf. on Computer Vision, ICCV*, 2009.

[226] L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3 :504–519, 1981.

[227] L. G. Shapiro and R. M. Haralick. Organization of relational models for scene analysis. *IEEE TPAMI*, PAMI-4(6) :595–602, Nov. 1982.

[228] L. G. Shapiro and R. M. Haralick. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1) :90–94, 1985.

[229] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9 :366–380, 1998.

[230] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[231] Y. Sheikh, E. A. Khan, and T. Kanade. Mode-seeking by medoidshifts. In *IEEE 11th Intl. Conf. on Computer Vision, ICCV*, pages 1–8, 2007.

[232] J. Shi and J. Malik. Self inducing relational distance and its application to image segmentation. In H. Burkhardt and B. Neumann, editors, *European Conference on Computer Vision - Volume I*, volume 1406 of *Lecture Notes in Computer Science*, pages 528–543. Springer, 1998.

[233] A. Shokoufandeh and S. Dickinson. Applications of bipartite matching to problems in object recognition. In *ICCV Workshop on Graph Algorithms and Computer Vision*, 1999.

[234] A. Shokoufandeh and S. J. Dickinson. A unified framework for indexing and matching hierarchical shape structures. In *4th Int. Workshop on Visual Form, LNCS 2059*, pages 67–84, 2001.

[235] S. Shukla and S. K. Rakse. Spam classification using new kernel function in support vector machine. *IJCSE - International Journal on Computer Science and Engineering*, 2(5) :1819–1823, August 2010.

[236] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1) :13–32, 1999.

[237] S. S. Skiena. *Implementing Discrete Mathematics : Combinatorics and Graph Theory with Mathematica, with programs by Steven Skiena and Anil Bhansali*. Addison-Wesley, 1990.

[238] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12) :1349–1380, 2000.

[239] C. Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12-13) :850–864, 2010.

[240] H. Spath. Cluster analysis algorithms for data reduction and classification of objects. *Ellis Horwood Limited, West Sussex*, 1980.

[241] S. N. Srihari. Handwritten address interpretation : A task of many pattern recognition problems. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(5) :663–674, 2000.

[242] S. Srinivasa and S. Kumar. A platform based on the multi-dimensional data model for analysis of bio-molecular structures. In *VLDB*, pages 975–986, 2003.

[243] Stefano Berretti, Alberto Del Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10) :1089–1105, 2001.

[244] P. N. Suganthan. Structural pattern recognition using genetic algorithms. *Pattern Recognition*, 35(9) :1883–1893, 2002.

[245] P. N. Suganthan, E. K. Teoh, and D. P. Mital. Pattern recognition by graph matching using the potts mft neural networks. *Pattern Recognition*, 28(7) :997–1009, 1995.

[246] P. N. Suganthan, E. K. Teoh, and D. P. Mital. Pattern recognition by homomorphic graph matching using hopfield neural networks. *Image and Vision Computing*, 13(1) :45–60, 1995.

[247] M. Suk and S. Oh. Region adjacency and its application to object detection. *Pattern Recognition*, 19 :161–167, 1986.

[248] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1) :11–32, 1991.

[249] D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli. Surftrac : Efficient tracking and continuous object recognition using local feature descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2937–2944. IEEE, 2009.

[250] R. Todeschini and V. Consonni. Handbook of molecular descriptors. *Weinheim : Wiley-VCH*, 2000.

[251] W. S. Torgerson. Multidimensional scaling I. Theory and method. *PSym*, 17 :401–419, 1952.

[252] W. S. Torgerson. *Theory and Methods of Scaling*. Johm Wiley and Sons, 1958.

[253] A. Torsello and E. R. Hancock. A skeletal measure of 2d shape similarity. *Computer Vision and Image Understanding*, 95(1) :1–29, 2004.

[254] A. Torsello and E. R. Hancock. Graph embedding using tree edit-union. *Pattern Recognition*, 40(5) :1393–1405, 2007.

[255] W. Tsai and K. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 9(12) :757–768, 1979.

[256] W. Tsai and K. Fu. Subgraph error-correcting isomorphism for syntactic pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13 :48–61, 1983.

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

[258] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10 :695–703, 1988.

[259] E. Valveny, S. Tabbone, O. R. Terrades, and E. Philippot. Performance characterization of shape descriptors for symbol representation. In W. Liu, J. Llados, and J.-M. Ogier, editors, *GREC*, volume 5046 of *Lecture Notes in Computer Science*, pages 278–287. Springer, 2007.

[260] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[261] V. Vapnik and V. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

[262] R. C. Veltkamp, H. Burkhardt, and H.-P. Kriegel, editors. *State-of-the-art in content-based image and video retrieval*. Kluwer Academic Publishers, 2001.

[263] S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 1449–1456. MIT Press, 2006.

[264] R. A. Wagner and M. J. Fisher. The string-to-string correction problem. *Journal of the ACM*, 21(1) :168–173, 1974.

[265] W. D. Wallis, P. Shoubridge, M. Kraetzl, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22(6/7) :701–704, 2001.

[266] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS*, pages 668–674. MIT Press, 2000.

[267] H. Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1) :17–20, 1947.

[268] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6 :1–34, 1997.

[269] R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6) :634–648, 1997.

[270] R. C. Wilson and E. R. Hancock. Levenshtein distance for graph spectral features. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2-Volume 02*, pages 489–492. IEEE Computer Society, 2004.

[271] R. C. Wilson, E. R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7) :1112–1124, 2005.

[272] A. Winter. Exchanging Graphs with GXL. In *Graph Drawing - 9th International Symposium*, Vienna, 2001. Springer Verlag.

[273] A. Wong, M. You, and S. Chan. An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 20 :628–636, 1990.

[274] A. Woznica. *Distance and Kernel Based Learning over Composite Representations*. PhD thesis, Université de Genève, 2008.

[275] X. Jiang, A. Munger, and H. Bunke. Computing the generalized median of a set of graphs. *IAPR Workshop on GBRPR, Austrian Computer Society*, pages 115–124, 1999.

[276] X. Jiang, A. Munger, and H. Bunke. On median graphs :properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10) :1144–1151, 2001.

[277] X. Xiaogang, Z. Sun, B. Peng, X. Jin, and W. Liu. An online composite graphics recognition approach based on matching of spatial relation graphs. *International Journal on Document Analysis and Recognition*, 7(1) :44–55, 2004.

[278] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8) :841–847, 1991.

[279] R. Xu and D. I. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3) :645–678, May 2005.

[280] X. Yan, P. S. Yu, and J. Han. Graph indexing : A frequent structure-based approach. In *SIGMOD '04 : Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 335–346. ACM, 2004.

[281] Y. Yang and X. Liu. A re-examination of text categorization methods. In M. A. Hearst, F. Gey, and R. Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

[282] S. Zampelli, Y. Deville, C. Solnon, S. Sorlin, and P. Dupont. Filtering for subgraph isomorphism. In C. Bessiere, editor, *CP*, volume 4741 of *Lecture Notes in Computer Science*, pages 728–742. Springer, 2007.

[283] M. Zaslavskiy, F. R. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12) :2227–2242, 2009.

[284] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *IEEE Conference on Computer Vision and Pattern Recognition,*. IEEE Computer Society, 2008.

[285] D. Zhang and G. Lu. Shape-based image retrieval using generic Fourier descriptor. *Signal Processing : Image Communication*, 17(10) :825–848, 2002.

[286] J. Zhang, B. Rivard, and D. M. Rogge. The successive projection algorithm (SPA), an algorithm with a spatial constraint for the automatic search of endmembers in hyperspectral data. *Sensors*, pages 1321–1342, 2008.

[287] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition : A literature survey. *ACM Computing Surveys*, 35(4) :399–458, 2003.

[288] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison–Wesley, Cambridge, MA, 1949.

[289] D. Zuwala and S. Tabbone. A method for symbol spotting in graphical documents. In H. Bunke and A. L. Spitz, editors, *Document Analysis Systems*, volume 3872 of *Lecture Notes in Computer Science*, pages 518–528. Springer, 2006.

Résumé

Les travaux de cette thèse se situent dans la cadre des approches structurelles pour la reconnaissance de formes. Plus précisément, nous avons porté notre choix sur les graphes. Le choix de la représentation structurelle est justifié par la grande capacité représentative des graphes par rapport à la représentation statistique (i.e. vecteurs). La première étape qui intervient dans l'étude de l'application des graphes dans le domaine des images est de définir une stratégie d'extraction de graphes représentatives d'images. Ensuite, il faut définir des fonctions nécessaires à la manipulation des bases de graphes. L'une des fonctions cruciales pour manipuler les graphes est la fonction de calcul des distances entre les graphes. En effet, le calcul de distances entre les graphes est un problème ouvert dans la littérature. De plus, il est considéré comme NP-complet. La plupart des solutions proposées dans la littérature présentent différentes limites d'utilisation telle que la taille des graphes, la prise en compte d'attributs, le temps de calcul. Outre la distance, le domaine des graphes souffre d'un manque d'algorithmes de classification (non-)supervisée appropriés. Dans ce sens, cette thèse présente un ensemble de contributions dont l'objectif est l'indexation de graphes. En premier lieu, nous montrons expérimentalement que choix de la représentation sous forme de graphes a un impact sur les performances. Ensuite, nous proposons une nouvelle approximation de la distance d'édition de graphes basée sur la notion de signature de noeuds. Nous introduisons aussi un algorithme de plongement de graphes. Cet algorithme consiste à représenter chaque graphe par un vecteur dans un espace euclidien. Ceci nous permet d'appliquer les algorithmes de classification des vecteurs sur les graphes par le biais du plongement. Dans le domaine de la classification non-supervisée (clustering), nous proposons un nouvel algorithme basé sur la notion du graphe médian et la notion du *mean-shift*. Enfin, nous proposons, une nouvelle méthode d'indexation de graphes basée sur la structure d'hypergraphe. Cette méthode permet aussi bien l'indexation que la navigation dans une base d'images représentées sous forme de graphes.

Mots-clés: Reconnaissance de formes structurelle, appariement de graphes, classification de graphes, indexation de graphes.