Contribution en appariement de graphes pour la recherche d'images par le contenu

par

Adel Hlaoui

Thèse présentée au Département d'informatique

en vue de l'obtention du grade de docteur ès sciences (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada.

Mai 2004

Le 10 janvier 2005,
Date

le jury a accepté la thèse de M. Adel Hlaoui dans sa version finale.

Membres du jury

M. Shengrui Wang
Directeur
Département d'informatique

M. Djemel Ziou
Membre
Département d'informatique

M. Béchir Ayeb
Membre
Département d'informatique

M. Mohamed Cheriet
Membre externe
Département de géniede la production automatisée - Laboratoire LIVIA

M. François Dubeau
Président-rapporteur
Département de mathématiques

Je dédie cette thèse à la mémoire de mon père

Mohamed

15 août 1933 - 19 mars 1984

et

à ma fille Inès

qui a eu deux mois le jour de ma soutenance.

Sommaire

Cette thèse s'inscris dans le cadre général de la reconnaissance de formes structurelles. Elle s'intéresse plus particulièrement à la modélisation des formes par les graphes. L'utilisation de graphes est motivée par le double intérêt qu'apportent ces derniers pour modéliser tous les objets d'une forme donnée et toutes les relations inter objets nécessaires pour la reconnaissance. Un exemple typique utilisé dans cette thèse est celui de *la recherche d'images par le contenu*(RIPC). Cependant, les techniques présentées dans cette thèse ont un champ plus vaste que la RIPC. La représentation des images par des graphes implique le recours à des algorithmes d'appariement de graphes afin de comparer et de détecter la similarité entre les images. Par ailleurs la recherche dans une base de données d'image nécessite une réorganisation préalable de la base afin de faciliter la recherche, ce qui nous conduit à faire appel à des techniques de classification des images représentées par des graphes.

Dans un premier temps, nous proposons un nouvel algorithme pour mettre en correspondance un graphe requête et un graphe modèle. L'idée de base est de diviser le processus de recherche des correspondances en plusieurs phases (K). À l'issue de chaque phase, l'ensemble des correspondances est extrait, évalué et finalement comparé à celui dont le coût de correspondance est minimal. Dans un deuxième temps, nous proposons un nouvel algorithme pour identifier un représentant appelé *Graphe Médian*, parmi un ensemble de graphes. Le rôle du graphe médian est capital pour la classification et la réorganisation d'une base de données image utilisant les graphes pour représenter son contenu. Finalement, nous proposons un système de recherche d'images par le contenu utilisant les graphes pour représenter leur contenu et les deux algorithmes précédemment décrits. D'une manière générale, les résultats présentés dans cette thèse montrent l'intérêt potentiel d'utiliser les graphes pour représenter les formes. Ces résultats semblent valider le choix

judicieux des graphes comme une solution de remplacement aux structures de données classiques à savoir les vecteurs. De plus, on voit clairement à travers les résultats obtenus que les algorithmes, développés dans cette thèse, pourront jouer un rôle primordial comme un outil de mesure de similarité dans un espace aussi complexe que les graphes.

Remerciements

Je remercie M. Shengrui WANG, qui m'a accueilli dans son équipe et qui a encadré mes travaux durant ces dernières années passées. Par ses conseils et son dévouement constant, il m'a aidé à développer mes aptitudes et mon intérêt pour la recherche. Je lui en suis très reconnaissant.

Je remercie M. Francois DUBEAU, Professeur à l'Université de Sherbrooke, pour avoir accepté de présider le jury de ma thèse.

Mes remerciements s'adressent ensuite aux membres du jury :

Monsieur Mohamed CHERIET, Professeur à l'École de Technologie Supérieure.

Monsieur Djemel ZIOU, Professeur à l'Université de Sherbrooke.

Monsieur Béchir EL AYEB, Professeur à l'Université de Sherbrooke.

Je remercie toutes les personnes que j'ai cotoyés dans l'équipe MOIVRE. Je n'ose pas nommer leurs noms de peur d'en oublier.

Je remercie également ma mére, mon frère et mes soeurs pour leur soutient et leur encouragement constant tout au long de mes études.

Enfin, j'ai une pensée toute particulière pour mon épouse Caroline, dont le soutien ne m'a jamais fait défaut.

Table des matières

Liste des tableaux

Table des figures

Introduction

Le domaine de la reconnaissance de forme a attiré, dans les dernières décennies, la curiosité d'une grande partie de la communauté scientifique. Plusieurs travaux ont été réalisés sur le plan théorique et pratique. Ces travaux ont conduit d'une part à l'apparition de nouvelles méthodes de reconnaissance et d'autre part à l'apparition de plusieurs systèmes de reconnaissance à la fois commercial et éducatif.

On distingue deux approches utilisées en reconnaissance de forme permettant de reconnaître et d'identifier une forme appartenant à une catégorie ou une classe : l'approche structurelle et l'approche statistique. La première approche utilise une description liée à la nature des formes étudiées. Cette approche permet de caractériser les objets et les relations intrinsèques entre ces objets et, ensuite, de comparer l'ensemble à l'aide des techniques différentes de celles utilisées dans les espaces métriques usuels. La deuxième approche, fondée sur l'étude statistique des mesures effectuées sur les objets à reconnaître, permet, dans un premier temps, d'identifier les objets et, dans un deuxième temps, de prendre une décision sur leur appartenance à une classe bien déterminée de type "plus forte probabilité d'appartenance à une classe".

Dans l'approche structurelle, la représentation des données est décrite par un ensemble de chaînes, d'arbres ou de graphes. Le problème de la reconnais-

sance est alors vu comme une mise en correspondance entre les deux formes. Cependant, dans l'approche statistique, la représentation des mesures effectuées sur les objets est décrite par un ensemble de vecteurs dans un espace fini. Le problème de la reconnaissance peut alors être formulé sommairement de la manière suivante : étant donné un vecteur inconnu, comment trouver la classe à laquelle on doit l'affecter ?

Une première comparaison entre les deux approches nous permet de dégager la puissance de la première approche à représenter judicieusement les formes en tenant compte à la fois du contenu des objets et de leurs relations intrinsèques. Cette constatation vient aussi du fait que la représentation vectorielle est facilement transférable en une représentation structurelle, ce qui n'est pas vrai dans le sens contraire.

Cependant, l'approche structurelle souffre d'une part d'un manque d'outils mathématiques puissants capables de mesurer le degré de similarité entre les objets par rapport à ceux utilisés dans l'approche statistique et d'autre part de la complexité élevée des algorithmes existants. Cela explique l'utilisation fréquente des méthodes statistiques dans le domaine de la reconnaissance de formes et plus précisément dans le domaine de la recherche d'images par le contenu. Malgré ces difficultés, nous assistons de nos jours à un regain d'intérêt pour l'approche structurelle et l'utilisation des graphes, en particulier comme outil de modélisation de formes (image, caractère, scène, parole, etc.). Ce regain d'intérêt a permis l'apparition de plusieurs travaux améliorant d'une part des méthodes existantes et d'autre part le développement des nouvelles méthodes, afin de manipuler et comparer les graphes.

Une analyse détaillée de ces différentes méthodes montre que le but ultime recherché est la réduction de la complexité exponentielle du problème de l'appariement de graphes. La complexité élevée du problème est causée par l'explosion combinatoire des mises en correspondances possibles pour trou-

ver un appariement entre une paire de graphes. Cette complexité augmente d'une manière linéaire avec le nombre de paires de graphes à comparer. Cela rend le problème encore plus difficile à résoudre. Nous avons divisé les méthodes d'appariement de graphes en deux classes (voir Figure 1). Dans la première classe, nous avons regroupé des méthodes dites *un-avec-un (one-to-one)*. La recherche de l'appariement est réalisée en comparant une seule paire de graphes. On vise uniquement à réduire la complexité exponentielle engendrée par la mise en correspondances des sommets et des arêtes des deux graphes. Dans la deuxième classe, on trouve des méthodes dites *un-avec-plusieurs (one-to-many)*. On vise à réduire la complexité engendrée à la fois par l'explosion combinatoire d'une seule comparaison d'une paire de graphes et par l'aspect linéaire d'une séquence de comparaisons de plusieurs paires de graphes. La littérature nous dévoile que la première classe a été la cible sans équivoque des chercheurs. Bien que, le nombre de méthodes appartenant à la deuxième classe soit relativement faible par rapport à celles issues de la première classe, les percées accomplies sont considérables.

Techniquement parlant, les méthodes appartenant à la première et à la deuxième classe reposent sur deux types d'approches. Une première approche est basée sur une *exploration déterministe* de l'espace de recherche. Une deuxième approche est basée sur une *exploration non déterministe* utilisant des techniques d'*optimisation stochastique*. Parmi ces techniques, on peut citer les algorithmes génétiques, les réseaux de neurones, les outils algébriques, les outils probabilistiques, etc. (voir Figure 1).

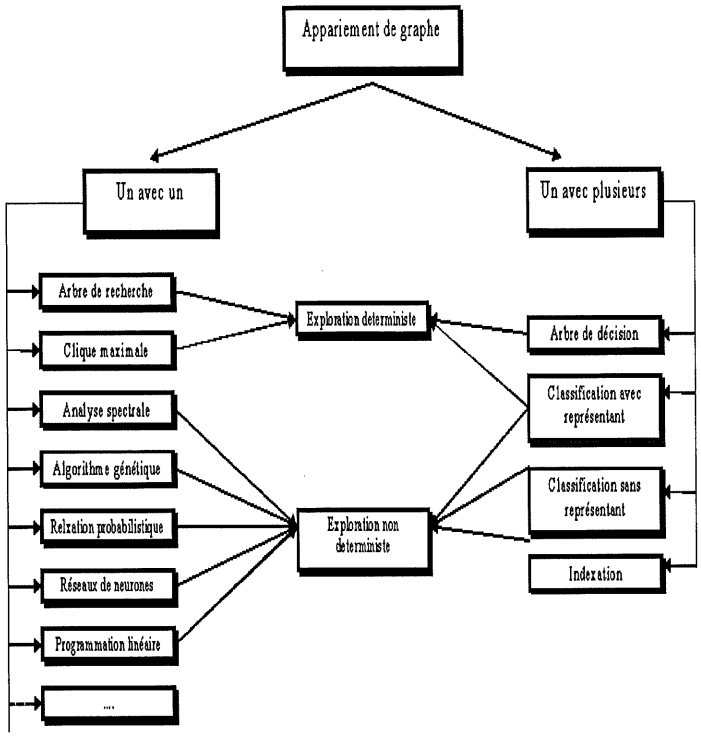


FIG. 1: *Classification des méthodes d'appariement de graphes.*

Dans la deuxième classe, nous avons distingué trois types de méthodes. La première méthode utilise une technique de décomposition des graphes en plusieurs sous-graphes. La recherche est ensuite effectuée sur un arbre de décision préalablement construit. La deuxième méthode utilise des techniques de *classification.* Enfin, la dernière méthode utilise les techniques d'*indexation* d'une base de données. Il est important de signaler que la méthode de classification est divisée, à son tour, en deux sous-classes. La classification est une technique qui permet de regrouper un ensemble de données dans différents groupes. Chaque groupe est identifié par un représentant. Une première sous-classe appelée classification avec représentant et une deuxième sous-classe appelée classification sans représentant. Vu la difficulté de trou-

ver un représentant, étant donné qu'il s'agit d'une structure aussi complexe que les graphes, la majorité des algorithmes existant appartiennent à cette deuxième sous-classe. Récemment, Bunke [10] a définit un nouveau concept appelé *"graphe médian"*. Le premier et le seul algorithme existant, à notre connaissance, capable de construire le graphe médian a été proposé par Jiang, Munger et Bunke [56].

On appelle graphe le couple $G(X, Y)$, où X est un ensemble de sommets et Y un ensemble d'arêtes. Dans cette définition, aucune étiquette ou pondération n'est associée aux sommets ou aux arêtes. Dans le cas contraire, le graphe deviendra un 4-uplet $G(X, Y, f, g)$ où f et g sont respectivement la fonction d'assignation des étiquettes aux sommets et aux arêtes. Le graphe est appellé "graphe étiqueté". Ces deux types de graphes ont un traitement différent lors de la recherche de l'appariement. On parle d'une *recherche exacte* pour le premier cas (graphe non étiqueté) et d'une *recherche inexacte* dans le deuxième cas (graphe étiqueté). La recherche exacte suit une loi d'exactitude absolue où l'imperfection est catégoriquement rejetée. La recherche inexacte, appelé aussi recherche approximative, autorise une tolérance relative d'imperfection. Malheureusement, les différents algorithmes existant dans la littérature ne s'appliquent pas de la même manière sur les deux types de graphes.

Classe un-avec-un

Les travaux de Barrow, Popplestone et Burstall [2] [3] d'une part et de Fischler et Enschlager [28] d'autre part, effectués au début des années 70, ont montré l'importance de l'utilisation des graphes pour décrire et représenter les formes, en particulier les images. Nous allons présenter les différents méthodes appartenant à cette classe.

Arbre de recherche

L'histoire des algorithmes d'appariement de graphes commence avec l'apparition de l'algorithme de Corneil et Gotlieb [19]. Le principe repose sur une énumération de l'espace de recherche. De nos jours, l'algorithme de Ullmann [110], reste le plus performant et le plus utilisé comme algorithme de référence. L'algorithme est basé sur une recherche arborescente avec rétropropagation. Le point fort de l'algorithme est l'introduction d'une procédure de vérification et de rejet par anticipation d'éventuelles mises en correspondances non compatibles. Ces deux algorithmes appartiennent à la première classe dans notre classification et utilisent la première approche (exploration déterministe) décrite ci-dessus. Ils visent la recherche exacte. Tout récemment, un autre algorithme baptisé VF a été proposé par Cordella et ses coéquipiers [17] [18]. L'idée de base consiste à représenter les mises en correspondances par un espace d'états. Un état représente une solution partielle de l'appariement recherché. Une transition représente une extension de l'appariement partiel. Le premier algorithme pour la recherche inexacte a été développé par Tsai et Fu [108]. Ce travail a été l'objet d'améliorations successives par différents auteurs [26] [95] [99]. Ces algorithmes utilisent des techniques d'éditions de graphe [6] [8]. Ces techniques sont inspirées des travaux menés sur les chaînes. Tout récemment, Bunke a établi un lien entre l'édition des graphes et la taille du plus grand sous-graphe commun entre deux graphes. Pour d'autres algorithmes, le lecteur est invité à consulter Conte *et al.* [19].

Analyse spectrale

L'utilisation massive de l'approche stochastique a véritablement commencé vers la fin des années 80. Umeyama [111] a présenté un algorithme qui repose sur une analyse des vecteurs propres des matrices d'adjacence des deux graphes. Malgré l'élégance de cette méthode et la rapidité avec laquelle elle permet de trouver une solution, son champ d'application se limite à des graphes de même taille et similaires. Pour éviter ce problème, Hancock et

Luo [77] ont proposé une autre méthode basée sur l'algorithme EM. D'un autre coté, les algorithmes génétiques ont ouvert une deuxième voie pour la conquête de la complexité polynomiale. Partant d'une population donnée, la mutation et le croisement, deux outils combinés, permettent une évolution contrôlée de la population vers l'appariement recherché. Une fonction de sélection permettra ensuite d'extraire les meilleurs mises en correspondances entre deux graphes. Plusieurs auteurs ont contribué à faire avancer cette technique. Nous distinguons en particulier le travail proposé par Wang *et al.* [114] ainsi que le travail de Cross et Hancock [20] [21].

Relaxation probabiliste

Le recours à des méthodes probabilistes date uniquement de quelques années. Une étude détaillée est décrite dans [37]. Une première tentative d'utiliser cette branche remonte à 1993 avec le travail de Kittler et Hancock [66]. C'est une méthode itérative utilisant la relaxation probabilistique. Une méthode qui tient compte uniquement des relations binaires dont la fonction d'erreur est une Gaussienne. En 1995, Christmas *et al.* [16] ont proposé une modélisation Bayésienne. Cette modélisation est améliorée en deux temps par Williams *et al.* en 1999 [115] et par Wilson et Hancock en 1999 [116]. Le modèle Bayèsien permet de définir des algorithmes itératifs fonctionnant sur le gradient. Finalement, différents types de réseaux de neurones ont été appliqués pour trouver le meilleur appariement entre deux graphes [106] [107]. L'idée de base est la suivante. Un neurone représente une mise en correspondance entre une paire de sommets et le poids d'une connexion représente une mesure de compatibilité entre deux noeuds. La fonction de la minimisation de l'énergie du réseau est définie en terme de compatibilité entre les mises en correspondance.

Classe un-avec-plusieurs

Une autre approche de réduction de la complexité est apparue au courant de la dernière décennie. Il s'agit d'impliquer dans le processus de recherche le graphe requête et l'ensemble de graphes d'une base de données. Dans ce qui suit, nous allons présenter les différents méthodes appartenant à cette classe.

Arbre de décision

Messmer et Bunke [81] [82] [83] [84] [85] [86] ont proposé un algorithme pour détecter le graphe ou le sous-graphe isomorphisme, basé sur la construction d'un arbre de décision. Chaque graphe appartenant à la base de données est décomposé en plusieurs sous-graphes. Plusieurs sous-graphes identiques seront représentés une et une seule fois dans l'arbre de décision. Le graphe requête est propagé dans l'arbre de décision pour déterminer le ou les graphes identiques. Dans un premier temps, une extension de cette méthode a été proposée par Shearer [102]. Au lieu de comparer un seul graphe requête avec un ensemble de graphes, l'auteur compare un ensemble de graphes requête avec les graphes de la base de données. Une autre extension a été proposée par le même auteur [103]. Il s'agit de détecter le plus grand sous-graphe en commun entre un graphe requête et une base de données de graphes.

Indexation

L'indexation consiste à calculer un code pour un graphe requête, à examiner une table d'affectation préalablement préparée et à sélectionner enfin le ou les graphes susceptibles d'être les plus similaires. La méthode proposée par Sossa et Horaud [105] consiste à caractériser un graphe par un polynôme $d(xI - L(G))$ avec $L(G)$ le laplacien de la matrice d'adjacence du graphe requête. Les codes du graphe requête sont les coefficients du polynôme. Ainsi, la comparaison entre les coefficients du polynôme laplacien de chaque graphe équivaut à détecter le graphe isomorphisme entre ces deux graphes. Cette mé-

thode exige que les deux graphes mis en comparaison aient la même taille. Seul le graphe isomorphisme est détecté. Une extension de cette méthode a été développée par le même auteur pour traiter des graphes qui n'ont pas le même nombre de sommets.

Classification avec et sans représentant

Les algorithmes appartenant à cette sous-classe permettent de réorganiser une base de données de graphe pour faciliter la comparaison. La classification est réalisée en séparant l'ensemble de graphes en différents groupes. Pour cela nous avons distingué trois techniques. La première technique proposé par Jiang, Munger et Bunke [56] [57] [58] [59] [61] consiste à identifier pour chaque ensemble un représentant appelé graphe médian. Cette identification est utilisée pour une répartition de la base de données de graphes à l'aide d'un algorithme classique tel que *k-means*. La deuxième technique consiste à calculer la distance entre chaque paire de graphes possibles. Le ou les graphes ayant des distances similaires appartiendront au même groupe. La troisième technique proposé par Luo et Hancock [75] [76] consiste à projeter les graphes dans un autre espace où la classification est facilement détectable. Bien que les deux dernières techniques n'aient pas besoin d'expliciter de représentant pour chaque groupe et qu'elles offrent une manière simple et rapide de réorganiser la base de données utilisée, leur capacité à mener correctement la classification des graphes reste à prouver. Le fait de projeter les graphes dans un autre espace induit nécessairement des pertes d'information. Selon nous, l'identification du graphe médian est un choix judicieux et important pour une classification réussie. Malgré, la difficulté de la tâche à trouver un représentant d'un ensemble de graphes, nous avons opté pour cette technique. Notre choix vient du fait que la manière la plus fidèle de réaliser la répartition est, sans aucun doute, de manipuler directement les graphes. Nous avons proposé un algorithme efficace et rapide par rapport au seul algorithme existant

dans la littérature basé sur les algorithmes génétiques.

Il existe dans la littérature plusieurs applications utilisant l'approche structurelle. Nous allons donner dans ce qui suit une idée des plus importantes contributions dans la littérature. Une description détaillée de ces différentes applications se trouve dans [92]. Nous allons commencer par décrire cinq applications importantes. Cette description sera suivie par une liste de pointeurs sur d'autres applications. La première application utilisant l'approche structurelle a été proposée par Edwin Petrakis [91] [92]. Dans ce travail, des images IRM ont été représentées par des graphes étiquetés. L'appariement de graphes est réalisé à l'aide de l'algorithme d'Ullmann. Josep Llados [73] [74] a présenté un autre travail intéressant qui consiste à combiner les techniques d'appariements de graphes et la transformée de Hough pour la reconnaissance des figures graphiques. Le troisième travail a été proposé par Aymeric Perchant [90] pour la reconnaissance structurelle de scènes en utilisant le morphisme de graphes d'attributs flous. Les différentes techniques présentées dans ce travail ont été appliquées à la reconnaissance de structures anatomiques saines et pathologiques sur des images IRM. Le quatrième travail a été présenté par Richard Wilson [118]. Il permet de détecter des routes dans des images infra-rouges et des images SAR en faisant appel à l'appariement de graphes. Un dernier travail a été proposé récemment par Kim Shearer [102] [103] pour l'indexation des vidéos. La méthode utilisée pour rechercher la similarité entre les images est inspirée de la méthode proposée par Messmer [82].

Nos objectifs

Comme nons l'avons mentionné plus haut, les méthodes issues de l'approche structurelle ont été répertoriées en deux grandes familles : des méthodes un-avec-un et des méthodes un-avec-plusieurs. Le recours à l'approche structurelle constitue une deuxième option sérieuse pour la reconnaissance

de formes. Cependant, l'approche souffre de l'explosion combinatoire lors de la recherche d'une solution optimale. Cette approche a été utilisée dans plusieurs domaines. Malheureusement, aucune utilisation n'a été décrite dans la littérature pour une application de grande envergure.

Dans cet esprit, l'objectif général de la thèse est de proposer des contributions dans le cadre de la reconnaissance de formes structurelles. Notre premier objectif est de développer un algorithme d'appariement de graphes appartenant à la première famille, rapide et efficace. L'idée de base de l'algorithme repose sur un processus de décomposition de l'espace de recherche en plusieurs phases. Le nombre de phases est le paramètre clé de l'algorithme qui détermine à la fois le degré d'efficacité de l'algorithme et le temps nécessaire pour trouver une solution acceptable. En d'autres termes, c'est le facteur qui permet de trouver une solution approximative proche de l'optimum dans un temps raisonnable.

Notre deuxième objectif est de proposer une nouvelle méthode appartenant à la deuxième famille décrite précédemment. Dans cette famille, on distingue deux types de méthodes : des méthodes indirectes, permettant une réorganisation intelligente de l'ensemble des graphes sans identification d'un représentant pour chaque sous-ensemble ; des méthodes directes, capables de réorganiser l'ensemble des modèles en plusieurs groupes en identifiant un représentant pour chaque représentant.

Notre troisième objectif est de proposer un outil de recherche d'images par le contenu utilisant l'approche structurelle, premièrement pour décrire et représenter le contenu de l'image, deuxièmement pour comparer et identifier les images en faisant appel à l'algorithme d'appariement de graphes proposé dans ce document et troisièmement pour réorganiser une base de données d'image afin de faciliter la recherche en faisant appel à l'algorithme du graphe médian décrit précédemment.

Structure du document

Cette thèse s'articule en 4 chapitres qui nous permettront de présenter les différents aspects de nos travaux.

Le premier chapitre de ce document donne quelques notions et concepts de base de la théorie des graphes. L'accent sera mis ensuite sur la présentation des différentes approches utilisées pour l'appariement de graphes.

Le deuxième chapitre présente le nouvel algorithme d'appariement de graphes. Il sera suivi d'une analyse de sa complexité et de l'expérimentation réalisée pour illustrer sa performance. Une comparaison avec quatre autres méthodes est présentée afin d'évaluer sa performance en terme de capacité d'appariement et de temps d'exécution.

Le troisième chapitre présente le nouvel algorithme de recherche du graphe médian. Il est suivi d'une expérimentation et d'une comparaison avec une autre méthode utilisant les algorithmes génétiques afin d'illustrer son efficacité à trouver une solution approximative sous-optimale dans un temps raisonnable.

Le quatrième chapitre présente dans un premier temps un algorithme de classification de graphes basé sur l'algorithme k-means et les deux algorithmes décrit respectivement dans les chapitres 2 et 3. Cet algorithme de classification est utilisé pour la recherche d'image par le contenu dans une base de données synthétiques. Dans un deuxième temps, nous allons présenter un système de recherche d'image par le

Enfin, la conclusion résume les différents résultats et ouvre quelques perspectives sur la suite de ce travail.

Chapitre 1

Graphes et Similarité

1.1 Introduction

L'utilisation des graphes comme un outil mathématique pour résoudre des problèmes complexes comme celui de Koenigsberg remonte au $17^{ème}$ siècle. Plus tard, les graphes ont été utilisés par Kirchoff comme un outil de modélisation de circuit électrique (lois de Kirchoff : 1854). Récemment, la communauté de vision par ordinateur et de reconnaissance de formes a utilisé les graphes comme un outil aussi de modélisation et de recherche d'image (Barrow, Popplestone et Burstall [2] [3]). De nos jours, on assiste à un regain d'intérêt pour l'utilisation des graphes après qu'ils ont été abandonnés vu la complexité et la difficulté de trouver des algorithmes efficaces et rapides pour manipuler et utiliser les graphes.

L'intérêt d'utiliser les graphes en analyse d'image se trouve certainement dans leur puissance à modéliser les dépendances entre objets présents sur une image et à tirer profit de la répartition spatiale de ces objets comme information potentielle pour décrire une image. La recherche d'une image représentée par un graphe est interprétée comme une recherche de similarité entre deux

ensembles ou sous-ensembles d'objets appartenant chacun à une image. Le calcul de similarité entre deux graphes d'un point de vue mathématique est réalisé par une recherche de morphisme de graphe. On distingue trois types de morphisme : l'isomorphisme de graphe, l'isomorphisme de sous-graphe et le plus grand commun sous-graphe. La recherche de morphisme de graphes est un problème difficile. En effet, l'isomorphisme de sous-graphes et le plus grand commun sous graphe sont deux problèmes NP-Complet, tandis que l'appartenance de l'isomorphisme de graphe à la classe P, NP-complet ou à une classe intermédiaire est jusqu'à présent un sujet ouvert.

La première classe regroupe des méthodes dites un-avec-un. Quatre méthodes appartenant à cette classe seront présentées ; recherche arborescente par rétropropagation, algorithme génétique, analyse spectrale des matrices d'adjacence et relaxation probabilistique. La deuxième classe regroupe des méthodes dites un-avec-plusieurs. Les méthodes appartenant à cette classe ont été divisées en deux sous-classes. Dans la première sous-classe, on trouve des méthodes qui procèdent par un calcul d'un représentant d'un ensemble de graphes appelé graphe médian. Dans la deuxième sous-classe, la comparaison est faite en utilisant diverses techniques sans recours à un calcul préalable du graphe médian. Etant donné que notre intérêt porte sur la première sous-classe, nous allons définir dans un premier temps le concept récemment introduit par Bunke et dans un deuxième temps présenter la seule méthode existante pour le calcul du graphe médian. Finalement, en guise de conclusion, nous résumerons le chapitre et fournirons un aperçu des tendances futures en appariement de graphes.

1.2 Notions de base : rappel et définitions

Dans cette section, nous présentons les principaux concepts et notions de base en appariement de graphes et de mesures de similarité entre les graphes.

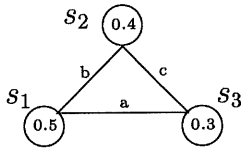
Définition 1.1 : On appelle graphe étiqueté la structure $G = (S, A, \alpha, \beta)$ où

- S est un ensemble fini de sommets,
- A est un ensemble fini d'arêtes,
- $\alpha : S \rightarrow L_S$ est une fonction d'assignation d'étiquette pour les sommets,
- $\beta : A \rightarrow L_A$ est une fonction d'assignation d'étiquette pour les arêtes.

où L_S et L_A sont deux ensembles d'étiquettes respectivement pour les sommets et les arêtes.

Exemple 1.1 :

$$G = (S, A, \alpha, \beta)$$
$$L_S = \{0.4, 0.3, 0.5\} \; L_A = \{a, b, c\}$$
$$S = \{s_1, s_2, s_3\} \; A = \{(s_1, s_2), (s_1, s_3), (s_2, s_3)\}$$
$$\alpha(s_1) = 0.5 \qquad \beta(s_1, s_2) = b$$
$$\alpha(s_2) = 0.4 \qquad \beta(s_1, s_3) = a$$
$$\alpha(s_3) = 0.3 \qquad \beta(s_2, s_3) = c$$



Le graphe $G' = (S', A', \alpha', \beta')$ est dit un sous-graphe de $G = (S, A, \alpha, \beta)$, $G' \subseteq G$, si et seulement si on a :

- $S' \subseteq S$,
- $A' = A \cap (S' \times S')$,
- $\alpha(s) = \alpha'(s) \quad s \in S'$,

– $\beta(a) = \beta'(a) \quad a \in A'$.

Définition 1.2 :

Étant donné deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$, une application $f : S \longrightarrow S'$ réalise un isomorphisme de graphe entre G et G' si et seulement si :

$$|S| = |S'| \text{ et } |A| = |A'|$$
$$\forall s' \in S', \exists! s \in S, \text{ tel que } f(s) = s',$$
$$\forall (s'_i, s'_j) \in A', \exists! (s_i, s_j) \in A, \text{ tel que } f(s_i) = s'_i, f(s_j) = s'_j.$$

L'isomorphisme de graphes peut être aussi réalisé en intégrant les deux fonctions d'assignation $\alpha(\alpha')$ et $\beta(\beta')$.

$$|S| = |S'| \text{ et } |A| = |A'|$$
$$\forall s' \in S', \exists! s \in S, \text{ tel que } f(\alpha(s)) = \alpha'(s'),$$
$$\forall (s'_i, s'_j) \in A', \exists! (s_i, s_j) \in A, \text{ tel que } \beta(f(s_i), f(s_j)) = \beta'(s'_i, s'_j).$$

La notion d'isomorphisme de sous-graphe est très proche de celle d'isomorphisme de graphe. C'est une fonction injective de $|S|$ dans $|S'|$.

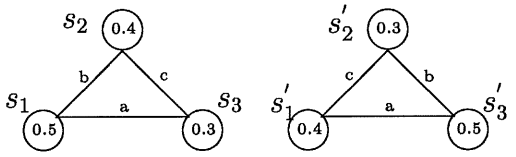
Une application $f : S \longrightarrow S'$ est un isomorphisme de sous-graphe entre G et G' si et seulement si :

$\exists S'' \subseteq S', \exists A'' \subseteq A'$, tels que la restriction de f aux graphes
$G = (S, A, \alpha, \beta)$ et $G'' = (S'', A'', \alpha', \beta')$ est un isomorphisme de graphe.

On distingue deux types d'isomorphisme : isomorphisme de graphe lorsque les deux graphes partagent le même nombre de sommets et d'arêtes et l'isomorphisme de sous-graphe dans le cas contraire. Un troisième type d'isomorphisme peut être distingué qu'on appelle le plus grand commun sous-graphe entre deux graphes. Ce dernier permet de trouver un isomorphisme de graphe entre deux sous-graphes (donc de même taille) appartenant à deux graphes donnés. Il s'agit du sous-graphe en commun possédant le plus grand nombre de sommets parmi tous les sous-graphes commun aux deux graphes.

Exemple 1.2 :

$$G = (S, A, \alpha, \beta) \qquad G' = (S', A', \alpha', \beta')$$
$$L_S = \{0.4, 0.3, 0.5\}\ L_A = \{a, b, c\} \qquad L_{S'} = \{0.4, 0.3, 0.5\}\ L_{A'} = \{a, b, c\}$$
$$S = \{s_1, s_2, s_3\} \qquad S' = \{s_1', s_2', s_3'\}$$
$$A = \{(s_1, s_2), (s_1, s_3), (s_2, s_3)\} \qquad A' = \{(s_1', s_2'), (s_1', s_3'), (s_2', s_3')\}$$
$$\alpha(s_1) = 0.5 \quad \beta(s_1, s_2) = b \quad \alpha'(s_1') = 0.4 \quad \beta'(s_1', s_2') = c$$
$$\alpha(s_2) = 0.4 \quad \beta(s_1, s_3) = a \quad \alpha'(s_2') = 0.3 \quad \beta'(s_1', s_3') = b$$
$$\alpha(s_3) = 0.3 \quad \beta(s_2, s_3) = c \quad \alpha'(s_3') = 0.5 \quad \beta'(s_2', s_3') = a$$
$$f(s_1) = s_3' \qquad f(s_2) = s_1' \qquad f(s_3) = s_2'$$



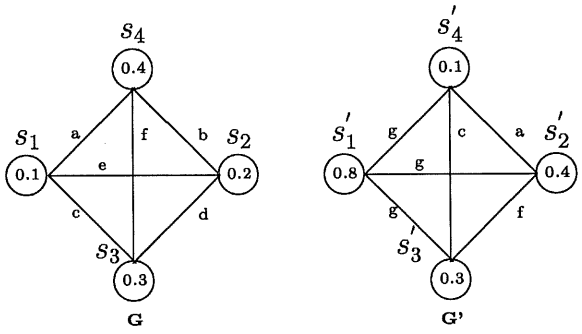
Définition 1.3 :

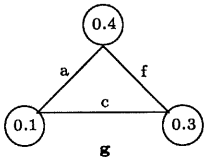
Soit G et G', deux graphes étiquetés. Le plus grand commun sous-graphe g entre G et G' est un sous-graphe à la fois de G et G' possèdant le plus grand nombre de sommets parmi tous les sous-graphes isomorphes entre G et G'.

Exemple 1.3 :

$$G = (S, A, \alpha, \beta) \qquad G' = (S', A', \alpha', \beta')$$
$$L_S = L_{S'} = \{0.1, 0.2, 0.3, 0.4, 0.8\}\ L_A = L_{A'} = \{a, b, c, d, e, f, g\}$$
$$S = \{s_1, s_2, s_3, s_4\} \qquad S' = \{s_1', s_2', s_3', s_4'\}$$
$$A = \{(s_1, s_2), (s_1, s_3), (s_1, s_3), (s_2, s_3), (s_2, s_4), (s_3, s_4)\}$$
$$A' = \{(s_1', s_2'), (s_1', s_3'), (s_1', s_3'), (s_2', s_3'), (s_2', s_4'), (s_3', s_4')\}$$
$$\alpha(s_1) = 0.1 \quad \beta(s_1, s_2) = e \quad \alpha'(s_1') = 0.8 \quad \beta'(s_1', s_2') = g$$
$$\alpha(s_2) = 0.2 \quad \beta(s_1, s_3) = c \quad \alpha'(s_2') = 0.4 \quad \beta'(s_1', s_3') = g$$
$$\alpha(s_3) = 0.3 \quad \beta(s_1, s_4) = a \quad \alpha'(s_3') = 0.3 \quad \beta'(s_1', s_4') = g$$
$$\alpha(s_4) = 0.4 \quad \beta(s_2, s_3) = d \quad \alpha'(s_4') = 0.1 \quad \beta'(s_2', s_3') = f$$

$$\beta(s_2, s_4) = b \qquad \beta'(s_2', s_4') = a$$
$$\beta(s_3, s_4) = f \qquad \beta'(s_3', s_4') = c$$





Définition 1.4 :

On appelle une opération d'édition δ une transformation d'un sommet ou d'une arête d'un graphe $G = (S, A, \alpha, \beta)$.
On définit six types d'opérations d'édition :

1. substitution de sommet : $\delta_{ss}(s_i)$
2. substitution d'arête : $\delta_{sa}(s_i, s_j)$
3. suppression de sommet : $\delta_{ds}(s_i)$
4. suppression d'arête : $\delta_{da}(s_i, s_j)$
5. insertion de sommet : $\delta_{is}(s_i)$
6. insertion d'arête : $\delta_{ia}(s_i, s_j)$

Exemple 1.4 :

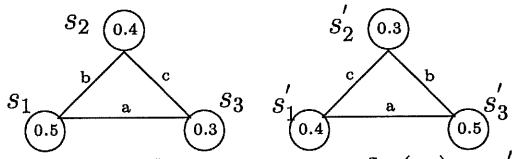
$$G = (S, A, \alpha, \beta) \qquad G' = (S', A', \alpha', \beta')$$
$$L_S = \{0.4, 0.3, 0.5\}\ L_A = \{a, b, c\} \qquad L_{S'} = \{0.4, 0.3, 0.5\}\ L_{A'} = \{a, b, c\}$$
$$S = \{s_1, s_2, s_3\} \qquad S' = \{s_1', s_2', s_3'\}$$

$$A = \{(s_1, s_2), (s_1, s_3), (s_2, s_3)\} \qquad A' = \{(s_1', s_2'), (s_1', s_3'), (s_2', s_3')\}$$

$$\alpha(s_1) = 0.5 \quad \beta(s_1, s_2) = b \qquad \alpha'(s_1') = 0.4 \quad \beta'(s_1', s_2') = c$$

$$\alpha(s_2) = 0.4 \quad \beta(s_1, s_3) = a \qquad \alpha'(s_2') = 0.3 \quad \beta'(s_1', s_3') = b$$

$$\alpha(s_3) = 0.3 \quad \beta(s_2, s_3) = c \qquad \alpha'(s_3') = 0.5 \quad \beta'(s_2', s_3') = a$$



Substitution du sommet s_1 : $\delta_{ss}(s_1) = s_1'$

Substitution de l'arête (s_1, s_2) : $\delta_{sa}(s_1, s_2) = (s_1', s_2')$

Définition 1.5 :

On appelle une séquence d'édition Δ un ensemble de transformation ordonné $\delta_t^1, ..., \delta_t^{|\Delta|}$ transformant un graphe G en un graphe G'
avec $t \in \{ss, sa, ds, da, is, ia\}$

Exemple 1.5 :

$$\Delta = \{\delta_{ss}^1, \delta_{sa}^2, \delta_{ss}^3\} \text{ avec } \Delta(G) = G'$$

G et G' les deux graphes représentés dans l'exemple 1.4.

Définition 1.6 :

On appelle f_c une fonction coût permettant d'associer pour chaque opération d'édition appartenant à Δ un nombre réel positif.

$$f_c : \Delta \rightarrow R$$

Exemple 1.6 :

$$f_c(\delta_{ss}^1) = d(0.5, 0.4) = 0.1$$
$$f_c(\delta_{sa}^2) = D(a, b) = 1$$

d est la distance Euclidienne tandis que D est la différence entre l'ordre de a et de b dans l'alphabet.

Définition 1.7 :

Étant donné deux graphes $G_1 = (S_1, A_1, \alpha_1, \beta_1)$ et $G_2 = (S_2, A_2, \alpha_2, \beta_2)$, on appelle une erreur de correction d'appariement de graphes (*ecag*) de G_1 à G_2, la fonction bijective $f : \overline{S}_1 \rightarrow \overline{S}_2$ où $\overline{S}_1 \subseteq S_1$ et $\overline{S}_2 \subseteq S_2$.

La fonction f peut être interprétée comme une transformation du graphe G_1 en G_2, en appliquant une séquence d'édition Δ.

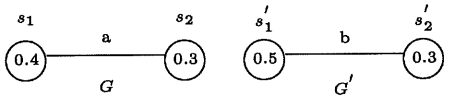
Le coût d'une séquence d'édition Δ relatif à une *ecag* est défini par

$$C(\Delta)_{ecag} = \sum_{i=1}^{|\Delta|} f_c(\delta_t^i)$$

avec $t \in \{ss, sa, ds, da, is, ia\}$.
La fonction f est une ecag optimale (*ecago*) si et seulement si, il n'existe aucune autre ecag f' transformant G_1 et G_2 dont le coût est inférieur à celui de f.

Exemple 1.7 :

Soient deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$



Avec $S = \{s_1, s_2\}$ et $S' = \{s_1', s_2'\}$

f_1 est une ecag entre G et G' telle que

$$f_1 : s_1 \rightarrow s_1'$$
$$f_1 : s_2 \rightarrow s_2'$$

La séquence d'opération d'édition relative à f_1 est

$$\Delta = \{\delta_{ss}^1, \delta_{ss}^2, \delta_{sa}^3\}$$

Avec $\delta_{ss}^1(s_1) = s_1'$, $\delta_{ss}^2(s_2) = s_2'$ et $\delta_{sa}^3(s_1, s_2) = (s_1', s_2')$

Le coût associé à f_1 est

$$C(\Delta)_{f_1} = f_c(\delta_{ss}^1) + f_c(\delta_{ss}^2) + f_c(\delta_{sa}^3)$$
$$f_c(\delta_{ss}^1) = d(0.4, 0.5) = 0.1$$

$$f_c(\delta^2_{ss}) = d(0.3, 0.3) = 0$$
$$f_c(\delta^3_{sa}) = D(a, b) = 1$$
$$C(\Delta)_{f_1} = 1.1$$

1.3 Algorithmes d'appariement de graphes

Plusieurs algorithmes d'appariement de graphes ont vu le jour, particulièrement dans les trente dernières années [19]. Ces algorithmes ont été conçus selon différentes philosophies, avec des outils différents et pour des buts différents. Notre objectif dans cette section est de détailler quelques méthodes fameuses qui ont fait leurs preuves en donnant des résultats satisfaisants sans pour autant aboutir à une solution générale du problème. Une étude comparative est réalisée dans les chapitres 2 et 3 intégrant ces méthodes et les méthodes développés dans le cadre de cette thèse. Pour un compte rendu plus complet, nous redirigeons le lecteur vers Hancock et Wilson [37], Messmer [83] et Llados [74].

1.3.1 Méthodes un-avec-un

Le but ultime des méthodes dites un-avec-un est de réduire la complexité de l'appariement de graphes en intégrant dans le processus uniquement deux graphes. L'appariement de graphe est réalisé par une recherche de l'isomorphisme de graphe, de sous graphe ou du plus grand commun sous graphe. Les méthodes appartenant à cette classe ont principalement utilisé deux approches : premièrement, une recherche arborescente déterministe utilisant des techniques heuristiques pour diriger le processus de recherche vers la solution du problème, deuxièmement, l'optimisation stochastique souvent employée dans la reconnaissance de formes statistiques. Dans ce qui suit, nous allons

détailler une méthode appartenant à la première approche et trois autres méthodes appartenant à la deuxième approche. La première méthode est basée sur une recherche arborescente par rétropropagation [108], une deuxième méthode est basée sur les algorithmes génétiques [114], une troisième est basée sur une analyse spectrale des matrices d'adjacence [111] et finalement une quatrième méthode basée sur la relaxation probabilistique [16].

Recherche arborescente par rétropropagation

La première méthode consiste en une recherche arborescente où chaque sommet de l'arbre représente un isomorphisme de sous-graphe partiel et une arête, son extension par une nouvelle mise en correspondance. La recherche de l'isomorphisme est conduite par l'application d'une séquence d'opérations d'édition notée Δ enchaînée sur un graphe G dont le coût globale noté $C(\Delta)$ est minimal. Les notions d'isomorphisme de sous-graphe, opération d'édition, et séquence d'opération ont été définies dans la section 1.2. Le but principal est de trouver une séquence d'opérations d'édition notée Δ transformant un graphe G en un graphe G' dont le coût $C(\Delta)$ est minimal.

Étant donné deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$, nous appelons appariement p un ensemble de mise en correspondance entre un sommet $s \in S$ et un sommet $s' \in (S' \cup \$)$ telle que chaque sommet des deux graphes est mis en correspondance une et une seule fois dans p. Le symbole $\$$ représente un sommet ou une arête imaginaire absent dans le deuxième graphe. Une séquence d'opérations Δ_p est un ensemble de transformations associé à un appariement p. Pour chaque paire de sommets (s, s') appartenant à p, il existe une opération d'édition dans Δ_p transformant le sommet s en s'. On distingue deux scénarios possibles :

1. $\alpha(s) \neq \alpha'(s')$: l'opération d'édition est une opération de substitution de $\alpha(s)$ par $\alpha'(s')$.

2. $s' = \$$: l'opération d'édition est une opération de suppression du sommet s de G.

Pour ce qui concerne la mise en correspondance des arêtes, trois scénarios possibles peuvent être générés suite à la mise en correspondance des sommets. Étant donné deux paires de sommets appartenant à p notées (s_1, s_1') et (s_2, s_2'), avec s_1 et s_2 appartenant à G et s_1' et s_2' appartenant à G' :

1. $\beta(s_1, s_2) \neq \beta'(s_1', s_2')$: l'opération d'édition est une opération de substitution de l'arête (s_1, s_2) par (s_1', s_2').
2. $(s_1, s_2) \in A$ et $(s_1', s_2') \notin A'$: l'opération d'édition est une opération de suppression de l'arête (s_1, s_2).
3. $(s_1, s_2) \notin A$ et $(s_1', s_2') \in A'$: l'opération d'édition est une opération d'insertion de l'arête (s_1, s_2) dans G.

L'algorithme de recherche de l'erreur de correction d'appariement de graphes (ecag) utilisant une recherche arborescente est formulé de la manière suivante : Une liste OUVERT est introduite afin de sauvegarder au fur et à mesure les appariements partiels p. La liste OUVERT est initialisée par un ensemble de mises en correspondance impliquant le premier sommet s_1 de G avec tous les sommets de G'. Un mise en correspondance supplémentaire est ajoutée à la liste OUVERT impliquant le sommet s_1 et le sommet imaginaire \$. Chaque mise en correspondance constitue un appariement contenant un seul élément. À partir de cette étape, un processus de recherche et de mise à jour de la liste OUVERT est exécuté jusqu'à ce que la liste soit vide. Le processus de recherche débute par une vérification de l'état de la liste. Il prend fin si la liste est vide sinon l'appariement avec le coût minimal est, dans un premier temps, examiné pour déterminer s'il forme un isomorphisme complet ou non ; ensuite il est retiré de la liste. Deux cas possibles sont alors envisageables :

1. L'appariement représente un isomorphisme de sous-graphe complet

c'est-à-dire impliquant tous les sommets du premier graphe G. Dans ce cas, son coût est comparé à celui d'un coût seuil initialisé au début du processus de recherche au coût le plus élevé possible. Le coût seuil est mis à jour dans le cas ou il est supérieur à celui de l'appariement courant.

2. L'appariement n'est pas complet, il est alors mis à jour par l'insertion d'une nouvelle mise en correspondance entre deux sommets n'appartenant pas encore à l'appariement. Une fois le coût de l'appariement calculé, il est de nouveau inséré à la liste OUVERT. Il faut noter que l'insertion de l'appariement est effectuée uniquement si son coût est inférieur au coût seuil.

Le processus de recherche décrit ci-dessus permet de trouver l'erreur de correction d'appariement de graphe optimale. Cependant, afin de trouver la solution optimale, le processus doit effectuer $O(|S||S'|^{|S|})$ mises en correspondance entre les sommets. Pour chaque mise en correspondance, le processus doit effectuer $O(|S|)$ mises en correspondance entre les arêtes. Ainsi $O(|S|^2|S'|^{|S|})$ opérations sont nécessaires pour déterminer la solution optimale.

La progression du processus de recherche, dont nous avons rendu compte dans la description de l'algorithme, est basée sur le coût minimal de l'appariement. La performance de l'algorithme peut être améliorée si d'autres indicateurs sont utilisés pour mieux renseigner l'algorithme et progresser dans la bonne voie. Ce qui réduit considérablement le temps nécessaire pour rechercher la solution optimale. L'algorithme de recherche A^*, par le recours à la notion du coût future permet d'informer le processus de recherche sur l'état qui doit être développé en premier ou le chemin à emprunter. Une méthode d'estimation du coût futur efficace a été proposé par Wong [119]. L'idée de base est d'estimer pour chaque appariement p un coût seuil. Chaque sommet

n'appartenant pas à p est donc apparié à un sommet du deuxième graphe qui lui non plus n'apparentant pas à p. Le coût de chaque opération de substitution de sommet ainsi que le coût de chaque opération de substitution d'arête relative à la mise en correspondance de sommet sont calculés pour former un coût minimum représentant un coût seuil qui doit être ajouté au coût réel de l'appariement p. Il est clair que la complexité du problème n'est pas réduite par l'estimation du coût futur. Au contraire il faut $O(|S|^2|S'|^{|S|+1})$ opérations pour trouver la solution optimale. Cependant, le processus de recherche sera bien informé et se dirigera vers la solution optimale plus rapidement qu'une recherche arborescente sans estimation du coût futur.

Algorithme génétique

Principalement, un processus génétique peut se voir comme un processus de recherche stochastique inspiré du monde réel. En partant d'une population d'individus, une solution à un problème donnée est trouvée par une transformation de la population de départ. Par transformation, on entend une reproduction, une mutation et un croisement des individus formant la population. L'approche génétique a été utilisé dans plusieurs domaines tels que le traitement d'image, la reconnaissance de formes, etc. D'une manière générale, un algorithme génétique peut être représenté par un ensemble de paramètres :

$$(E, O, P, F, C, T)$$

où

1. E : Encodage du problème.
2. O : Opérateurs génétiques.
3. P : Probabilité associée à chaque opérateur.
4. F : Fonction d'évaluation.

5. C : Condition d'arrêt.

6. T : Taille de la population.

Wang et al. [114] ont proposé un algorithme basé sur l'approche génétique pour résoudre le problème de la recherche de l'ecag. L'encodage du problème est un paramètre clé et spécifique pour chaque problème et nécessite une attention particulière. La recherche de l'ecag entre deux graphes est un problème de mise en correspondance entre les sommets des deux graphes suivi d'une mise en correspondance entre les arêtes. Un individu dans la population doit refléter cette mise correspondance nécessaire à l'évaluation et à la recherche de l'ecag. Un individu ou un chromosome représente un ensemble de mise en correspondance entre les sommets. Bien évidement, chaque chromosme doit former un isomorphisme entre les deux graphes. La popluation sera initialisée par un ensemble d'individu généré aléatoirement excepté un seul individu baptisé "*statusmatching*". Le but de cette exception est d'éclairer le processus génétique sur le bon chemin pour trouver le plus vite possible la meilleure solution.

On appelle le "*status*" $s(v)$ d'un sommet appartenant à un graphe donné G la somme de l'étiquette attribuée au sommet v avec toutes les étiquettes des arêtes rattachées au même sommet v.

$$s(v) = \alpha(v) + \sum_{v' \in S} \beta(v, v') \tag{1.1}$$

On appelle $S(G)$ la liste de tous les status d'un graphe donné G et $CanS(G)$ la liste canonique triée telle que

$$CanS(G) = \{s_i | j > i, s_i > s_j\} \tag{1.2}$$

avec s_i et s_j respectivement les sommets d'indice i et j.

Le status matching des deux graphes G et G' n'est autre que la "matrice" formée des deux vecteurs $CanS(G)$ et $CanS(H)$ triés par ordre des sommets.

Dans un problème de recherche d'isomorphisme exact, le status matching ainsi défini permet de trouver s'il existe ou non un isomorphisme entre les deux graphes. Par contre, dans un problème de recherche de l'ecag, le status matching est en mesure d'accélérer le processus de recherche sans aucune garantie de déboucher sur l'ecag optimale. Pour les autres paramètres, le choix des différentes valeurs est une question purement expérimentale qui va être décrit dans le chapitre 2, section 2.3.

Analyse spectrale des matrices d'adjacence

L'algorithme proposé par Umeyama [111] est basé sur une approche analytique utilisant les matrices d'adjacence des deux graphes. L'idée de base de cette méthode est de trouver l'ensemble des mises en correspondance à partir d'une matrice de permutation P. Étant donné deux graphes G et G', la minimisation de la matrice de permutation P permet de déterminer l'ensemble des mises en correspondance entre les graphes G et G'.

$$J(P) = \| PA_G P^T - A_{G'} \| \tag{1.3}$$

Où A_G et $A_{G'}$ représentent respectivement les matrices d'adjacence de G et G'. Ainsi, les deux graphes G et G' sont isomorphes si et seulement si on a $J(P) = 0$ c'est-à-dire $PA_G P^T = A_{G'}$. La minimisation de cette matrice P conduit à trouver une solution optimale. Cependant, comme la recherche d'une solution optimale est un problème combinatoire, une solution approximative est alors proposée afin d'éviter l'explosion combinatoire du problème. L'utilisation des matrices orthogonales dans (1.3), à la place des matrices d'adjacence A_G et $A_{G'}$, permet de trouver une solution approximative au problème.

$$A_G = U_G \Lambda_G U_G^T A_{G'} = U_{G'} \Lambda_{G'} (U_{G'})^T \tag{1.4}$$

Où U_G et $U_{G'}$, deux matrices diagonales associées respectivement à G et G', et

$$\Lambda_G = diag(\alpha_i)\Lambda_{G'} = diag(\beta_i) \tag{1.5}$$

avec α_i et β_i sont respectivement les valeurs propres de A_G et $A_{G'}$.

L'inconvénient majeur de cet algorithme est que son champ d'application est réduit à des graphes de même taille et que son efficacité est limitée à des graphes similaires.

Relaxation probabiliste

Les algorithmes d'optimisation Bayésienne offrent une autre alternative au problème de l'appariement de graphes. Ce type d'algorithme permet d'atteindre une solution au problème dans un temps polynomial. Cependant, aucune garantie sur l'optimalité de la solution n'est acquise. De plus le processus peut à un moment donné cesser d'évoluer vers la solution du problème (minimum local). L'optimisation consiste à minimiser une fonction de similarité. La similarité entre deux éléments est mesurée en terme de probabilité et non pas d'une manière binaire comme c'est le cas pour la relaxation discrète. L'idée de base de cette méthode est de mettre à jour la probabilité de la mise en correspondance entre deux sommets à chaque itération. La probabilité à l'itération n de la mise en correspondance entre un sommet v et un sommet w est $P^n(v \rightarrow w)$. La mise à jour des probabilités des mises en correspondance des sommets est définie par :

$$P^{(n+1)}(v \rightarrow w) = \frac{P^{(n)}(v \rightarrow w)Q(v \rightarrow w)}{\sum_{u \in V} P^{(n)}(u \rightarrow w)Q(u \rightarrow v)} \tag{1.6}$$

$Q(v \rightarrow w)$ mesure la similarité des arêtes en tenant compte de toutes les arêtes partant du sommet v.

$$Q(v \rightarrow w) = \prod_{y \in A_w} \sum_{x \in V} P^{(n)}(x \rightarrow y)R(v, w, x, y) \tag{1.7}$$

Avec $A_w = \{u \in V^{'} | (u, w) \in E^{'}\}$ est l'ensemble des arêtes partant de w et R est la fonction de compatibilité entre les mises en correspondance (v, w) et (x, y).
R est défini par :

$$R(v, w, x, y) = \frac{P(x \rightarrow y | v \rightarrow w)}{P(x \rightarrow y)P(v \rightarrow w)} \tag{1.8}$$

1.3.2 Méthodes un-avec-plusieurs

Par opposition aux méthodes un-avec-un, les méthodes appartenant à la deuxième classe incluent, dans le processus de réduction de la complexité du problème, un ensemble de graphes. On cherche à comparer un graphe donné avec un ensemble de graphes. Pour cela, une des deux alternatives existantes pour réaliser cet objectif, est de construire un représentant de l'ensemble de graphes. Nous allons d'abord définir le concept de graphe médian, puis nous détaillerons la seule méthode existante pour construire le graphe médian.

Graphe Médian

Le concept du graphe médian joue un rôle très important dans la représentation des objets. Lorsque les objets sont représentés par un ensemble de chaînes, un problème tel que la classification des objets, se résume à une recherche de la chaîne médiane [61]. Bunke et ses coéquipiers ont récemment élargi le concept du médian pour les graphes. Étant donné un ensemble de graphes S, le graphe médian est défini comme le graphe équidistant à tous les graphes appartenant à l'ensemble S. En d'autres termes, c'est le graphe dont la somme des distances avec tous les graphes de S est la plus faible. Le graphe résultant de l'opération de recherche est appelé *graphe médian de l'ensemble* si ce dernier appartient à l'ensemble S. Dans le cas contraire, le graphe trouvé est appelé *graphe médian généralisé*. La difficulté de la re-

cherche ou de la construction du graphe médian dépend premièrement de la condition d'appartenance ou non à l'ensemble S et deuxièmement de la nature du calcul complexe de la distance entre deux graphes. La complexité de la construction des deux concepts est exponentielle. Cependant, la complexité du graphe médian de l'ensemble est exponentielle en fonction de la taille des graphes et polynomiale en fonction de la cardinalité de l'ensemble S. La complexité du graphe médian généralisé est doublement exponentielle en fonction de la taille des graphes et de la cardinalité de l'ensemble S. Afin de rendre le processus de recherche faisable dans la pratique, des méthodes approximatives devront être utilisées pour atténuer la complexité élevée du problème.

Définition 1.8 :

Soit S un ensemble de graphes étiquetés. Soit C l'ensemble de tous les graphes qui peuvent être construits à partir de l'ensemble des étiquettes E_S et E_A respectivement pour les sommets et les arêtes. Sachant que $S \subseteq C$, le graphe médian généralisé et le graphe médian de l'ensemble sont définis respectivement par :

$$\overline{g} = arg \min_{g \in C} SOD(g, S) \tag{1.9}$$

$$\widehat{g} = arg \min_{g \in S} SOD(g, S) \tag{1.10}$$

avec

$$SOD(g, S) = \sum_{i=1}^{|S|} d(g, g_i) \tag{1.11}$$

Les deux concepts minimisent la somme des distances SOD entre le graphe médian et l'ensemble des graphes de S. La seule différence réside dans l'espace des graphes utilisé pour la recherche. Plusieurs méthodes existent pour le calcul de la distance entre deux graphes. Une des méthodes qui peut être appliquée pour le calcul de cette distance est l'algorithme proposé dans

le chapitre II (Un nouvel algorithme pour l'appariement de graphes basé sur la mise en correspondance des sommets). Pour d'autres méthodes le lecteur peut consulter le travail de Shenker [96].

Algorithme génétique

Les algorithmes génétiques s'inscrivent dans le cadre des méthodes d'optimisation discrètes. L'idée de base est inspirée de la théorie de l'évolution biologique. Le recours aux algorithmes génétiques a montré qu'ils ont un potentiel considérable pour résoudre des problèmes dits complexes. Tel est le cas pour l'appariement de graphes. En partant d'une population donnée, le processus s'engage à faire évoluer cette population vers une autre proche de la solution du problème. L'évolution ou la convergence de la population est réalisée principalement par le biais de deux opérateurs appelés le croisement et la mutation. La prise de décision sur la qualité de la population est basée sur l'analyse d'une fonction "$fitness$".

L'unique algorithme dans la littérature pour construire le graphe médian a été proposé par Jiang [56] [58] [59]. Le point saillant dans cet algorithme est la manière de représenter une solution donnée. En effet, un chromosome représente les différentes mises en correspondances entre un graphe candidat et l'ensemble des graphes appartenant à S. La fonction de fiteness n'est autre que la somme des distances SOD. Chaque chromosome est un tableau d'entiers dont la taille est égale au nombre total des sommets de tous les graphes de S. La valeur d'un élément u dans le tableau indique l'indice d'un sommet v d'un graphe k appartenant à S. L'élément u est un sommet du graphe candidat. Une mise en correspondance entre le graphe candidat et le graphe k est représentée par les deux sommets d'indices u et v. Une valeur supérieure ou égale à 1 indique que la mise en correspondance est réalisée par une opération de substitution. Une valeur égale à zéro indique que le sommet

v est inséré dans le graphe candidat. La contribution du sommet u dans la fonction SOD est donnée par :

$$\sum_{i=1}^{k_1} c_{ns}(u, v^i) + \sum_{i=1}^{k_2} c_{nd}(u) \tag{1.12}$$

avec $c_{ns}(u, v^i)$ est le coût de substitution du sommet u par le sommet v du graphe i et $c_{nd}(u)$ est le coût de la suppression du sommet u. Le nombre total de sommets dans tous les graphes de S est égale à $k_1 + k_2$.

1.4 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base en appariement de graphe utilisés en reconnaissance de formes. L'appariement de graphe consiste à identifier toutes les mises en correspondance entre les sommets d'un premier graphe (requête) avec ceux d'un deuxième graphe (modèle). L'identification est réalisée en recherchant un isomorphisme entre les deux graphes. Nous avons distingué trois types d'isomorphisme. L'isomorphisme de sous-graphe, l'isomorphisme de graphe et le plus grand commun sous-graphe.

L'isomorphisme dans le cas général est une fonction bijective entre les sommets d'un premier graphe et les sommets d'un deuxième graphe. Cette fonction assure non seulement la mise en correspondance entre les sommets mais aussi la mise en correspondance entre les arêtes des deux graphes quand l'on tient compte des mises en correspondance entre les sommets. Dans le cas où les deux graphes ont le même nombre de sommets et arêtes, l'isomorphisme trouvé est un isomorphisme de graphe, sinon c'est un isomorphisme de sous-graphe. Dans le cas où seulement un sous ensemble de sommets d'un graphe requête est isomorphe à un sous ensemble de sommets d'un graphe modèle et qu'il est le plus grand parmi tous les sous-ensembles isomorphes à d'autres sous-ensembles, on parle alors du plus grand commun sous-graphe.

Nous avons regroupé les méthodes d'appariement de graphes en deux classes : des méthodes permettant de comparer un graphe avec un autre et des méthodes permettant de comparer un graphe avec plusieurs graphes. Pour la première classe, nous avons décrit quatre méthodes que nous avons jugées intéressantes. Pour la deuxième classe, nous avons tout d'abord commencer par décrire le concept du graphe médian, puis nous avons détaillé la seule méthode existante. Les méthodes présentées dans ce chapitre ont fait l'objet d'une étude comparative avec nos méthodes développées dans le cadre de ce travail. Sans être exhaustif, nous avons essayé de recenser les méthodes d'appariements de graphe existant dans la littérature. Pour cela, nous avons dressé un tableau récapitulatif qui présente, pour chaque méthode, la nature de la solution recherchée, la complexité et un pointeur vers sa référence. La nature de la solution donne le champ d'application de la méthode. Nous distinguons deux types de méthodes, les méthodes dites exactes et les méthodes inexactes.

Méthode	E :I	C	Ref.
Méthodes déterministe			
Corneil & Gotlieb	E	NP	[19]
Ullmann	E	NP	[110]
Tsai & Fu	I	NP	[108]
Sanfeliu & Fu	I	NP	[95]
Shapiro & Haralick	I	NP	[99]
Wong & You	I	NP	[121]
Eshera & Fu	I	NP	[26]
VF	E	NP	[17]
Scmidt et Druffel	I	NP	[98]
Relaxation distcréte	I	NP	[116]
Algorithme de McKay	E	NP	[79]
Node mapping	I	NP	[39]
Graphe d'association	E	NP	[89]
Méthodes non déterministes			
Algorithme génétique	I	P	[114]
Méthodes spectrales	I	P	[111]
Programmation linéaire	I	P	[1]
Simulated annealing	I	P	[38]
EM	I	P	[22]
Relaxation probabiliste	I	P	[16]
Réseaux de neurone	I	P	[106]

TAB. 1.1: Algorithmes d'appariement de graphes.

Chapitre 2

Un nouvel algorithme pour l'appariement de graphes basé sur la mise en correspondance des sommets

2.1 Introduction

Dans ce deuxième chapitre, nous proposons un algorithme d'appariement de graphes. L'algorithme développé permet de trouver une solution approximative, souvent très proche de l'optimale. L'appariement trouvé représente un isomorphisme de graphe ou de sous-graphe entre deux graphes donnés, en utilisant une décomposition du processus de recherche en plusieurs phases dont le nombre est K. Le paramètre K indique le nombre maximal de phases qui peut être utilisé pour trouver l'isomorphisme. La valeur de K peut varier entre 1 et le nombre de sommets dans le plus grand graphe. L'efficacité de l'algorithme résulte de la faible valeur de K utilisée dans l'algorithme, ce qui

réduit d'une manière considérable l'espace de recherche.

Dans ce chapitre, nous allons exposer en premier lieu un algorithme initial d'appariement de graphes basé sur la mise en correspondance des sommets et des arêtes, utilisant la substitution comme outil de calcul de similarité entre les deux graphes (Section 2.2). Nous proposons ensuite une version générale de cet algorithme, introduisant les opérations d'insertion et de suppression des sommets et des arêtes. Dans cette version générale, nous avons introduit de nouvelles techniques heuristiques dans le but d'accélérer le processus de recherche. Ces nouvelles techniques contribuent de manière significative à l'élimination des *"mauvais"* appariements et à guider rapidement le processus de recherche vers la solution du problème. Dans la section 2.3, nous présentons les différentes expérimentations réalisées. Les expériences menées dans le cadre de ce chapitre comportent deux volets. Premièrement, nous montrons l'efficacité de l'algorithme sur des graphes générés aléatoirement. Deuxièmement, nous ferons état d'une étude comparative incluant l'algorithme développé dans ce chapitre et les quatre algorithmes décrits dans le chapitre 1. Nous concluons enfin notre exposé dans la section 2.4.

2.2 Appariement de graphes basé sur la mise en correspondance des sommets

L'appariement de graphes consiste à chercher des mises en correspondance entre les sommets d'un premier graphe et les sommets d'un deuxième graphe, d'une part, et des mises en correspondance entre les arêtes des deux graphes, d'autre part. Un appariement de graphes généré par application des opérations d'édition est interprété comme une distance (ecag) entre les deux graphes mis en correspondance. Notre but est de rechercher l'appariement dont la distance est la plus faible. Une première solution consiste à minimi-

ser cette distance en intégrant dans le processus de recherche à la fois les mises en correspondance des sommets et les mises en correspondance des arêtes. Malheureusement, cette idée n'est pas en mesure de donner des résultats même pour des graphes de faibles tailles. Le processus de minimisation est un problème difficile. Le temps nécessaire pour trouver une solution croît exponentiellement en fonction de la taille des deux graphes. Une solution approximative s'impose pour éviter la complexité élevée du problème.

L'idée de base de notre algorithme repose sur une séparation entre la mise en correspondance des sommets et la mise en correspondance des arêtes. Afin de réduire l'espace de recherche, la recherche de l'appariement est réalisée en minimisant tout d'abord le coût des mises en correspondance des sommets. L'algorithme procède par une exploration de l'espace de recherche en plusieurs phases (K). Chaque phase est composée de plusieurs étapes. Nous estimons que quelques phases sont nécessaires pour atteindre la minimisation du coût global. L'hypothèse utilisée dans l'algorithme stipule que le processus de recherche est guidé par la mise en correspondance des sommets en premier lieu, suivi d'une mise en correspondance entre les arêtes des deux graphes. Intuitivement, cette hypothèse est très raisonnable dans les applications pratiques, vu que la similarité entre deux objets en premier lieu est une similarité entre leur contenu suivi d'une similarité spatiale des objets.

2.2.1 Algorithme de base

Etant donné deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$, l'algorithme proposé dans ce chapitre permet de trouver l'ecag en K phases d'itération. Dans chaque phase, exception faite pour la première phase, $|S|$ étapes sont nécessaires pour explorer et analyser tous les appariements en cours. Ici, on suppose que le graphe G est plus petit ou égal que G' en terme de nombre de sommets, c'est-à-dire $|S| \leq |S'|$. En utilisant une valeur très

grande pour K, l'algorithme est en mesure de trouver l'*ecago, l'erreur de correction de l'appariement de graphes optimal.* Il faut signaler que la valeur de K ne peut jamais dépasser en aucun cas la valeur de |S|. Cependant, l'algorithme est capable de trouver de très bons appariements (souvent optimaux) en utilisant une valeur de K beaucoup moins grande que $|S|$ (ex. $K = 3$, $K = 5$, etc.). Ce qui est un avantage considérable, permettant à la fois de ne pas explorer tout l'espace de recherche et d'introduire une flexibilité sur la qualité du résultat attendu.

Le processus d'appariement est itératif et se déroule comme suit. Dans la première phase, l'algorithme examine les appariements qui comportent les meilleures mises en correspondance des sommets. Par "meilleur", on entend les mises en correspondance dont la valeur est la plus faible parmi l'ensemble des mises en correspondance. L'ensemble des mises en correspondances entre deux graphes comporte $|S| \times |S'|$ éléments. L'appariement dont l'ecag est minimale sera retenu. Dans la deuxième phase, l'algorithme examine les appariements qui comportent les deuxièmes meilleures mises en correspondance des sommets. L'appariement dont l'ecag est minimal sera retenu et comparé à celui de la première phase. Celui dont la valeur de l'ecag est minimale sera retenu. Ce processus de recherche est répété K fois.

Description de l'algorithme

Formellement l'algorithme est décrit de la manière suivante : étant donné deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$, une matrice $P = (p_{ij})$ de dimension $|S| \times |S'|$ est introduite. Chaque élément ou mise en correspondance p_{ij} dans P représente la similarité entre le sommet i du premier graphe et le sommet j du deuxième graphe. Une deuxième matrice binaire $B = (b_{ij})$ de dimension $|S| \times |S'|$ est introduite afin d'identifier à chaque instant le ou les mises en correspondance les plus prometteuses qui formeront

le ou les différents appariements. L'algorithme commence par l'initialisation de la matrice P suivi de celle de B. Chaque élément p_{ij} de la matrice P est initialisé à $p_{ij} = d(\alpha(s_i), \alpha'(s'_j))$. La matrice B est tout d'abord initialisée à zéro, ensuite un seul élément dans chaque ligne est choisi et mis à jour à 1 selon le critère suivant : un élément mis à 1 dans B correspond à l'élément dans la matrice P qui a la valeur la plus faible de tous les éléments de sa ligne.

Au cours de la première phase, l'algorithme identifie l'unique appariement à partir de la matrice B. Si l'appariement forme un isomorphisme, on calcule l'erreur engendrée par les mises en correspondance des différents sommets à partir de la matrice P. L'erreur trouvée est additionnée à l'erreur générée à partir des mises en correspondances des arêtes relatives aux mises en correspondances des sommets. L'appariement ainsi que son erreur seront retenus. Un appariement est considéré valide si et seulement si il représente un isomorphisme de graphes ou de sous-graphes.

Au cours de la deuxième phase, et de façon similaire pour toutes les autres phases, l'algorithme identifie les mises en correspondance possédant les deuxièmes meilleures valeurs dans la matrice P. Pour optimiser le processus de recherche, l'identification est réalisée en plusieurs étapes. Chaque étape représente le traitement d'une ligne dans la matrice P. Maintenant, pour chaque élément identifié, la valeur de son correspondant dans la matrice B est mise à jour à 1. L'algorithme identifie tous les différents appariements qui peuvent être générés, calcule leurs erreurs respectives et compare le meilleur appariement à celui trouvé dans la phase précédente. Dans le cas où aucun appariement n'a été retenu dans la première phase, le meilleur des appariements valides de la deuxième phase sera retenu. Le même processus est répété $|S|$ fois pour achever le traitement de la deuxième phase.

La différence entre la première phase et les autres phases est que dans la

première phase, l'examen des différents appariements identifiés est réalisé une fois que toutes les lignes ont été initialisées tandis que pour les autres phases, l'examen est réalisé ligne par ligne (d'ou la notion d'étape). Un appariement est considéré comme valide et peut être examiné si et seulement si chaque élément, sommet ou arête est mis en correspondance avec un et un seul élément. Une matrice B qui a une ligne dont la valeur de tous ces éléments est égale à zéro, ne permet pas de générer un appariement. Ce qui explique que l'examen des appariements dans la première phase est réalisé une fois que toutes les lignes ont été initialisées.

Une implémentation directe de cet algorithme ne permet pas d'éviter la redondance des appariements. Chaque appariement généré à partir de la matrice B dans une phase donnée sera de nouveau généré dans les phases suivantes. Afin d'éviter ce phénomène de redondance, nous introduisons une procédure permettant de garder en mémoire à chaque étape une copie de la matrice B notée B' et d'ajuster la matrice B afin de générer uniquement de nouveaux appariements. À la fin de l'identification et la sélection, la procédure met à jour la matrice B.

Pour chaque phase (exception faite pour la première phase) et pour chaque ligne de la matrice B, l'algorithme copie la matrice B dans B' et identifie le ou les meilleurs mises en correspondance en ignorant les mises en correspondance qui ont été déjà sélectionnées. Une fois l'élément identifié et avant de mettre à jour sa valeur dans la matrice B, tous les éléments de la ligne examinée seront mis à zéro. La mise à jour des éléments identifiés peut alors avoir lieu non pas dans la matrice B uniquement mais aussi dans la matrice B'. De cette manière et en tout temps, la ligne examinée comportera uniquement un et un seul élément à 1, tous les autres sont à 0. De ce fait, aucun appariement sélectionné dans une phase précédente ne sera sélectionné une deuxième fois. En pratique, nous avons besoin uniquement d'un

tableau pour sauvegarder et restaurer les mises en correspondance déjà visitées. Seule la copie de la ligne en cours est suffisante pour éviter le problème de redondance.

Algorithme :Appariment_Sommets_Base()

Entrées :

1) Deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$.

2) Le nombre de phases K.

Sortie :

1) Le meilleur appariement p.

Début

1. Initialiser P :

   Pour chaque i, j avec $1 \leq i \leq |S|$ et $1 \leq j \leq |S'|$ faire

   $p_{ij} = d(\alpha(s_i), \alpha'(s'_j))$.

2. Initialiser B :

   Pour chaque i, j avec $1 \leq i \leq |S|$ et $1 \leq j \leq |S'|$ faire

   $b_{ij} = 0$.

3. Pour $Phase\_Courante$ allant de 1 jusqu'a K faire

   Si $(Phase\_Courante == 1)$

   Alors

   Pour chaque i avec $1 \leq i \leq |S|$ faire

   $j = arg \min_{1 \leq k \leq |S'|} p_{ik}$.

   $b_{ij} = 1$.

   Fin Pour

   $p = Evaluation\_Appariement(B)$.

   Sinon

   Pour chaque i avec $1 \leq i \leq |S|$ faire

$B' = B$.

Pour chaque j avec $1 \leq j \leq |S'|$ faire

$b_{ij} = 0$.

Fin Pour

$j = arg \min_{1 \leq k \leq |S'|, b'_{ik}=0} p_{ik}$.

$b_{ij} = 1$ et $b'_{ij} = 1$.

$p = Evaluation\_Appariement(B)$.

$B = B'$

Fin Pour.

Si chaque $(b_{ij} == 1)$ avec $1 \leq i \leq |S|$ et $1 \leq j \leq |S'|$

Alors $Phase\_Courante = K$.

Fin

Procédure : Evaluation-Appariement(B)

Sortie :

1) Le meilleur appariement p.

Début

Pour chaque appariement valide de B faire

1. Calculer l'$ecag_s$.
2. Calculer l'$ecag_a$.
3. Si $ecag < ecag_s + ecag_a$

   Alors $ecag = ecag_s + ecag_a$.

   Mettre à jour le meilleur appariement p.

Fin

2.2.2 Extension de l'algorithme

Description de l'algorithme

Dans sa version de base, l'algorithme d'appariement de graphe proposé dans ce chapitre inclut uniquement la substitution pour éditer les sommets et les arêtes des deux graphes. Nous allons proposer dans cette sous-section une extension de cet algorithme pour introduire les autres opérations d'édition à savoir l'insertion et la suppression. D'un autre coté, nous allons présenter des techniques heuristiques introduites pour diriger rapidement le processus vers la solution du problème. Cela nous permet de réduire considérablement le temps nécessaire pour trouver une solution. Dans la première partie de cette sous-section, nous allons décrire l'introduction de l'insertion et de la suppression pour l'édition des graphes. La deuxième partie sera consacrée aux techniques heuristiques.

L'introduction de l'insertion et de la suppression des sommets et des arêtes ouvre la porte aux diverses transformations possibles d'un graphe donné en un autre graphe sans aucune restriction sur la taille des deux graphes. Cependant, en pratique, cette introduction rend le processus de recherche plus complexe. Étant donné deux graphes, la transformation d'un graphe en un autre, ce qui induit un calcul de similarité ou de distance, peut être effectuée de plusieurs façons. Nous avons opté pour une transformation qui tient compte de la structure des deux graphes. Lorsque les deux graphes ont le même nombre de sommets et d'arêtes, seule la substitution sera appliquée pour transformer le premier graphe. Lorsque le premier graphe est plus petit en terme de nombre de sommets, la substitution et la suppression seront appliquées pour transformer le deuxième graphe. Enfin, lorsque le premier graphe est plus grand en terme de nombre de sommets, la substitution et l'insertion seront appliquées pour transformer le deuxième graphe. Ce schéma

de transformations recouvre les différents scénarios quant à la structure des deux graphes.

Étant donné deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha, \beta)$. Dans le cas où $|S| < |S'|$, la matrice P comporte $(|S'| \times |S'|)$ éléments. Les $|S|$ premières lignes seront initialisées en calculant la similarité entre les sommets du premier graphe avec ceux du deuxième graphe. Les $(|S'| - |S|)$ dernières lignes seront inutilisées par la valeur des étiquettes des sommets du deuxième graphe. À partir des $|S|$ premières lignes, le processus choisira les $|S|$ sommets du premier graphe qui seront appariés avec $|S|$ sommets du deuxième graphe. À partir des $(|S'| - |S|)$ dernières lignes, le processus choisira les sommets du deuxième graphe qui seront supprimés. Dans le cas où $|S| > |S'|$, la matrice P comporte $(|S| \times |S|)$ éléments. Les $|S'|$ premières colonnes seront initialisées en calculant la similarité entre les sommets du premier graphe avec ceux du deuxième graphe. Les $(|S| - |S'|)$ dernières colonnes seront initialisées par la valeur des étiquettes des sommets du premier graphe. À partir des $|S'|$ premières colonnes, le processus choisira les $|S'|$ sommets du deuxième graphe qui seront appariés avec $|S'|$ sommets du premier graphe. À partir des $(|S| - |S'|)$ dernières colonnes, le processus choisira les sommets du premier graphe qui seront supprimés.

Pour des raisons de commodité, nous allons présenter l'algorithme en traitant le cas ou $|S| \leq |S'|$. En d'autres termes, le premier graphe est égal au deuxième ou plus petit que lui pour ce qui est du nombre de sommets. Il faut noter que pour chaque insertion (resp. suppression) d'un sommet, une ou plusieurs autres insertions (resp. suppressions) seront suivies pour corriger les arêtes. Le coût total de ces différentes insertions (resp. suppression) des arêtes sera additionné au coût engendré par la correction des sommets.

Algorithme :Appariement_Sommets_Base_Extension()

Entrées :

1) Deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$.

2) Le nombre de phase K.

Sortie :

1) Le meilleur appariement p.

Début

1 : Initialiser P :

Pour $i = 1, ..., |S|$ faire

Pour $j = 1, ..., |S'|$ faire

$p_{ij} = d(\alpha(s_i), \alpha'(s_j))$.

FinPour

FinPour

Pour $i = |S| + 1, ..., |S'|$ faire

Pour $j = 1, ..., |S'|$ faire

$p_{ij} = abs(\alpha'(s_j))$.

FinPour

FinPour

2 : Initialiser B :

Pour $i; j = 1, ..., |S'|$ faire

$b_{ij} = 0$.

FinPour

Pour $i = 1, ..., |S'|$ faire

$j = arg \min_{1 \leq k \leq |S'|} p_{ik}$.

FinPour

$3 : Phase\_Courante = 1.$

Tant que $Phase\_Courante \leq K$ faire

Si $(Phase\_Courante == 1)$

Alors $Extraction\_Evaluation\_Appariement(B, 0)$

Sinon Pour $i = 1, ..., |S'|$ faire

$B'(i, .) = B(i, .).$

Pour $j = 1, ..., |S'|$ faire

$b_{ij} = 0.$

Fin Pour

$j = arg \min_{1 \leq k \leq |S'|, b'_{ik}=0} p_{ik}.$

$Extraction\_Evaluation\_Appariement(B, 1).$

$B(i, .) = B'(i, .).$

FinPour

FinSi

$Phase\_Courante++.$

FinTant que

Fin

L'algorithme tel qu'il a été décrit dans sa version de base consiste à extraire un ou plusieurs appariements dans chaque phase et à chaque étape, vérifier pour chaque appariement s'il réalise un isomorphisme ou non. Dans l'affirmative, l'algorithme calcule son ecag et retient celui qui possède l'ecag minimal. La première technique introduite consiste à contrôler les appariements au fur et à mesure de leur construction. En partant d'un appariement vide, une nouvelle mise en correspondance est insérée dans l'appariement à chaque étape. Le contrôle de l'appariement consiste à vérifier, à chaque étape, si l'appariement partiel forme un isomorphisme partiel ou non. Si ce n'est pas le cas, tous les appariements futurs qui contiennent cet appariement partiel

ne seront pas pris en compte. Dans le cas où l'appariement partiel forme un isomorphisme, son ecag partiel est calculée et comparée au meilleur appariement. Cet appariement partiel sera rejeté dans le cas où son ecag est plus grande que le meilleur. De la même manière, tous les appariements futurs qui contiennent cet appariement partiel ne seront plus sélectionnés. La deuxième partie de cette technique compare l'ecag partiel avec l'ecag du meilleur appariement. L'algorithme ne peut bénéficier de cette deuxième partie uniquement lorsqu'il réussit à trouver le premier appariement. Cette restriction sera allégée par la deuxième technique.

La première technique permet de contrôler l'évolution de la construction de l'appariement. À chaque étape, un appariement partiel est vérifié. Une deuxième technique a été introduite pour mémoriser le plus grand appariement partiel au fur et à mesure qu'on avance dans le processus de recherche. Dans de rares cas, le processus de recherche est incapable de trouver un appariement formant un isomorphisme après K phases. La technique consiste à forcer la construction d'un appariement en utilisant le plus grand appariement partiel minimisant l'erreur de correction ecag trouvée à la fin de la phase K. Un sous-ensemble de mises en correspondance est sélectionné pour compléter l'appariement. Cette technique peut être aussi employée dès la fin de la première phase $Phase\_Courante == 1$. L'algorithme peut bénéficier alors de l'apport de la première technique (deuxième partie) dès la fin de la première phase. Une fois qu'un appariement a été trouvé, l'algorithme abandonne cette deuxième technique. L'appariement trouvé sera utilisé pour vérifier les appariements partiels.

Procédure : $Extraction\_Evaluation\_Appariement(B, a)$
Début
Si $(a == 0)$

Alors Pour $i = 1, ..., |S'|$ faire

$M = Extraire\_Appariement()$.

FinPour

Si (M est un isomorphisme)

Alors

Calculer l'$ecag_s$.

Calculer l'$ecag_a$.

$ecag = ecag_s + ecag_a$.

FinSi

Sinon Pour $k = 1, ..., Max$ (Max est le nombre d'appariments possibles)

Pour $i = 1, ..., |S'|$ faire

$M = Extraire\_Appariement()$.

Fin Pour

Pour $i = 1, ..., |S'|$ faire

Si ($M(1, .., i)$ est un ismorphsime partiel)

Alors

Calculer l'$ecag_s$ partielle.

Si ($ecag_s < ecag$)

Alors

Calculer l'$ecag_a$ partielle.

Si ($ecag_s + ecag_a < ecag$)

Alors

Si ($i == |S'|$)

Alors

$ecag = ecag_s + ecag_s$.

Sauvegarder l'appariment M.

Fin Si

Sinon ($i = |S'| + 1$)

Fin Si

Sinon ()$i = |S'| + 1$)
FinSi
FinSi
FinPour
FinPour
FinSi
Fin

2.2.3 Complexité

Théorème

La complexité de l'algorithme pour un nombre de phases égale à K est de l'ordre de $O(|S|^2 K^{|S|})$.

Démonstration

Étant donné deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$. À l'étape i de la phase K, la ligne i de la matrice B contient un seul élément à 1, les $i-1$ premières lignes contiennent K éléments à 1 et les $(|S| - i)$ dernières lignes contiennent $K - 1$ éléments à 1. À la fin de cette étape, la matrice B contient $(i^{|S|} - (i-1)^{|S|})$ éléments égales à 1. À la fin de cette phase K, le nombre d'appariements issu uniquement de cette phase est $(K^{|S|} - (K-1)^{|S|})$. Le cumul du nombre d'appariements pour les K phases est $K^{|S|}$. La mise en correspondance des arêtes pour chaque appariement nécessite $O(|S|^2)$ opérations. Ainsi la complexité de l'algorithme est $O(|S|^2 K^{|S|})$.

2.2.4 Exemple

Dans cette sous-section, nous présenterons un exemple pour détailler la démarche adoptée pour la recherche de l'isomorphisme de sous-graphe selon l'algorithme présenté dans ce chapitre. La figure 2.1 montre deux graphes G et G'. Le premier et le deuxième graphe comportent respectivement trois et quatre sommets. Le nombre à l'intérieur du cercle représente l'étiquette du sommet tandis que celui à l'extérieur représente l'ordre du sommet. Le nombre à coté de chaque ligne reliant deux sommets représente l'étiquette de l'arête.

$$G = (S, A, \alpha, \beta) \qquad G' = (S', A', \alpha', \beta')$$

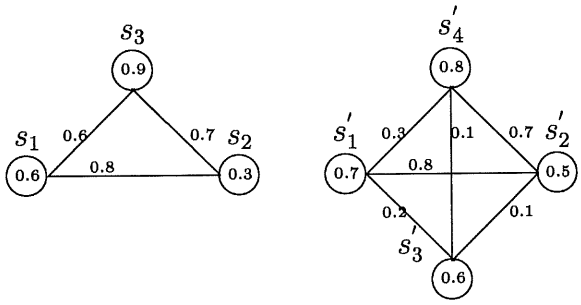
$$L_S = L_{S'} = \{0.3, 0.5, 0.6, 0.7, 0.8, 0.9\} \; L_A = L_{A'} = \{0.1, 0.2, 0.3, 0.6, 0.7, 0.8\}$$

$$S = \{s_1, s_2, s_3\} \qquad S' = \{s_1', s_2', s_3', s_4'\}$$

$$A = \{(s_1, s_2), (s_1, s_3), (s_2, s_3)\}$$

$$A' = \{(s_1', s_2'), (s_1', s_3'), (s_1', s_3'), (s_2', s_3'), (s_2', s_4'), (s_3', s_4')\}$$

$$\alpha(s_1) = 0.6 \qquad \beta(s_1, s_2) = 0.8 \qquad \alpha'(s_1') = 0.7 \qquad \beta'(s_1', s_2') = 0.8$$
$$\alpha(s_2) = 0.3 \qquad \beta(s_1, s_3) = 0.6 \qquad \alpha'(s_2') = 0.5 \qquad \beta'(s_1', s_3') = 0.2$$
$$\alpha(s_3) = 0.9 \qquad \beta(s_2, s_3) = 0.7 \qquad \alpha'(s_3') = 0.6 \qquad \beta'(s_1', s_4') = 0.3$$
$$\alpha'(s_4') = 0.8 \qquad \beta'(s_2', s_3') = 0.1$$
$$\beta'(s_2', s_4') = 0.7$$
$$\beta'(s_3', s_4') = 0.1$$



0.1	0.1	0	0.2
0.4	0.2	0.3	0.5
0.2	0.4	0.3	0.1

TAB. 2.1: Matrice P.

0	0	1	0
0	1	0	0
0	0	0	1

TAB. 2.2: Matrice B.

Afin de faciliter la compréhension de l'algorithme, nous allons utiliser la version de base qui inclut uniquement l'opération de substitution. L'algorithme décrit plus haut comporte trois étapes. La première ainsi que la deuxième étape concernent l'initialisation des deux matrices. Le noyau de l'algorithme est présenté dans la troisième étape qui fait appel à la procédure *Evaluation_Appariement*().

À la fin de l'étape 1, la matrice P contiendra la similarité entre les sommets des deux graphes. Le tableau 2.1 montre le contenu de la matrice P après initialisation.

À la fin de l'étape 2, tous les éléments de la matrice B sont initialisés à 0. L'étape 3, comporte K phases. Supposons que K est égale à 3. Lors de la première phase, l'algorithme dans un premier temps initialise de nouveau la matrice B, en affectant un seul élément à 1 pour chaque ligne. Le tableau 2.2 montre le contenu de la matrice B après cette initialisation.

Dans un deuxième temps, l'algorithme extrait à partir de cette matrice, le seul appariement existant, soit $(s_1, s_3^{'}), (s_2, s_2^{'}), (s_3, s_4^{'})$. Ensuite, le coût associé à cet appariement est calculé en regardant la matrice P, soit 0.3 (1.5 en regardant les arrêtes). Le processus alors passe à la phase 2, c'est-à-dire

1	0	0	0
0	1	0	0
0	0	0	1

1	0	1	0
0	1	0	0
0	0	0	1

TAB. 2.3: Matrice B et B', $Phase\_Courante == 2$, $ligne == 1$.

1	0	1	0
0	0	1	0
0	0	0	1

1	0	1	0
0	1	1	0
0	0	0	1

TAB. 2.4: Matrice B et B', $Phase\_Courante == 2$, $ligne == 2$.

$Phase\_Courante == 2$. Maintenant, le traitement sera effectué en analysant la matrice B ligne par ligne, en commencant par la ligne 1. D'abord, une copie de la matrice B est sauvegardée dans une matrice intermédiaire B'. Ensuite, tous les éléments de la première ligne seront mis à 0. Le processus cherche alors l'élément qui possède la valeur la plus faible dans P et qui n'a pas été sélectionné dans la phase précédente. Le contenu de la matrice B et B' est donné par le tableau 2.3.

À partir de la matrice B, l'algorithme extrait le seul appariement existant et réalisant un isomorphisme, soit $(s_1, s_1^{'}), (s_2, s_2^{'}), (s_3, s_4^{'})$ et calcule son coût, soit 1.3. En comparant son coût à celui obtenu dans la première phase, ce dernier sera retenu. Le processus continue la recherche en analysant la deuxième ligne de la matrice B. Le contenu de la matrice B et B' est donné par le tableau 2.4. À partir de la matrice B, l'algorithme extrait le seul appariement existant, soit $(s_1, s_1^{'}), (s_2, s_3^{'}), (s_3, s_4^{'})$ et calcule son coût, soit 1.4. En comparant son coût à celui obtenu dans la première phase, ce dernier ne sera pas retenu. Le processus continue la recherche en analysant la troisème ligne de la matrice B. Le contenu de la matrice B et B' est donné par le tableau 2.5.

1	0	1	0
0	1	1	0
1	0	0	0

1	0	1	0
0	1	1	0
1	0	0	1

TAB. 2.5: Matrice B et B', $Phase\_Courante == 2$, $ligne == 3$.

2.3 Résultats expérimentaux

Nous avons expérimenté le nouvel algorithme d'appariement présenté dans ce chapitre sur une base de graphes aléatoires. L'expérimentation est composée de deux parties. Dans la première partie, nous allons montrer le comportement de notre algorithme vis à vis de l'algorithme de recherche arborescente. Deux indicateurs illustrent la performance des deux algorithmes. Le temps d'exécution pour trouver le meilleur appariement et le nombre de solutions optimales atteintes. Comme il a été dit auparavant, la recherche arborescente est une méthode optimale. Pour cela, nous avons pris dans les deux expérimentations cette méthode comme méthode témoin sur la qualité de la solution obtenue. La deuxième expérimentation est une étude comparative de notre algorithme avec quatre autres méthodes.

2.3.1 Graphes aléatoires

Dans le but d'étudier le comportement de l'algorithme dans la pratique, nous avons réalisé une série d'expérimentations sur des graphes aléatoires. Le choix d'une base de graphes aléatoires a été motivé par le contrôle simple et efficace qu'offrent les graphes aléatoires. Nous avons généré 1000 paires de graphes. Un nombre aléatoire de sommets a été généré pour chaque graphe variant de 2 à 10. Pour chaque sommet et arête dans un graphe, nous avons généré un nombre réel aléatoire compris entre 0 et 1. Nous avons varié le

Nombre de phases	Nombre d'appariements optimal	Temps moyen en secondes
1	609	2.14
2	827	3.69
3	940	6.14
4	971	11.04
5	1000	16.28
A*	1000	186.57

TAB. 2.6: Comparaison entre A* et l'algorithme proposé.

nombre de phases K de 1 à 5 dans l'algorithme. Pour chaque paire de graphes et pour les deux algorithmes, nous avons comptabilisé le temps nécessaire pour rechercher l'isomorphisme. À la fin de chaque test, nous avons calculé le temps moyen nécessaire pour rechercher l'isomorphisme pour l'ensemble des graphes. Nous avons aussi comptabilisé le nombre de paires de graphes dont notre algorithme a réussi à trouver l'isomorphisme optimal. L'expérimentation a été réalisée sur une machine SUN, Ultra 60, dont la cadence de l'horloge des processeurs est 450 Mhz et une mémoire vive interne de 512 Mb.

Le tableau 2.6 montre la performance des deux algorithmes. Il est clair que pour l'algorithme proposé, plus le nombre de phases augmente, plus le temps moyen nécessaire pour rechercher l'isomorphisme augmente. On remarque aussi l'évolution du nombre d'appariements optimaux trouvés en fonction du nombre de phase K. Pour cette expérimentation, l'appariement optimal est atteint pour l'ensemble des paires de graphes pour $k = 5$. Cependant, le temps nécessaire pour trouver l'optimalité pour ce cas précis $K = 5$ est nettement inférieur à celui enregistré pour A*. Le tableau 2.7 montre le temps nécessaire en secondes pour trouver le meilleur appariement en utilisant le

Nombre de sommets	8×12	10×15	13×18
Temps en secondes	39	761	1345

TAB. 2.7: Temps nécessaire pour trouver le meilleur appariement.

nouvel algorithme avec $K = 5$. L'algorithme A* n'a pas été utilisé dans cette expérience étant donné qu'il est incapable de trouver une solution optimale. L'algorithme A* ne peut pas supporter plus de 10 sommets par graphes. La taille de l'espace de recherche pour ce genre de problème croît exponentiellement avec la taille des graphes à apparier. Cependant, l'algorithme A* pour l'isomorphisme exact et inexact et l'algorithme de Ullmann [110] pour l'isomorphisme exact restent parmi les meilleurs et les plus utilisés comme indicateurs d'optimalité de la solution trouvée.

2.3.2 Étude comparative

Dans cette section, nous présenterons une étude comparative afin de dégager les points forts et faibles de notre algorithme. La comparaison inclut l'algorithme proposé, l'algorithme basé sur la relaxation probabiliste [16], l'algorithme basé sur les outils génétiques [114], l'algorithme basé sur l'analyse spectrale des matrices d'adjacence [111] et l'algorithme basé sur la recherche arborescente [108]. Deux indicateurs ont été pris en considération pour l'évaluation et la comparaison, le temps nécessaire pour trouver une solution ainsi que le taux d'appariement optimal atteint.

Nous avons considéré l'algorithme basé sur la recherche arborescente comme témoin d'optimalité. Cet algorithme déterministe génère une solution optimale, cependant il n'est pas en mesure de donner une solution pour des graphes de taille 10 sommets ou plus. Cet algorithme souffre de la complexité exponentielle. L'algorithme basé sur les outils génétiques a été pro-

Paramètre	Valeur
Nombre d'itérations	200
Population	30
Selection	Ordre
Valeur d'inhibition	4
Croisement	PMX
Mutation	SWAP

TAB. 2.8: Paramètres génétiques utilisés pour l'expérimentation.

posé par Wang et al [114]. Il permet de trouver une solution approximative. L'algorithme nécessite l'initialisation de plusieurs paramètres. Le tableau 2.8 montre les valeurs des différents paramètres. Pour plus de détails concernant la réalisation voir [114]. Nous avons utilisé les mêmes conditions d'expérimentation.

La base de données de graphes contient six ensembles. Les six ensembles contiennent chacun 1000 paires de graphes ayant respectivement 4, 5, 6, 7, 8 et 9 sommets. Pour chaque sommet et chaque arête, nous avons assigné aléatoirement une étiquette à partir de l'ensemble des étiquettes $L_S = L_A = \{1, 2, ..., 9\}$. Le nouvel algorithme a été expérimenté avec $K = 3$.

Le tableau 2.9 montre que l'algorithme de recherche arborescente est bel et bien un algorithme optimal. Pour le dernier ensemble dont les graphes ont 9 sommets, cet algorithme est incapable de donner une solution. Cet algorithme explore l'espace de recherche en entier pour trouver une solution. L'étendue de l'espace de recherche croît exponentiellement en fonction de la taille des graphes à apparier. Pour ce cas, nous avons comparé les autres algorithmes entre eux et l'optimalité a été attribuée à celui dont l'erreur de correction, ecag est la plus faible. On remarque tout d'abord que la performance de l'algorithme d'analyse spectrale à trouver une solution optimale dégrade

Taille d'un graphe	Nouvel algorithme	Relaxation probab.	Algorithme génétique	Analyse spectrale	Arbre de recherche
4	1000	597	855	330	1000
5	1000	483	639	97	1000
6	976	311	478	43	1000
7	920	191	289	8	1000
8	865	125	199	3	1000
9	680	160	240	0	-

TAB. 2.9: Nombre d'appariements optimal atteint pour les différents algorithmes.

d'une manière significative au fur et mesure que la taille des graphes augmente. Cela prouve que cette solution n'est pas intéressante lorsqu'il s'agit de graphes non similaires. La relaxation ainsi que l'algorithme génétique offrent une solution acceptable pour des graphes de 4 et 5 sommets. Cependant, la performance de ces deux algorithmes diminue au fur et à mesure que la taille des graphes augmentent. Le nouvel algorithme offre une solution très proche de l'optimale et ce même pour des graphes avec 7 et 8 sommets. Les nouvelles techniques heuristiques introduites dans la deuxième version de notre algorithme ont montré leurs capacités à améliorer considérablement le taux d'optimalité atteint.

Le tableau 2.10 montre que les algorithmes basés respectivement sur l'analyse spectrale et l'approche génétique sont très efficaces en termes de temps de calcul. La recherche arborescente a un temps raisonnable pour des graphes de taille de 4, 5 et 6 sommets. Cependant, le temps nécessaire pour trouver une solution optimale avec cet algorithme augmente d'une manière exponentielle lorsque la taille des graphes augmente. Notre algorithme nécessite peu de temps pour trouver une solution pour des graphes de taille allant jusqu'à 8

Taille d'un graphe	Nouvel algorithme	Relaxation probab.	Algorithme génétique	Analyse spectrale	Arbre de recherche
4	0.002	0.011	0.001	0.001	0.001
5	0.007	0.029	0.001	0.001	0.003
6	0.074	0.065	0.002	0.001	0.034
7	1.063	0.128	0.002	0.001	8.959
8	16.884	0.228	0.003	0.001	352
9	135.974	0.368	0.004	0.001	-

TAB. 2.10: Le temps d'exécution de chaque algorithme. (en secondes sur un PC Pentium IV, 256 Mo).

sommets. Nous avons enregistré des temps très raisonnables pour des graphes ayant 14 sommets en introduisant les nouvelles techniques heuristiques.

Dans une étude précédente que nous avons effectuée pour comparer la performance de notre algorithme avec d'autres algorithmes, nous avons généré des paires de graphes avec différents degrés de similarité. La génération de paires de graphes consiste à générer le premier graphe aléatoirement, ensuite à générer le deuxième en introduisant un bruit sur le premier. Nous avons utilisé cinq niveaux de bruit. Le niveau zéro consiste simplement à recopier le premier pour générer le deuxième. Nous avons constaté un bon taux d'appariement pour les algorithmes basés sur l'analyse spectrale et l'approche génétique pour des graphes de 11 et 13 sommets. L'algorithme proposé a montré une deuxième fois sa supériorité en affichant des taux d'optimalité proches de 100% dans la majorité des tests. Pour plus de détails voir [42].

2.4 Conclusion

Nous avons proposé dans ce chapitre un nouvel algorithme d'appariement de graphes. L'algorithme permet d'extraire l'isomorphisme de graphes ou de sous-graphes. L'algorithme repose sur la notion d'opérateurs d'édition pour trouver l'isomorphisme. Trois types d'opérateurs peuvent être utilisés pour transformer un graphe en un autre. La substitution, la suppression et l'insertion. L'idée de base de notre algorithme repose sur une séparation entre la mise en correspondance des sommets et la mise en correspondance des arêtes. L'algorithme divise le processus de recherche en plusieurs phases. Dans chaque phase, le meilleur appariement est extrait. Ensuite, il est évalué et finalement comparé au meilleur de tous le processus. Nous avons présenté tous d'abord une version traitant l'opération de substitution. Une extension a été présentée généralisant l'algorithme sur toutes les autres opérations d'édition. L'algorithme développé offre de nouvelles techniques pour rediriger rapidement le processus de recherche vers la solution du problème. Les expériences présentées dans ce chapitre montre la supériorité de notre algorithme pour traiter le problème d'appariement de graphes similaires ou non similaires.

Chapitre 3

Un nouvel algorithme de calcul du graphe médian

3.1 Introduction

Le concept de graphe médian introduit par Jiang *et al.* [56] est un concept fort important en reconnaissance de formes structurelles. Le graphe médian permet de représenter l'information essentielle et utile d'un ensemble de graphes par un seul prototype ou représentant. Ce concept permet aussi d'établir un lien solide entre la reconnaissance de formes structurelles et la reconnaissance de formes statistiques. Par exemple, il permet une réutilisation des techniques et méthodes de clustering utilisées par l'approche statistique dans la reconnaisance de formes structurelles. En plus, des méthodes hybrides peuvent être développées entre les deux approches en utilisant ce concept adapté aux graphes. Le médian est un concept classique pour les vecteurs ou les chaînes. Cependant, les spécificités structurelles des graphes rendent ce concept difficile à définir et à utiliser. Peu d'algorithmes sont développés pour construire ou rechercher le graphe médian d'un ensemble de

graphes. L'algorithme de Jiang [56] fut le premier et le seul permettant de calculer le graphe médian d'un ensemble de graphes. Nous avons jugé qu'un effort peut être déployé dans cette direction.

La recherche d'un graphe médian est un processus de minimisation qui consiste à construire un graphe dont la somme des distances avec tous les graphes appartenant à un ensemble S est minimale. Ce processus nécessite de chercher un sous ensemble d'étiquettes pour les sommets et les arêtes appartenant respectivement à E_s et E_a et d'effectuer des mises en correspondance entre les sommets et les arêtes du graphe médian à construire et ceux de chacun des graphes de S. Au terme de ce processus, chaque appariement entre le graphe médian et un graphe de S est un isomorphisme de graphes ou de sous-graphes selon la taille des deux graphes. On suppose dans ce chapitre que la taille du graphe médian est un paramètre dont la valeur est préalablement donnée ou calculée. On désigne par M le nombre de sommets dans le graphe médian. Le graphe médian recherché minimise la somme des distances SOD, soit

$$\overline{g} = arg \min_{g \in C} SOD(g, S) \tag{3.1}$$

où

$$SOD(g, S) = \sum_{i=1}^{|S|} d(g, g_i) \tag{3.2}$$

Dans la littérature, la distance entre deux graphes peut être calculée de deux manières, soit en calculant l'erreur de correction d'appariement de graphes (*ecag*), soit en recherchant le plus grand sous-graphe commun. Nous avons présenté dans le chapitre 2, un algorithme permettant de calculer cette distance en utilisant les opérations d'édition. Dans ce chapitre, nous utiliserons l'ecag pour calculer la distance entre deux graphes.

Jiang *et al.* [56] ont démontré que le nombre de sommets dans le graphe médian ne peut pas dépasser le nombre total des sommets des graphes de S,

soit $N_S = \sum_{i=1}^{|S|} |S_i|$. Une première solution consiste à vérifier tous les graphes possibles qui peuvent être générés séquentiellement de taille M. Le processus de vérification consiste d'abord à calculer pour chaque graphe généré la somme des distances avec tous les graphes de S, puis à retenir le graphe qui a la plus faible valeur de SOD. Malheureusement, cet algorithme n'est pas applicable même pour un ensemble S très petit et des graphes de faible taille.

Afin d'éviter l'explosion combinatoire du problème, nous avons adopté une stratégie qui consiste à décomposer l'algorithme de recherche du graphe médian en plusieurs sous-problèmes. L'idée est que chaque sous-problème permet de résoudre une difficulté particulière et que son traitement est indépendant des autres processus. Afin de définir les différents processus et leurs rôles, nous avons dégagé les différentes difficultés à surmonter pour trouver une solution au problème. La complexité élevée du problème est causée, premièrement par le nombre de graphes dans l'ensemble S et deuxièmement par la taille des graphes. Ces deux facteurs influent directement sur le calcul de SOD et la taille du graphe médian. Le temps de calcul de SOD augmente exponentiellement lorsque la taille des graphes augmente et linéairement lorsque le nombre de graphes augmente. La taille du graphe médian augmente lorsque la taille des graphes de S augmente. La décomposition adoptée comporte trois processus : un premier permettant de réduire la taille de l'ensemble d'étiquettes des sommets E_s, un deuxième processus permettant d'estimer la taille du graphe médian et un troisième permettant de calculer la distance SOD entre un graphe médian de taille fixe M et l'ensemble des graphes de S. Dans cette même étape, la construction du graphe médian est réalisée. Ce troisième processus constitue le noyau de l'algorithme. L'algorithme général pour construire le graphe médian généralisé peut être décrit de la manière suivante :

Algorithme : Graphe_Médian_Généralisé()
Entrées : L'ensemble des graphes S.
Sortie : Le graphe médian généralisé.

Début

1 : Réduction_Ensemble_Étiquettes()
2 : Cardinalité_Graphe_Médian()
3 : Minimisation_SOD()

Fin

Dans ce qui suit, nous allons proposer en détail les trois processus cités ci-dessus (Section 3.2). La section 3.3 décrit les résultats expérimentaux. Enfin, une conclusion sera donnée dans la section 3.4.

3.2 Algorithme du graphe médian

3.2.1 Réduction de l'ensemble des étiquettes

Les ensembles des étiquettes respectivement pour les sommets E_s et les arêtes E_a constituent le point de départ pour la construction du graphe médian. La taille de ces deux ensembles est un facteur déterminant dans la complexité du problème. Plus l'ensemble est grand plus il y a des sous-ensembles à extraire et à évaluer pour le graphe médian. Notre but est de réduire la taille de ces deux ensembles sans altérer la qualité de la solution à atteindre. Dans ce qui suit, nous allons expliquer la méthode adoptée pour réduire la taille de l'ensemble des sommets E_s. Pour l'autre ensemble E_a, la même démarche est utilisée. Notre démarche comporte deux parties, une première déterministe et une seconde heuristique.

L'ensemble des étiquettes E_s contient toutes les étiquettes des différents

graphes de S. La taille de E_s n'est autre que le nombre de sommets de tous les graphes de S. Un certain nombre de ces étiquettes appartenant à E_s peuvent être dupliquées plusieurs fois ($Dfois$). Or si la taille du graphe médian recherché est égale à M, seulement, dans le pire des cas, M étiquettes ayant la même valeur peuvent être utilisées pour construire le graphe médian. Dans le cas où $D > M$, les $D - M$ étiquettes ayant la même valeur n'auront aucun impact sur la solution finale. Ainsi, dans un premier temps, nous examinerons l'ensemble E_s et pour les étiquettes qui sont dupliquées plus que M fois, $D - M$ étiquettes seront éliminées de l'ensemble E_s. Cette méthode est une démarche simple et facile à mettre en oeuvre. Cependant, son utilisation est conditionnée par la nature des étiquettes. Si les étiquettes ne sont pas suffisamment dupliquées, aucune réduction n'est réalisée avec cette méthode.

Nous avons envisagé une deuxième technique de réduction. Elle consiste à regrouper les étiquettes en plusieurs groupes, puis à utiliser uniquement les centres de chaque groupe pour le reste du processus. Il existe une multitude d'algorithmes différents pour classifier un ensemble de données. Nous avons utilisé l'algorithme *k-means* [78] pour réduire la taille de l'ensemble des étiquettes E_s. Notre choix de k-means a été motivé par l'universalité, l'efficacité et la puissance de cet algorithme qui permet une bonne répartition des données. Le nombre de classes a été fixé à $2M$ où M est le nombre de sommets du graphe médian. Ce choix revêt évidemment un caractère expérimental. Il s'agit d'un compromis entre la réduction de la complexité du problème de la recherche et le maintien de la qualité du médian calculé. Le problème dépend beaucoup de l'application. Il faut encore des études approfondies pour trouver des règles générales. Une fois cette réduction est réalisée, le nouvel ensemble des sommets E_s sera représenté par L_s.

3.2.2 Cardinalité du graphe médian

Partant du théorème de Bunke [56] qui stipule que la cardinalité du graphe médian est comprise entre 1 et le nombre total des sommets des graphes de S, soit $N_S = \sum_{i=1}^{|S|} |S_i|$, un choix judicieux de la cardinalité s'impose afin de réduire le temps de calcul de SOD. L'approche adoptée dans cette thèse pour le calcul de SOD est l'utilisation des opérations d'édition, soit la substitution, l'insertion et la suppression respectivement des sommets et des arêtes. Etant donnés deux graphes $G = (S, A, \alpha, \beta)$ et $G' = (S', A', \alpha', \beta')$, la distance entre deux sommets (resp. arêtes) mis en correspondance est la distance Euclidienne entre les deux étiquettes. Le calcul de la distance entre deux graphes peut être interprété comme le coût de la transformation du graphe G en G' en utilisant les opérations d'édition. Le coût de la transformation n'est autre que la somme de toutes les transformations élémentaires appliquées sur les sommets (resp. les arêtes) du premier graphe. Le coût de chaque opération d'édition est décrit comme suit :

- Substitution de sommets : $f_c(\delta_{ss}) = f_c(u_1, v_2) = d(\alpha(u_1), \alpha'(v_2))$
- Substitution d'arêtes : $f_c(\delta_{sa}) = f_c((u_1, v_1), (u_2, v_2)) = d(\beta(u_1, v_1), \beta'(u_2, v_2))$
- Insertion de sommets : $f_c(\delta_{is}) = f_c(u) = \alpha(u)$
- Insertion d'arêtes : $f_c(\delta_{ia}) = f_c(u, v) = \beta(u, v)$
- Suppression de sommets : $f_c(\delta_{ps}) = f_c(u) = \alpha(u)$
- Suppression d'arêtes : $f_c(\delta_{pa}) = f_c(u, v) = \beta(u, v)$

Le coût d'une substitution d'un sommet u de G mis en correspondance avec un sommet u' de G' est la distance Euclidienne entre les deux étiquettes respectivement de u et de u'. L'opération d'insertion est appliquée dans le cas où la taille du premier graphe G est inférieure à celle de G'. Pour chaque sommet de G' qui n'existe pas dans G, un sommet est inséré dans G ayant la même étiquette et le même nombre d'arêtes que le sommet dans le deuxième

graphe. Le coût de chaque insertion inclut la valeur de l'étiquette ainsi que la valeur de toutes les étiquettes des différentes arêtes ajoutées. L'opération de suppression est appliquée dans le cas ou la taille du premier graphe G est supérieure à celle de G'. Pour chaque sommet de G qui n'existe pas dans G', un sommet est supprimé de G ainsi que toutes les arêtes qui lui sont rattachées. Le coût de chaque suppression inclut la valeur absolue de l'étiquette du sommet supprimé ainsi la valeur absolue de toutes étiquettes des différentes arêtes supprimées.

Notre but est de minimiser le coût de la transformation de G en G'. D'après l'analyse des coûts des différentes opérations d'édition proposée ci-dessus, la substitution offre un coût relativement faible par rapport à l'insertion et la suppression, ce qui nous amène à dire que la meilleure cardinalité minimisant SOD est obtenue si le nombre d'insertions et de suppressions dans la séquence de transformation est très bas. Donc, deux graphes ayant le même nombre de sommets et d'arêtes n'ont besoin ni d'insertion ni de suppression. Or, les graphes dans S n'ont pas forcément la même taille. Dans ces conditions et pour utiliser le moins possible d'insertions et de suppressions, la cardinalité du graphe médian doit être aussi proche que possible que les cardinalités des différents graphes dans S. Nous avons opté pour une cardinalité égale à la moyenne de toutes les cardinalités des graphes de S. Notre hypothèse est partiellement validée expérimentalement dans la section 3.3.1. N_S est le nombre de sommets dans tous les graphes de l'ensemble S. Le choix de M peut être aussi donner par l'utilisateur ou par une autre technique capable de mieux approximer la cardinalité du graphe médian. Un travail théorique futur peut approfondir ce choix purement expérimental.

$$M = \frac{1}{|S|} N_S \tag{3.3}$$

3.2.3 Minimisation de SOD

Le problème de minimisation de $SOD(g, S)$ pour obtenir le graphe médian $\overline{g}$ est très complexe. Les difficultés sont causées par les facteurs suivants :

1. Un grand espace de recherche pour les sommets du $\overline{g}$ et un autre grand espace pour les arêtes du $\overline{g}$ (en pratique ce dernier peut-être beaucoup plus grand que le premier).
2. Le graphe $\overline{g}$ doit être apparié à chacun des graphes de S pour permettre le calcul de la distance $d(\overline{g}, g_i)$. En soi, c'est un processus complexe dont nous avons discuté précédemment.
3. Il n'y a pas d'indice pour permettre d'évaluer explicitement si une étiquette est meilleure qu'une autre pour construire $\overline{g}$ puisque la "qualité" d'une étiquette n'est pas forcément déterminée par la présence de cette étiquette dans chaque graphe g_i de S.

En d'autres termes, le calcul du graphe médian implique la recherche des étiquettes pour les sommets et les arêtes, et l'intégration de ces étiquettes pour former le graphe médian. La principale difficulté résulte du fait que les deux parties (la recherche et l'intégration) sont intiment liées.

Les observations suivantes nous permettent de voir le problème d'une autre perspective et nous présenterons des voies pour simplifier le processus de recherche et d'intégration. La distance, entre le graphe médian $\overline{g}$ et le graphe g_i, $d(\overline{g}, g_i)$ peut être décomposée comme suit :

$$d(\overline{g}, g_i) = d_s(\overline{g}, g_i) + d_a(\overline{g}, g_i) \tag{3.4}$$

où $d_s(\overline{g}, g_i)$ est la distance entre $\overline{g}$ et g_i relative aux sommets, tandis que $d_a(\overline{g}, g_i)$ est la distance entre $\overline{g}$ et g_i relative aux arêtes. La somme des distances $SOD(\overline{g}, S)$ devient alors

$$SOD(\overline{g}, S) = \sum_{i=1}^{|S|} d_s(\overline{g}, g_i) + \sum_{i=1}^{|S|} d_a(\overline{g}, g_i) \tag{3.5}$$

$$SOD(\overline{g}, S) = SOD_s(\overline{g}, S) + SOD_a(\overline{g}, S) \tag{3.6}$$

Ici, $SOD_s(\overline{g}, S) = \sum_{i=1}^{|S|} d_s(\overline{g}, g_i)$ représente la partie de SOD engendrée par les distances entre les sommets du $\overline{g}$ et ceux de chaque graphe g_i de S. Par ailleurs, $SOD_a(\overline{g}, S) = \sum_{i=1}^{|S|} d_a(\overline{g}, g_i)$ représente la partie de SOD engendrée par les distances entre les arêtes du $\overline{g}$ et ceux de chaque graphe g_i de S. La séparation de SOD en SOD_s et SOD_a nous permet de constater que la minimisation de SOD peut se faire en deux phases. La première phase consiste à chercher un sous ensemble de sommets dont chaque élément doit être mis en correspondance avec un sommet de $g_i (i = 1, .., |S|)$. Cet appariement entre les sommets du (potentiel) $\overline{g}$ et g_i impose également des mises correspondances entre les arêtes de $\overline{g}$ et les différentes arêtes de g_i de S. Par conséquent, la deuxième phase de la minimisation consiste seulement à trouver les étiquettes appropriées pour les arêtes de $\overline{g}$ afin que SOD_a soit minimisée.

La démarche adoptée pour minimiser SOD en 2 phases permet de formaliser le problème en un problème de recherche dans un espace d'états dont chaque élément est un sous-ensemble de M étiquettes provenant de l'ensemble des étiquettes L_s. L'ensemble d'états est représenté par L_s^M. Alors que la recherche des étiquettes pour les arêtes joue un rôle secondaire et la partie de l'intégration se fait naturellement. Cependant, il faut remarquer que pour trouver la solution optimale, l'exploration de l'espace d'état L_s^M tout entier demeure nécessaire. Une solution approximative s'impose pour éviter l'explosion combinatoire. Nous avons adopté une approche similaire à celle utilisée dans l'algorithme d'appariement de graphes présenté dans le chapitre 2. L'approche consiste à minimiser d'abord SOD_s, et ensuite minimiser SOD_a. Bien que cette approche ne garantisse pas que le $\overline{g}$ obtenu soit le minima de SOD, elle a l'avantage de réduire de façon significative la complexité de l'algorithme. Comme pour l'algorithme d'appariement de graphes, cette approche repose aussi sur une hypothèse similaire (implicite) que le graphe médian doit, avant tout, être en moyenne similaire à tous les

graphes de S en ce qui concerne les sommets.

Pour minimiser SOD_s, commençons à examiner plus en détail l'expression $SOD_s(\overline{g}, S) = \sum_{i=1}^{|S|} d_s(\overline{g}, g_i)$ où $d_s(\overline{g}, g_i)$ est calculée comme suit

$$d_s(\overline{g}, g_i) = \sum_{k=1}^{M} d_s(\overline{s}_k, s_{f_i(s_k)}) + C_s(\overline{g}, g_i) \tag{3.7}$$

Le calcul de la distance entre deux graphes est effectué par application des opérations d'édition. Dans le cas où les deux graphes ont le même nombre de sommets, seule la substitution est utilisée. Dans le cas où le premier graphe est plus grand que le deuxième en terme de nombre de sommets, la substitution et l'insertion seront utilisées. Finalement, dans le cas où le premier graphe est plus petit que le deuxième, la substitution et la suppression seront utilisées de telle sorte que le processus de transformation opère toujours sur le deuxième graphe. C'est-à-dire que lors d'une substitution, l'opération d'édition transforme le sommet du deuxième graphe en changeant la valeur de son étiquette. Pour la suppression, l'opération d'édition transforme le deuxième graphe en supprimant le sommet en question et toutes les arêtes partant de ce sommet. Pour l'insertion, l'opération d'édition transforme le deuxième graphe en insérant le sommet manquant par rapport au premier graphe ainsi que toutes les arêtes manquantes partant du sommet dans le premier graphe. Dans notre cas précis, seul le graphe g_i est transformé. La première partie de l'équation ci-dessus, le terme $d_s(\overline{s}_k, s_{f_i(s_k)})$ calcule la similarité entre un sommet $\overline{s}_k$ de $\overline{g}$ et le sommet mis en correspondance dans le graphe g_i. Ce dernier est noté $s_{f_i(s_k)}$ avec $f_i(.)$ la fonction qui retourne la correspondance d'un sommet s_k dans un graphe g_i. Le deuxième terme $C_s(\overline{g}, g_i)$ donne le coût associé à la suppression (resp. insertion) d'un ou plusieurs sommets dans g_i. Le nombre de sommets supprimés (resp. insérés) est la différence entre le nombre de sommets dans les deux graphes.

$$SOD_s(\overline{g}, S) = \sum_{i=1}^{|S|} [\sum_{k=1}^{M} d_s(\overline{s}_k, s_{f_i(s_k)}) + C_s(\overline{g}, g_i)] \tag{3.8}$$

$$SOD_s(\overline{g}, S) = \sum_{k=1}^{M} \sum_{i=1}^{|S|} d_s(\overline{s}_k, s_{f_i(s_k)}) + \sum_{i=1}^{|S|} C_s(\overline{g}, g_i) \tag{3.9}$$

$$SOD_s(\overline{g}, S) = \sum_{k=1}^{M} \sum_{i=1}^{|S|} d_s(\overline{s}_k, s_{f_i(s_k)}) + C_s(\overline{g}, S) \tag{3.10}$$

En réalité, le terme $C_s(\overline{g}, S)$ est une somme des différents coûts associés à la suppression des sommets du graphe g_i qui n'ont pas été mis en correspondance avec un sommet de $\overline{g}$. Chaque sommet supprimé d'un g_i contribue par sa propre étiquette dans le coût total C_s. Dans le cas où des insertions ont été effectuées lors de la transformation, chaque sommet non apparié de $\overline{g}$ contribue par sa propre étiquette au coût total C_s. Ainsi, le terme $C_s(\overline{g}, S)$ peut s'écrire de la manière suivante :

$$C_s(\overline{g}, g_i) = \sum_{k=1}^{|V_i|-M} d_s(\phi, s_k) + \sum_{k=1}^{M-|V_i|} d_s(\overline{s}_k, \phi) \tag{3.11}$$

Avec ϕ un sommet imaginaire inséré dans le graphe dont la taille est la plus faible afin de calculer le coût associé aux différentes opérations de suppression ou d'insertion.

Les discussions ci-dessous nous permettent de voir le processus de minimisation de SOD_s comme la répartition de l'ensemble des sommets des différents graphes g_i en plusieurs groupes. La répartition est décrite comme suit :

- Les sommets appariés à un sommet $\overline{s}_k$ du graphe médian $\overline{g}$ formeront un groupe noté R_k avec $k = 1, .., M$. Chaque élément d'un même groupe R_k provient d'un et un seul graphe g_i de S.
- L'ensemble des sommets qui n'ont pas été appariés avec aucun sommet $\overline{s}_k$ formeront un groupe noté R_0.

Ainsi, on aurait $M+1$ groupes de sommets. Pour le groupe R_0, la contribution à la somme des distances SOD_s n'est autre que le deuxième terme de l'équation (3.10), tandis que la contribution des autres groupes est simplement la somme des distances par rapport au sommet $\overline{s}_k$, c'est-à-dire le premier terme de la même équation. Ces remarques sont très importantes

car elles suggèrent le processus suivant pour extraire des candidats pour $\overline{g}$ afin de minimiser SOD_s :

1. Extraire M groupes d'étiquettes (sommets) à partir de l'ensemble de graphes S de telle sorte que 1) chaque groupe contienne au plus un sommet provenant d'un graphe quelconque g_i ; 2) chaque groupe contienne au plus M éléments ; Si le graphe g_i est plus petit que $\overline{g}$ alors un ou plusieurs sommets de ce dernier ne seront pas mis en correspondance avec un sommet de g_i ; 3) les éléments de chaque groupe doivent être les plus similaires possibles ; 4) les étiquettes qui n'appartiennent à aucun groupe $(R_1, ..., R_M)$ doivent avoir les étiquettes les plus petites possibles.
2. Pour chaque groupe, sélectionner une étiquette représentative (on a intentionnellement choisi de ne pas préciser l'ensemble pour ce choix) et déterminer la contribution de cette étiquette à la somme des distances SOD_s. Si un groupe contient moins de $|S|$ éléments, alors déterminer la contribution des sommets qui devront être insérés dans chaque graphe g_i où $|V_i| < M$.
3. Calculer la contribution de chaque sommet qui doit être supprimé en regardant le premier groupe R_0. Pour chaque paire de sommets de $\overline{g}$, définir un ensemble d'arêtes. Chaque arête provient d'un et un seul graphe g_i. Prendre une décision quant à l'insertion ou non d'une arête entre deux sommets de $\overline{g}$. Associer une étiquette à partir de L_a à chaque arête insérée en minimisant la somme des distances SOD_a.
4. Trouver des techniques afin que les tâches du point 1 et point 2 puissent être effectuées afin de générer des groupes R_k qui ont le plus de chance de résulter en un $\overline{g}$ qui minimise SOD.

Nous avons conçu un algorithme de recherche médian en cherchant à partitionner les sommets des graphes de l'ensemble S selon les directives /

contraintes énumérées ci-dessus. L'assise de notre algorithme est la génération des candidats provisoires les plus probables qui pourraient devenir des candidats du graphe médian. Ici, un candidat provisoire (CP) est un sous-ensemble de M étiquettes (de L_s) qui a été apparié à au moins un graphe de S. Le terme "provisoire" est utilisé ici dans le sens que le candidat n'est pas encore apparié à tous les graphes de S. Un candidat provisoire est généré à partir de L_s toujours selon la meilleure similarité (distance possible) qu'il permet par rapport à un graphe de S (voir paragraphe suivant pour plus de détails). Notre algorithme est conçu de telle sorte qu'un candidat provisoire ne peut être généré qu'une seule fois par rapport à un graphe de S. Par conséquent, si le même candidat provisoire est généré $|S|$ fois, le candidat aurait été apparié à tous les graphes de S (en ce qui concerne les sommets pour être plus précis). C'est à ce moment que le candidat provisoire devient un candidat médian et la minimisation de SOD_s peut être effectuée pour permettre d'évaluer la qualité du candidat, c'est-à-dire la distance SOD.

La génération d'un candidat provisoire selon la similarité est effectuée en suivant une approche inspirant de notre algorithme d'appariement de graphes (chapitre 2). L'idée repose sur l'exploitation d'une (grande) matrice de similarité P de taille $|L_s| \times |L_s|$, où L_s représente l'ensemble de tous les sommets - étiquettes des graphes de S. La matrice P peut être vu aussi comme une matrice de blocs

$$P = (P^{(1)}, P^{(2)}, ..., P^{(|S|)}) \tag{3.12}$$

où chaque bloc $P^{(i)}$ correspond à une matrice de similarité composée de L_s et les sommets du graphe g_i(cette matrice est de taille $|L_s| \times |g_i|$). L'élément p^i_{lc} de $P^{(i)}$ est la similarité entre l'étiquette l de L_s et l'étiquette c de g_i.

$$p^i_{lc} = d(e^{L_s}_k, e^{g_i}_l) \, avec \, l = 1, 2, ..., |L_s| \, et \, c = 1, 2, ..., |g_i| \tag{3.13}$$

La matrice P est initialisée au début de l'algorithme. Le fil conducteur au plus bas niveau est l'examen successif des plus petits éléments de la matrice P. Nous avons besoin d'une autre matrice binaire B qui enregistre les résultats de ce parcours et qui fournit des entrées à l'extraction des candidats provisoires. La matrice B a exactement la même taille que P et est organisée de la même manière, c'est-à-dire en $|S|$ blocs

$$B = (B^{(1)}, B^{(2)}, , B^{(|S|)}) \tag{3.14}$$

La matrice B est initialisée à zéro partout. Un élément de p_{lc}^i est "marqué" si p_{lc}^i est le plus petit élément de P qui n'est pas encore été "marqué". Pour marquer un élément de P, l'élément correspondant de B est affecté à 1. L'extraction d'un candidat provisoire peut commencer dès que un des bloc de B (ex. $B^{(i)}$) contient suffisamment de 1.

Avant d'entrer dans les détails, il est utile de constater que si certains éléments d'un $B^{(i)}$ deviennent 1 au fur et à mesure que l'algorithme avance, c'est parce que ces éléments correspondent aux meilleurs mises en correspondance entre les étiquettes de L_s et les étiquettes de g_i. Par conséquent, les premiers candidats provisoires extraits de $B^{(i)}$ sont les meilleurs candidats provisoires par rapport à g_i. Les conditions suivantes sont nécessaires pour permettre une extraction de candidats provisoires :

1. Si $|g_i| \geq M$, il faut que $B^{(i)}$ ait au moins M éléments à 1 qui se trouvent sur différentes lignes et différents colonnes (Pour assurer que le candidat provisoire forme un isomorphisme).
2. Si $|g_i| < M$, il faut que $B^{(i)}$ ait au moins $|g_i|$ éléments à 1 qui se trouvent sur différents lignes et différents colonnes.

Sous la condition 1, on peut extraire des candidats provisoires (au moins un), un sous ensemble de M éléments de L_s appariés respectivement aux différents sommets de g_i. Tous les sommets non appariés de g_i sont considérés

appariés à ϕ. Sous la condition 2, on peut extraire des candidats provisoires (au moins un), un sous ensemble de $|g_i|$ éléments de L_s, appariés respectivement aux différents sommets de g_i. Par convention, ce candidat provisoire ($|g_i| < M$) peut être confondu à n'importe quel candidat provisoire qui l'inclut dans le processus de génération d'un candidat de graphe médian.

Il est facile de comprendre qu'une des deux conditions ci-dessus sera satisfaite pour un i quelconque au fur et à mesure que les éléments de P seront marqués. Il est facile aussi de comprendre qu'une fois une des deux conditions est satisfaite pour une matrice $B^{(i)}$, elle reste satisfaite. Donc, un mécanisme est nécessaire pour éviter l'extraction répétée d'un même candidat provisoire par rapport à g_i. Ceci est détaillé dans l'algorithme. On peut remarquer maintenant que la $1^{ère}$ fois qu'un même candidat provisoire est généré deux fois par rapport à g_{i1} et g_{i2}, ce candidat provisoire serait un bon candidat de choix pour le médian de $\{g_{i1}, g_{i2}\}$. En généralisant ce raisonnement, la $1^{ère}$ fois qu'un même candidat est généré $|S|$ fois (pour chacun des graphes de S), il devient un bon candidat pour le médian de S. En fait, ce candidat réalise une bonne réparation des sommets des différents graphes de S.

Algorithme : Minimisation_SOD()

Entrées :

1 : L'ensemble des graphes S.

2 : La taille du graphe médian M.

Sorties :

1 : Un graphe médian généralisé.

2 : L'ensemble des mises en correspondance.

3 : La somme des distances : SOD.

Début

1 : Initialisation de P et B :

Pour $i = 1, ..., |L_s|$ faire

Pour $j = 1, ..., |E_s|$ faire

$p_{ij} = d(L_s(i), E_s(j))$.

$b_{ij} = 0$.

FinPour

FinPour

2 : $f = 1$.

3 : Répéter

$(l, c) = arg \min_{1 \leq i \leq |L_s|, 1 \leq j \leq |E_s| \, et \, b_{l,c}=0} p_{ij}$.

$b_{lc} = 1$.

$j = Graphe\_Associe(c)$.

$ncp = Calcul\_Nombre\_Candidats\_provisoires(B, j)$.

Pour $k = 1, ..., npc$ faire

Extrait un candidat provisoire CP.

$d_s(CP, g_j) = \sum_{i=1}^{M} p_{CP^l(i), CP^c(i)} + C_s(CP, g_j)$.

Si $(CP \notin F)$

Alors Insérer CP dans F.

Former un nouvel ensemble $\Re_f(R_0, ..., R_M)$.

$f++$.

Sinon Si $g_j \# \Re_f.Liste - Graphes$

Alors Mettre à jour l'ensemble $\Re(R_0, ..., R_M)$.

Fin Si

Fin Si

Si $(|R_1| == |S|)$

Alors Pour $i = 1, ..., |S|$

Calculer $d_a(CP, g_i)$.

Fin Pour.

$SOD_s(CP, S) = \sum_{i=1}^{|S|} d_s(CP, g_i)$.

$SOD_a(CP, S) = \sum_{i=1}^{|S|} d_a(CP, g_i)$.
$SOD(CP, S) = SOD_s(CP, S) + SOD_a(CP, S)$.
Mettre à jour le meilleur médian.
Fin Si
Fin Pour
Valider la condition d'arrêt.
Jusqu'à $Condition = VRAI$

Fin

La première étape de l'algorithme consiste à initialiser les deux matrices P et B. Le nombre de lignes dans les deux matrices correspond au nombre d'éléments dans l'ensemble d'étiquettes L_s. Le nombre de colonnes correspond dans les deux matrices au nombre d'éléments dans l'ensemble E_s. Dans l'implémentation, le nombre de colonnes est plus grand que $|E_s|$. En fait, dans chaque graphe appartenant à S dont le nombre de sommets est inférieur au nombre de sommets du graphe médian recherché M, on ajoute un nombre de colonnes égal à la différence entre les deux tailles. La deuxième étape consiste simplement à initialiser la variable f qui permet de comptabiliser le nombre total de candidat provisoire extrait.

La troisième étape, noyau de l'algorithme, consiste à extraire des candidats provisoires, évaluer leurs distances et retenir le graphe médian minimisant la distance SOD. Le processus commence par sélectionner un élément de P. Un élément est une mise en correspondance entre deux sommets. Cet élément n'a jamais été sélectionné auparavant ($b_{ij} = 0$). Pour ne pas le sélectionner dans des étapes subséquentes, la valeur de b_{lc}) doit passer à 1. La procédure *Graphe_Associé* détermine l'indice j du graphe associé à cet élément. La valeur de j varie entre 1 et $|S|$. La procédure *Calcul_Nombre_Candidats_provisoires* détermine à partir de la matrice B le

nombre de candidats provisoires pouvant être extrait. Il faut rappeler qu'un candidat provisoire est un sous-ensemble de M éléments de L_s.

Maintenant, pour chaque candidat provisoire, le processus sauvegarde les différentes mises en correspondance du candidat dans une structure de données temporaires CP. Cette structure est un tableau de M éléments. Chaque élément est composé de deux champs pour contenir les indices l et c. Le calcul de la distance des sommets entre ce candidat provisoire et le graphe associé de S d'indice j est réalisé en regardant la matrice P. Le terme $C_s(CP, g_j)$ est calculé et additionné à la distance $d_s(CP, g_j)$. Ce terme est égal à zéro si $M = |g_j|$. Nous avons introduit une deuxième structure F. L'ensemble F permet de renseigner le processus sur l'extraction ou non d'un candidat dans les itérations précédentes. Si le candidat provisoire CP_f est généré pour la première fois alors il inséré dans F. Un nouvel ensemble $\Re_f$ est cré pour contenir les informations relatives à ce candidat provisoire :

1. Un élément est inséré à la liste $\Re_f.Liste\_Graphes$ pour indiquer l'indice du graphe associé g_j.
2. Les différentes mises en correspondances. Chaque mise en correspondance est inésrée dans un groupe R_k.
3. La distance partielle $d_s(CP_f, g_j)$.

Dans le cas, où le candidat provisoire appartient à F, l'algorithme met à jour l'ensemble $\Re$. La mise à jour consiste à insérer un élément dans la liste $\Re_f.Liste\_Graphes$ et une mise en correspondance pour chaque groupe R_k et la distance $d_s(CP, g_j)$.

Lorsque un groupe R_k d'un candidat provisoire contient $|S|$ éléments, son candidat provisoire est apparié à tous les graphes de S. On peut alors calculer la distance engendrée par la mise en correspondance des arêtes $d_a(CP, g_j)$ avec $1 \leq j \leq |S|$ et calculer la somme des distance $SOD(CP_f, S)$. Le candidat provisoire est déclaré graphe médian potentiel. La condition d'arrêt

de l'algorithme peut être validé simplement par la sélection de tous les éléments de P. Cette condition d'arrêt n'est pas suffisante pour éviter l'explosion combinatoire du problème. Nous avons développé d'autres techniques pour prédire d'avance le moment opportun pour arêter le processus de recherche. Par exemple, lorsqu'un premier graphe médian est détecté, l'arrêt du processus est une solution approximative acceptable qui peut donner de bonnes solutions au problème dans la plupart du temps.

La complexité de la recherche du graphe médian dépend du nombre de graphes de S, de la taille des graphes $|V_i|$ et de la taille du graphe médian M. À partir de l'ensemble d'étiquettes L_s, le processus de recherche consiste à extraire M étiquettes. Ensuite chaque sous-ensemble doit être apparié individuellement avec tous les graphes g_i de S. Le nombre de sous-ensembles est égal à $C_{L_s}^{M}$. Le nombre d'itérations nécessaires pour apparier le graphe médian avec un graphe g_i de taille $|V_i|$ est $M^{|V_i|}$. Donc, le nombre d'itérations totale pour les différents graphes de S est $C_{L_s}^{M} \times \sum_{i=1}^{|S|} M^{|V_i|}$. La taille de l'ensemble d'étiquettes $|L_s|$ est égal à $2 \times M$. Le nombre de sous-ensembles devient alors $\frac{1}{(2M!)}$ et le nombre d'itérations totales égal à $\frac{1}{(2M!)} \times \sum_{i=1}^{|S|} M^{|V_i|}$.

3.3 Résultats expérimentaux

Nous avons expérimenté l'algorithme de recherche du graphe médian présenté dans ce chapitre sur une base de graphes aléatoires. Afin de démontrer l'efficacité de l'algorithme, nous avons conduit trois expérimentations. La première concerne l'hypothèse adoptée sur la cardinalité du graphe médian choisi. La deuxième concerne la performance de l'algorithme en termes de temps de recherche. Finalement, la troisième est une étude comparative avec l'algorithme de recherche du graphe médian proposé par Jiang [56] [58] [59].

3.3.1 Cardinalité du graphe médian

Dans le but de justifier expérimentalement le choix adopté dans ce travail concernant la cardinalité du graphe médian, nous avons généré 10 ensembles de graphes contenant respectivement $5, 10, 15, 20, 25, 30, 35, 40, 45$ et 50 graphes. Chaque graphe contient un nombre de sommets générés aléatoirement. Ce nombre varie entre 3 et 8. Pour chaque sommet et arête, nous avons choisi une étiquette aléatoirement à partir de $\{0.1, 0.2, ..., 0.9\}$. Pour chaque ensemble de graphe, nous avons varié la cardinalité du graphe médian de 3 à 8 avec un pas de 1. Pour chaque ensemble de graphes et pour chaque cardinalité particulière, nous comptabilisons la valeur cumulative de SOD pour l'ensemble des graphes. À la fin des différents tests, nous calculons la moyenne de SOD. Le tableau 3.1 montre la valeur moyenne de chaque SOD trouvée. À partir de ce tableau, on remarque qu'il y a une forte corrélation entre la valeur moyenne du nombre de sommets et le minima du SOD. En fait, dans la majorité des cas, la valeur minimale de SOD est réalisée pour une cardinalité égale à la moyenne du nombre de sommets dans chaque ensemble. Dans le tableau 3.1, la première colonne décrit la cardinalité du graphe médian utilisée. Nous avons affiché entre parenthèses en deuxième ligne, la moyenne du nombre de sommets pour chaque ensemble, précédé par le nombre de graphes dans chaque ensemble. La valeur 7.4 représente la valeur de la somme des distances SOD obtenue avec un graphe médian de taille 3 et un ensemble de graphes contenant 5 graphes. La moyenne des sommets de tous les graphes dans cette ensemble est 4.8. Dans la même colonne le meilleur résultat (6.2) a été enregistré par un graphe médian de taille 5.

	Nombre de graphes dans chaque ensemble (Nombre moyen de sommets par graphe)									
Taille	5 (4.8)	10 (4)	15 (7)	20 (6)	25 (6)	30 (6)	35 (4.2)	40 (4.3)	45 (4.1)	50 (5)
3	7.4	12.5	28.4	32.1	39	51.5	36.3	**48.7**	49.7	58.2
4	6.4	**12**	25.3	26.3	36.2	51.2	**33.1**	48.8	**49.1**	**55.8**
5	**6.2**	12.4	**19.8**	25.2	34.1	47.3	38.2	56.7	55.3	63.1
6	7.2	15	21.2	**24.1**	**31.2**	**42.1**	42.3	77	64	77.7
7	9.4	19	30.3	29	34.4	56.1	52.3	98	72	111.1
8	9.7	19.3	29.7	33.3	38.7	57.2	66.3	126.3	98.7	134.1

TAB. 3.1: Taux d'appariement optimal.

3.3.2 Performance de l'algorithme en termes de temps de recherche

Dans les algorithmes de recherche combinatoire, le temps nécessaire pour effectuer la recherche du graphe médian est très important. Il est impératif qu'un tel algorithme puisse répondre dans un temps raisonnable acceptable par l'utilisateur. Dans cette expérience, nous avons généré six ensembles de graphe comportant respectivement 10, 20, 30, 40 et 50 graphes. Pour chaque graphe nous avons généré aléatoirement les paramètres suivants :

- Un nombre de sommets compris entre 3 et 8.
- Pour chaque sommet et arête, une étiquette prise à partir de $\{0.1, 0.2, ..., 0.9\}$.

Pour chaque ensemble, nous avons varié la cardinalité du graphe médian à rechercher entre 3 et 8. Nous avons fixé le nombre de classes pour la classification de l'ensemble des étiquettes L_s à $2 \times M$, avec M la cardinalité du graphe médian (section 3.2.2).

	Nombre de graphes dans chaque ensemble					
Taille	10	15	20	30	40	50
3	0.063	0.109	0.125	0.156	0.219	0.390
4	0.172	0.219	0.250	0.375	1.078	1.453
5	2.344	3.891	5.532	8.703	11.750	14.750
6	11.953	18.250	28.547	40.360	54.594	72.422
7	49.313	76.281	104.891	149.766	221.156	272.313
8	292.843	441.578	616.266	896.255	1226	1542

TAB. 3.2: Temps en secondes nécessaire pour rechercher le graphe médian. (sur un PC Pentium IV, 256 Mo)

À partir de le tableau 3.2, nous pouvons observé que le temps nécessaire pour rechercher le graphe médian croit linéairement en fonction du nombre de graphes de S et croit d'une manière exponentielle en fonction de la taille du graphe médian. Il est clair qu'à partir de ce tableau le temps nécessaire pour rechercher le graphe médian avec une cardinalité égal à la moyenne des cardinalités des graphes dans l'ensemble est nettement inférieur à celui d'une cardinalité très grande.

3.3.3 Étude comparative avec l'algorithme de Jiang

Dans cette sous-section, nous allons exposer les résultats obtenus en comparant notre algorithme de recherche du graphe médian avec ceux obtenus avec l'algorithme de Jiang utilisant l'approche génétique. Ce dernier algorithme a été décrit dans le premier chapitre section 1.3.2. Pour cela, nous avons conduit sept tests utilisant respectivement des bases de 5, 10, 15, 20, 30, 40 et 50 graphes. Le nombre de sommets dans chaque graphe a été fixé a 5. Le nombre de classes dans l'algorithme de k-means est fixé au double

		Nombre de graphes dans chaque ensemble						
		5	10	15	20	30	40	50
Nouvel	SOD_s	2.9	7.4	10.2	12.4	20.4	28.4	36.5
	SOD_a	10.3	22.8	35.1	47	70.1	96.2	120.1
algorithme	SOD	13.2	30.2	45.3	59.4	90.5	124.6	156.6
Algorithme	SOD_s	4.9	12.6	17.8	23.8	37.3	49.2	61.2
	SOD_a	11.3	23	32	47.8	69.2	98.3	125.2
génétique	SOD	16.2	35.6	49.8	71.6	106.5	147.5	186.4

TAB. 3.3: Valeur de la somme des distances SOD calculée avec le nouvel algorithme et celui de l'algorithme génétique de Jiang.

	Nombre de graphes dans chaque ensemble						
	5	10	15	20	30	40	50
Nouvel algorithme	3.350	7.310	10.110	13.840	20.8	35.4	59.840
Algorithme génétique	3.070	8.120	14.660	22.680	38.2	51.6	81.3

TAB. 3.4: Temps en secondes nécessaire pour la recherche du graphe médian.

de la taille de chaque graphe c'est-à-dire 10 classes. Pour chaque sommet et arête, nous avons affecté une étiquette prise aléatoirement à partir de $\{0.1, 0.2, ..., 0.9\}$. Concernant l'algorithme de Jiang, nous avons fixé les paramètres génétiques suivants dans les quatre expériences comme suit :

- Nombre de populations : 40
- Probabilité de croisements : 0.9
- Probabilité de mutations : 0.1
- Nombre d'itérations : 100

Les tableaux 3.3 et 3.4 montrent respectivement la valeur de SOD pour chaque test et le temps en secondes nécessaire pour trouver le meilleur graphe médian. À la lumière de ces résultats, nous observons que notre algorithme présente une SOD plus faible que celle obtenue à l'aide de l'algorithme de Jiang. Cette différence est linéairement croissante en fonction du nombre de graphes utilisés. D'un autre coté, le nouvel algorithme présente un temps d'exécution plus faible que l'algorithme de Jiang. Le temps d'exécution réalisé par le nouvel algorithme à partir du quatrième test (nombre de graphes est supérieur ou égal à 20) est nettement inférieur que celui réalisé à l'aide de l'algorithme de Jiang. Ceci dit, les deux algorithmes affichent en général de meilleures performances avec une supériorité dans certains cas pour notre algorithme. Cette supériorité est due a l'approche déterministe de notre algorithme. La performance de notre algorithme peut être améliorée en utilisant un autre outil de filtrage des meilleures étiquettes que l'algorithme k-means, et en améliorant la technique de recherche du graphe médian. Pour l'algorithme de Jiang, les paramètres génétiques jouent un rôle très important dans la performance du processus de recherche. Un meilleur choix de ces paramètres peut aussi améliorer sa performance d'une manière considérable.

3.4 Conclusion

Pour conclure ce chapitre, nous avons proposé un algorithme de recherche du graphe médian d'un ensemble de graphes S. C'est un concept proche du barycentre de graphe. On recherche le graphe qui est équidistant de tous les graphes appartenant à l'ensemble S. Ce graphe n'appartient pas forcément à cet ensemble.

La recherche du graphe médian est un problème complexe. Une solution approximative s'impose. Notre algorithme présente trois étapes pour trouver

le graphe médian généralisé. La première étape transforme l'ensemble des étiquettes de départ E_s et E_a. À l'issue de cette étape, seules les étiquettes potentiellement candidates seront retenues. La deuxième étape permet de définir la structure idéale du graphe médian en termes de nombre de sommets. La troisième étape constitue le noyau de notre algorithme. Elle permet de construire le graphe médian en affectant une étiquette à chaque sommet et arête. Le graphe médian cherche à minimiser la somme des distances SOD. Toutes les mises en correspondance entre les sommets et les arêtes du graphe médian et les sommets et les arêtes des différents graphes de S sont générées dans cette même étape.

Le nouvel algorithme a fait l'objet d'une étude comparative avec celui proposé par Jiang [56] utilisant les algorithmes génétiques. Les deux algorithmes ont un comportement similaire en terme de rapidité pour des ensembles de petites tailles. Toutefois, le nouvel algorithme est beaucoup plus rapide pour des ensembles dépassant une trentaine de graphes. Notre algorithme montre aussi une supériorité pour trouver un graphe médian proche de l'optimal par rapport au deuxième algorithme.

Chapitre 4

Reconnaissance de formes et appariement de graphes

4.1 Introduction

Plusieurs travaux ont été effectués dans le passé, utilisant les graphes pour la modélisation et la reconnaissance de formes. Ces travaux ont touché plusieurs domaines tels que la reconnaissance de caractère [94], la reconnaissance des objets 3D [32] [120], l'interprétation des diagrammes schématiques [69], la détection des routes à partir des images radar [118], l'analyse des image IRM et la reconnaissance des tumeurs [90] [91] [92], la reconnaissance des composantes d'un plan d'architecte [73] [74], l'appariement de séquences vidéo [102], la reconnaissance des visages [71], etc. Cette grande diversité d'applications montre le potentiel des graphes pour la modélisation et la recherche dans les bases de données. Cependant, aucun système commercial ou éducatif de grande envergure utilisant les graphes n'a encore vu le jour. La principale raison selon nous est le manque de vulgarisation des algorithmes de recherche et de manipulation de graphes. Le mariage entre la reconnaissance de formes et

les outils de modélisation et de manipulation des graphes est un domaine en pleine effervescence. De plus en plus, on voit apparaître des workshops spécialisés (GbR et SSPR) qui s'intéressent davantage à ce mariage dont l'effort est sans aucun doute bénéfique pour les deux partenaires.

Dans ce qui suit, nous allons présenter dans un premier temps un algorithme de classification d'un ensemble de graphes. L'algorithme développé est une adaptation de l'algorithme k-means pour les graphes. Pour valider l'algorithme expérimentalement, nous avons construit une base de données d'images. Chaque image représente la lettre F légérement modifiée par rapport à une image de référence. Dans un deuxième temps, nous allons présenter une application de recherche d'image dans une base d'images synthétiques. Des expérimentations ont été effectuées pour comparer des images et pour regrouper l'ensemble des images dans plusieurs groupes en utilisant respectivement l'algorithme d'appariement de graphes proposé dans le chapitre 2 et l'algorithme de classification de graphes présenté dans ce chapitre. Nous proposons en dernier lieu, une application de recherche d'images par le contenu pour des images réelles.

4.2 Classification de graphes

Les graphes sont des structures de données puissantes pour décrire des objets ainsi que les relations entre ces objets. Une telle structure de donnée permet de représenter plus adéquatement le contenu des objets et la répartition spatiale des objets. Une collection de formes représentées par des graphes nécessite une classification préalable. Cette classification permet de séparer les formes en plusieurs groupes qui partagent les mêmes caractéristiques. Une telle classification permet de réduire le temps de recherche d'une forme particulière. Elle permet de réduire le nombre de formes candidates

pour sélectionner le ou les formes les plus similaires. Dans ce contexte nous avons développé une variante de l'algorithme k-means pour les graphes basée sur l'algorithme d'appariement de graphes et l'algorithme du graphe médian décrit respectivement dans le chapitre 2 et 3. L'algorithme k-means développé permet de regrouper un ensemble de graphes en k groupes. La distance entre deux graphes est calculée en utilisant l'algorithme d'appariement de graphes. La construction du représentant (centre) de chaque groupe est réalisée à l'aide de l'algorithme du graphe médian.

Algorithme : k_means_graphe()

Entrées :

1) S : L'ensemble des graphes.
2) k : Le nombre de classe.

Sorties :

1) Les différents classes $C_1, C_2, ..., C_k$.
2) Les différents centres $\overline{g}_1, \overline{g}_2, ..., \overline{g}_k$.

Début

1. Choisir aléatoirement k graphes $\overline{g}_i$ décrivant les différents centres initiales.
2. Répeter
   Répartir les différents graphes selon les $\overline{g}_i$.
   Calculer les nouveaux graphes médians $\overline{g}_i$.
   Jusqu'à aucun changement dans $\overline{g}_i \quad (1 \leq i \leq k)$.

Fin

Pour valider l'algorithme ci-dessus, nous avons construit une base de données de 100 images. Chaque image représente la lettre F. La lettre F est for-

mée de trois segments. Deux segments horizontaux et un troisième segment vertical. Chaque image est modélisée par un graphe. Le nombre de sommets dans chaque graphe n'est autre que le nombre de terminaisons de toutes les lignes. En d'autres termes, le nombre de sommet est égal à cinq. Dans le cas où il y a une intersection entre le segment vertical et le deuxième segment horizontal, un sommet supplémentaire est ajouté au graphe pour décrire ce point d'intersection. Une arête dans le graphe représente un segment ou demi-segment dans l'image. Chaque sommet est représenté par les coordonnées x et y du point de terminaison qu'il représente. Chaque arête est représentée par 0 ou 1 pour décrire la présence ou non du segment dans l'image. Pour décrire la similarité et mesurer la distance entre deux sommets, la distance Euclidienne est utilisée entre les deux points représentés par les deux sommets mis en correspondance. Le coût de la suppression ou de l'insertion d'un sommet ou d'une arête est une constante.

Nous avons utilisé la version de l'algorithme k-means décrite plus haut pour séparer les images en plusieurs classes. Nous avons fixé le nombre de classes à 3. Le nombre de phases (K) dans l'algorithme d'appariement de graphes est fixé à 5. La figure 4.1 (première rangée) montre des images appartenant à la première classe. Les cinq images appartenant à cette classe présentent une intersection au même niveau (deuxième intersection). De plus, tous les segments ont approximativement la même longueur. La figure 4.1 (deuxième rangée) montre des images appartenant à la deuxième classe. Toutes les images appartenant à cette classe ont un deuxième segment horizontal court qui ne traverse pas le segment vertical. En d'autres termes, tous les graphes qui représentent ces images ont cinq sommets. La figure 4.1 (troisième rangée) montre des images appartenant à la troisième classe. Le deuxième segment horizontal a pratiquement la même longueur dans toutes

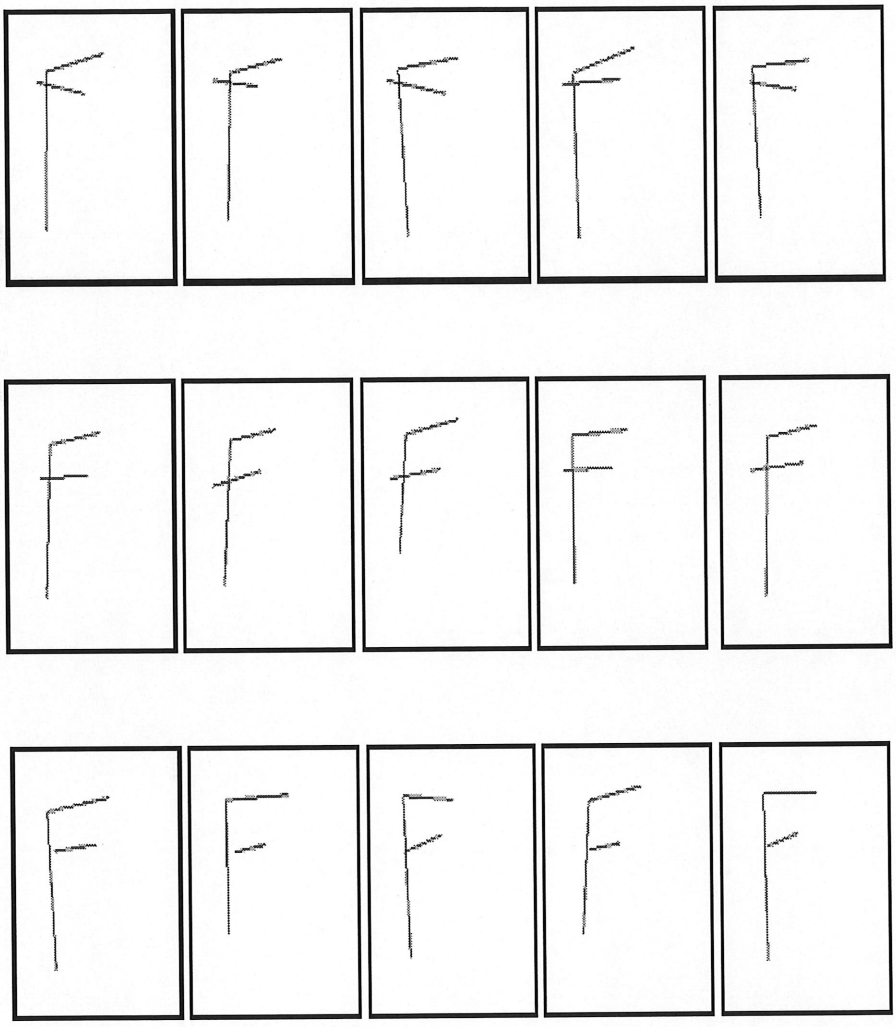


FIG. 4.1: *Classification d'un ensemble d'images représentant la lettre F dans trois classes.*

les images et ils sont tous au même niveau par rapport au premier segment horizontal.

4.3 Recherche d'images synthétiques

Dans cette section, nous allons présenter une application de recherche d'image par le contenu. La base de données comporte des images synthétiques générées d'une manière semi-aléatoire. Les détails de la construction de la base de données, la modélisation des images par des graphes, la recherche d'images par le contenu ainsi que l'organisation de la base de données seront exposées dans les sous sections qui suivent. La recherche d'images est effectuée en faisant appel à l'algorithme d'appariement présenté dans le chapitre 2. L'organisation de la base de données est réalisée en faisant appel à la fois à l'algorithme d'appariement de graphes et l'algorithme de graphe médian. L'organisation de la base de données est un processus de classification des images en plusieurs groupes d'images selon un ou plusieurs critères.

4.3.1 Construction de la base de données

L'objectif principal est de valider les deux algorithmes décrits respectivement dans le chapitre 2 et 3. Nous avons rapporté dans ces chapitres une première évaluation avec des graphes générés aléatoirement. Dans cette section, nous allons étendre notre évaluation à des données formant des images synthétiques. Le but de cette évaluation est d'une part de faire un pas vers le cas réel (image du monde réel) et d'autre part étendre le champ d'application de l'algorithme à plusieurs étiquettes par sommet ou arête. Cette extension, permettra à l'utilisateur d'interroger le système de reconnaissance selon plusieurs critères en combinant les différents caractéristiques des objets et des

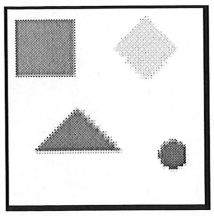


FIG. 4.2: *Image de la base de données.*

relations entre objets. Notre tâche en premier lieu est donc de construire une base d'images synthétiques, pour cela nous avons adopté les hypothèses suivantes.

1. Chaque image comportera au moins 2 objets et au plus 4 objets.
2. Un objet n'est autre qu'une forme géométrique régulière : carré, triangle, losange ou un cercle.
3. La forme, la couleur, la taille et l'emplacement sont les caractéristiques de base d'un objet.

En plus, le nombre d'images dans la base de données a été fixé préalablement à 1000 et il n'y a pas de superposition entre les objets d'une même image. La construction d'une telle base de données d'image vise premièrement à bien distinguer la ou les images qui sont visuellement très proches et deuxièmement à faciliter la modélisation des images par des graphes, ce qui constitue le sujet de la sous-section suivante.

4.3.2 Modélisation des images par des graphes

L'étape de la modélisation des images par des graphes est une des plus importantes dans un processus de recherche d'image par le contenu. La première tâche consiste à définir les caractéristiques pertinentes à extraire de l'image. Une deuxième tâche consiste à définir la structure du graphe mo-

1	Haut - Gauche
2	Haut
3	Haut - Droite
4	Gauche
5	Droite
6	Bas - Gauche
7	Bas
8	Bas - Droite

TAB. 4.1: Tableau de correspondance.

délisant l'image en question. L'image telle que décrite dans la section précédente est une collection d'objets réguliers non superposés. Chaque objet dans l'image est décrit par la forme, la couleur et la taille. La structure du graphe modélisant une image permettra d'obtenir une représentation fidèle de l'image en question. Chaque sommet dans un graphe représentera un objet dans une image. Chaque arête reliant deux sommets dans un graphe représentera une relation entre deux objets dans une image. Un sommet est décrit par trois étiquettes représentant respectivement la forme, la couleur et la taille de l'objet. Une arête est décrite par deux étiquettes représentant respectivement la distance entre les deux objets et la position relative. Nous avons utilisé la distance de Hausdorff [53] pour calculer la distance entre deux objets. La position relative entre deux objets est déterminée selon le tableau de correspondance ci-dessous. (tableau 4.1).

La position relative entre deux objets sera donnée en fonction de l'emplacement d'un objet par rapport à l'autre. Pour localiser l'emplacement d'un objet, nous avons subdivisé l'image en 4 zones. Ensuite, nous avons affecté un indice de zone pour chaque objet. Un objet appartient à une zone particulière si et seulement si, la plus grande partie des ses pixels appartenant à

cette zone. Finalement, la position relative est attribuée suivant l'indice de chaque objet.

4.3.3 Recherche d'images par le contenu

L'algorithme d'appariement de graphes décrit dans le chapitre 2 permet de rechercher l'isomorphisme de graphes ou de sous-graphes existant entre deux graphes. L'algorithme a été décrit pour des graphes comportant une étiquette par sommet et par arête. Une première tâche donc consiste à adapter l'algorithme à plusieurs étiquettes par sommet ou par arête. D'autre part, nous avons utilisé dans l'algorithme de base la distance Euclidienne pour mesurer la similarité entre deux étiquettes, ce qui induit une similarité globale entre les deux graphes. Or, dans notre cas précis, ce type de distance n'est pas applicable pour certains types d'attributs (exemple la forme). Une deuxième adaptation est donc nécessaire pour calculer l'erreur globale entre deux graphes.

L'erreur globale ou l'erreur de correction notée *ecag* est composée de l'erreur engendrée par l'appariement des sommets noté $ecag_s$ et de l'erreur engendrée par l'appariement des arêtes noté $ecag_a$.

$$ecag = \sum_{i=1}^{|V|} ecag_s + \sum_{i=1}^{|V|} ecag_a \tag{4.1}$$

Avec $|V|$ la taille du graphe le plus petit entre les deux graphes.

L'appariement des sommets consiste à mesurer le degré de similarité ou la distance entre les valeurs des étiquettes **forme**, **taille** et **couleur**, noté respectivement $ecag_{sf}$,$ecag_{st}$ et $ecag_{sc}$. Nous avons introduit pour chaque terme de cette dernière équation un paramètre pour contrôler son usage lors de la définition de la requête.

$$ecag_s = \alpha ecag_{sf} + \beta ecag_{st} + \gamma ecag_{sc} \tag{4.2}$$

L'erreur engendrée par la forme $ecag_{sf}$ prend la valeur 1 si les deux objets(appartenant à deux images différentes) mis en correspondance n'ont pas la même forme et 0 dans le cas contraire. L'erreur engendrée par la taille $ecag_{st}$ est décrite de la manière suivante

$$ecag_{st} = \frac{|T_1 - T_2|}{T_1 + T_2} \tag{4.3}$$

Où T_1 et T_2 sont la taille respectivement du premier et deuxième objet mis en correspondance.

L'erreur engendrée par la couleur $ecag_{sc}$ est décrite de la manière suivante

$$ecag_{sc} = \frac{\sqrt{(C_{l1} - C_{l2})^2 + (C_{u1} - C_{u2})^2 + (C_{v1} - C_{v2})^2}}{\sqrt{(C_{l1} + C_{l2})^2 + (C_{u1} + C_{u2})^2 + (C_{v1} + C_{v2})^2}} \tag{4.4}$$

où C_{l1}, C_{u1}, C_{v1}, C_{l2}, C_{u2} et C_{v2} sont respectivement les bandes L, U et V du premier et deuxième objet mis en correspondance.

D'autre part, l'appariement des arêtes consiste à mesurer le degré de similarité ou la distance entre les valeurs des étiquettes distance et position relative entre deux objets, appelé respectivement $ecag_{ad}$ et $ecag_{ap}$.

$$ecag_a = \delta ecag_{ad} + \epsilon ecag_{ap} \tag{4.5}$$

L'erreur engendrée par la position relative de deux arêtes $ecag_{ap}$ prend la valeur 1 si les deux arêtes des deux graphes mis en correspondance n'ont pas la même valeur et 0 dans le cas contraire.

L'erreur engendrée par la distance $ecag_{ad}$ est décrite de la manière suivante

$$ecag_{ad} = \frac{|D_1 - D_2|}{D_1 + D_2} \tag{4.6}$$

Où D_1 et D_2 sont respectivement la distance entre deux objets de la même image concernant la première et la deuxième image.

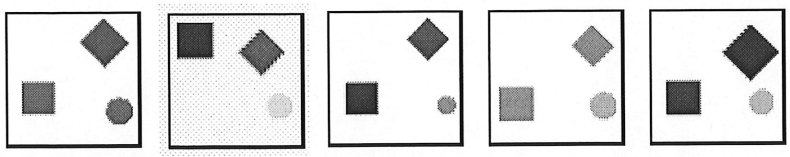


FIG. 4.3: *Recherche d'images selon la forme ; L'image requête et les quatre images les plus semblables.*

Les différents paramètres introduits dans les équations 4.2 et 4.5 offrent à l'usager une grande flexibilité pour décrire sa requête selon ses besoins. Cette flexibilité permet à l'usager d'introduire un ou plusieurs caractéristiques et un degré d'importance pour chaque caractéristique.

Recherche d'images selon la forme

Dans cette première expérience, la recherche est effectuée selon la forme des objets. On cherche des images dans la base de données qui ont exactement un carré, un losange et un cercle. La valeur attribuée au paramètre α est 1, tous les autres paramètres sont nuls. La figure 4.3 illustre l'image requête et le résultat de la recherche. Les quatre images résultats sont semblables à l'image requête. L'erreur relative à ces images est nulle.

Recherche d'image selon la taille

Dans cette deuxième expérience, la recherche est effectuée selon la taille des objets. On cherche des images de la base de données qui ont quatre objets. La valeur attribuée au paramètre β est 1, tous les autres paramètres sont nuls. L'image requête illustrée par la première image de la figure 4.4 comporte quatre grands objets. Les quatre images résultats présentent des objets ayant la même grosseur que les objets dans l'image requête.

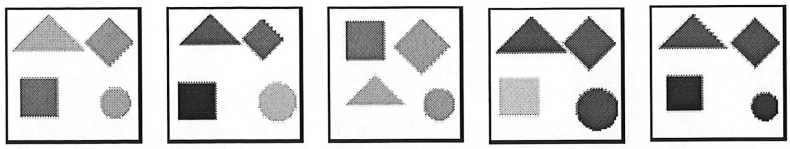


FIG. 4.4: *Recherche d'images selon la taille ; L'image requête et les quatre images les plus semblables.*

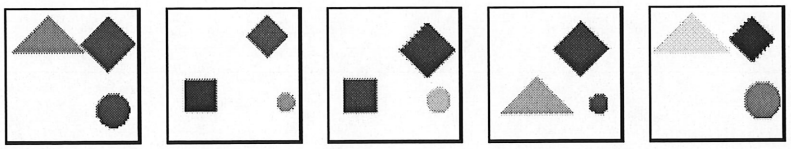


FIG. 4.5: *Recherche d'images selon la couleur ; L'image requête et les quatre images les plus semblables.*

Recherche d'images selon la couleur

Dans cette troisième expérience, la recherche est effectuée selon la couleur des objets. La valeur attribuée au paramètre γ est 1, tous les autres paramètres sont nuls. La figure 4.5 illustre l'image requête et quatre images résultats. Les trois premières images résultats ont approximativement les mêmes couleurs que l'image requête tandis que la quatrième image présente un objet (le triangle) dont la couleur est très différente de celles dans l'image requête. On voit dans cet exemple, que les cinq images ne partagent pas les mêmes types d'objets.

Recherche d'images selon la position relative

Dans cette quatrième expérience, la recherche est effectuée selon la position relative des objets. On cherche des images de la base de données qui ont trois objets. Le premier objet se trouve dans le coin haut droit de l'image.

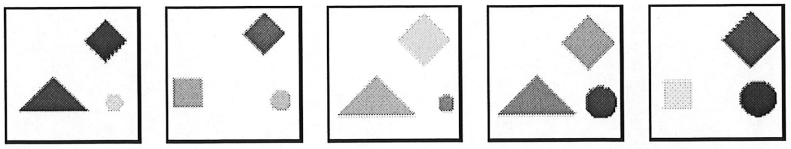


FIG. 4.6: *Recherche d'images selon la position relative; L'image requête et les quatre images les plus semblables.*

Le deuxième objet se trouve en dessous du premier et le troisième se trouve dans le coin bas gauche de l'image. La valeur attribuée au paramètre ϵ est 1, tous les autres paramètres sont nuls. Les quatre images résultats présentent la même disposition des objets. Les images résultats ne partagent pas forcément la même forme des objets.

Recherche d'images selon la forme et la position relative

Dans cette cinquième expérience, la recherche est effectuée selon la forme et la position relative des objets. On cherche des images de la base de données qui ont trois objets et ayant la même disposition que l'image requête. La valeur attribuée au paramètre α et ϵ est 0.5, tous les autres paramètres sont nuls. Les quatre images résultats présentent la même disposition des objets ainsi la même forme d'objets que l'image requête. Il faut noter qu'à partir d'un certain rang des images résultats, les images ont tendances a perdre de précision concernant ou bien la position relative ou bien la forme.

4.3.4 Organisation de la base de données

Dans cette sous-section, nous rapportons les résultats obtenus à partir de la classification de la base de données décrite plus haut en plusieurs classes selon de nombreux critères. L'algorithme de classification utilisé a été décrit

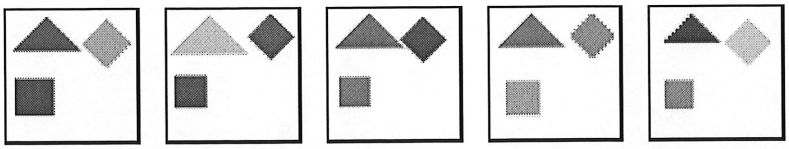


FIG. 4.7: *Recherche d'images selon la distance ; L'image requête et les quatre images les plus semblables.*

dans la section 4.2. Cet algorithme utilise pour calculer la distance entre deux graphes l'algorithme présenté dans le chapitre 2 et pour recalculer les centres des différentes classes (médians) l'algorithme présenté dans le chapitre 3. Pour cela, nous avons fixé le nombre de classes à 3, le nombre de phases dans l'algorithme d'appariement de graphes à 5.

Classification selon la forme

Dans cette première expérience le critère choisi pour répartir les différentes images en 5 classes est la forme. Les figures 4.8 et 4.9 montrent respectivement le médian de la classe et quatre images appartenant à la classe 1 et 2. La classe 1 comporte des images qui ont quatre objets (un triangle, un carré, un losange et un cercle). Cette classe comporte aussi des images qui ont trois et quatre objets. Cependant, les images appartenant à cette classe ont au moins trois objets semblables à l'image du médian. La quatrième image de la classe 2 comporte un objet de moins par rapport au médian. Cependant, elle comporte trois objets (triangle, cercle et losange) semblables au médian.

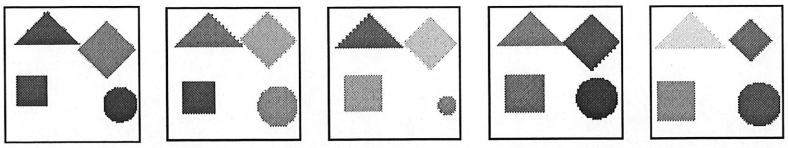


FIG. 4.8: *Répartition selon la forme; Le médian de la première classe ainsi que quatre images de la même classe.*

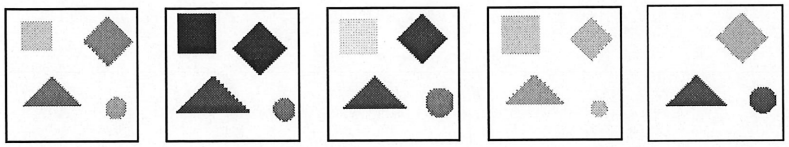


FIG. 4.9: *Répartition selon la forme; Le médian de la deuxième classe ainsi que quatre images de la même classe.*

Classification selon la forme et la position relative

La deuxième expérience montre l'efficacité de l'algorithme à bien répartir les différentes images selon deux critères à savoir la forme et la position relative des objets. Les figures 4.10 et 4.11 montrent le médian de la classe et quatre images appartenant respectivement à la classe 1 et 2. Les différentes images de la première et la deuxième classe partagent les mêmes caractéristiques. La grosseur ainsi que la couleur n'ont pas été respecté pour la classification. Outre ces images la classe 1 et 2 comportent d'autres images qui n'ont pas nécessairement exactement la même formes d'objets cependant ils partagent la même position relative. Le deuxième scénarios est aussi possible.

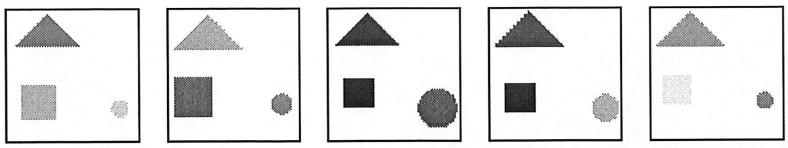


FIG. 4.10: *Répartition selon la forme et la position relative ; Le médian de la première classe ainsi que quatre images de la même classe.*

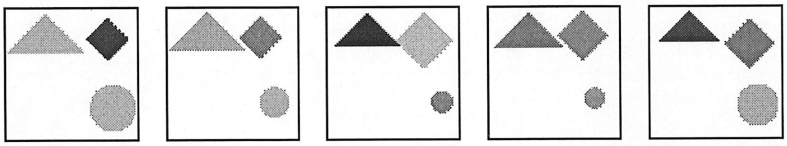


FIG. 4.11: *Répartition selon la forme et la position relative ; Le médian de la deuxième classe ainsi que quatre images de la même classe.*

Temps de recherche séquentielle et à l'aide des clusters

Plus la base de données est grande plus le temps de recherche est important. Il est donc impérativement de trouver le moyen de réduire ce temps de recherche et rendre le processus de recherche faisable dans un temps raisonnable. La classification décrite ci-dessus permet de réduire le temps de recherche en réduisant le nombre de comparaisons possibles. Par conséquent, une recherche exhaustive sera évitée et remplacé par une recherche intelligente sans perte significative de la qualité des résultats trouvés. Afin de mieux voir l'impact direct de la recherche à travers les classes par rapport à une recherche exhaustive nous avons mené deux expérimentations. Dans la première, nous avons mené une recherche exhaustive permettant de trouver les images les plus semblables à une image requête. Une deuxième expérimentation a été conduite dans le même but et avec la même image requête mais en utilisant les classes et les médians préalablement recherchés.

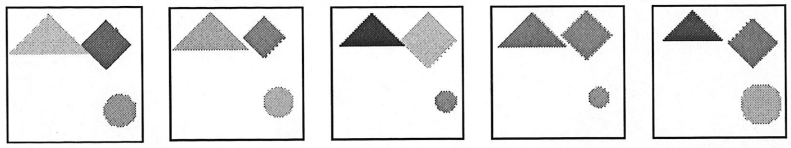


FIG. 4.12: Résultat de la recherche de l'image requête (la première) selon la forme et la position relative.

La figure 4.12 montre le résultat de la première expérimentation, l'image requête et les quatre images les plus semblables. Le critère de la classification et de la recherche exhaustive est la forme et la position relative. Les quatre images les plus semblables appartiennent à la même classe illustrée dans la figure 4.11. Ce résultat montre que le processus de classification a bien séparé les différentes images de la base de données.

Nous avons mené une deuxième expérimentation qui permet de rechercher la même image requête en utilisant les classes. Dans un premier temps, nous avons comparé l'image requête avec les différents médians des différentes classes au nombre de 8. Nous avons conclu que le médian de la deuxième classe est le plus proche de notre image requête. Ensuite, nous avons mené une recherche exhaustive restreinte aux images de cette deuxième classe. Les quatre images les plus semblables à notre image requête sont les mêmes que celles obtenus en utilisant la recherche exhaustive. Nous avons aussi comptabilisé le temps de recherche nécessaire pour trouver les quatre premières images les plus semblables à l'aide d'une recherche exhaustive et en utilisant les classes et les médians des différentes classes. Le temps nécessaire pour trouver les images les plus similaires est de 0.28 secondes en utilisant les classes. Cependant, 2 secondes sont nécessaires pour trouver les images les plus semblables avec une recherche exhaustive. Le gain en temps de calcul est très significatif. D'autres tests ont été effectué sur des graphes aléatoires figurent dans [44].

4.4 Recherche d'images réelles

Dans la section précédente, nous avons présenté une application permettant de modéliser, de rechercher et d'organiser une base d'images synthétiques à l'aide des graphes. Dans cette section, nous allons présenter une deuxième application permettant de modéliser et de rechercher des images réelles. Le système de reconnaissance d'images par le contenu décrit dans cette section comporte deux parties (Figure 4.13). L'objectif de cette section est non pas de construire un système de recherche d'images par le contenu robuste et efficace mais plutôt une application, traitant des images réelles, faisant appel à la modélisation structurelle des images et aux algorithmes développés dans cette thèse. Le système conçu est divisé en deux parties. La première partie (*Segmentation*) comporte quatre processus : un algorithme de classification de pixels de l'image, un algorithme de détection des régions dominantes de l'image, un algorithme de fusion des régions et un algorithme de construction des graphes. La deuxième partie est consacrée à la recherche d'images requête dans la base de données d'images en faisant appel à l'algorithme d'appariement de graphes. Dans ce qui suit, nous allons présenter les quatre algorithmes pour réaliser la segmentation et la modélisation des images par les graphes. Des expérimentations ont été conduites pour mettre en évidence la performance de l'algorithme d'appariement de graphes basé sur les sommets pour la recherche d'images par le contenu.

En partant d'une image réelle, la première partie de notre système consiste à automatiser le processus de modélisation et de construction du graphe correspondant. Ce processus nécessite une méthode de segmentation. Il existe une multitude de méthodes de segmentation. Cependant, pour des raisons de commodité, nous avons opté pour le développement d'un algorithme de

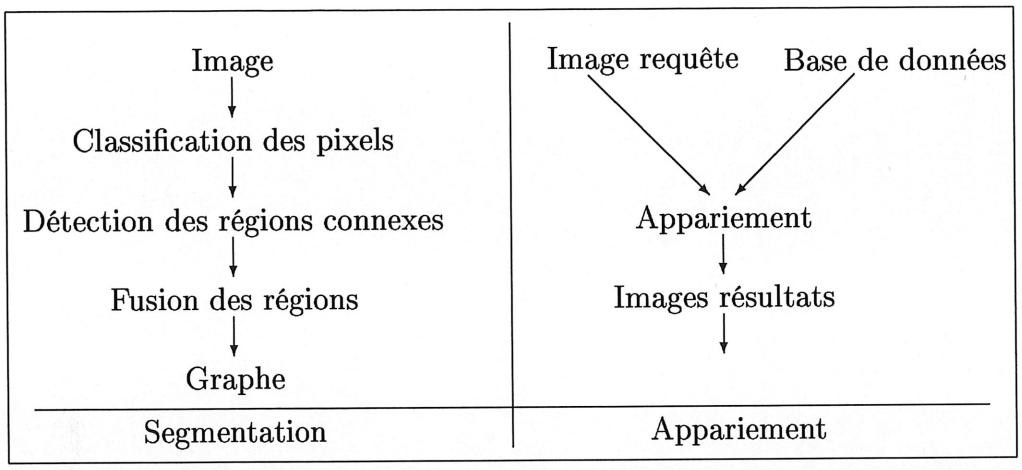


FIG. 4.13: *Système de recherche d'images par le contenu.*

segmentation. La construction de graphe est alors facile à automatiser. La première tâche de notre algorithme de segmentation consiste à classifier les différents pixels de l'image en plusieurs classes. Cette étape a été réalisée en faisant appel à l'algorithme FCM. À la base de cette classification, la détection des régions connexes est effectuée. Cette détection est réalisée en deux temps. Tous d'abord l'algorithme cherche toutes les régions connexes de l'image. Puis, l'algorithme fusionne les petites régions avec des régions (dominantes) préalablement sélectionnées selon leurs tailles. La dernière étape consiste à construire le graphe correspondant.

4.4.1 Classification des pixels

Dans cette sous-section, nous étudions brièvement l'algorithme de classification des pixels utilisé. Cet algorithme a été développé par Sun et Wang [105]. Il est basé sur l'algorithme C-moyenne flou (FCM) utilisant la stratégie "Trial-and-error" pour déterminer le nombre de classes pour un ensemble de données.

Le C-moyenne flou (FCM) :

Cet algorithme a été inventé en 1973. Plusieurs travaux ont été réalisés proposant d'autres définitions pour la norme utilisée et d'autres prototypes en ce qui concerne le centre des classes. De nos jours, le FCM est le plus utilisé pour la classification floue dans les applications de reconnaissance et de recherche des données. Le FCM de base consiste à minimiser

$$Min(J_m(U,V;X)) = \sum_{k=1}^{n} \sum_{i=1}^{C} u_{ik}^{m} ||x_k - v_i||_A^2 \tag{4.7}$$

Avec
$X = \{x_1, , x_n\}$ l'ensemble de données ;
n le nombre de données ;
C est le nombre de classes ;
$V = \{v_1, , v_C\}$ les centres des classes ;
$U = (u_{ik})_{C \times n}$ est la matrice définissant la classe de membership ;
m est un paramètre appelé le flou.

Dans ce qui suit, nous introduisons un algorithme efficace pour déterminer le nombre de classes. Dans l'algorithme FCM, le nombre de classe est un paramètre clé donné par l'utilisateur. Dans la pratique, la détermination du nombre de classes est la première tâche à réaliser. Cette tâche est un problème très difficile et complexe pour la classification des données. Pour cela, nous faisons appel à une stratégie simple et efficace appelé "Trial-and-error". L'idée de base de l'algorithme utilisant cette stratégie consiste à réaliser la classification pour plusieurs valeurs de C, puis à évaluer chaque résultat à l'aide de la fonction index $Vd(c)$ et finalement à choisir le nombre optimal de classes. Dans chaque étape, l'algorithme répartit l'ensemble des données en c classes, avec $C_{min} \leq c \leq C_{max}$. Il évalue ensuite le résultat avec un index de validité et finalement divise la "mauvaise" classe en deux classes.

4.4.2 Détection et fusion des régions

Une fois que la classification des pixels est terminée, chaque pixel aura une étiquette qui indique le numéro de sa classe d'appartenance. Le but de cette étape est de déterminer les régions connexes qui seront utilisées lors de la modélisation et la recherche d'image. La détection est réalisée en trois phases. La première phase consiste à déterminer toutes les régions connexes de l'image. Les pixels de chaque région connexe ont une couleur similaire. En d'autres termes, les pixels de chaque région appartiennent à la même classe suivant la classification faite lors de l'étape précédente. La deuxième phase consiste à déterminer les régions *dominantes* pour chaque représentant d'une classe. Une région dominante est une région possédant le plus grand nombre de pixels parmi toutes les régions relatives à une classe donnée. À la fin de cette deuxième phase, le nombre de régions dominantes détecté est égal au nombre de classes. La dernière phase consiste à choisir parmi les régions restantes, celles qui peuvent être considérées comme intéressante de point de vue du nombre de pixels. Les régions sélectionnées seront considérées aussi comme dominantes. Une région sélectionnée doit satisfaire l'équation 4.8 pour être considérée comme dominantes.

$$n_a > (n_d \alpha) \tag{4.8}$$

Avec n_a est le nombre de pixels de la région candidate pour obtenir une place parmi les dominantes, n_d est le nombre de pixels de la région dominante relative à cette région candidate (les deux régions appartiennent à la même classe). Le paramètre α contrôle le nombre de régions candidates pour une place parmi les dominantes.

Pour fusionner les régions restantes (r_i) avec les régions dominantes. Le critère de fusion est la couleur. Une région (r_i) est fusionnée avec une région dominante (r_d) si et seulement si l'équation ci-dessous est satisfaite. Le pa-

ramétre N et D indiquent respectivement le nombre de régions à fusionner et le nombre de résgions dominantes.

$$r_i = arg \min_{i \in N,\ d \in D} (R_i - R_d)^2 + (G_i - G_d)^2 + (B_i - B_d)^2) \tag{4.9}$$

4.4.3 Modélisation par des graphes

La modélisation des images est réalisée de la manière suivante. Chaque région de l'image est représentée par un sommet du graphe et la relation spatiale entre deux régions est matérialisée par une arête. Chaque sommet est représenté par quatre étiquettes. Les trois premières étiquettes décrivent les composantes RGB de la région. La quatrième étiquette décrit la taille de la région. Chaque arête est décrite par deux étiquettes, la distance entre les deux régions correspondantes et la position relative. Cette dernière étiquette est assigniée manuellement pour le moment. Il est clair qu'il est possible d'inclure d'autres caractéristiques pour décrire les régions (ex. texture). La figure 4.14 décrit une image de la base de données et la même image une fois segmentée. La figure 4.15 montre le graphe correspondant. Les trois premiers nombres dans le grand cercle décrivent la valeur des trois composantes RGB de la région tandis que le dernier nombre indique la taille de la région. Le nombre à coté de chaque ligne (arête) indique la distance entre les deux régions modélisées par les deux sommets reliés par cette arête.

4.4.4 Expérimentation

La base d'images utilisée dans l'expérience contient 100 images. Chaque image a été segmentée en plusieurs régions. Ensuite, chaque image a été modélisée par un graphe. Chaque graphe contient entre 3 et 20 sommets. Les étiquettes décrivant un sommet sont respectivement la taille et la couleur de la région correspondante. L'étiquette décrivant une arête est la distance entre

FIG. 4.14: *Image de la base de données et son image segmentée.*

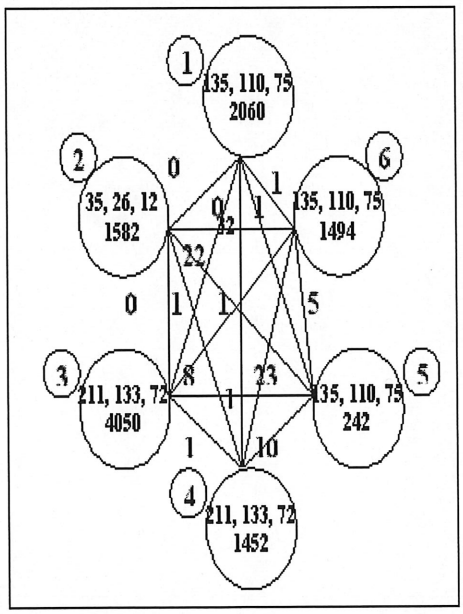


FIG. 4.15: *Le graphe correspondant à l'image de la figure 4.14.*

FIG. 4.16: *Segmentation de l'image.*

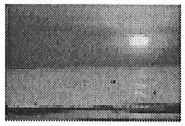


FIG. 4.17: *Image requête.*

les deux régions reliant les deux sommets reliés par cette arête. La figure 4.16 décrit le résultat de la segmentation. La figure 4.18 illustre le résultat de la recherche d'une image requête illustrée par la figure 4.17. La première et la deuxième image résultat sont très semblables à l'image requête. Cependant, la troisième ne ressemble pas à l'image requête mais par contre, elle partage la même couleur avec l'image requête.

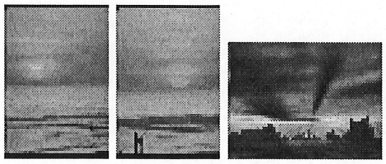


FIG. 4.18: *Résultats de la recherche.*

4.5 Conclusion

Nous avons présenté deux applications de recherche d'images par le contenu utilisant les graphes pour la modélisation et la représentation des données. Pour mesurer la similarité entre les images, l'algorithme d'appariement de graphes basé sur les sommets décrit dans le chapitre 2 a été utilisé. Pour la classification des images, nous avons développé une version de l'algorithme k-means appliquée aux graphes. Cette version utilise les deux algorithmes présentés dans le chapitre 2 et 3. La première application porte sur la recherche d'image par le contenu dans une base d'images synthétiques. Les résultats obtenus montrent que l'approche structurelle (graphe) est une autre approche fiable et donne d'aussi bons résultats en RIPC que l'approche statistique.

La deuxième application exposée dans ce chapitre, permet d'introduire les graphes dans le paradigme de la recherche d'image réelle par le contenu. Pour cela, nous avons développé un algorithme de segmentation d'image en plusieurs régions. L'algorithme comporte essentiellement trois étapes. La première étape consiste à identifier les couleurs dominantes en appliquant un algorithme de classification flou (FCM). La deuxième étape consiste à identifier toutes les régions connexes et à élire des régions dominantes. La troisième étape consiste à fusionner les petites régions avec les régions dominantes les plus proche.

Les applications proposées dans ce chapitre et celles citées dans la section 4.1 témoignent de la féconde contribution des graphes à la reconnaissance de formes. Un travail futur peut être entamé, qui approfondira l'aspect spécifique de chaque application.

Conclusion

Cette thèse s'inscrit dans le cadre général de la reconnaissance de formes structurelles. Elle s'intéresse plus particulièrement à la modélisation des formes par les graphes. Un graphe est un ensemble de sommets et d'arêtes où chaque sommet ou arrête est caractérisé par un ensemble d'étiquettes. Trois axes de recherche ont formé l'épine dorsale de la thèse : l'appariement de graphes, la recherche d'un graphe médian représentant un groupe de graphes pour la classification d'un ensemble de graphes et le développement d'un système de recherche d'images par le contenu.

L'appariement de graphes est un problème de mise en correspondance des sommets de deux graphes qui tient compte des mises en correspondance des arêtes. C'est un problème NP-complet. Nous avons proposé un algorithme pour rechercher le meilleur appariement entre deux graphes. Tous les algorithmes d'appariement de graphes partagent le même problème d'explosion combinatoire de l'espace de recherche. Pour atténuer cette complexité, des solutions polynomiales ont été développées. Quant à l'optimalité, aucune garantie n'est acquise. Nous avons tenté de trouver une solution au problème permettant de trouver des bons appariements sans explorer tout l'espace de recherche.

Le graphe médian est un nouveau concept en reconnaissance de formes structurelles. C'est un concept très important à pour objectif la représenta-

tion d'un groupe de graphes par un seul représentant. La recherche du graphe médian est une tâche difficile et très coûteuse en temps de calcul. À notre connaissance un seul algorithme existe dans la littérature. L'importance de ce concept réside dans son éventuelle utilisation comme un pont entre l'approche structurelle et l'approche statistiques.

La recherche d'images par le contenu est un domaine en pleine expansion. Les systèmes existant utilisent souvent l'approche statistique. Notre objectif était de développer une application dans laquelle la modélisation du contenu des images est réalisée à l'aide des graphes. Cette application nécessiste aussi les algorithmes d'appariement de graphes et du graphe médian décrits dans cette thèse. L'utilisation des graphes pour représenter le contenu d'une image présente plusieurs avantages. Un des avantages est la simplicité de définir des requêtes combinant en même temps le contenu des objets à rechercher et les relations des objets.

Contributions principales

Les contributions principales de notre travail peuvent donc se résumer suivant ces trois axes :

Nouvel algorithme d'appariement de graphes :

L'algorithme d'appariement de graphes développé [39] [46] permet de trouver l'isomorphisme de graphes ou de sous-graphes. L'idée de base de l'algorithme est la séparation du processus de recherche des mises en correspondance des sommets et des arêtes. L'algorithme permet alors de trouver une solution proche de l'optimal dans un temps raisonnable et d'éviter la complexité élevée du problème. L'algorithme a montré sa performance par rapport à d'autres méthodes. Son point fort réside dans la manière de

trouver la solution contrôlée par un seul paramètre K. L'algorithme est indépendant des données (similaires ou non), ce qui ouvre la porte à son utilisation dans plusieurs domaines en reconnaissance de formes.

Nouvel algorithme de recherche du graphe médian :

Le concept de graphe médian est très important en reconnaissance de formes. Représenter un groupe de graphes par un seul candidat est une tâche difficile qui n'a pas été explorée suffisamment dans le passé. Nous avons développé [42] [45] [47] un algorithme pour réaliser cette tâche. Nous avons converti le problème de la recherche du graphe médian en un problème d'optimisation. Nous avons proposé une solution heuristique pratique qui permet de contrôler la complexité élevée du problème. Une utilisation immédiate de notre algorithme dans la classification [44] d'un ensemble de graphes permet une meilleure répartition de cet ensemble. Cette répartition tire profit de l'aptitude du graphe médian à représenter fidèlement un groupe de graphes.

Appariement de graphes et reconnaissance de formes :

Nous avons développé [40] [48] une application pour la recherche d'images par le contenu. Des images synthétiques ont été utilisées et modélisées par des graphes. L'algorithme d'appariement de graphes a été appliqué pour chercher le ou les images les plus similaires à une image requête. Nous avons enregistré des résultats satisfaisants. L'usager a le choix d'inclure plusieurs caractéristiques en une seule requête. De plus, des caractéristiques décrivant les objets et les relations inter-objets peuvent être combinées dans une seule requête. Nous avons aussi développé [41] [43] une application pour la recherche d'images réelles. Cette application est une tentative pour une modélisation et représentation automatique du contenu d'une image réelle par un graphe.

Perspectives

Les perspectives ouvertes par notre travail sont nombreuses car chaque contribution présentée dans ce document peut être enrichie indépendamment. D'une manière générale, l'approche structurelle, en particulier les graphes, est un axe de recherche ouvert, qui mérite une attention plus approfondie.

La première perspective concerne l'algorithme d'appariement de graphe proposé. L'algorithme recherche le meilleur appariement en comparant les sommets dans un premier temps, ensuite les arêtes. L'algorithme sélectionne les mises en correspondance suivant leur degré de similarité. On peut envisager d'inclure l'approche de relaxation probabilistique pour choisir les meilleures mises en correspondance. L'algorithme gardera la même technique de recherche en profitant du choix judicieux de l'approche Bayésienne.

Concernant l'algorithme de graphe médian, une première perspective pouvant être envisagée qui consiste à trouver un moyen rapide et efficace pour déterminer le nombre optimal de sommets et d'arêtes dans un graphe médian. Une deuxième perspective consiste à améliorer la construction du graphe médian selon les sommets. Une idée similaire à la première perspective est d'inclure dans le choix des mises en correspondance l'approche Bayésienne.

Une autre perspective consiste à développer un outil de génération automatique de graphe. Cet outil aura comme tâche l'extraction d'un ou plusieurs graphes à partir des documents web ou des documents XML. Peu d'outils existent de nos jours permettant une telle génération. Cet outil opère en amont de nos algorithmes développés dans le cadre de ce travail.

Bibliographie

[1] H. A. Almohamed and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE PAMI*, 15(5) : 522-525, 1993.

[2] H. G. Barrow and R. J. Popplestone. Relational descriptions in picture processing. *Machine Intellignece*, 4 :377-396, 1971.

[3] H. G. Barrow and R. M. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters*, 4(4) :83-84, 1976.

[4] S. Berretti, A. D. Bimbo and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. *IEEE PAMI*, 23(10) :1089-1105, 2001.

[5] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters* 19 :255-259, 1998.

[6] H. Bunke. Error correcting graph matching : on the influence of the underlying cost function. *IEEE PAMI*, 21(9) :917-922, 1999.

[7] H. Bunke and A. Kandel. Mean and maximum common subgraph of two graphs. *Pattern Recognition Letters*, 21 :163-168, 2000.

[8] H. Bunke and X. Jiang. Graph matching and similarity. In Teodorescu, H.-N., Mlynek, D., Kandel, A., Zimmermann, H.-J. Edition :Intelligent Systems and Interfaces, Kluwer Academic Publishers. 2000.

[9] H. Bunke, X. Jiang and A. Kandel. On the minimum common supergraph of two graphs. *Computing*, 65 :13-25, 2000.

[10] H. Bunke. Recent advances in structural pattern recognition with applications to visual form analysis. Visual Form 2001, Springer Verlag, LNCS 2059, 11-23, 2001.

[11] H. Bunke. Graph matching for visual object recognition. *Spatial Vision*, 13 :335-340, 2000.

[12] H. Bunke, S. Gunter and X. Jiang. Towards bridging the gap between statistical and structural pattern recognition : two new concepts in graph matching. Advances in Pattern Recognition - ICAPR 2001, Springer Verlag, LNCS 2013, 1-11, 2001.

[13] H. Bunke and S. Gunter. Weighted mean of a pair of graphs. *Computing*, 67 :209-224, 2001.

[14] C. K. Chen and D. Y. Y. Yun. Unifying graph-matching problem with a practical solution. In Proceedings of International Conference on Systems, Signals, Control, Computers, September 1998. http ://citeseer.nj.nec.com/chen98unifying.html.

[15] 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), May 2004.

[16] W. J. Christmas, J. Kittler and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE PAMI*, 17(8) :749-764, 1995.

[17] L.P. Cordella, P. Foggia, C. Sansone, F.Tortorella and M. Vento. Graph Matching : A fast algorithm and its evaluation. In Proceedings of the 14^{th} IEEE ICPR, pages 1582-1584, Brisbane, Austria, August, 16-20 :1998.

[18] L.P. Cordella, P. Foggia, C. Sansone, F.Tortorella and M. Vento. An improved algorithm for matching large graphs. In Proceedings of the 3^{rd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 149-159, Ischia, Italy, May 23-25 :2001.

[19] D. G. Corneil and C. G. Gotlieb. An efficient algorithm for graph isomorphism, *Journal of the ACM*, 17(1) :51-64, 1970.

[20] A. D. J. Cross, R. C. Wilson and E. R. Hancock. Genetic search for stuctural matching. *Pattern Recognition*, 30 :953-970, 1997.

[21] A. D. J. Cross and E. R. Hancock. Convergence of a hill climbing genetic algorithm for graph matching. *Pattern Recognition*, 33 :1863-1880, 2000.

[22] A. D. J. Cross and E. R. Hancock.Graph matching with a dual step EM algorithm. *IEEE PAMI*, (20) :1236-1253, 1998.

[23] F. W. DePiero, M. M. Trivedi and S. Serbin. Graph matching using a direct classification of node attendance, *Pattern Recognition*, 29(6) :1031-1048, 1996. http ://citeseer.nj.nec.com/depiero96graph.html.

[24] S. Dickinson, M. Pellilo and R. Zabih. Introduction to the special section on graph algorithms in computer vision. *IEEE PAMI*, 23(10) :1049-1052, 2001.

[25] R. Englert and R. Glantz. Towards the clustering of graphs. In Proceedings of the 2^{nd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 125-133, Haindorf, Austria, May 10-12 :1999.

[26] M. A. Eshera and K. S. Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. *Jounal of the ACM*, 8(5) :604-618, 1986.

[27] M. Fernandez and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22 :753-758, 2001.

[28] M. Fischer and R. R. Elschlager. The representation and matching pictorial structures. *IEEE Transactions on Computers*, 22(1) :67-92, 1973.

[29] P. Foggia, C. Sansone and M. Vento. A database of graphs for isomorphism and sub-graph isomorphism benchmarking. In Proceedings of the 3^{rd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 176-187, Ischia, Italy, May 23-25 :2001.

[30] P. Foggia, C. Sansone and M. Vento. A performance comparison of five algorithms for graph isomorphism. In Proceedings of the 3^{rd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 188-199, Ischia, Italy, May 23-25 :2001.

[31] R. Giugno and D. Shasha. GraphGrep : A fast and universal method for querying graphs. In Proceedings of the 16^{th} IEEE ICPR, Quebec, Canada, August, 11-15 :2002.

[32] E. Gmur and H. Bunke. 3D object recognition based subgraph matching in polynomial time. In R. Mohr, Th. Pavlidis, and A. Sanfeliu, editors, *Structural Pattern Recognition Analysis*, 131-147, World Scientific Publ. Co. 1989.

[33] S. Gunter and H. Bunke. Validation indices for graph clustering. In Proceedings of the 3^{rd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 229-238, Ischia, Italy, May 23-25 :2001.

[34] S. Gunter and H. Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23 :401-417, 2002.

[35] S. Gunter and H. Bunke. Adaptive self-organizing map in the graph domain. *Hybrid Methods in Pattern Recognition*, World Scientific, ISBN 981-02-4832-6, 47 :61-74, 2002.

[36] M. Hagenbuchner, M. Gori, H. Bunke, A. C. Tsoi and C. Irniger. Using attributed plex grammars for the generation of image and graph databases. *Pattern Recognition Letters*, 24(8) : 1081-1087, 2003.

[37] E. Hancock and R. C. Wilson. Graph-based methods for vision : A yorkist manifesto. IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR), Windsor, Canada, LNCS 2396, pages 31-46, 2002.

[38] L. Herault, R. Horaud, F. Veillon and J. J. Niez. Symbolic image matching by simulated annealing. In Proceedings of Britsih Machine Vision Conference, 319-324, 1990.

[39] A. Hlaoui and S. Wang. A new algorithm for inexact graph matching. In Proceedings of the 16^{th} IEEE ICPR, Quebec, Canada, August, 11-15 :2002.

[40] A. Hlaoui and S. Wang. A new algorithm for graph matching with application to content-based image retrieval. IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR), Windsor, Canada, LNCS 2396, pages 291-300, 2002.

[41] A. Hlaoui, H. Sun and S. Wang. Image retrieval using fuzzy segmentation and a graph matching technique. In the 2^{nd} International Conference on Machine Learning and Cybernetics, Beijing, China, 2002.

[42] A. Hlaoui and S. Wang. A new median graph algorithm. In Proceedings of the 4^{th} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 239-249, York, UK, June 30 /July 2 :2003.

[43] A. Hlaoui and S. Wang. Graph representation in content-based image retrieval. AICCSA 2003, Tunis - Tunisie, 16-18 Juillet 2003.

[44] A. Hlaoui and S. Wang. Graph clustering algorithm with application to content-based image retrieval. In the 3^{nd} International Conference on Machine Learning and Cybernetics, Beijing, China, 2003.

[45] A. Hlaoui and S. Wang. A direct approach to graph clustering. In proceedings of the IASTED Conference NCI 2004, Grindelwald, Suisse.

[46] A. Hlaoui and S. Wang. A node-mapping-based algorithm for graph matching. Soumis à International Journal of Pattern Recognition and Image Analysis.

[47] A. Hlaoui and S. Wang. Median graph computation for graph clustering. Soumis à the international Journal of Soft Computing - A Fusion of Foundations, Methodologies and Applications.

[48] A. Hlaoui and S. Wang. Image Retrieval Systems Using Graph Matching. Rapport de Recherche, No. 275, Département de mathématiques et d'informatique, Université de Sherbrooke, 2001.

[49] B. Huet and E. Hancock. Shape recognition from large image libraries by inexact graph matching. *Pattern Recognition Letters*, 20 :1259-1269, 1999.

[50] B. Huet, A. D. J. Cross And E. R. Hancock. Shape Retrieval by Inexact Graph Matching. IEEE International Conference on Multimedia Computing and Systems (ICMCS'99), Florence, Italy, page 772-776, 7-11 June 1999. http ://www.eurecom.fr/ huet/cv.html

[51] B. Huet and E. R. Hancock. Inexact Graph Retrieval. IEEE CVPR99 Workshop on Content-based Access of Image and Video Libraries

(CBAIVL-99), Fort Collins, Colorado USA, pages 40-44, June 22, 1999. http ://www.eurecom.fr/ huet/cv.html

[52] B. Huet, A. D. J. Cross and E. R. Hancock. Graph Matching for Shape Retrieval. Advances in Neural Information Processing Systems 11, Edited by M.J. Kearns, S.A. Solla and D.A. Cohn, MIT Press, June 1999. http ://www.eurecom.fr/ huet/cv.html

[53] D. P. Huttenlocher, G. A. Klanderman and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE PAMI*, 15(9) :850-863, 1993. http ://citeseer.nj.nec.com/huttenlocher93comparing.html.

[54] Q. Iqbal and J. K. Aggarwal. Using structure in content-based image retrieval. In proceedings of the IASTED International Conference Signal and Image Processing(SIP), Oct. 18-21, 1999, Nassau, Bahamas, pages 129-133.

[55] C. Irniger and H. Bunke. Graph matching : filtering large databases of graphs using decision trees. In Proceedings of the 3^{rd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 239-249, Ischia, Italy, May 23-25 :2001.

[56] X. Jiang, A. Munger and H. Bunke. On median graphs : properties, algorithms, and applications. *IEEE PAMI*, 23(10) :1144-1151, 2001.

[57] 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, pp. 143-151, 2002.

[58] X. Jiang, A. Munger and H. Bunke. Synthesis of representative graphical symbols by computing generalized median graph. In Proceedings of the 3^{rd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 187-194, Ischia, Italy, May 23-25 :2001.

[59] X. Jiang, A. Munger and H. Bunke. Computing the generalized median of a set of graphs. In Proceedings of the 2^{nd} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 115-124, Haindorf, Austria, May 10-12 :1999.

[60] X. Jiang and H. Bunke. Optimal quadratic-time isomorphism of ordered graphs. *Pattern Recognition*, 32 :1273-1283, 1999.

[61] X. Jiang, L. Schiffmann and H. Bunke. Computation of median shapes. In Proceedings of the 4^{th} Asian Conference on Computer Vision, Taipei, Taiwan, 300-305, 2000.

[62] X. Jiang and H. Bunke. Optimal vertex ordering of graphs. *Information Processing Letters*, 72 :149-154, 1999.

[63] X. Jiang. and H. Bunke. On the coding of ordered graphs, *Computing*, 61(1) :23-38, 1998.

[64] A. Juan and E. Vidal. Fast Median Search in Metric spaces. Advances in Pattern Recognition. A. Amin and D. Dori, eds. pages 905-912, Springer-Verlag.

[65] H. Kawaji, Y. Yamaguchi, H. Matsuda and A. Hashimoto. A graph-based clusetring method for a large set of sequences using a graph partitioning algorithm. *Genome Informatics*, 12 : 93-102, 2001.

[66] J. Kittler and E. R. Hancock. Combining evidence in probabilistic relaxation. *IEEE PRAI*, 3 :29-51, 1989.

[67] S. Kosinov and T. Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR), Windsor, Canada, LNCS 2396, pages 133-142, 2002.

[68] M. Lazarescu, H. Bunke and S. Venkatesh. Graph matching : Fast elimination using machine learning techniques. Advances in Pattern Recognition, LNCS 1876, Springer Verlag, 236-245, 2000.

[69] S. W. Lee and J.H. Groen. Translation, rotation and scale invariant recognition of hand-drawn symbols in schematic diagrams. Internat. J. Pattern Recognition Artifi. Intell. 4(1), 1-15.

[70] G. Levi. A note on the derivation of maximal common subgraphs of two directed and undirected graphs. Calcolo 9, pages 341-354, 1972.

[71] T. Leung, M. Burl, and P. Perona. Finding Faces in Cluttered Scenes Using Labelled Random Graph Matching. In Proceedings of the 5^{th} International Conference of Computer Vision, pages 637-644, 1995.

[72] C. Liu, K. Fan, J. Horng and Y. Wang. Solving weighted graph matching problems by modified microgenetic algorithm. IEEE International Conference on Systems, Man, and Cybernetic(ICSMC'95), pages 638-643, 1995.

[73] J. Llados, E. Marti and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE PAMI*, 23(10) :1137-1143, 2001.

[74] J. Llados. Combining Graph Matching and Hough Transform for Hand-Drawn Graphical Document Analysis. Application to Architectural Drawings. Thèse de doctorat de l'université de Barcelone.http ://www.cvc.uab.es/ josep/.

[75] B. Luo, R. C. Wilson, E. R. Hancoke. The Independant and Principal Component of Graph Spectra. In Proceedings of the 16^{th} IEEE ICPR, Quebec, Canada, August, 11-15 :2002.

[76] B. Luo, R. C. Wilson, E. R. Hancoke. Spectral Feature Vectors for Graph Clustering. IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition(S+SSPR), Windsor, Canada, LNCS 2396, pages 83-93, 2002.

[77] B. Luo and E. R. Hancoke. Structural matching using the EM algorithm and singular value decomposition. *IEEE PAMI*, 23(10) :1120-1136, 2001.

[78] J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proc. of The Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I, Statistics, L.M. Le Cam and J. Neyman (Eds.). University of California Press, 1967.

[79] B. D. McKay. Practical graph isomorphism. Congressus Numerantium, 30 :45-87, 1981.

[80] K. Messer and J. Kittler. A region-based image database system using colour and texture. *Pattern Recognition Letters*, 20 :1323-1330, 1999.

[81] B.T. Messmer and H. Bunke. A network based approach to exact and inexact graph matching. Technical report IAM-93-021, University of Bern, Institut fur Informatik und angewandte Mathmatik, 1993.

[82] B.T. Messmer and H. Bunke. Subgraph isomorphism detection in polynomial time on pre-processed model graphs. In ACCV, pages 373-382, 1995.

[83] B.T. Messmer. Efficient graph matching algorithms for preprocessed model graphs. Thesis, University of Bern, 1996 http ://citeseer.nj.nec.com/114601.html

[84] B.T. Messmer and H. Bunke. Fast error-correcting graph isomorphism based on model recompilation. http ://citeseer.nj.nec.com/messmer96fast.html

[85] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE PAMI*, 20(5) :493-504, 1998.

[86] B.T. Messmer and H. Bunke. Efficient subgraph isomorphism detection : a decomposition approach. *IEEE Knowledge and Data Engineering*, 12(2) :307-323, 2000.

[87] L. Mico and J. Oncina. An approximate median search algorithm for non-metric spaces. *Pattern Recognition Letters*, 22 :1145-1151, 2001.

[88] K. Oflazer. Error-tolerant retrieval of trees. *IEEE PAMI*, 19(12) :1376-1380, 1997.

[89] M. Pellilo, K. Siddiqi and S. W. Zucker. Matching hierarchical structures using association graphs. *IEEE PAMI*, 21(11) :1105-1120, 1999.

[90] A. Perchant. Morphisme de graphes d'attributs flous pour la reconnaissance structurelle de scènes. Application à la reconnaissance de structures anatomiques saines et pathologiques sur des IRM, rapport de Thèse, 6 novembre 2000.

[91] E. G. M. Petrakis, C. Faloutsos. Similarity searching in medical image databases. *IEEE Knowledge and Data Engineering*, 9(3) :435-447, 1997.

[92] E. G. M. Petrakis, C. Faloutsos and K. Lin. ImageMap : An Image Indexing Method Based on Spatial Similarity. *IEEE Knowledge and Data Engineering*, 14(5) :979-987, 2002.

[93] A. Robles-Kelly and E. Hancock. In Proceedings of the 16^{th} IEEE ICPR, Quebec, Canada, August, 11-15 :2002.

[94] J. Rocha and T. Pavlidis. A shape analysis model with applications to a character recognition system. *IEEE PAMI*, 16(4) :393-404, 1994.

[95] A. Sanfeliu and K.S. Fu, A distance measure between attributed relational graphs for pattern recognition. *IEEE SMC*, 13(3) :353-362, 1983.

[96] A. Schenker, M. Last, H. Bunke and A. Kandel. Comparision of distance measures for graph-based clustering of documents. In Proceedings of the 4^{th} IAPR Workshop on Graph-based Representations in Pattern Recognition, pages 202-213, York, England, 30^{th} June - 2^{nd} July :2003.

[97] D. Shasha, J. T. L. Wang and R. Giugno. Algorithmics and applications of tree and graph searching. International Conference on Management of Data and Symposium on Principles of Database Systems, pages 39-52, 2002.

[98] D.C. Schmidt and L.E. Druffel. A fast backtracking algorithm to test directed graphs for isomorphism using distances matrices. *Journal of the ACM*, 23 :433-445, 1976.

[99] L. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE PAMI*, 3(5) :504-518, 1981.

[100] H. Sun, S. Wang and Q.Jiang, "FCM-based model selection algorithms for determining the number of clusters", to appear in *Pattern Recognition*, 2004.

[101] L. Shapiro. Relational matching. In T. Y. Young, editor, Handbook of Pattern Recognition and Image Processing : Computer Vision, pages 475-496. Academic Press, 1993.

[102] K. Shearer, S. Venkatesh and H. Bunke. Video sequence matching via decision tree path following. *Pattern Recognition Letters*, 22 :479-492, 2001.

[103] K. Shearer, H. Bunke and S. Venkatesh. Video indexing and similarity by largest common subgraph detection using decision trees. *Pattern Recognition*, 34 :1075-1091, 2001.

[104] M. Skomorowski. Use of random graph parsing for scene labelling by probablistic relaxation. *Pattern Recognition Letters*, 20 :949-956, 1999.

[105] H. Sossa and R. Horaud. Model indexing : The graph-hashing approach. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Urbana-Champaign, Illinois, USA, pages 811-814, June 1992.

[106] 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.

[107] 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.

[108] W.H. Tsai and K.S. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE SMC*, 9(12) :757-768, 1979.

[109] W.H. Tsai and K.S. Fu. Subgraph error-correcting isomorphisms for syntactic pattern recognition. *IEEE SMC*, 13 :48-62, 1983.

[110] J. R. Ulmann. An algorithm for subgraph isomorphism, Journal of the association for Computing Machinery, 23(1) :31-42, January 1976.

[111] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE PAMI*, 10(5) :695-703, 1988.

[112] W. D. Wallis, P. Shoubridge, M. Kraetz and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22 :701-704, 2001.

[113] J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang and K. M. Currey. An algorithm for finding the largest approximately common substructures of two trees. *IEEE PAMI*, 20(8) :889-895, August 1998.

[114] Y. Wang, K. Fan and J. Horng. Genetic-based search for error-correcting graph isomorphism. *IEEE SMC*, 27(4) :588-597, 1997.

[115] M. L. Williams, R. C. Wilson and E. R. Hancock. Multiple graph matching with bayesian inference. *Pattern Recognition Letters*, 18 :1275-1281, 1997.

[116] R. C. Wilson, E. N. Evans and E. R. Hancock. A bayesien compatibility model for graph matching. *Pattern Recognition Letters*, 17 :263-276,1996.

[117] R. C. Wilson and E. R. Hancock. Graph matching with hierarchical discrete relaxation. *Pattern Recognition Letters*, 20 :1041-1052, 1999.

[118] R. C. Wilson. Inexact graph matching using symbolic constraints, thèse de doctorat, 7 novembre 1996. Université de York, Royaume-Uni.

[119] A. K. C. Wong, S. W. Lu and M. Rioux. Recognition and shape synthesis of 3-D objects based on attributed hypergraphs. *IEEE PAMI*, 11(3) :279-289, 1989.

[120] E. K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3) :287-303, 1992.

[121] A. K. C. Wong and M. You. Entropy and distance of random graphs with application to structural pattern recognition. *IEEE PAMI*, 7(3) :509-609, 1985.