UNIVERSITÉ FRANÇOIS RABELAIS
TOURS

École Doctorale : Santé, Sciences
et Technologies

Année Universitaire : 2007-2008

**THÈSE POUR OBTENIR LE GRADE DE
DOCTEUR DE L'UNIVERSITÉ DE TOURS**

Discipline : Informatique

présentée et soutenue publiquement

par :

Rashid Jalal QURESHI

le 4 mars 2008

Reconnaissance de formes et symboles graphiques complexes dans les images de documents

---

Directeur de thèse : Professeur Hubert CARDOT

Co-encadrement : Jean-Yves RAMEL

JURY

Hubert CARDOT	*Directeur de thèse*	Professeur des universités à l'Université de Tours
Pierre HEROUX	*Examinateur*	Maître de Conférences à l'Université de Rouen
Josep LLADOS	*Examinateur*	Associate Professor à l'Universitat Autònoma de Barcelona, Espagne
Jean-Marc OGIER	*Rapporteur*	Professeur des universités à l'Université de La Rochelle
Jean-Yves RAMEL	*Examinateur*	Maître de Conférences à l'Université de Tours
Karl TOMBRE	*Rapporteur*	Professeur des universités à l'Institut National Polytechnique de Lorraine

Remerciements

Une graine ne peut pousser correctement que si elle est plantée dans une terre fertile, et est régulièrement arrosée. Je souhaite remercier très vivement la personne sans laquelle tout ceci n'existerait pas. Je voudrais exprimer ma gratitude et mes sincères remerciements à Jean Yves Ramel, Maître de conférences au Laboratoire d'informatique de l'Université François Rabelais de Tours, mon co-directeur de thèse, qui m'a transmis le goût de la recherche. Ses encouragements tout le long des trois années passées m'ont été très précieux et je pense sincèrement que ses conseils me serviront largement dans le futur. Je le remercie pour avoir encadré ce travail et pour avoir corrigé ce manuscrit dans les meilleurs délais.

Je tiens à remercier les membres du jury qui ont accepté de juger ce travail. Mes remerciements s'adressent à Karl Tombre et Jean-Marc Ogier, respectivement Professeur à École des Mines de Nancy (Institut National Polytechnique de Lorraine) et à l'Université de La Rochelle - qui m'ont fait l'honneur d'être les rapporteurs de ma thèse, qui ont, dans des délais courts, fait l'effort de trouver du temps dans leur agenda pour rapporter sur mon mémoire, qu'ils trouvent ici l'expression de ma sincère gratitude. Leurs travaux dans le domaine de l'analyse du document m'ont vraiment aidé depuis que j'ai commencé ma recherche. Leur grande expérience, leurs compétences et contributions au niveau international m'apparaissent exemplaires.

Je remercie également, Josep Lladós, Professeur à CVC Universitat Autònoma de Barcelona, et Pierre Héroux, Maître de Conférences à l'Université de Rouen, pour l'intérêt qu'ils ont toujours porté à mon travail et pour le plaisir qu'ils m'ont fait en acceptant d'être examinateurs. Je leur suis donc extrêmement reconnaissant d'avoir trouvé du temps pour juger mon travail.

Je remercie très chaleureusement Hubert CARDOT, Professeur au Laboratoire d'informatique de l'Université François Rabelais de Tours, mon directeur de thèse, pour la confiance qu'il m'a accordée en m'accueillant dans le laboratoire d'informatique (LI) et pour sa contribution décisive dans l'élaboration de ce travail.

Mes plus vifs remerciements vont également aux H.E.C « Higher Education Commission, Government of Pakistan » et à SFERE (Société Française d'exportation des Ressources Éducatives) pour m'avoir fourni une Bourse et l'opportunité de venir en France pour obtenir une thèse. Je remercie spécialement M. Georges PIERRON, Directeur de Gestion des Programmes SFERE, pour la confiance qu'il m'a accordée et Mme. France Lamiscarre pour sa disponibilité et encouragements.

Un remerciement spécial va à Sébastien DELEST, avec qui j'ai exploré la France et partagé d'agréables moments entre nous.

Je tiens aussi à remercier vivement tous les membres de l'équipe Reconnaissance des Formes et Analyse d'Image (RFAI) pour les conseils et les encouragements qu'ils m'ont dispensés. J'ai ainsi pu travailler dans les meilleures conditions et ces trois années de recherche ont été passionnantes. J'ai une pensée amicale pour mes trois collègues de bureau : Moustafa, Mohammad Arif et Cédric.

Enfin, je tiens à remercier ma famille et mes parents pour leur attention constante mais discrète, leur intérêt pour mes travaux et leur enthousiasme m'incitant à toujours chercher de nouveaux défis et de nouveaux horizons.

À mes parents

Auxquels je dois ce que je suis. Que dieu vous protège.

Daddy, your affection lives on as the most cherished memory in my life…

Tous l'univers visible est comme un magasin d'images et signes auxquels l'imagination donne une place et valeur relatives

The whole visible universe is but a storehouse of images and signs to which the imagination will give a relative place and value;

Charles Baudelaire (1821-1867)

Résumé

Ce travail de thèse se situe à la croisée de trois thématiques de recherche : la mise en place de représentations structurelles pour représenter le contenu d'images de documents, la reconnaissance structurelle des formes et graphiques complexes et la localisation des symboles dans les images des documents.

Pour répondre aux problématiques de l'analyse d'images de documents, nous nous sommes particulièrement intéressés aux méthodes structurelles de reconnaissance des formes et avons choisi d'utiliser les graphes comme outil de représentation des contenus des images. La raison principale de ce choix réside dans leur capacité à intégrer le contexte global du document lors de l'analyse des différentes parties ainsi que leur robustesse aux transformations affines (lorsque les attributs sont bien définis). Leur capacité de description des relations entre éléments et la multitude d'outils permettant leur manipulation constitue également des justifications pour ce choix.

Pour ce qui concerne la représentation du contenu des images, nous sommes reparti des travaux de J-Y. Ramel que nous avons complétés et généralisés à différents types de formes et donc de documents. La nouvelle représentation obtenue améliore et simplifie à la fois la tâche de localisation mais aussi la tâche de reconnaissance de formes graphiques contenues dans les documents. Concernant l'étape de reconnaissance, nous présentons trois stratégies originales pour la mise en correspondance de graphes, combinant les approches structurelle et statistique. Elles aident à la résolution du problème de complexité et évitent un temps de calcul exponentiel intolérable. Les nouvelles techniques d'appariement de graphes que nous proposons sont basées sur des fonctions de similarité qui utilisent aussi bien des valeurs numériques que symboliques pour produire un score. Ces mesures de similarité ont de nombreuses propriétés intéressantes comme un fort pouvoir discriminant, une invariance aux transformations affines et une faible sensibilité au bruit. Nous évaluons aussi les taux de reconnaissance obtenus afin de conclure sur la meilleure technique permettant de reconnaître les formes complexes en contexte parmi toutes celles que nous avons étudiées.

Une nouvelle approche générique et automatique est également présentée pour la localisation des symboles graphiques dans les images de documents. A notre connaissance, il s'agit de l'approche qui nécessite le minimum de connaissances a priori sur les domaines ou sur le type de symboles présents dans les images. Notre approche nécessite très peut d'interaction avec l'utilisateur et localise toutes les régions susceptibles de contenir des symboles en une seule passe. Comme pour nos autres propositions, nous évaluons cette nouvelle approche sur différents types d'images de documents (électroniques, architectures et logiques).

Table des matières

Chapitre 3 : Représentation structurelle d'images de documents graphiques et localisation de symboles

Chapitre 4 : De l'appariement de graphes symboliques à l'appariement de graphes numériques

Table des Figures

Table des Tableaux

1 Introduction générale

Ce travail de thèse se situe à la croisée de trois thématiques de recherche : la mise en place de représentations structurelles pour représenter le contenu d'images de documents, la reconnaissance structurelle des formes et graphiques complexes et la localisation des symboles graphiques dans les images des documents.

Les graphes, en particulier les graphes attribués, sont des outils universels utilisés très largement dans le domaine de la vision par ordinateur et de la reconnaissance de formes. Le problème de l'appariement de graphes est de trouver une correspondance entre les sommets d'un graphe et les sommets d'un autre graphe qui satisfasse à certaines contraintes ou critères d'optimalité [Bunke, 1999].

Il y a selon nous deux grandes catégories de méthodes de comparaison de graphes. La première catégorie inclut les méthodes qui utilisent une approche symbolique pour la mise en correspondance, par exemple, la recherche d'isomorphisme exact entre graphes ou entre sous-graphes, et la recherche de sous-graphes similaires avec tolérance d'erreur. Dans ce dernier cas, l'idée est de définir une distance entre graphes afin de permettre, par exemple, de rechercher l'image la plus similaire à une image requête fournie par l'utilisateur.

Cette catégorie de méthodes propose des solutions en un temps raisonnable mais plus approximatives. Un compromis doit donc être établi selon l'application visée. De plus, si l'on relaxe les contraintes au niveau de l'appariement des éléments des graphes, les méthodes approximatives deviennent beaucoup plus robustes contre le bruit et les distorsions présentes dans les images. Nous avons donc choisi de travailler sur ce type de méthodes afin de pouvoir reconnaître des formes complexes en contexte.

La reconnaissance de symboles graphiques attire la curiosité des scientifiques depuis de nombreuses années déjà. Les symboles sont des signes ou des formes avec une signification particulière associée à des domaines spécifiques d'application. Ils peuvent correspondre simplement à des formes binaires en 2D composées de lignes, d'arcs et de formes pleines ou alors correspondre à des formes plus complexes en niveaux de gris ou même en couleur réparties de manière isolée ou connectée à l'intérieur d'images. D'un point de vue applicatif, une grande partie des travaux de recherche dans ce domaine est

consacrée à la reconnaissance de dessins techniques [Adam, 2000] [Tombre, 1997] et à la rétro conversion automatique des schémas d'architecture afin qu'ils soient archivables dans un format appréhendable par les systèmes de DAO [Dosch, 2000]. L'objectif est alors de reconnaître les symboles représentant des éléments de bâtiment comme les murs, les portes, les fenêtres, les meubles, les escaliers, etc. Pour cela, les systèmes logiciels se basent la plupart du temps sur un dictionnaire de symboles connus a priori qui définissent les configurations de formes habituellement utilisées. Bon nombre d'articles, complètement ou en partie consacrés aux problèmes liés à la reconnaissance de symboles dans les images de documents ont été publiés récemment. Différentes approches ont été conçues pour différents types d'applications. Mais les procédés requis pour résoudre les différents problèmes particuliers restent pour l'instant utilisables uniquement sur un type d'images précis et peuvent rarement être utilisés ou adaptés pour d'autres applications. Malgré ces nombreux travaux, les problèmes scientifiques restant à résoudre sont encore nombreux, surtout lorsque l'on s'attache à produire des algorithmes et méthodes d'analyse et d'indexation génériques.

Dans le cadre de cette thèse, nous avons concentré nos efforts sur les points suivants :

La sélection du mode de représentation des images est certainement l'une des opérations les plus importantes permettant une réduction importante de la quantité d'information à analyser. Le but est de faciliter la tâche des étapes suivantes tout en fournissant un maximum d'informations fiables sur le contenu du document initial.

Ensuite, la localisation concerne l'exploitation des représentations produites pour identifier différentes composantes qui ont des probabilités fortes d'être des symboles. L'objectif des recherches dans ce cadre vise à élaborer des algorithmes destinés à fournir les zones d'intérêt d'une image (correspondant à de l'information utile) tout en négligeant les zones sans information utile.

Enfin, la procédure de classification/interprétation permet de déclencher les actions, soit de reconnaissance directe, soit de construction de clusters (regroupements des parties similaires) pour ensuite soumettre les représentants de chaque cluster à un modèle de caractérisation afin de leur associer une ou plusieurs signatures ou étiquettes pour finalement aboutir à l'interprétation ou à l'indexation du contenu du document.

Il est à noter que ces différentes opérations sont fortement interdépendantes. Le fameux paradoxe du ***reconnaître pour mieux segmenter et segmenter pour mieux reconnaître*** apparaît et le choix d'une méthode de localisation/segmentation peut donc influencer la définition du processus d'interprétation.

Pour répondre à toutes ces problématiques, nous nous sommes particulièrement intéressés aux méthodes structurelles de reconnaissance des formes et avons choisi d'utiliser les graphes comme outil de représentation des contenus des images. La raison principale de ce choix réside dans leur capacité à intégrer le contexte global du document lors de l'analyse des différentes parties ainsi que leur robustesse aux transformations affines (lorsque les attributs sont bien définis). Leur capacité de description des relations entre éléments et la multitude d'outils permettant leur manipulation constituent également des justifications pour ce choix.

En ce qui concerne la représentation du contenu des images, nous sommes repartis des travaux de J-Y. Ramel que nous avons complétés et généralisés à différents types de formes et donc de documents. Nos travaux sur ce point sont présentés dans le chapitre. 3 et concernent particulièrement la sélection des attributs descriptifs à associer aux arcs et aux nœuds du graphe décrivant le contenu de l'image. La nouvelle représentation obtenue améliore et simplifie à la fois la tâche de localisation mais aussi la tâche de reconnaissance de formes graphiques contenues dans les documents.

Concernant l'étape de reconnaissance, nous présentons trois stratégies originales pour la mise en correspondance de graphes, combinant les approches structurelle et statistique. Elles aident à la résolution du problème de complexité et évitent un temps de calcul exponentiel intolérable. Les nouvelles techniques d'appariement de graphes que nous proposons sont basées sur une fonction de similarité qui utilise aussi bien des valeurs numériques que symboliques pour produire un score de similarité. Ces mesures de similarité ont de nombreuses propriétés intéressantes comme un fort pouvoir discriminant, une invariance aux transformations affines et une faible sensibilité au bruit. Nous évaluons les taux de reconnaissance obtenus afin de conclure sur la meilleure technique permettant de reconnaître les formes complexes en contexte parmi toutes celles que nous avons étudiées.

Ce manuscrit suit l'organisation suivante :

Le **Chapitre 1** dresse un état de l'art sur l'usage des graphes en analyse d'images et en reconnaissance des formes. Nous présentons les définitions et les notations utiles pour la lecture de ce document. Nous citons ensuite les principales approches de représentation de formes utilisant les graphes existantes dans la littérature. Nous détaillons les principales méthodes à base de graphes utilisées en reconnaissance de formes et en analyse d'images. Nous terminons le chapitre 1 par une discussion sur la nécessité de proposer un nouveau système générique et capable d'interpréter complètement des documents graphiques et formes complexes en contexte.

Le **Chapitre 2** s'intéresse à l'état actuel des recherches en reconnaissance des symboles graphiques isolés et leur localisation dans les images de documents graphiques. Nous mettons en évidence aussi bien les avancées réalisées depuis quelques années que les lacunes qui restent à combler. Une synthèse des résultats des concours GREC est également présentée.

Le **Chapitre 3** se scinde en deux grandes parties. La première est dédiée aux différentes approches utilisables pour la représentation des documents graphiques et décrit en détail la représentation générique que nous proposons, construite à base de graphes relationnels attribués. Parmi les différentes primitives graphiques utilisables, nous nous focalisons sur celles issues d'un processus de vectorisation afin d'obtenir une représentation complète des formes fines et pleines présentes dans une image. La seconde partie décrit une nouvelle approche générique et automatique pour la localisation des symboles graphiques dans les images de documents sans connaissance a priori. Nous présentons les résultats expérimentaux obtenus par cette nouvelle approche et ce chapitre reprend en partie les résultats publiés dans [Qureshi, 2007b] [Qureshi, 2007e].

Le **Chapitre 4** aborde les différentes méthodes que nous proposons pour résoudre le problème de la mise en correspondance de graphes attribués en temps polynomial. Les images représentées par des graphes sont comparées à l'aide d'une nouvelle méthode pour déterminer le meilleur appariement possible entre deux représentations structurelles. Notre méthode peut être vue comme un double processus : la sélection d'appariements entre sommets, et ensuite le calcul d'un score à partir de la définition d'une mesure de similarité adaptable en fonction des caractéristiques des formes à

reconnaître et des images à analyser. Une optimisation utilisant une technique de signatures de graphes est également proposée pour réduire les temps de calcul. Nous terminons le chapitre 4 par une comparaison des résultats obtenus par notre méthode avec ceux trouvés dans la littérature. Ce chapitre reprend en partie les résultats publiés dans [Qureshi, 2005], [Qureshi, 2006a] [Qureshi, 2006b], [Qureshi, 2006c] [Qureshi, 2006d] [Qureshi, 2007a], [Qureshi, 2007b], [Qureshi, 2007c], [Qureshi, 2007d].

Enfin, la dernière partie de ce manuscrit conclut la manuscrit et mettant en évidence nos apports et en dégageant de nouvelles perspectives de recherche pour les années à venir.

Chapitre 1

Représentation et reconnaissance de formes à base de graphes : un état de l'art

Les représentations sous forme de graphes structurels ont été appliquées dans un grand nombre de problèmes d'analyse d'images et de reconnaissance de formes. Cependant, les techniques de mise en correspondance des graphes posent encore des problèmes de complexité ainsi que des problèmes de sensibilité au bruit. La détection d'isomorphismes de sous-graphes ou la recherche du plus grand sous-graphe commun ou encore la distance d'édition entre graphes sont des problèmes NP-complet. Ceci implique un temps de calcul exponentiel intolérable lors de la résolution des problèmes qui nous intéressent. Un autre problème en reconnaissance des formes est la variabilité des objets en cas de bruit et de distorsion dans les images à analyser. Ainsi, il est possible de trouver différentes représentations structurelles pour un même objet surtout lorsque les primitives, le mode de construction et les attributs de description sont mal choisis. C'est la raison pour laquelle des réflexions sur les modes de représentations des contenus des images de documents sont toujours aussi importantes que les méthodes de reconnaissance employées pour résoudre de tels problèmes. Afin de pouvoir mieux situer nos travaux par la suite, après un bref rappel sur la terminologie utilisée en théorie des graphes, ce chapitre décrit les principales méthodes à base de graphes utilisées en reconnaissance des formes et en analyse d'images et plus particulièrement lorsqu'il s'agit d'images de documents.

1.1 Définitions élémentaires

Visuellement, un graphe est un schéma constitué par un ensemble de points et par un ensemble de flèches reliant chacun de ceux-ci. Les points sont appelés les sommets du graphe et les flèches les arcs du graphe[1]. Les sommets d'un graphe peuvent, par exemple, représenter des parties d'objets et les arcs des relations entre ces différentes parties. Les relations peuvent être de natures diverses, par exemple, géométriques, spatiales, temporelles ou conceptuelles. Les sommets et les arcs peuvent comporter des étiquettes et attributs pour ajouter des informations sur les objets, par exemple, la longueur d'un trait, la couleur d'une région, l'angle entre les traits, ou la distance entre deux points.

Les graphes peuvent être orientés ou non. Un graphe orienté « directed graph, digraph » est un couple *(V, E),* dans lequel *V* est un ensemble dont les éléments sont les sommets (ou nœuds), *E* est un ensemble de couples d'éléments de *V* ($E \subseteq V \times V$) – dans lesquels l'ordre a de l'importance, c'est-à-dire *(a, b)* et *(b, a)* sont différents. Les éléments de *E* constituent les arcs de *G*.

Pour un arc orienté *(a, b),* "*a*" est la source ou l'origine, et "*b*" est l'arrivée ou le but. Dans un graphe non orienté «undirected graph», l'ordre n'a pas d'importance, la paire *(a, b)* joue le même rôle que la paire *(b, a).*

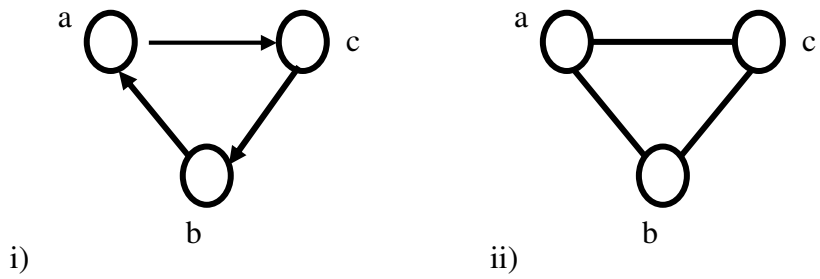


Fig. 1.1 : i) Un graphe orienté, ii) Un graphe non orienté.

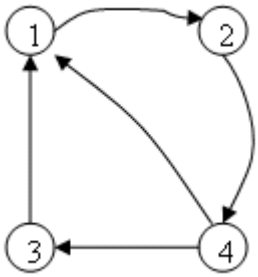
Deux représentations différentes peuvent être employées pour décrire des graphes : la matrice d'adjacence et la liste d'adjacence.

[1] http://hypergeo.free.fr/article.php3?id_article=70

- *Représentation par matrice d'adjacence*

Dans cette représentation, le graphe est décrit par une matrice booléenne indexée en ligne et en colonne par les sommets du graphe. Pour deux sommets *u* et *v*, l'entrée (*u, v*) de la matrice détermine si l'arête (*u, v*) de *u* à *v* est présente dans le graphe.

Exemple :



	1	*2*	*3*	*4*
1	0	**1**	0	0
2	0	0	0	**1**
3	**1**	0	0	0
4	**1**	0	**1**	0

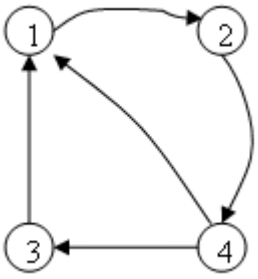
Fig. 1.2 : Un graphe et sa matrice d'adjacence.

Avec une représentation d'un graphe par matrice d'adjacence, le calcul des chemins dans le graphe se résume à un calcul de puissances successives de la matrice d'adjacence. Notons, également qu'il est possible d'utiliser plusieurs matrices d'adjacence non booléennes pour représenter un graphe attribué.

- *Représentation par listes d'adjacence*

Dans cette représentation, chaque sommet contient une liste des sommets qui lui sont adjacents.

Exemple :



1 : (1, 2)
2 : (2, 4)
3 : (3, 1)
4 : (4, 3), (4, 1)

Fig. 1.3 : Un graphe et ses listes d'adjacence.

1.2 Rappels terminologiques

1.2.1 Sous-graphe et graphe partiel

Pour un sous-ensemble de sommets A inclus dans V, le sous-graphe de G induit par A est le graphe $G' = (A, E(A))$ dont l'ensemble des sommets est A et l'ensemble des arcs $E(A)$ est formé de tous les arcs de G ayant leurs deux extrémités dans A. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que tous les arcs incidents à ces sommets.

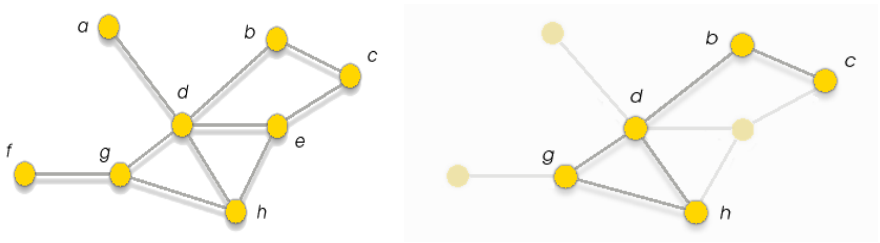


Fig. 1.4 : Un exemple de graphe (à gauche) et de sous-graphe (à droite). [2]

Soit $G = (V, E)$ un graphe, le graphe $G' = (V, E')$ est un graphe partiel de G, si E' est inclus dans E. Autrement dit, on obtient G' en enlevant un ou plusieurs arcs au graphe G.

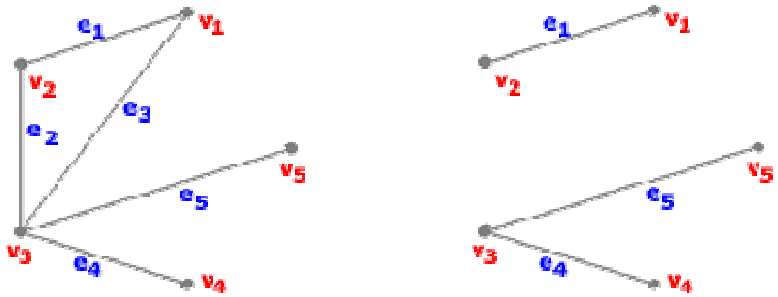


Fig. 1.5 : Le graphe G (à gauche) et un graphe partiel de G (à droite). [2]

[2] http://apprendre-en-ligne.net/graphes/graphes/sousgraphe.html

1.2.2 Graphe complet et clique

Un graphe complet est un graphe dont les sommets sont tous reliés deux à deux par un arc. Dans un graphe *G*, on nomme clique un sous-graphe complet de *G*, c'est-à-dire une partie de l'ensemble des sommets de *G* telle que le graphe induit par *G* sur cette partie soit un graphe complet. Un des problèmes centraux de la théorie des graphes consiste à chercher la clique de taille maximum dans un graphe. Un graphe complet à *n* sommets contient $n.(n-1)/2$ arcs. On note K_n le graphe complet d'ordre *n*, c'est-à-dire contenant *n* sommets[3].

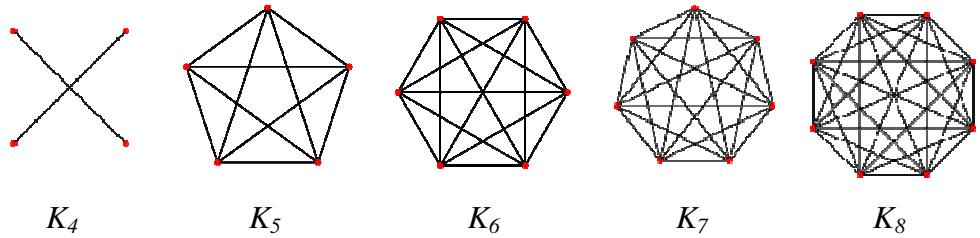


Fig. 1.6 : Quelques exemples de graphes complets.

Pour un graphe d'ordre *n*, il existe deux cas extrêmes pour l'ensemble de ses arcs : soit le graphe n'a aucun arc, soit tous les arcs possibles pouvant relier les sommets *2* à *2* sont présents (graphe complet). L'ordre de la plus grande clique de *G* est noté $\omega(G)$. Dans le graphe présenté figure 1.7, il y a deux cliques d'ordre 3 définies par les ensembles de sommets *{d, g, h}* et *{d, e, h}*

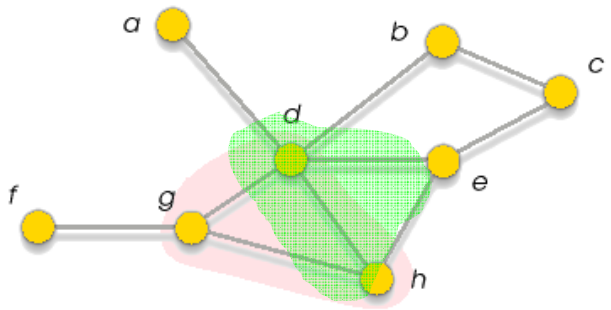
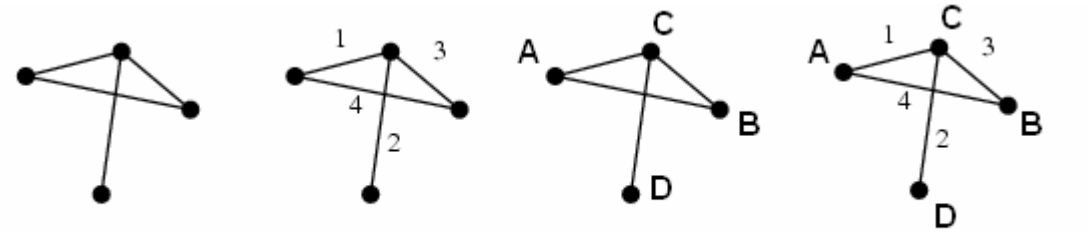


Fig. 1.7 : La clique {*d, g, h*} est représentée en rose et *{d, e, h}* en vert[4].

[3] http://fr.wikipedia.org/wiki/Graphe_complet
[4] http://gilco.inpg.fr/~rapine/Graphe/Graphe/clique.html

1.2.3 Graphe étiqueté et multi-étiqueté

Soit A_v et A_E respectivement les ensembles des étiquettes des sommets et des arcs. On peut alors définir un graphe multi-étiqueté G par un 4-tuple $G = (V, E, \alpha, \beta)$ où : V est l'ensemble fini des sommets, $E \subseteq V \times V$ est l'ensemble des arcs, $\alpha : V \rightarrow A_V^i$ est une fonction qui assigne les étiquettes aux sommets, $\beta : E \rightarrow A_E^j$ est une fonction qui assigne les étiquettes aux arcs. Ici, i varie de 1 à δ et j varie de 1 à Ω, où δ et Ω représentent respectivement le nombre d'étiquettes associées aux sommets et aux arcs. Ces étiquettes peuvent être des attributs symboliques (lettres, mots) ou des valeurs numériques.



Un graphe sans étiquette | Etiquettes sur les arcs | Etiquettes sur les sommets | Etiquettes sur les arcs et les sommets

Fig. 1.8: Différentes possibilités d'étiquetage d'un graphe.

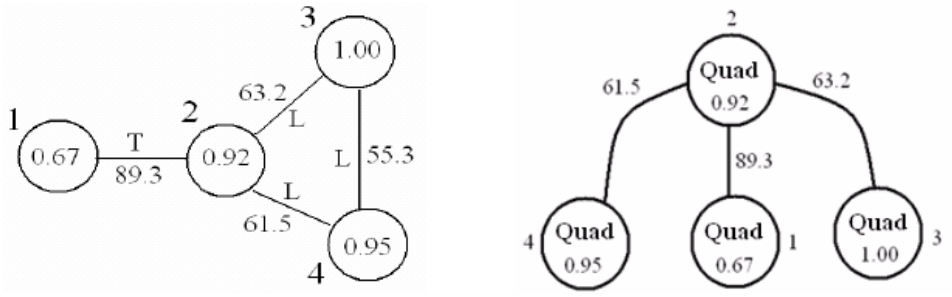


Fig. 1.9 : Exemples des graphes multi-étiquetés.

1.2.4 Graphe biparti

Un graphe est biparti si ses sommets peuvent être divisés en deux ensembles *X* et *Y*, de sorte que chaque arc du graphe relie un sommet dans *X* à un sommet dans *Y*. Dans l'exemple ci-dessous, un graphe *G = (V, E)* est décrit par l'ensemble de ses sommets *V = {1, 2, 3, 4, 5}* et l'ensemble de ses arcs *E = {(1,2), (1,4), (2,3), (2,5), (3,4), (4,5)}*, on a bien deux parties *X = {1, 3, 5}* et *Y = {2, 4}*, ou vice versa.

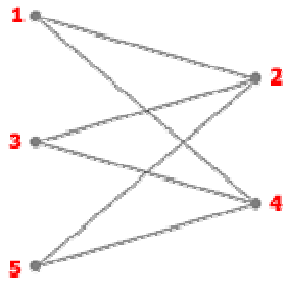


Fig. 1.10 : Un graphe biparti.

1.2.5 Graphe planaire

Dans la théorie des graphes, un graphe planaire est un graphe quelconque qui a la particularité de pouvoir se représenter sur un plan sans qu'aucun arc n'en croise une autre.

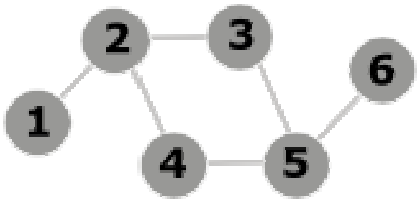


Fig. 1.11 : Un graphe planaire[5]

[5]http://www.techno-science.net/?onglet=glossaire&definition=6486

1.2.6 Graphe et arbres

Les arbres sont un cas particulier de graphes. Etant donné un graphe non orienté et connexe, un arbre couvrant ce graphe est un sous-ensemble qui est un arbre et qui connecte tous les sommets ensemble. Un graphe peut comporter plusieurs arbres couvrants différents. Lorsqu'on peut associer un poids à chaque arc, le poids de l'arbre couvrant correspond à la somme des poids des arcs de l'arbre couvrant. Un arbre couvrant de poids minimal (ACM) est un arbre couvrant dont le poids est inférieur ou égal à celui de tous les autres arbres couvrants du graphe. Un arbre couvrant de poids minimum est en général différent de l'arbre des plus courts chemins (SPT « Shortest Paths Tree ») construit, par exemple, avec l'algorithme de parcours en largeur (ou BFS, pour Breadth First Search[7]) ou avec l'algorithme de Dijsktra [Dijkstra, 1959]. Un arbre des plus courts chemins est bien un arbre couvrant, mais il minimise la distance de la racine à chaque sommet et non la somme des poids associés aux arcs[6].

Un arbre des plus courts chemins (SPT) de racine *A,* sa hauteur est *7*, son poids est *14*

Un arbre couvrant minimal (ACM) sa hauteur est *10*, son poids est *10*

Fig. 1.12 : L'arbre courant minimum (ACM) et le *Shortest Path Tree* (SPT)

1.3 Représentation d'images et de formes à base de graphes

Le graphe est un outil qui est utilisé de façon très diverse en traitement des images et en reconnaissance de formes. Leur utilisation remonte au début des années 70 pour l'interprétation automatique de dessins polygonaux [Pavliedis, 1972] et se poursuit aujourd'hui [Bunke, 1997a] [Dickinson, 2001] [Conte, 2004]. Le principe est généralement le suivant : la forme ou l'image est définie par un ensemble de primitives qui constituent les sommets du graphe. Des relations binaires de compatibilité entre primitives constituent les arcs du graphe.

[6]http://fr.wikipedia.org/wiki/Arbre_couvrant_de_poids_minimal
[7]http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/breadthSearch.htm

Nous montrons, au cours des sections suivantes, plusieurs représentations d'images et d'objets sous forme de graphes ainsi que leurs avantages et inconvénients. Nous essayons, en particulier de comprendre comment et pourquoi choisir une représentation plutôt qu'une autre en tenant compte des spécificités des domaines d'applications, des problèmes liés à la modélisation et des performances attendues.

1.3.1 Graphe de pixels

En traitement d'images, la représentation la plus couramment utilisée est le graphe de pixels. Chaque pixel représente un nœud du graphe et chaque arc est défini par la 4- ou la 8-connexité du pixel. Dans la technique du codage par arbre de prédiction, l'image est considérée comme un graphe dans lequel chaque pixel est relié à ses voisins. Il est donc nécessaire de disposer de la totalité de l'image avant d'envisager tout traitement. On peut ensuite chercher dans ce graphe un arbre recouvrant dont le codage est de taille minimale.

Récemment P. Franco [Franco, 2003] a utilisé cette approche pour la reconnaissance de caractères et de symboles graphiques. Cette approche est détaillée plus précisément dans le chapitre qui traite de la reconnaissance de symboles. La figure 1.13 illustre néanmoins ce type de représentation.

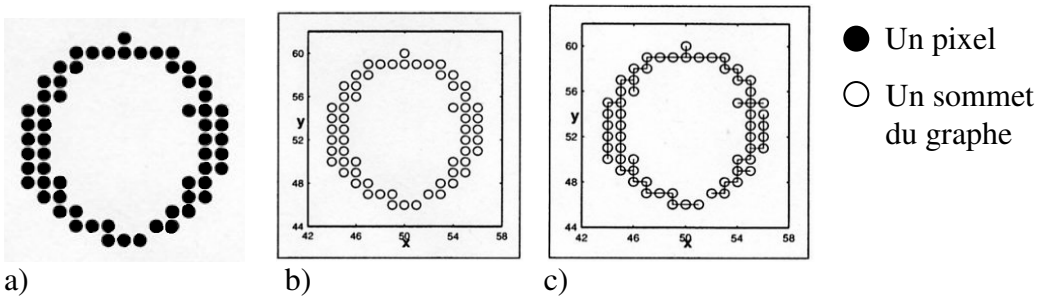


Fig. 1.13 : (a) Le caractère O (b) Le caractère *O* dans un repère (c) Un arbre de poids minimum (ACM).

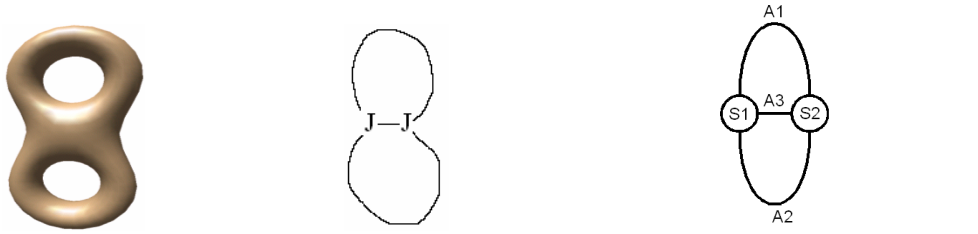
Ce type de graphe n'est qu'un changement de représentation des données de l'image et ne cherche pas à synthétiser l'information présente dans l'image. Ainsi, ces modèles non simplifiés sont souvent si gros que des algorithmes classiques de théorie des graphes ont un coût prohibitif qui ne permet pas leur utilisation dans les cas réels.

1.3.2 Graphe des points caractéristiques

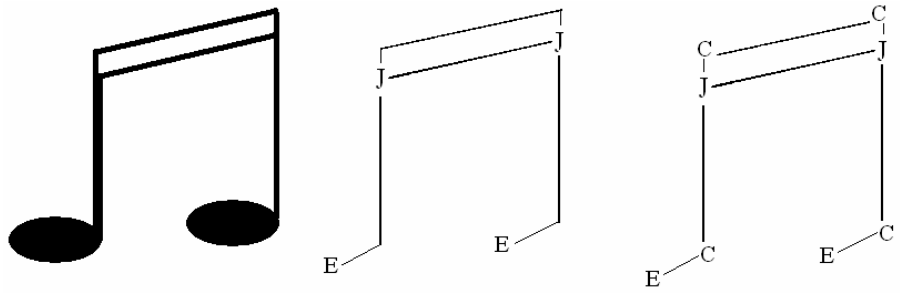
Comme les graphes de pixels sont bien souvent de tailles trop importantes pour être analysés, il est préférable de n'utiliser que certains points caractéristiques de la forme à décrire. Ainsi, certaines méthodes utilisent des outils de squelettisation pour définir les principales parties d'un objet. Ensuite, le graphe du squelette fait intervenir les éléments suivants :

- les extrémités (considérés comme des sommets)
- les jonctions (considérés comme des sommets)
- les branches du squelette (considérés comme des arcs)

Dans la figure 1.14a, le squelette de la forme présente les deux sommets S1 et S2 et les trois arcs A1, A2 et A3. Le squelette étant intimement lié à la forme d'origine, il est alors possible, à partir du squelette, de déterminer les principaux composants d'une forme et de guider la segmentation d'un maillage 3D [Brunner, 2004] ou la reconnaissance d'une forme.



a) Modèle 3D — Squelette — Graphe des points caractéristiques



b) Doubles croches

Graphe du squelette avant découpage récursif :
4 segments,
2 extrémités (E)
2 jonctions (J)

Graphe du squelette après découpage récursif :
8 segments,
2 extrémités (E)
2 jonctions (J)
4 coupures (C)

Fig. 1.14 : Exemples de squelettisation et d'utilisation de graphes (a)dans le cas de maillage ou de volume 3D (b) pour un symbole graphique 2D [Ramel, 1992].

Pour faciliter la recherche d'isomorphismes, Djamel Merad [Merad, 2004] propose de transformer les squelettes en graphes acycliques orientés. Dans sa méthode, un graphe acyclique non orienté est généré à partir du squelette (figure 1.15). Chaque nœud dans le graphe du squelette représente un ensemble de points du squelette possédant la même distance (appelée poids) au bord de l'objet. Les arcs représentent les chemins de connexion entre points du squelette. En appliquant un algorithme pour extraire l'arbre couvrant minimal (ACM), un arbre acyclique non orienté est obtenu. Pour construire une telle structure hiérarchique, les arcs sont orientés des nœuds à poids forts vers les nœuds de poids faibles. Le graphe orienté (figures 1.15.c et 1.15.d) est ainsi créé. La racine du graphe est le nœud qui a le plus fort poids. S'il existe plusieurs nœuds ayant le poids le plus fort, celui qui possède le plus de descendants directs sera choisi. La recherche d'isomorphismes peut ensuite être accélérée par associations de nœuds de niveau similaire dans les deux graphes en priorité.

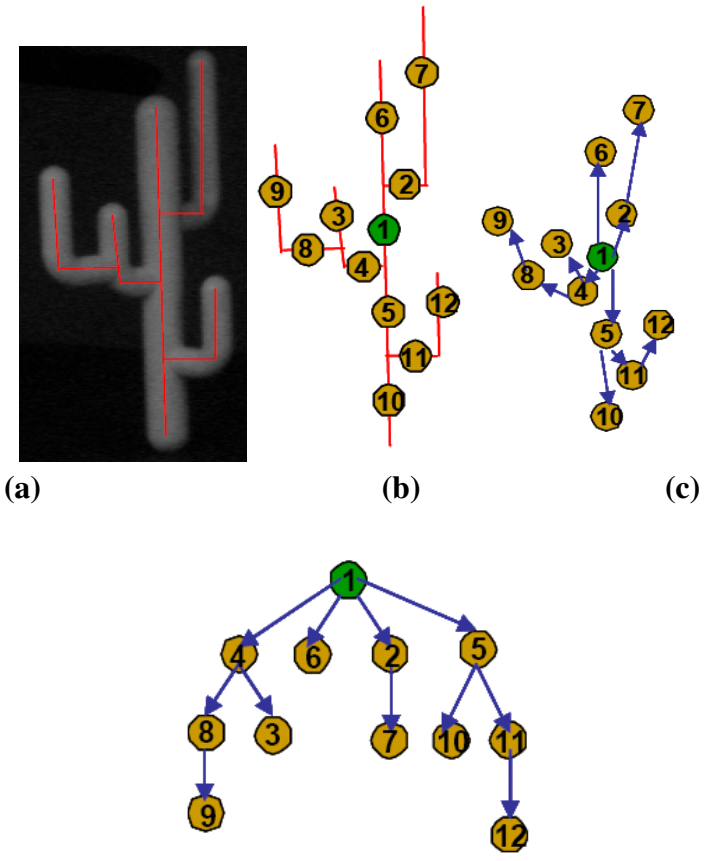


Fig. 1.15 : Construction de l'ACM à partir du graphe du squelette [Merad, 2004].

Notons que, les méthodes habituelles de construction des squelettes sont généralement assez rapides mais également instables car les informations stockées dans les petits segments ne sont pas nécessairement représentatives quand on considère des agrégats plus gros.

1.3.3 Graphe de primitives

Les représentations utilisant des graphes peuvent se baser sur des informations de plus haut niveau que les pixels ou que les points significatifs. Certains graphes représentent l'image sous forme d'une collection de primitives extraites des images comme par exemple les descriptions vectorielles dans le cas des documents graphiques. Généralement, les primitives correspondent aux sommets du graphe et les relations topologiques entre primitives correspondent aux arcs entre ces sommets. Les types de connexions (l'étiquette donnée aux arcs) varient selon l'application. Par exemple, [Liu, 2004] propose de représenter les caractères chinois par des graphes dans lesquels les sommets représentes les traits et les arcs les relations topologiques comme P, X, T ou L dépendant du type d'intersection entre les traits. Dans le graphe de vecteurs proposé par Han [Han, 1994], les sommets représentent les vecteurs liés aux contours et les arcs associés aux étiquettes *P* ou *A* précisent si les vecteurs sont parallèles ou adjacents (figure 1.18). Dans [Ramel, 2000] le système effectue, dans un premier temps, une vectorisation des contours afin d'obtenir un ensemble de quadrilatères représentatifs des formes fines présentes dans l'image. Les quadrilatères et leurs relations de voisinage sont utilisés pour construire un graphe structurel étiqueté. Dans cette représentation structurelle, les quadrilatères constituent les sommets du graphe tandis que les arcs décrivent la nature de la relation topologique (figure 1.16) entre primitives (comme parallèle, intersection, jonction, etc..). Pour reconnaître une entité, il est nécessaire de la comparer à des graphes de référence. Dans l'application décrite, la localisation des symboles consiste à trouver dans un graphe tous les sous-graphes qui correspondent à des modèles d'entités mécaniques par recherche d'isomorphismes. On verra que cette approche est assez robuste aux distorsions vectorielles, mais moins tolérante au bruit car le nombre de quadrilatères peut fluctuer. Cette approche a servi de base à nos travaux de recherche.

De la même façon, des graphes de traits sont souvent utilisés pour la représentation des caractères (principalement chinois et japonais). Les nœuds représentent les tracés constituant un caractère et les arcs représentent les relations topologiques entre les traits (figure 1.17).

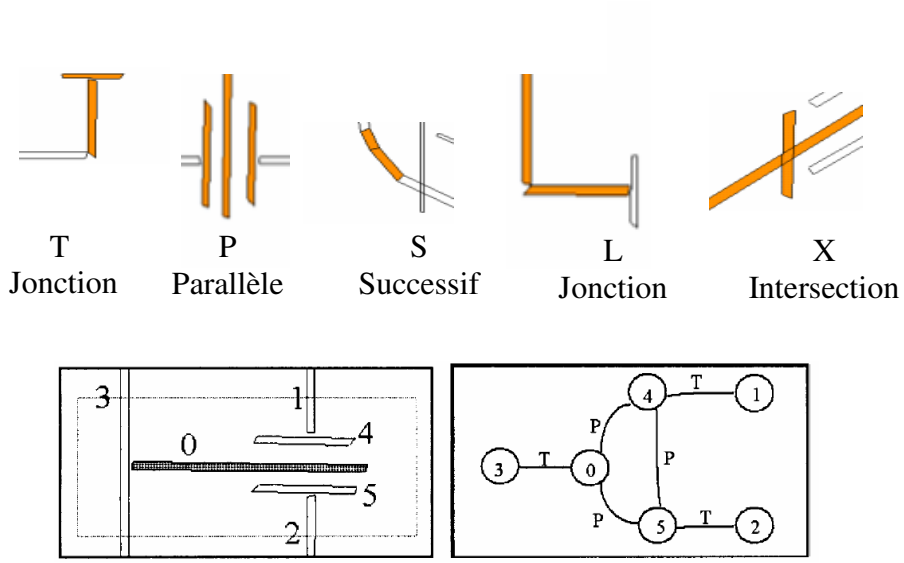


Fig. 1.16 : Exemples de relations topologiques entre primitives.

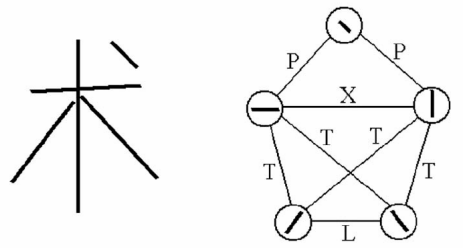


Fig. 1.17 : Représentation d'un caractère chinois basée sur un graphe étiqueté [Liu, 2004].

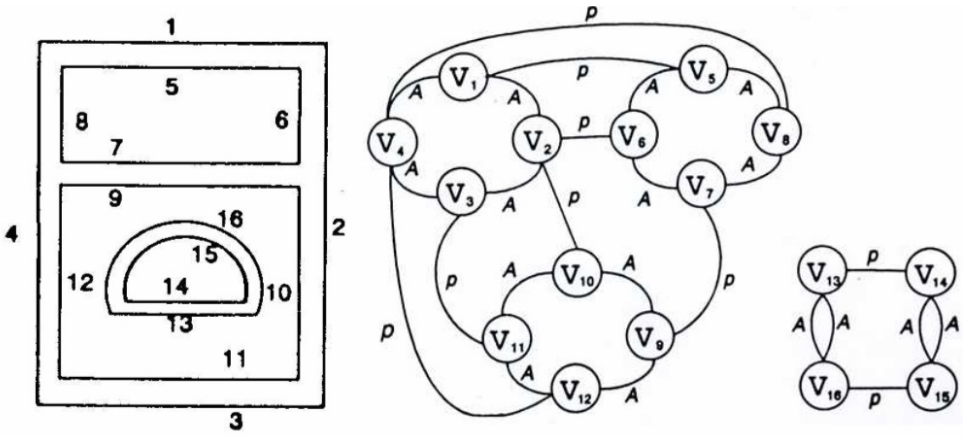


Fig. 1.18: Le graphe de vecteurs [Han, 1994].

1.3.4 Graphe d'adjacence de régions

La description du contenu d'une image peut se faire par une représentation topologique dans laquelle les relations contextuelles tiennent une grande importance. Quand l'objectif est de segmenter l'image en un nombre restreint de régions mais néanmoins suffisant pour que chaque zone soit bien représentative des pixels qui la composent, deux approches à base de graphes existent. Les relations de voisinages entre régions peuvent être des relations d'inclusion entre régions et sous-régions, des relations entre régions de même niveau ou une combinaison des deux. Par exemple, une représentation plus compacte de l'image est obtenue avec l'aide de graphes d'adjacence de régions (GAR) construite à partir d'un découpage en zones homogènes de l'image. Dans ce cas, on associe souvent aux nœuds représentant les régions des attributs comme le niveau de gris moyen, la surface, la taille, des indices texturaux, etc. Quant aux arcs entre deux régions adjacentes, on leur associe la longueur du contour commun ou une mesure de contraste, ou une information de position (dessus, à l'intérieur de, à côté de ...).

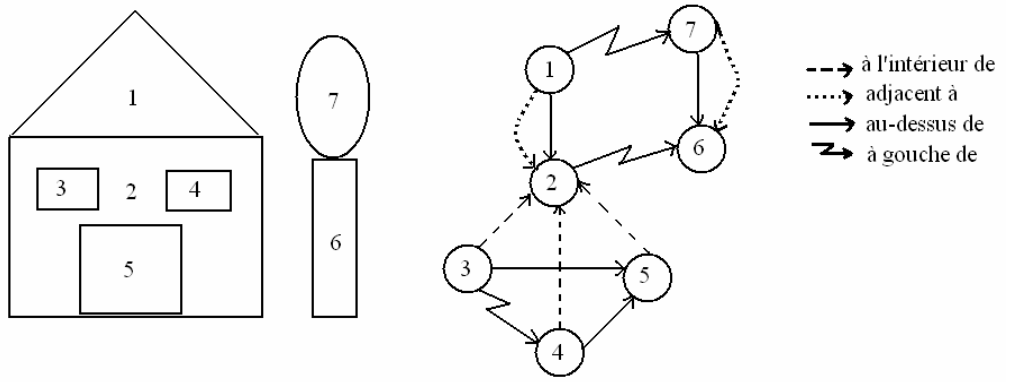


Fig. 1.19 : Un exemple de graphe d'adjacence de régions extrait de [Ramel, 1992].

Pour obtenir ce type de graphes, les méthodes existantes proposent souvent de partir d'un état initial où chaque pixel de l'image constitue une région élémentaire, puis de fusionner progressivement les régions connexes qui satisfont un certain critère d'homogénéité [Haris, 1998]. Comme exemple de critère simple, on peut considérer l'écart entre les valeurs moyennes des niveaux de gris des pixels dans les deux régions. Si cet écart est inférieur à un certain seuil, on peut fusionner les deux régions. Il convient néanmoins de faire attention car en réduisant le nombre de régions, on accroit la probabilité de fusionner des régions sans réelle correspondance ce qui fait perdre de la pertinence au modèle. Le résultat est une image où tous les pixels d'une région sont

marqués de la même valeur [Hlaoui, 2003]. Les arcs représentent les relations d'adjacence possibles entre les régions ainsi obtenues.

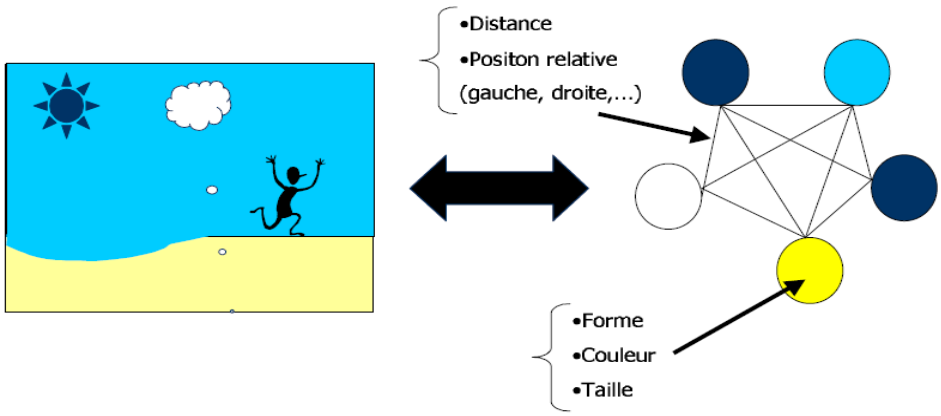


Fig. 1.20 : Un graphe attribué qui représente les relations entre les régions de l'image [Hlaoui, 2003].

Les GAR fournissent une description spatiale des données qui se prête intuitivement aux primitives graphiques. Les primitives graphiques sont les objets graphiques élémentaires manipulés dans les systèmes d'analyse de documents. Le problème est alors d'extraire un type de primitives graphiques donné, dans des conditions données. Par exemple, la figure1.21 est basée sur des primitives graphiques polygones décrivant un Clip Art. Cependant, il est difficile de construire des critères et mesures pour la détermination des régions constituant les objets.

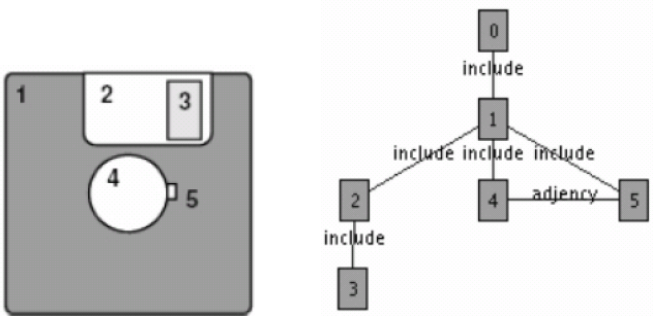
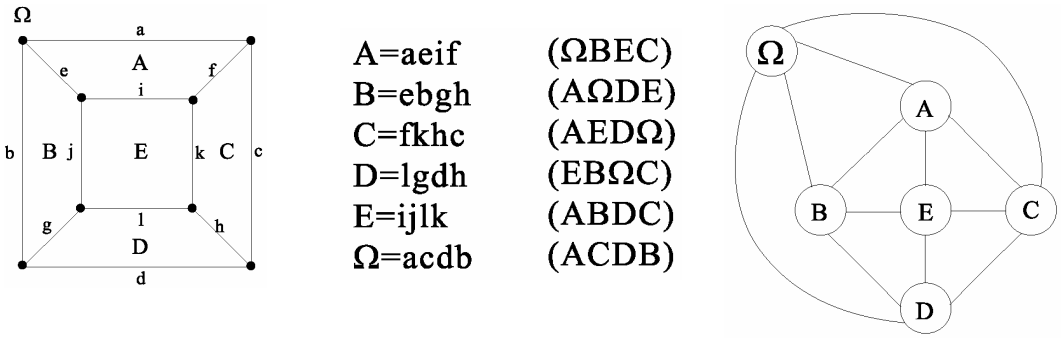


Fig. 1.21 : Graphe de polygones [Fonseca, 2004].

Joseph Llados [Lladós, 2001a] propose d'utiliser des GAR pour décrire les schémas techniques. Dans un premier temps, un graphe de points caractéristiques (jonctions, extrémités, …) est construit. Les sommets sont les points caractéristiques et les arcs correspondent aux traits entre ces points. Le graphe de régions est alors généré en cherchant les boucles dans le 1er graphe, un GAR est ainsi obtenu dans lequel les arcs représentent les relations d'adjacence entre les boucle (régions).



(a) (b) (c)

Fig. 1.22 : a) Graphe planaire b) chaînes de caractères représentant les régions c) graphe d'adjacence de régions.

La figure 1.22 montre un exemple de GAR construit à partir du graphe planaire. Le graphe original possède 5 régions (A, B, C, D, et E) et une région particulière : la région externe qui est infinie notée par Ω. Une chaîne de caractères décrivant les contours est ensuite associée à chaque nœud. La technique semble sensible au bruit. En cas de région non-fermée, l'auteur propose d'ajouter quelques pixels (pont de pixels) entre les deux points à connecter, mais cet artifice ne répond que partiellement au problème.

Récemment, Mathieu Delalandre [Delalandre, 2004] a présenté une méthode de construction de graphes hybrides (i.e. graphes d'inclusions et de voisinages). Cette représentation utilise tout d'abord des méthodes de marquage de composantes connexes et d'extraction d'occlusions. Les composantes et occlusions constituent les primitives de description permettent la construction de différents types de graphes : graphes d'inclusions et graphes de voisinage sous contrainte de distance. Ces graphes sont fusionnés lors de la génération d'un graphe hybride.

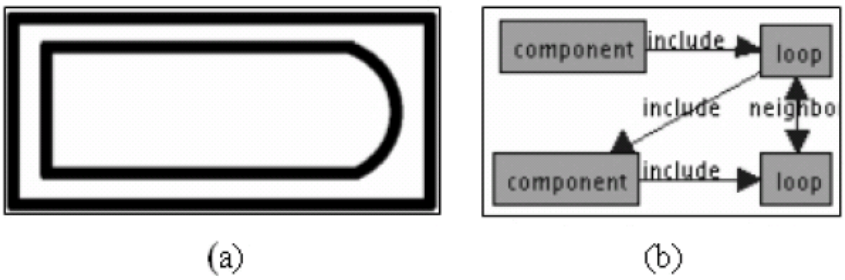


Fig. 1.23 : (a) Symbole et (b) graphe hybride associé [Delalandre, 2005].

Il existe bien d'autres méthodes de représentations utilisant les graphes et il est donc impossible d'être exhaustif. Il nous paraît cependant important de rappeler l'importance que revêt le choix du mode de représentation utilisé vis-à-vis de la généricité du système que l'on désire. Le choix des attributs (numériques et/ou symboliques) à associer aux arcs et aux nœuds, leur nombre influe énormément, tout autant que la densité (nombre d'arcs et de sommets), sur les performances qui seront obtenues quels que soient les outils qui pourront être utilisés par la suite pour manipuler, analyser ou comparer ces graphes.

1.4 Méthodes de mise en correspondance exacte de graphes

Il y a deux grandes catégories de méthodes d'appariement de graphes : les méthodes qui fonctionnent selon une approche symbolique (comme par exemple la recherche d'isomorphisme exact entre graphes ou entre sous-graphes) et les méthodes de comparaison de graphes avec tolérance d'erreur plutôt adaptées aux approches numériques. La plupart du temps, les méthodes symboliques utilisent soit un arbre de recherche avec retour-arrières « *backtracking* », soient les matrices d'adjacence des graphes. La deuxième catégorie inclut les méthodes dans lesquelles une fonction d'énergie est minimisée avec quelques heuristiques pour arriver à la solution. Tous les algorithmes d'appariement de graphes de la première catégorie sont assez performants en termes de résultats fournis mais partagent le même problème d'explosion combinatoire de l'espace de recherche (un problème *NP*-complet au pire). La deuxième catégorie propose des solutions plus approximatives mais en des temps plus raisonnables.

1.4.1 Isomorphisme de graphes et de sous-graphes

Etant donné deux graphes : un graphe modèle G_M et un graphe de données G_D, la procédure de comparaison implique la vérification de leur ressemblance. Généralement, on peut définir le problème d'appariement de graphes comme suit : étant donné deux graphes $G_M = (V_M, E_M)$ et $G_D = (V_D, E_D)$, avec $|V_M| = |V_D|$, le problème est de trouver une bijection $f : V_D \rightarrow V_M$ telle que si $(u, v) \in E_D$ alors $(f(u), f(v)) \in E_M$. Si une certaine correspondance f existe, on l'appelle un isomorphisme, et G_D est dit isomorphe à G_M.

Un sous-type important de ce problème est l'appariement de sous-graphes, dans lequel nous avons deux graphes $G = (V, E)$ et $G' = (V', E')$, tels que $V' \subset V$ et $E' \subseteq E$, dans ce cas le but est de trouver une correspondance $f' : V' \rightarrow V$ telle que si $(u, v) \in E'$ alors $(f'(u), f'(v)) \in E$. Si une telle correspondance existe, elle est appelée isomorphisme de sous-graphes.

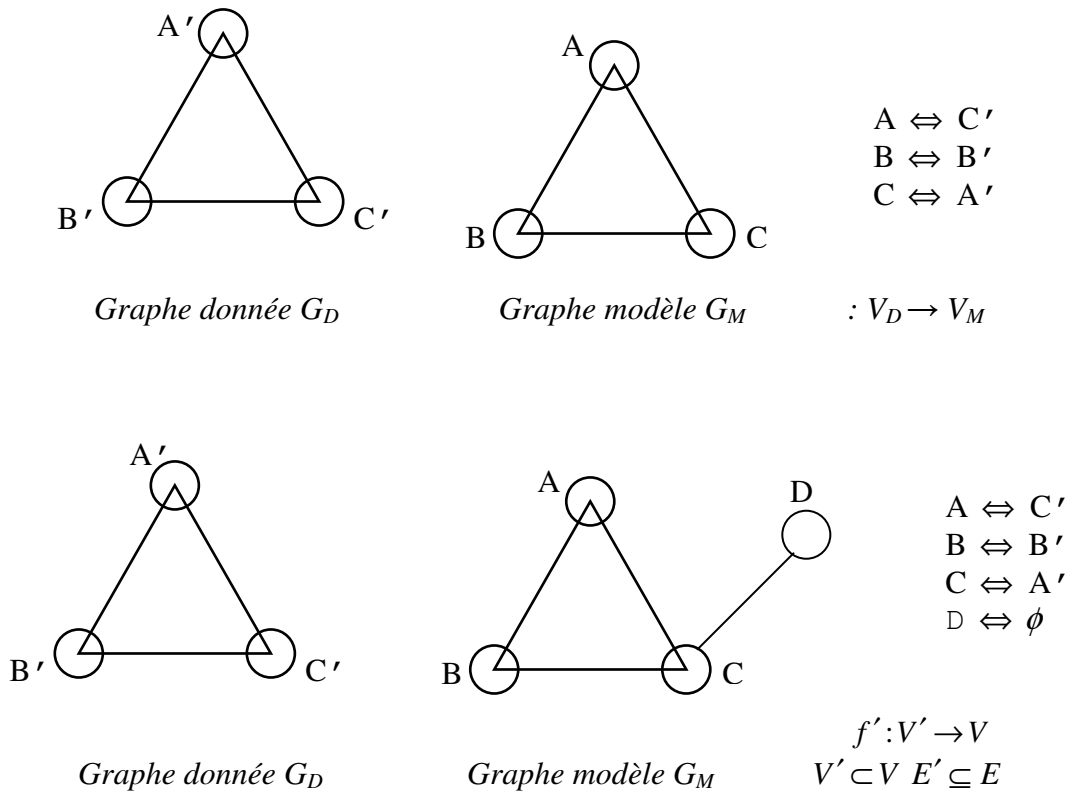


Fig. 1.24 : Un exemple d'isomorphisme de graphes et d'isomorphisme de sous-graphes.

La complexité théorique de la recherche d'isomorphismes de graphes n'est, à ce jour, pas complètement établie. Il n'a jamais été montré que le problème appartenait à la classe des problèmes *NP*-complet mais aucun algorithme de résolution de complexité polynomiale n'a été trouvé [Garey, 1979]. Il existe des algorithmes polynomiaux pour certains graphes particuliers. Par contre, le problème de la recherche d'un isomorphisme de sous-graphes est un problème *NP*-complet [Alt, 1984] et exponentiel pour les graphes généraux [Messmer, 1995]. Lorsque les graphes sont planaires, la complexité devient polynomiale (cette situation est assez fréquente en traitement d'images).

Il existe deux grands algorithmes de recherche fréquemment utilisés présentés dans les sections suivantes.

- **Résolution à partir du graphe d'association**

La première solution consiste à rechercher des cliques maximales dans un graphe d'association G_A. Etant donné deux graphes $G_M = (V_M, E_M)$ et $G_D = (V_D, E_D)$, le graphe d'association $G_A (V_A, E_A)$ est construit de sorte que chaque sommet $v_A \in V_A$ corresponde à l'association entre deux sommets $v_M \in V_M$ et $v_D \in V_D$ possédant les mêmes attributs et que chaque arc $E_A \subseteq V_A \times V_A$ corresponde à l'arc entre ces sommets compatibles (i.e., reliés dans les graphes d'origine G_M et G_D par des arcs identiques). Formellement :

$V_A = \{(v_M, v_D) \mid v_M \in V_M, v_D \in V_D$ et *étiquette*(v_M) = *étiquette*$(v_D)\}$

$E_A =$ consiste en l'ensemble d'arcs e_A, $e_A = (v_A, v'_A)$ avec
$v_A = (v_M, v_D)$, $v'_A = (v'_M, v'_D)$ tel que :

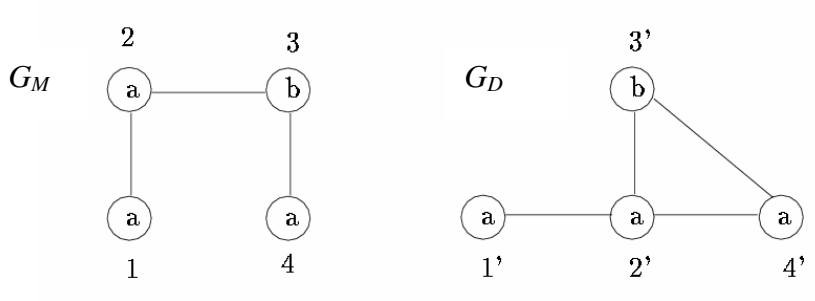
a) $v_M \neq v'_M$ et $v_D \neq v'_D$

b) s'il existe un arc $e_M = (v_M, v'_M) \in E_M$ alors il faut qu'il existe un arc
$e_D = (v_D, v'_D) \in E_D$, *étiquette*$(e_M)$= *étiquette*$(e_D)$

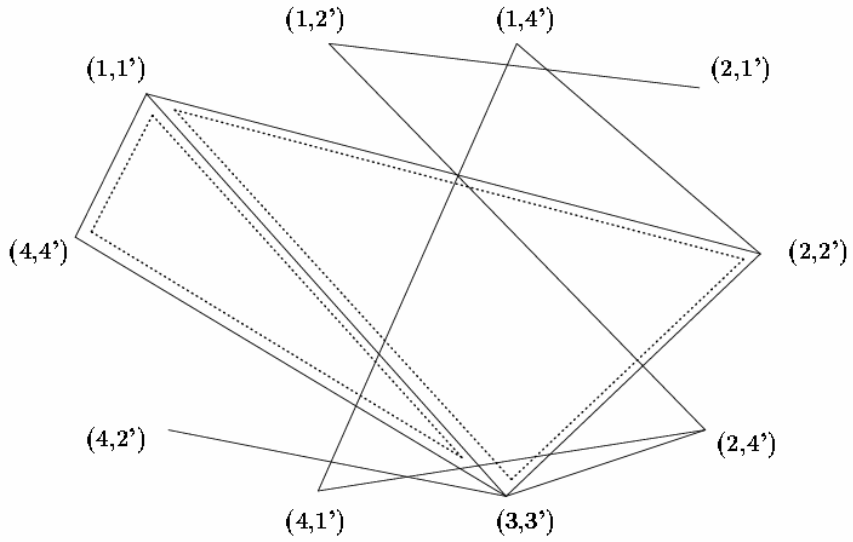
c) s'il n'y a pas un arc $(v_M, v'_M) \in E_M$ alors il faut qu'il n'existe pas un arc $(v_D, v'_D) \in E_D$

Chaque clique du graphe d'association correspond à un isomorphisme de sous-graphe. On recherche dans G_A la plus grande ou la "meilleure" clique maximale au sens d'un critère donné. L'approche est utilisée par [Falkenhainer, 1990] et [Myaeng, 1992].

La figure.1.25 montre le graphe d'association construit à partir de deux graphes G_M et G_D (Fig. 1.25a). Il y a deux cliques maximales, chacune constituée de 3 sommets, C_1 [(1, 1'), (2, 2'), (3, 3')] et C_2 [(1, 1'), (3, 3'), (4, 4')] représentée par les traits en pointillés dans la figure 1.25b.



a) Deux graphes G_M, G_D



b) Le graphe d'association pour G_M et G_D

Fig. 1.25: Recherche d'isomorphisme à partir du graphe d'association [Messmer, 1995].

Le problème principal de la mise en correspondance de graphes par détection de cliques maximales est sa complexité en temps. Pour deux graphes G_M et G_D avec n et m sommets respectivement, la taille du graphe d'association dépend fortement au nombre d'étiquettes dans G_M et G_D . Si $n \leq m$ et les étiquettes dans les deux graphes sont différentes, ce qui correspond aux meilleurs cas, la complexité sera seulement en $O(nm+n^2)$. Cependant, dans le pire des cas, quand les étiquettes dans les deux graphes sont identiques, la complexité augmente alors jusqu'à $O((nm)^n)$ [Messmer, 1995].

- **Résolution à partir d'arbre de recherche :**
  Algorithmes « Backtrack-Search (BT) » et «Forward-Checking (FC) »

Concernant les algorithmes basés sur la technique de parcours d'arbres de recherche, celui proposé par Corneil et Gotlieb[Corneil, 1970] est considéré comme l'un des premiers détaillé dans la littérature. Cet algorithme a été utilisé dans le domaine de la chimie pour détecter des similarités entre différentes formules chimiques. L'approche est basée sur un algorithme nommé « *depth-first backtracking search* ». Etant donné deux graphes G_M et G_D, les sommets de G_M sont mis en correspondance avec les sommets de G_D, itérativement, sous la condition que la structure des arcs de G_M soit préservée dans G_D durant l'appariement des sommets. Donc, si le nombre de sommets dans les deux graphes est identique et si tous les sommets ont été mis en correspondance avec succès, il existe alors un isomorphisme entre ces deux graphes. La limite de cette méthode est liée au fait que la recherche dans le graphe est exhaustive (recherche par force brute). Donc, cet algorithme est très coûteux: $O(N^4)$.

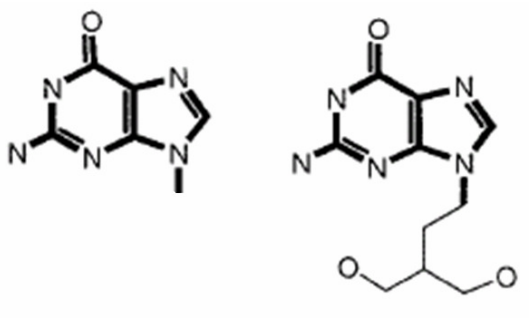


Fig. 1.26 : Les structures chimiques comme exemple d'isomorphisme de graphe/ sous-graphe [Raymond, 2002].

Quelques années plus tard, Ullman [Ullman, 1976] a amélioré cette méthode. L'algorithme d'Ullman est basé sur la procédure récursive de backtracking en combinaison avec une procédure de raffinement de l'espace de recherche (la routine *forward checking*). L'entrée de l'algorithme est composée d'un graphe modèle $G_M = (V_M, E_M)$ et d'un graphe de données $G_D = (V_D, E_D)$.

L'algorithme d'Ullman cherche à trouver tous les isomorphismes de sous-graphes possibles du graphe G_D à partir du graphe G_M. Pour réduire le nombre d'appariements à tester, une approche d'appariement incrémentale des sommets a été proposée. A chaque étape, un couple de sommets est ajouté dans l'appariement courant et la condition d'isomorphisme de sous-graphes est testée. Si, pour un certain couple de sommets

l'isomorphisme n'était pas trouvé, le système fait un retour en arrière « *backtrack* » et le sommet du graphe modèle utilisé précédemment va être mis en correspondance avec un autre sommet afin de tester une nouvelle condition d'isomorphisme. L'espace de recherche se réduit très efficacement grâce à la technique de « *forward checking* ». Elle a pour but de détecter par avance des instanciations inconsistantes en appliquant un critère de test de consistance sur les arcs pendant la recherche.

La complexité de l'algorithme d'Ullman dépend de la taille des graphes. Supposons que la taille du graphe modèle G_M est n et celle du graphe G_D est m, le coût de l'algorithme dans le meilleur des cas est $O(mn)$. Dans le pire des cas, lorsque la connexité du graphe est très élevée, le coût augmente jusqu'a $O(m^n n^2)$. Par conséquent, le temps de traitement explose exponentiellement. La figure. 1.27 fournit l'algorithme d'Ullman.

Un graphe attribué G défini par un 4-tuple $G = (V, E, L_V, L_E)$ où : V est l'ensemble fini des sommets, $E \subseteq V \times V$ est l'ensemble des arcs, $L_V : V \to \sum_V \times A_V^k$ est une fonction qui assigne les étiquettes aux sommets, $L_E : E \to \sum_E \times A_E^l$ est une fonction qui assigne les étiquettes aux arcs.

Algorithme d'Ullman (G, G')

Entrée: deux graphes attribués $G = (V, E, L_V, L_E)$, $G' = (V', E', L'_V, L'_E)$.
Sortie: $f[1 \ldots |V|]$ vecteur représentant un isomorphisme de sous-graphes du G à G'.
/* $f[i] = j$ montre que le sommet $v_i \in V$ a été associé avec le sommet $w_j \in V'$ */

soit P = [1… |V|; 1…|V'|] une matrice de compatibilité entre le sommet du G et G'.
0. **début**
1. **pour** i = 1 à |V|
2. **pour** j = 1 à |V'|
3. **si** $L_V(v_i) = L'_V(w_j)$ **alors** $P[i, j] := 1$;
4. **sinon** $P[i, j] := 0$;
5. **pour** i = 1 à |V|
6. $f[i] := 0$;
7. **retourner** Backtracking(P, 1, f)
8. **fin.**

Algorithm Backtracking (P, i, f)
Entrée: Une matrice de compatibilité P entre les sommets de deux graphes G et G', le niveau courant i dans l'arbre de recherche, un vecteur f représentant l'isomorphisme partiel jusqu'au niveau i-1.

Sortie : l'isomorphisme partial f jusqu'au niveau i.

0. **début**
1. **si** $i > |V|$ **alors retourner** f comme un isomorphisme de sous graphe du G à G'.
2. **sinon**
3. **pour** $j = 1$ à $|V'|$
4. **si** $P[i, j] = 1$ **alors**
5. **début**
6. $f[i] := j$;
7. $P' := P$;
8. **pour** $k = i + 1$ à $|V|$ $P'[k, j] := 0$;
9. **si** Forward_checking(P', i , f) **alors**
retourner Backtracking(P', $i + 1$, f) ;
10. **sinon** $f[i] := 0$;
11. **fin.**
12. **fin.**

Algorithm Forward checking(P, i, f)
Entrée: Une matrice de compatibilité P entre les sommets de deux graphes G et G', le niveau courant i dans l'arbre de recherche, un vecteur f représentant l'isomorphisme partiel jusqu'au niveau i-1.

Sortie: VRAI si pour chaque sommet $v_k \in V$ ($k > i$) il existe au moins un appariement compatible avec appariements des sommets existants présents dans f, FAUX, sinon.

0. **début**
1. **pour** $k = i + 1$ à $|V|$
2. **pour** $j = 1$ à $|V'|$
3. **si** $P[k, j] = 1$ **alors**
4. **pour** $l = 1$ à i -1
5. **début**
6. vérifier les contraintes suivantes de consistance des arcs:
(1) pour chaque arc $e = (v_k, v_l) \in E$
existe-il un arc $e' = (w_j, w_{f[l]}) \in E'$ tel que $L_E(e) = L'_E(e')$
(2) pour chaque arc $e' = (w_j, w_{f[l]}) \in E'$
existe-il un arc $e = (v_k, v_l) \in E$ tel que $L_E(e) = L'_E(e')$
7. **si** (1) ou (2) sont FAUX **alors** $P[k, j] := 0$;

11. **fin**
12. **si** il existe une ligne k dans P telle que $P[k, j] = 0$ pour $j = 1 \ldots |V'|$ **alors** **retourner** FAUX;
13. **sinon retourner** VRAI;
12. **fin**.

Fig. 1.27 : Algorithme d'Ullman et la procédure Forward Checking [Lladós, 1997].

- **Résolution à partir de graphes ordonnés**

Un algorithme de recherche d'isomorphismes de graphes en temps quadratique a été proposé par X. Jiang et Horst Bunke [Jiang, 1999]. Les graphes ordonnés « *Ordered graphs* » sont représentés par des codes et l'isomorphisme est déterminé en comparant les codes des deux graphes. Un graphe ordonné, comme son nom l'indique, est constitué d'arcs qui peuvent être ordonnés pour chaque sommet. Selon l'auteur, les graphes planaires sont un exemple de graphes ordonnés.

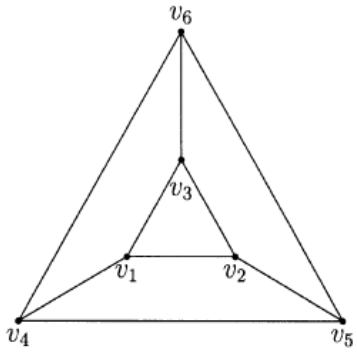


Fig. 1.28 : Un exemple de graphe planaire.

Définition 1. Un graphe ordonné est un triplet *(V, E, L)* où *(V, E)* définit un graphe. Pour chaque sommet $v_i \in V$, les arcs entrant dans v_i $(v_iv_1, v_iv_2, v_iv_3, \ldots, v_iv_k)$ peuvent être représentés par un cycle $L(v_i)$. Par exemple, les arcs connectés à v_1 dans la figure 1.28 correspondent au cycle (v_1v_2, v_1v_4, v_1v_3)

Les graphes ordonnés peuvent être représentés par des codes et l'isomorphisme est déterminé en comparant les codes des deux graphes. Ces codes sont générés à l'aide de circuits d'Euler. Il est nécessaire de transformer le graphe initial en un graphe d'Euler G_d. Pour un graphe G non-orienté, si tous les arcs sont doublés et si on donne un sens opposé à chaque arc dans chaque paire de sommets. Le graphe résultant est un graphe d'Euler G_d dans lequel chaque sommet possède le même nombre d'arcs entrants et sortants. Il est alors possible de construire un chemin cyclique passant par chaque arc de G_d une fois et seulement une fois. Un tel chemin correspond à un chemin dirigé cyclique dans le graphe G original qui traverse chaque arc une fois et seulement une fois dans chaque direction.

Quelques exemples des circuits Eulériens du graphe de la figure.1.28 sont fournis ci-dessous. Nous supposons que les arcs autour d'un sommet sont ordonnés dans le sens des aiguilles d'une montre « clockwise ».

$$P(v_1\, v_2) = v_1\ v_2\ v_3\ v_1\ v_3\ v_6\ v_4\ v_1\ v_4\ v_5\ v_2\ v_5\ v_6\ v_5\ v_4\ v_6\ v_3\ v_2\ v_1$$
$$P(v_2\, v_1) = v_2\ v_1\ v_4\ v_5\ v_2\ v_5\ v_6\ v_3\ v_2\ v_3\ v_1\ v_3\ v_6\ v_4\ v_6\ v_5\ v_4\ v_1\ v_2$$
$$P(v_2\, v_3) = v_2\ v_3\ v_1\ v_2\ v_1\ v_4\ v_5\ v_2\ v_5\ v_6\ v_3\ v_6\ v_4\ v_6\ v_5\ v_4\ v_1\ v_3\ v_2$$

Ici, $P(v_i\ v_j)$ dénote le circuit d'Euler obtenu, à partir de l'arc v_iv_j (de v_i vers v_j).

Un procédé de numérotation des sommets du circuit est ajouté de sorte qu'un code soit obtenu. La règle de codage est comme suit : l'étiquette du sommet initial d'un arc initial est 1, l'étiquette de son sommet terminal est 2, l'étiquette du prochain sommet est 3, et ainsi de suite. L'idée principale est d'effectuer un re-étiquetage des sommets dans l'ordre de rencontre pour la première fois. Le code est ainsi donné par un vecteur de la dimension $2m+1$, où m est le nombre d'arcs dans le graphe. Pour les circuits d'Euler donnés ci-dessus, les codes correspondants sont :

$C(v_1\ v_2) = C(v_2\ v_3) =$ 1231345156264654321
$C(v_2\ v_1) =$ 1234145616265654321

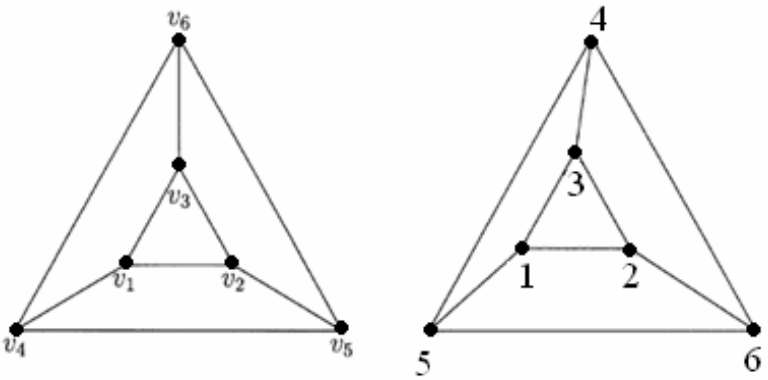


Fig. 1.29 : Un graphe ordonné (à gauche) et sa version ré-étiquetée C (v_1 v_2) (à droite).

A partir de ce codage, un algorithme simple pour déterminer tous les isomorphismes entre les deux graphes ordonnés G_1 et G_2 est fourni ci-dessous (figure 1.30):

1. Choisir aléatoirement un arc $v_i\, v_j$ du G_1 et calculer $P(v_i\, v_j)$ et $C(v_i\, v_j)$
2. pour chaque arc $v'_i\, v'_j$ du G_2 faire :
   - *2.1* Calculer $P(v'_i\, v'_j)$ *et* $C(v'_i\, v'_j)$.
     Si $C(v_i\, v_j) \neq C(v'_i\, v'_j)$, refaire étape-2 avec l'arc suivant de G_2.
   - 2.2 /* Isomorphisme existe */
     Vérifier, si l'ordre est préservé dans cet isomorphisme
     Si, le test est vrai, alors il y a un isomorphisme ordonné
     entre G_1 *et* G_2 qui associe chaque sommet de P $(v_i v_j)$
     avec le sommet correspondant dans $P(v'_i\, v'_j)$.

Fig. 1.30 : Pseudo-code pour déterminer tous les isomorphismes entre deux graphes ordonnés ($P(x, y)$ renvoie le circuit eulérien, $C(x, y)$ renvoie la chaîne associée).

Si le nombre de sommets pendant la génération de $C(v'_i\, v'_j)$ n'est pas identique au nombre de sommets dans $C(v_i\, v_j)$, la procédure de codage est immédiatement arrêtée et on recommence avec l'arc suivant dans G_2. Cet algorithme fonctionnant en temps quadratique [Jiang, 1999] peut être comparé à celui d'Ullmann [Ullman, 1976]. La figure. 1.31 illustre les différences sur 50 paires de graphes aléatoires $G(n, m)$, avec n sommets et m arcs.

Lorsque le graphe devient de taille importante $(m \geq 230)$, le temps de calcul exigé par la méthode d'Ullmann [Ullman, 1976] augmente exponentiellement tandis que l'algorithme de comparaison de graphes ordonnés reste stable. Évidemment, le code dépend du choix de l'arc pris comme point de départ. Par conséquent, un graphe ordonné ne peut pas être représenté par un code unique, mais il y a $2m$ codes possible. Cependant, les auteurs spécifient quelques conditions à vérifier pour savoir quelle chaine (parmi les $2m+1$ chaines) peut être qualifiée de vrai code.

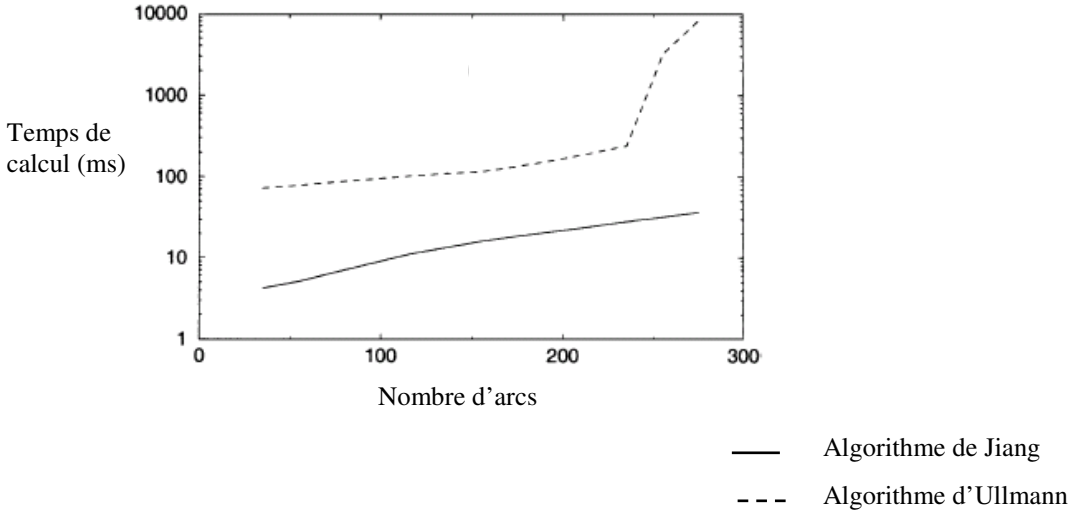


Fig. 1.31 : Comparaison des temps de calcul pour Jiang et Ullmann.

- **Résolution à partir d'arbres de décision est des matrices d'adjacence**

Messmer a proposé un algorithme impressionnant pour la détection d'isomorphismes de graphe/sous-graphe [Messmer, 1995] en utilisant une approche basée sur les arbres de décision et les matrices d'adjacence.

Soit G = $(V, E, \mu, \nu, L_\nu, L_e)$

- V l'ensemble de sommets
- $E \subseteq V \times V$ l'ensemble des arcs
- $\mu : V \rightarrow L_\nu$ est une fonction qui associe les étiquettes aux sommets
- $\nu : E \rightarrow L_e$ est une fonction qui associe les étiquettes aux arcs
- L_ν, L_e sont des ensembles finis d'étiquettes symboliques.

Si un graphe contient n sommets, $V = \{v_1, v_2, \ldots, v_n\}$, on peut représenter le graphe par sa matrice d'adjacence $M = (m_{i,j})$, $i, j = 1, \ldots, n$, où $m_{ii} = \mu(v_i)$ et $m_{ij} = v((v_i, v_j))$ pour $i \neq j$. La matrice d'adjacence M d'un graphe G n'est pas unique, et toute permutation de M représente également une matrice d'adjacences valide de G. Par exemple, A, B, C, D, E, et F dans la figure. 1.32 sont les 6 matrices d'adjacence possibles pour le graphe G.

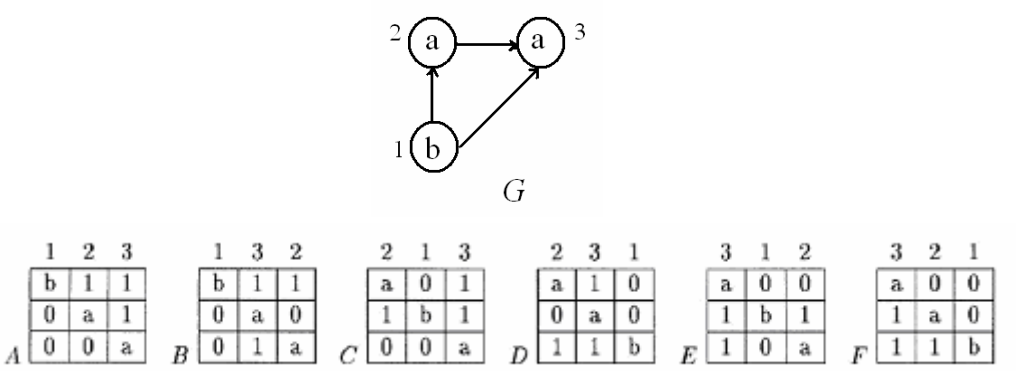


Fig. 1.32 : Un graphe et ses matrices d'adjacence.

Soit G_1 et G_2 deux graphes et M_1 et M_2 leur matrice d'adjacence respectivement, G_1 et G_2 sont isomorphes s'il existe une matrice P (une permutation de leur matrice d'adjacence) telle que :

$$M_2 = P\, M_1\, P^T \qquad \text{où } P^T \text{ est la transposée de la matrice } P$$

Pour chaque graphe modèle G_M toutes les matrices d'adjacence (ensemble de permutations de la matrice d'adjacence) sont générées et représentées par un arbre de décision (représentation ligne-colonne de la matrice d'adjacence). Par exemple, dans la figure 1.33, les matrices d'adjacence de deux graphes modèles G_1 et G_2 sont représentés par un arbre de décision commun.

Pour comparer un graphe aux graphes modèles, on utilise alors la matrice d'adjacence du graphe à tester en essayant de parcourir un chemin contenant des étiquettes identiques dans l'arbre. Les matrices de permutations qui correspondent au chemin trouvé dans l'arbre représentent l'isomorphisme.

Cette approche a une complexité quadratique en temps par rapport au nombre de sommets dans les graphes en entrée. Néanmoins, la taille de l'arbre de décision augmente exponentiellement avec la taille des graphes modèles. L'implémentation actuelle décrite par l'auteur montre que la méthode ne peut supporter que des petits graphes avec un nombre réduit de modèles. [Bunke, 2000a].

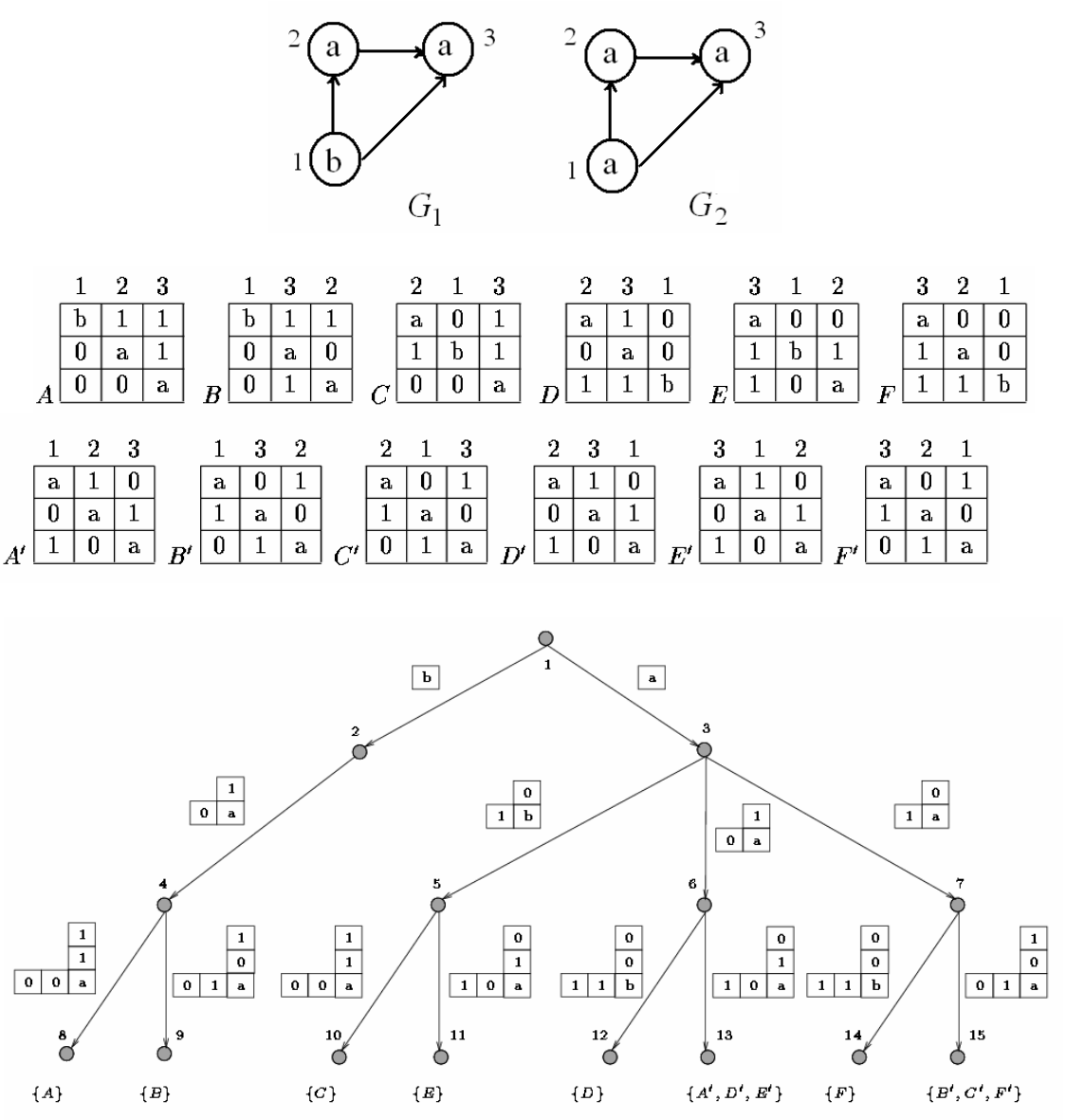


Fig. 1.33 : Arbre de décision construit à partir des matrices d'adjacence de deux graphes modèles G_1 et G_2 (représentation ligne-colonne) [Messmer, 1995].

1.4.2 Méthodes à base de décomposition des graphes

Lorsqu'un problème est complexe, il est souvent nécessaire de le décomposer en plusieurs parties, puis de résoudre ces parties de façon plus ou moins indépendante et enfin de les recombiner. Comme il a été prouvé que le temps pour trouver un isomorphisme de sous-graphe augmente de façon exponentielle en fonction du nombre de sommets dans les deux graphes, la communauté scientifique a essayé de décomposer le graphe en morceaux, puis de chercher l'isomorphisme morceau par morceau. Il y a deux contributions principales exploitant cette idée : celle de Messmer [Messmer, 1995] et celle de Sonbaty [Sonbaty, 1998]. Les deux propositions utilisent le principe de décomposition du graphe de manières différentes. Messmer décompose tous les graphes modèles et construit une structure spéciale pour, par la suite, rechercher l'isomorphisme pour un graphe donné. Sonbaty décompose les graphes modèles individuellement et utilise ensuite une distance d'édition. Les deux algorithmes sont détaillés dans la suite.

L'idée principale de l'algorithme de Sonbaty [Sonbaty, 1998] est de décomposer les graphes en de plus petits graphes qu'il appelle les graphes attribués relationnels de base « Basic Attributed Relational Graphs » (BARG). Les BARG sont stockés sous forme d'arbres et sont composés d'un nœud et de tous les nœuds reliés à ce nœud. La figure 1.34 montre un exemple de graphe (avec cinq nœuds) décomposé dans cinq graphes de base (figure 1.34(b)-(f)). À chaque étape de décomposition, un nœud du graphe original est choisi pour être le nœud racine et son graphe de base est généré selon la structure du graphe original.

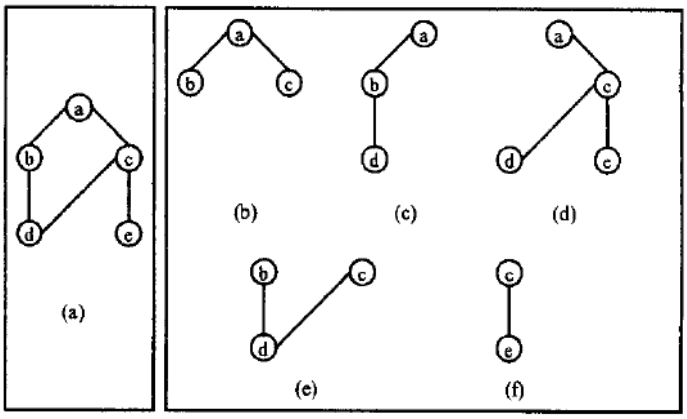


Fig. 1.34 : Les graphes attribués de base pour les nœuds a, b, c, d, et e.

Le mise en correspondance entre deux graphes est alors réalisée au niveau des sous-graphes décomposés sur la base de transformations correctrices d'erreurs « *Error-correcting transformations*» équivalentes à une distance d'édition. Les transformations correctrices d'erreurs effectuées pour calculer la distance entre deux BARG, sont : l'insertion de nœud (n_i), la suppression de nœud (n_d), l'insertion de branche (b_i), la suppression de branche (b_d), la substitution d'étiquette de nœud (n_s) et la substitution d'étiquette de branche (b_s). Les coûts correspondant à chaque opération sont : W_{ni}, W_{nd}, W_{bi}, W_{bd}, W_{ns}, et W_{bs} respectivement. Ainsi, pour deux BARG *U* et *V*, le coût de mise en correspondance de *U* et *V* est

$$Distance\ (U, V) = W_{ns} * dist\ (n_i, v_j) + min\ (W_{ni}, W_{nd}) * abs\ (k - p) + min(W_{bi}, W_{bd}) * abs(k - p) + min(dist\_bipartie)$$

Où

- $dist(n_i, v_j)$ est la distance entre les attributs des sommets n_i et v_j
- k et p sont le nombre d'arcs connectés au nœud racine des BARG, et *dist_bipartie* est une distance pondérée entre les arcs $b_{f(\text{i à k})}$ du BARG « *U* » et les arcs $e_{h(1 \text{ à } p)}$ du BARG « *V* ».

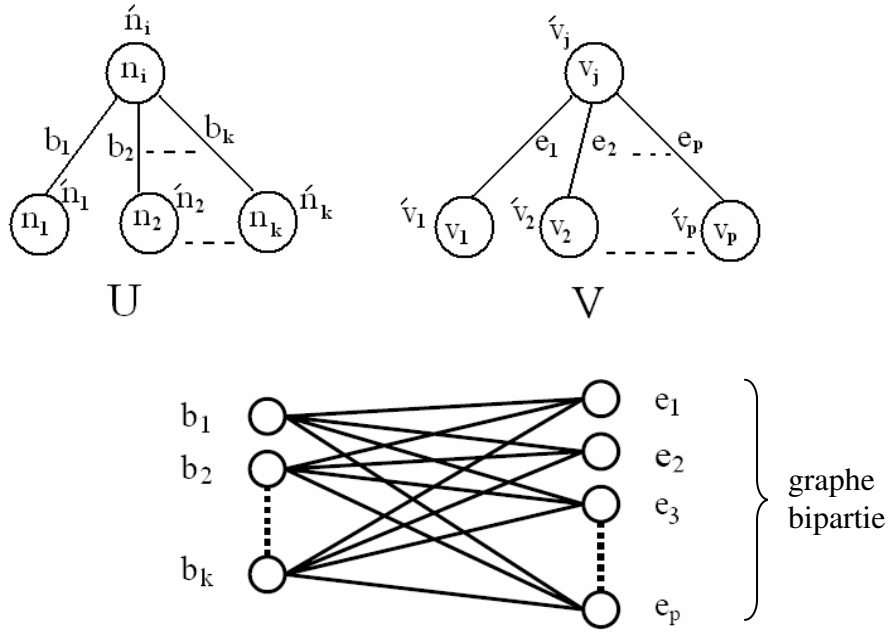


Fig. 1.35 : Deux BARGs *U*, *V et* le graphe biparti pondéré associé utilisé pour calculer la distance entre arcs.

Le pseudo-code de l'algorithme de calcul de distance entre graphes est détaillé ci-dessous :

Algorithme de Sonbaty
Entrée : deux graphes G_M *(Graphe Modèle),* G_D *(Graphe Donnée)*
Sortie : Cout d'appariement
0. **début**
1. Décompose (G_M)
2. Décompose(G_D)
3. Pour i = 1 à Nombre_de_sommet(G_M)
4. Pour j = 1 à Nombre_de_sommet(G_D)
5. Matrice_de_Distance[i, j] = distance entre $BARG_i$ de G_D et $BARG_j$ de G_M
6. Construire le graphe biparti pondéré équivalent à la Matrice_de_Distances
7. Cout_appariement = Minimum du tous les graphes biparti pondérés.
8. **fin**

Fig. 1.36 : Algorithme de Sonbaty [Sonbaty, 1998].

En effet, à partir de matrice de distances, un graphe biparti pondéré est à nouveau construit dans lequel chaque paire de sommets (un du graphe de donnée et un du graphe modèle est associée par un arc qui a un poids équivalent à la distance entre ces deux sommets.

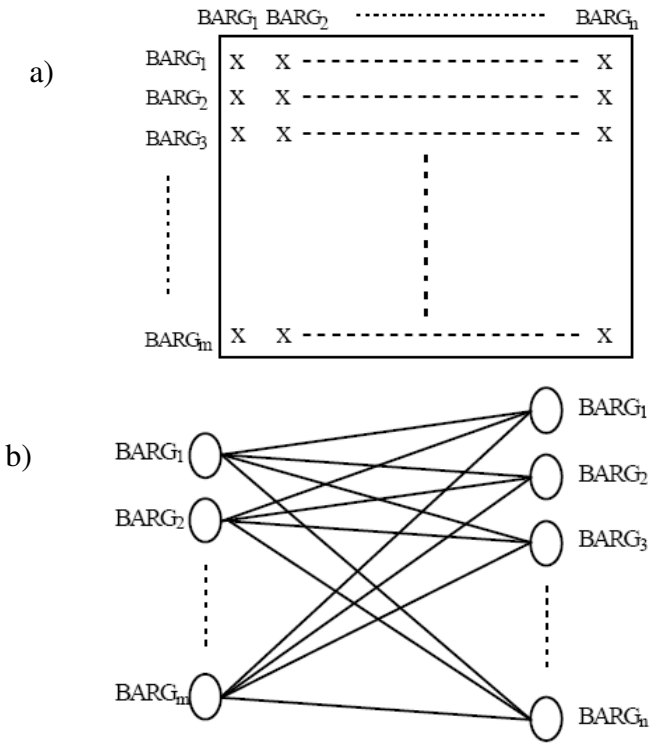


Fig. 1.37 : a) Matrice de distances entre les BARG du graphe de Donnée et les BARG du graphe Modèle, b) Graphe biparti pondéré correspond à la matrice de distances.

La complexité moyenne de l'algorithme est O ($M^2 * N^2$) pour deux graphes attribués G_1 et G_2 avec M et N comme nombre de sommets.

Messmer [Messmer, 1995] a également remarqué que le temps de calcul dépend principalement du nombre de graphes modèles présents dans le dictionnaire des formes à reconnaitre. Il considère alors que l'indexation ou l'organisation de ces graphes sous une forme hiérarchique est nécessaire. Messmer propose de décomposer tous les graphes modèles en ensemble de sous-graphes et de construire une représentation dans laquelle tous les sous-graphes sont présents. Si un sous-graphe est commun à plus d'un graphe modèle, il n'est présent qu'une seule fois dans cette représentation. Cette étape de décomposition est faite hors ligne « off line » pour gagner du temps.

Un algorithme de décomposition sous optimal peut être utilisé qui décompose un graphe en petits morceaux jusqu'à obtenir un seul sommet. Tous les graphes modèles sont décomposés avec le même algorithme et un ensemble de sous-graphes est obtenu. Ensuite, ces sous-graphes sont insérés dans une structure hiérarchique, appelée réseau de sous-graphes, ou chaque sommet représente un sous-graphe unique des graphes modèles. La figure 1.38 montre le réseau généré à partir de la décomposion des graphes g_1 et g_2. Au niveau-0 du réseau, les sommets 1, 2, 3, 4, 5 ne représentent qu'un sommet des graphes modèles. Les autres niveaux représentent les sous-graphes qui comportent plus qu'un sommet. Les graphes modèles complets sont liés au niveau final. Chaque sommet du réseau possède un numéro unique pour l'identifier. Le sommet " 7 " est le plus grand sous-graphe commun entre les graphes g_1 et g_2.

La recherche d'isomorphismes d'un graphe à reconnaître commence au niveau-0. Après chaque test d'isomorphisme de sous graphes, le système marque le sous-graphe soit comme *vivant,* soit comme *mort.* Les sous-graphes marqués comme vivants sont les candidats pour les étapes suivantes. Donc, progressivement le réseau est parcourir et les sous-graphes qui satisfont la condition d'isomorphisme vis-à-vis du graphe requête sont identifiés. Au final, une fusion de tous les graphes vivants donne le sous-graphe final pour déterminer la correspondance avec un graphe modèle. Dans la figure 1.38, on voit bien que le graphe g_2 (sommet ''9'' du réseau) a donné le plus grand sous-graphe pour un graphe requête g_3, représenté par {1254}.

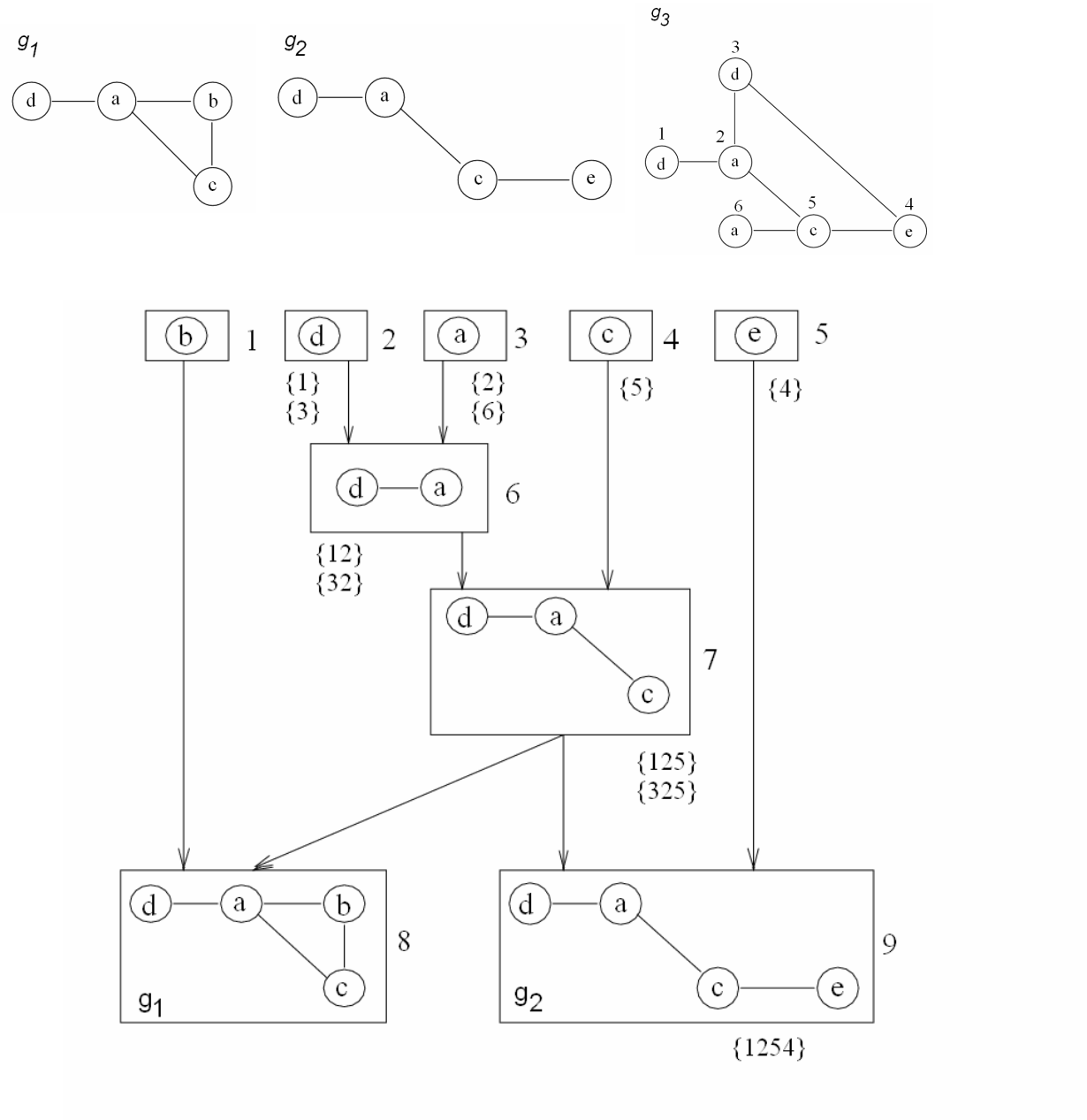


Fig. 1.38 : Décomposition en sous-graphes de deux graphes g_1 et g_2, (g_3 est un graphe à comparer).

Il y a deux limitations principales à cette approche : premièrement, il faut que les graphes ne soient pas denses (pas trop d'arcs entre les sommets). Deuxièmement, le système est beaucoup moins performant quand les graphes modèles ne sont pas similaires. Ce qui est de plus en plus vrai lorsque le nombre d'attributs associés aux nœuds et arcs augmente.

1.4.3 Algorithmes VF et VF2

L'algorithme le plus récent pour la recherche d'isomorphismes de graphes et sous-graphes est l'algorithme VF de Cordella[Cordella, 1998]. L'auteur a défini une heuristique qui est basée sur l'analyse d'un ensemble de sommets adjacents aux sommets déjà présents dans l'appariement courant. Grâce à un calcul rapide de cette heuristique, la méthode a montré une amélioration significative par rapport à l'algorithme d'Ullman (Tab 1.1). De ce fait, le nom de l'algorithme est une abréviation anglaise : « Very Fast algorithm ». Une autre version de l'algorithme est VF2 [Cordella, 2001] qui réduit l'espace mémoire nécessaire ce qui entraîne une réduction de complexité de l'ordre $O(N^2)$ à $O(N)$ par rapport au nombre de sommets dans les graphes.

L'algorithme VF2 est donné dans la figure. 1.39. Etant donné deux graphes $G_1(N_1, B_1)$ et $G_2(N_2, B_2)$, la mise en correspondance des graphes est décrite par la notion de « State Space Representation ». L'objectif est de trouver un appariement $M \subset N_1 \times N_2$ (un ensemble de couples (n, m), $n \in G_1$ et $m \in G_2$) tel que M soit bijectif et préserve la structure des arcs entre les deux graphes. Un état s du processus représente une solution partielle de la correspondance entre deux graphes. M est l'ensemble des solutions partielles, $M(s)$ est un sous-ensemble de M représentant la solution partielle courante à l'état s correspondant aux sous-graphes $G_1(s)$ et $G_2(s)$ de G_1 et G_2 respectivement.

L'idée principale est qu'à chaque état du processus (à partir de l'état initial s_0 jusqu'à l'état de solution s_n), l'algorithme vérifie des conditions de fiabilité pour l'état actuel. Si l'état s est susceptible d'être étendu à certaines étapes dans l'avenir, on garde cet état. Les états qui ne satisfont pas les conditions peuvent être éliminés efficacement à l'avance. Les conditions utilisées s'appellent *k-look-ahead rules* (normalement $k = 1$ ou $k = 2$). Ces règles dépendent forcément de chaque application. Cependant, dans leur application, les auteurs proposent d'utiliser deux groupes principaux de règles : « Syntactic feasibility rules » et «Semantic feasibility rules » [Pham, 2005]. Une fonction de fiabilité $F(s, n, m)$ est une fonction booléenne utilisée pour réduire l'espace de recherche. Lorsque la valeur renvoyée est vraie, c'est une indication d'isomorphisme de l'état s' obtenu en ajoutant le couple candidat (n, m) à l'état s ($s' = s \cup p$, où $p = (n, m)$).

$\wedge$ (ET)

$$F(s, n, m) = F_{syn}(s, n, m) \wedge F_{sem}(s, n, m)$$

La fonction de fiabilité syntaxique F_{syn} dépendant seulement de la structure des graphes, et la fonction de fiabilité sémantique F_{sem} dépend des attributs des graphes.

Algorithme VF2(s)
Entrée : un état intermédiaire s ou l'état initial s_0 tel que $M(s_0)$ = NULL
Sortie: Les appariements entre les deux graphes

0. **début**
1. **si** $M(s)$ comprend tous les sommets de G_2 **alors** renvoyer $M(s)$
2. **sinon**
3. Calculer $P(s)$ ensemble de couples candidats pour l'inclusion dans $M(s)$
4. **pour chaque** $(n, m) \in P(s)$
5. **si** $F(s, n, m)$ **alors**
6. Calculer l'état s' (obtenu en ajoutant le couple (n, m) à $M(s)$)
7. Appeler VF2 (s')
8. **fin si**
9. **fin pour**
10. Stocker la structure de données
11. **fin si**
12. **fin**

Fig. 1.39 : L'algorithme VF2 de Cordella [Cordella, 2001]

L'ensemble de couples candidats $P(s)$ est obtenu en considérant l'ensemble de sommets associés entre $G_1(s)$ et $G_2(s)$.

Soit $T_1^{out}(s)$ et $T_2^{out}(s)$ l'ensemble des sommets pas encore présents dans l'appariement courant, mais correspondant à des sommets cibles pour les arcs qui sortent des graphes $G_1(s)$ et $G_2(s)$ respectivement. Soit, $T_1^{in}(s)$ et $T_2^{in}(s)$ l'ensemble des sommets pas encore présents dans l'appariement courant, mais qui correspondent à des sommets sources pour les arcs qui se terminent dans les graphes $G_1(s)$ et $G_2(s)$. Soient,

$$T_1(s) = T_1^{in}(s) \cup T_1^{out}(s) \quad \text{et} \quad T_2(s) = T_2^{in}(s) \cup T_2^{out}(s).$$

Alors, pour un graphe $G(N, B)$ et un sommet $n \in N$, si l'ensemble des sommets prédécesseurs et des sommets successeurs de n sont dénotés par $Pred(G, n)$ et $Succ(G, n)$, alors la fonction de fiabilité et les définitions des 5 règles pour tester les isomorphismes sont les suivantes :

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

$$R_{\text{pred}}(s,n,m) \Longleftrightarrow$$
$$(\forall n' \in M_1(s) \cap \text{Pred}(G_1,n) \exists m' \in \text{Pred}(G_2,m) \mid (n',m') \in M(s)) \wedge$$
$$(\forall m' \in M_2(s) \cap \text{Pred}(G_2,m) \exists n' \in \text{Pred}(G_1,n) \mid (n',m') \in M(s)),$$

$$R_{\text{succ}}(s,n,m) \Longleftrightarrow$$
$$(\forall n' \in M_1(s) \cap \text{Succ}(G_1,n) \exists m' \in \text{Succ}(G_2,m) \mid (n',m') \in M(s)) \wedge$$
$$(\forall m' \in M_2(s) \cap \text{Succ}(G_2,m) \exists n' \in \text{Succ}(G_1,n) \mid (n',m') \in M(s)),$$

$$R_{\text{in}}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{\text{in}}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{\text{in}}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{\text{in}}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{\text{in}}(s))),$$

$$R_{\text{out}}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{\text{out}}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{\text{out}}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{\text{out}}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{\text{out}}(s))),$$

$$R_{\text{new}}(s,n,m) \Longleftrightarrow$$
$$\text{Card}(\tilde{N}_1(s) \cap \text{Pred}(G_1,n)) \geq \text{Card}(\tilde{N}_2(s) \cap \text{Pred}(G_2,n)) \wedge$$
$$\text{Card}(\tilde{N}_1(s) \cap \text{Succ}(G_1,n)) \geq \text{Card}(\tilde{N}_2(s) \cap \text{Succ}(G_2,n)).$$

Les règles ci-dessus sont utilisées pour trouver l'isomorphisme de sous-graphes. Pour tester l'isomorphisme des graphes, il faut remplacer l'opérateur(≥) par l'opérateur (=) dans les 3 règles (R_{in}, R_{out}, R_{new}). Le tableau 1.1 compare les complexités des algorithmes VF2 et Ullman montrant bien la supériorité de VF2.

Tab. 1.1 : Comparaison de complexité par rapport au temps de calcul et à l'espace utilisé.

	VF2		Ullman	
Complexité	au mieux	au pire	au mieux	au pire
Temps	$O(N^2)$	$O(N!\,N)$	$O(N^3)$	$O(N!\,N^2)$
Espace	$O(N)$	$O(N)$	$O(N^3)$	$O(N^3)$

Remarque: ($N = |V_1| + |V_2|$)

Le « *Nauty algorithm* » de McKay [McKay, 1981] doit également être cité car il est considéré aussi comme un des algorithmes les plus rapides permettant de trouver un

isomorphisme. L'algorithme assez complexe est basé sur la théorie des ensembles « group theory » [McKay, 1978]. Il se base sur les groupes d'automorphismes et les symétries présentes dans un graphe. Un automorphisme de graphes est un isomorphisme de graphes avec lui-même. L'ensemble des automorphismes Aut(G) est défini par un groupe de permutations finies. Une symétrie dans un problème de satisfaction de contraintes est une fonction bijective qui transforme des solutions en solutions. Les symétries sont importantes car elles créent des sous-arbres symétriques dans l'arbre de recherche. S'il n'y a pas de solutions, l'absence de solutions est prouvée plusieurs fois dans des sous-arbres symétriques. Si des solutions existent, beaucoup de solutions symétriques seront énumérées dans des sous-arbres également symétriques. La détection et l'élimination des symétries permettent, par conséquent, d'éliminer ces sous-arbres symétriques et d'accélérer la résolution des problèmes de satisfaction de contraintes. Les symétries sont présentes naturellement dans les graphes puisqu'une permutation peut être vue comme un automorphisme dans un graphe [Stéphane, 2006].

Une comparaison de ces algorithmes [Ullman, 1976] [McKay, 1981] [Cordella, 1998], [Cordella, 2001] est présentée dans [Foggia, 2001]. L'auteur compare leur complexité et conclut avec la remarque qu'il n'existe aucun algorithme qui marche mieux que tous les autres. Le choix dépend du type de graphes traités. Par exemple, l'algorithme Nauty est beaucoup plus performant pour les graphes aléatoires de taille importante alors que pour les graphes non denses et plutôt réguliers VF2 est le meilleur.

1.4.4 Recherche du plus grand sous-graphe commun

La recherche du plus grand sous-graphe commun entre deux graphes est une autre façon de mesurer une similarité entre les graphes malgré des distorsions. Le plus grand sous graphe commun de deux graphes G_1 et G_2, mcs(G_1, G_2) est un sous-graphe des deux graphes G_1 et G_2, vérifiant la condition qu'il contient le plus grand nombre de sommets parmi tous les sous-graphes communs pour ces deux graphes.

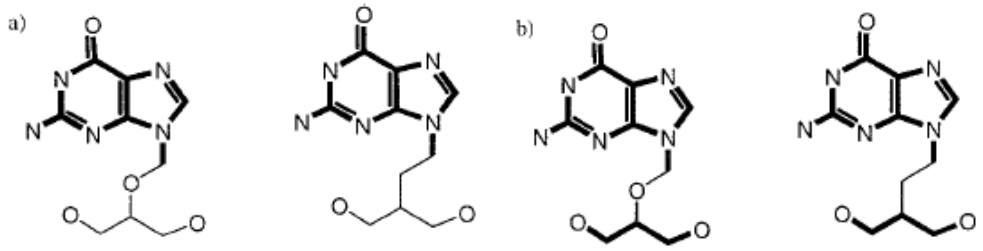


Fig. 1.40 : a) Plus grand sous-graphe commun connecté b) Plus grand sous graphe commun non connecté [Raymond, 2002].

Le problème de recherche du plus grand sous-graphe commun est similaire à la recherche de la clique maximale dans un graphe d'association [Koch, 2001]. Le graphe d'association généré à partir de $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ est un graphe $G = (V, E)$ non orienté et défini comme :

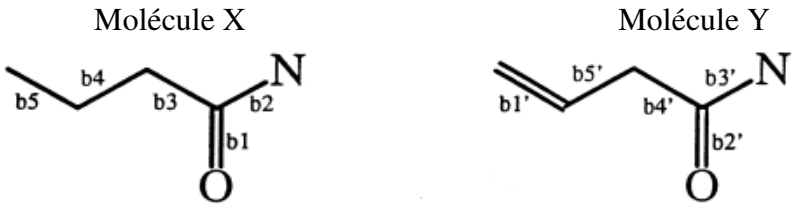
$$V = V_1 \times V_2 \text{ et}$$
$$E = \{((i,h),(j,k)) \in V \times V : i \neq j, h \neq k, et(i,j) \in E_1 \Leftrightarrow (h,k) \in E_2\}$$

Le détection de clique maximale est un problème NP difficile et oblige donc à utiliser des algorithmes approximatifs [Bomze, 1999]. Dans certaines conditions, la vitesse peut être augmentée en utilisant des heuristiques, par exemple, l'algorithme « branch and bound parallèle » [Shinano, 1998] organise les graphes sous forme hiérarchique [Pelillo, 1999].

Les trois algorithmes les plus connus pour trouver le plus grand sous-graphe commun (MCS) incluent l'algorithme de McGregor [McGregor, 1982], l'algorithme de Durand-Pasari [Durand, 1999] et l'algorithme de Balas-Yu [Balas, 1986]. Ces algorithmes sont tout à fait semblables en plusieurs aspects. Ils appartiennent tous à la catégorie des algorithmes exacts ou optimaux, en opposition aux algorithmes approximatifs ou sous-optimaux, ils fournissent donc toujours la solution optimale.

Ils exécutent une recherche, en profondeur d'abord, avec l'aide de certaines heuristiques pour élaguer les chemins de recherche inutiles. La différence entre ces 3 algorithmes se situe seulement dans l'information représentant chaque état de l'espace de recherche et dans le type d'heuristiques adoptées. Nous avons choisi de détailler l'approche de Durand [Durand, 1999] qui est la plus récente. Cet algorithme a été implémenté dans le domaine de la chimie pour trouver des isomorphismes de sous-graphes entre les molécules chimiques. Pour comparer les molécules X et Y (fig. 1.41(a)), leurs structures sont représentées par les graphes étiquetés X et Y respectivement, dans lesquels les sommets représentent des atomes, et les arcs dénotent des liens/liaisons. Les étiquettes de sommets et d'arcs indiquent des types d'atomes et de liaisons (fig. 1.41 (b))

(a) Molécules X et Y



(b) Les graphes étiquetés X et Y

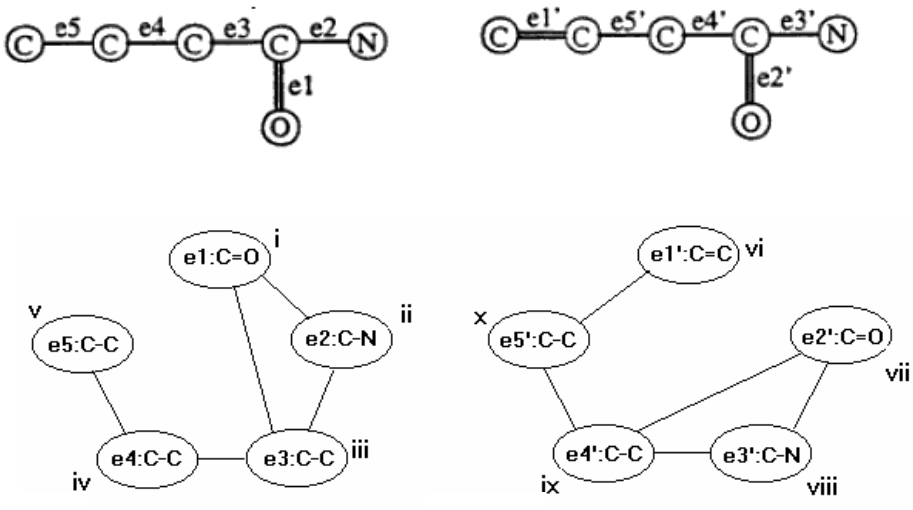
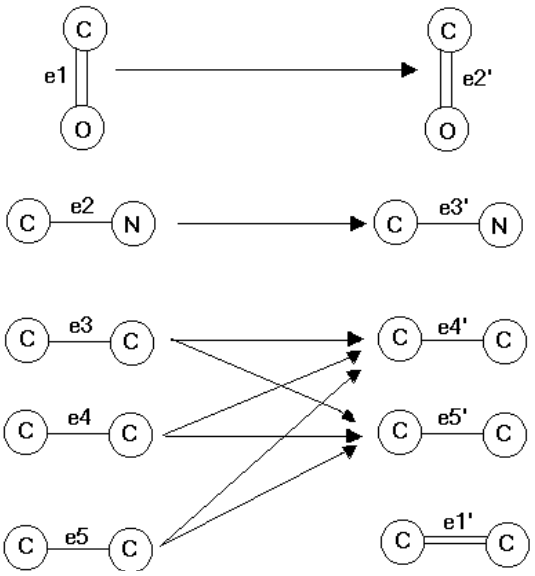
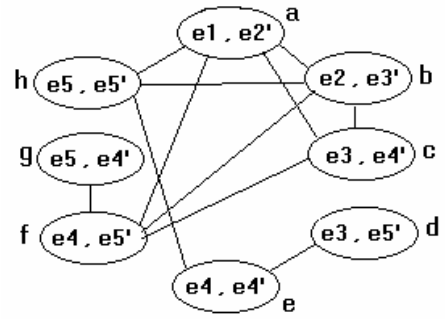


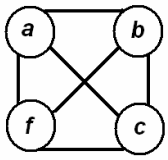
Fig. 1.41 : Les molécules X et Y et leurs structures représentées par les graphes.



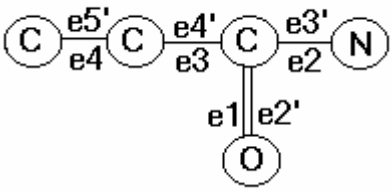
a) Mise en correspondance des arcs qui deviennent les sommets dans le graphe de compatibilité



clique maximale



b) Graphe de compatibilité et clique maximale associées



c) La plus grand sous-structure commun (MaCS)

Fig. 1.42: Illustration de la routine TopSim donnant la structure commune maximale [Durand, 1999].

La routine TOPSIM décrite figure 1.42 compare X et Y et identifie leur sous-graphe commun maximal (MaCS), ce qui revient à trouver la clique maximale dans un graphe de compatibilité. Etant donné une paire de graphes X et Y, la similarité MSI(X, Y) est définie comme :

$$MSI(X, Y) = [\ |MaCS(X,Y)| / NAB(X)\ ] * [\ |MaCS(X,Y)| / NAB(Y)\ ]$$

La dissimilarité entre les graphes X et Y est mesurée par la distance topologique, TD(X, Y), définie comme :

$$TD(X, Y) = NAB(X) + NAB(Y) - 2|MaCS(X, Y)|$$

|MaCS(X,Y)| est le nombre de sommets et d'arcs dans la structure commune maximum(MaCS) de X et Y. NAB(X) est le nombre de sommets et d'arcs dans le graphe X, et NAB(Y) est le nombre de sommets et les arcs dans le graphe Y.

$$NAB(X) = A_6B_5 = 11 \quad NAB(Y) = A_6B_5 = 11 \quad MaCS(X,Y) = A_5B_4 = 9$$

$$MSI(X, Y) = [9 /11] * [9 /11] = 0.66$$

$$TD(X, Y) = 11 + 11 – 2*9 = 4$$

Une analyse des performances de ces trois algorithmes est présentée dans [Conte, 2003] ; la comparaison a été effectuée sur une grande base de données de graphes étiquetés produits synthétiquement. L'auteur conclut que l'algorithme de McGregor [McGregor, 1982] est le meilleur si les graphes sont non denses (peu d'arcs) et/ou de petite taille. Pour les graphes plus denses ou de taille plus importante, au contraire, l'algorithme de Durand-Pasari [Durand, 1999] fonctionne mieux. Si le MaCS a une structure plus régulière, (maillée par exemple) l'algorithme de McGregor est dans la plupart des cas le meilleur algorithme. L'algorithme de Durand-Pasari fonctionne mieux seulement pour de petits graphes denses. Quand le degré d'irrégularité grandit, l'algorithme de Balas-Yu donne de meilleurs résultats pour les grands graphes denses. Cependant, la recherche de MaCS reste un problème NP-complet et la complexité des algorithmes reste dans le pire cas exponentielle par rapport au nombre de sommets dans les graphes.

L'algorithme [Messmer, 1995] de recherche d'isomorphisme de sous-graphe à base d'arbres (de décision) peut aussi être adapté [Messmer, 1999] pour effectuer une recherche du plus grand sous graphe commun.

Le temps nécessaire pour parcourir l'arbre de décision est $O(2^n n^3)$, où n est le nombre des nœuds dans le graphe requête. Ainsi, la complexité de ce procédé est sensiblement plus haute que $O(n^2)$, nécessaire pour la détection d'isomorphisme de sous-graphes. Cette remarque confirme que la recherche du plus grand sous- graphe commun est une tâche plus compliquée que la détection d'isomorphisme de sous-graphe[Bunke, 2000b].

1.5 Méthodes de mise en correspondance inexacte de graphes

1.5.1 Présentation du problème

Dans certaines applications, le codage d'un objet par un graphe attribué peut légèrement fluctuer à cause de bruits et de distorsions. Il est alors nécessaire d'introduire un modèle d'erreur ou d'intégrer la notion de tolérance durant l'appariement des graphes. Ainsi, le terme "inexacte" appliqué à certains problèmes de mise en correspondance de graphes veut dire qu'il n'est pas toujours possible de trouver un isomorphisme entre les deux graphes. C'est le cas lorsque le nombre de noeuds est différent dans les deux graphes (modèle, données). Ceci peut aussi résulter de l'aspect schématique du modèle ou, en analyse d'images par exemple, de la difficulté de segmenter correctement l'image en des entités significatives.

Par conséquent, le problème de mise en correspondance de graphes ne consiste plus en la recherche d'un résultat exact de mise en correspondance mais en la recherche de similarité entre graphes. Les moyens utilisables pour résoudre ce type de problèmes sont actuellement la distance d'édition entre graphes « *Graph Edit Distance* » ou l'isomorphisme de sous graphe avec tolérance d'erreur « *error correcting subgraph isomorphism* ». La mise en correspondance vise alors la recherche d'une correspondance non bijective entre le graphe de données et le graphe modèle. Dans ce qui suit, on suppose que $|V_M| < |V_D|$.

1.5.2 Distance d'édition entre graphes

Cette technique est une extension de l'approche "distance d'édition entre chaînes de caractères" « string edit distance » [Wagner, 1974]. Elle a été introduite dans le domaine des graphes par H. Bunke [Bunke, 1999].

L'idée de base est de transformer un graphe en un autre graphe en utilisant un ensemble d'opérations de transformations sur les sommets et les arcs du graphe. Habituellement, il y a six types d'opérations d'édition possibles. La substitution d'étiquette d'un sommet, la substitution d'étiquette d'un arc, l'insertion d'un sommet, l'insertion d'un arc, la suppression d'un sommet et la suppression d'un arc. Un coût est associé à chaque opération et la distance d'édition correspond à la somme de tous les coûts des transformations permettant de passer d'un graphe à un autre. Différentes séquences d'opérations peuvent être considérées, chacune ayant un coût différent et la séquence qui donne le plus petit coût est considérée comme la transformation générant la meilleure mise en correspondance.

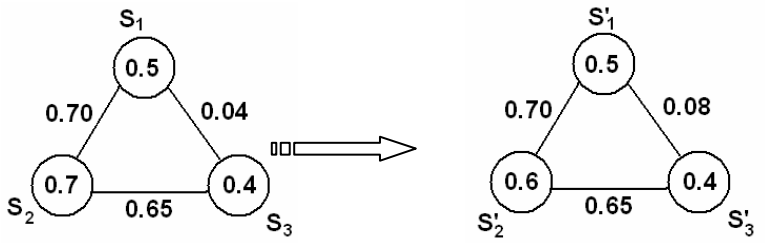
Si "e" est une opération d'édition parmi les opérations définies précédemment, et $C(e)$ son coût, le coût total de séquence complète $s = e_1, e_2,\ldots, e_n$ peut se calculer en effectuant la somme de tous les coûts individuels :

$$C(s) = \sum_{i=1}^{n} C(e_i)$$

La distance d'édition entre les deux graphes g_1 et g_2 correspond à :

$d(g_1, g_2) = min\{C(s)$ *avec s une* séquence d'opérations d'édition pour transformer g_1 en $g_2\}$

La figure. 1.43 donne un exemple de transformation dans lequel le coût minimal correspond à la différence entre les deux valeurs numériques associées à chaque sommet ou arc modifié.



Graphe G_1 Graphe G_2

Fig. 1.43: Distance d'édition entre graphes.

La séquence de coût minimal correspond à une substitution de sommet et une substitution d'arc. Elle produit l'appariement suivant :

$S_2 \rightarrow S_2' \quad [0.1]$
$(S_1, S_3) \rightarrow (S_1', S_3') \quad [0.04]$

Coût de la séquence : 0.1 + 0.04 = 0.14

Appariements Sommets : $S_1 \rightarrow S_1'$, $S_2 \rightarrow S_2'$, $S_3 \rightarrow S_3'$

Arcs: $(S_1, S_2) : (S_1', S_2')$, $(S_1, S_3) : (S_1', S_3')$, $(S_2, S_3) : (S_2', S_3')$

Les coûts doivent normalement être des valeurs réelles positives ou nulles mais peuvent dépendre de l'application. Il est alors difficile de trouver aussi bien automatiquement que par apprentissage les valeurs optimales de ces coûts.

Récemment, Bunke a démontré que le plus grand sous-graphe commun et la distance d'édition entre graphes donnent des résultats équivalents lorsqu'une fonction particulière de coût est utilisée. Une première approche est présentée dans [Bunke97b]. Le lien entre la distance d'édition et le plus grand sous-graphe commun (mcs) alors le suivant :

$$d(g_1, g_2) = |g_1| + |g_2| - 2\,|mcs(g_1, g_2)|$$

$mcs(g_1, g_2)$ est le plus grand sous-graphe commun de g_1 et g_2, avec $d(g_1, g_2)$ la distance d'édition calculée, $| g_1 |$, $| g_2 |$ et $|mcs(g_1, g_2)|$ représentent le nombre du sommets de g_1, g_2 et $mcs(g_1, g_2)$ respectivement.

Dans [Bunke, 1998], une autre solution est proposée caractérisée par une contrainte sur le coût de suppression ou d'insertion d'un élément (sommet ou arc) qui doit être supérieur au coût de substitution d'un élément.

Dans ce 2ème cas, le plus grand sous-graphe commun de deux graphes g_1 et g_2 et leur distance d'édition sont liés par :

$$d(g_1, g_2) = 1 - \frac{|mcs(g_1, g_2)|}{\max(|g_1|, |g_2|)}$$

Kelly et Hancock [Kelly, 2002] ont montré comment utiliser les vecteurs propres de la matrice d'adjacence du graphes pour convertir un graphe en une chaîne de caractères et puis calculer la similarité entre les graphes par distance d'édition entre chaînes de caractères. L'année suivante, la technique a été améliorée en utilisant une sérialisation spectrale couplée à une distance d'édition [Kelly, 2003]. En bio-informatique, le problème de conversion d'un graphe en une chaîne est appelé sérialisation. Le principe de la sérialisation est de mettre en ordre les ensembles de sommets d'un graphe de manière à ce que les sommets qui ont une corrélation forte entre eux soient proches les uns des autres. Une fois les graphes convertis en chaînes de caractères à l'aide des vecteurs propres de la matrice d'adjacence, les possibilités d'utilisation de distance d'édition sont très nombreuses. Dans [Geusebroek, 1999], les auteurs utilisent ce type de techniques pour la segmentation des tissus. De même, dans [Foggia, 1999], la reconnaissance des chiffres manuscrits à partir d'une base de données de caractères standards est résolue grâce à ce type de méthodes.

L'algorithme de recherche d'isomorphisme de sous-graphes avec tolérance d'erreur proposé par Joseph Lladós [Lladós, 2001b] essaye de mettre en correspondance un graphe requête avec un graphe modèle sur la base de transformations appliquées à des graphes d'adjacence de régions (*GAR*) $G_D = (V_D, E_D)$ et $G_M = (V_M, E_M)$ respectivement. Quand les graphes sont des GAR, l'opération de substitution de sommets implique réellement une substitution d'une région entière de G_D par une région entière de G_M. Il devient alors possible, dans certains cas, des heuristiques, pour guider un algorithme de recherche d'isomorphismes à partir de la définition d'une distance d'édition adaptée.

L'idée intuitive de l'algorithme est de réaliser un appariement incrémental des régions. L'algorithme proposé est de type « Branch and Bound » et la génération des états successifs est guidée par le coût des opérations d'édition.

1.5.3 Méthodes à base d'arbre de décision

Récemment, la méthode de recherche d'isomorphismes de sous-graphe à base d'arbre de décision de [Messmer, 1998B] a été adaptée de manière à tolérer des erreurs. En effet, il est possible de prévoir la correction d'erreurs durant la création de l'arbre de décision. Pour chaque graphe modèle, des exemples avec distorsions sont générés et compilés dans l'arbre de décision. Le nombre d'exemples dépend de l'erreur maximale admissible. Durant la reconnaissance (la comparaison), l'arbre de décision est employé de manière classique. La complexité de temps d'exécution de ce procédé reste quadratique par rapport au nombre de sommets dans le graphe requête. Cependant, la taille de l'arbre de décision augmente exponentiellement avec le nombre de sommets dans les graphes modèles et dépend donc beaucoup du degré de déformation considéré. Par conséquent, cette approche est limitée aux graphes de très petite taille. De plus, il est très difficile de prévoir les types d'erreurs qui se produiront dans les cas réels.

Dans une deuxième approche, les corrections d'erreurs sont considérées durant l'exécution seulement. L'arbre de décision représentant l'ensemble de graphes modèles n'incorpore aucune information au sujet des erreurs possibles. Cette fois-ci, c'est le graphe requête qui est transformé pour produire un ensemble de copies déformées du graphe. Chaque graphe est alors classifié par l'arbre de décision. La complexité en temps d'exécution de cette méthode est $O(\delta n^{2(\delta+1)})$ où n est le nombre de nœuds dans le graphe requête et δ est un seuil qui définit le nombre maximum d'opérations d'édition admissibles pour effectuer les déformations [Bunke, 2000b].

1.6 Mesure de similarité entre graphes

1.6.1 Méthodes de mesure de similarité sans appariement

- **Sondage de graphe (graph probing)**

Le concept de sondage de graphe a été récemment présenté par Lopresti [Lopresti, 2001]. Il s'agit d'une combinaison très intéressante entre méthode structurelle et approche statistique. Dans l'approche proposée, les documents (page web) sont représentés par des graphes orientés. Pour trouver la similarité entre les graphes représentant les documents, différentes caractéristiques numériques sont extraites à partir de leur représentation sous forme de graphe. Une fois des vecteurs numériques obtenus, il devient possible d'étudier les similarités et/ou les corrélations entre les caractéristiques extraites. Les (sondes) correspondent aux caractéristiques qui fournissent des informations sur les sommets et les arcs. Par exemple, le nombre de sommets avec une certaine étiquette peut correspondre à une sonde.

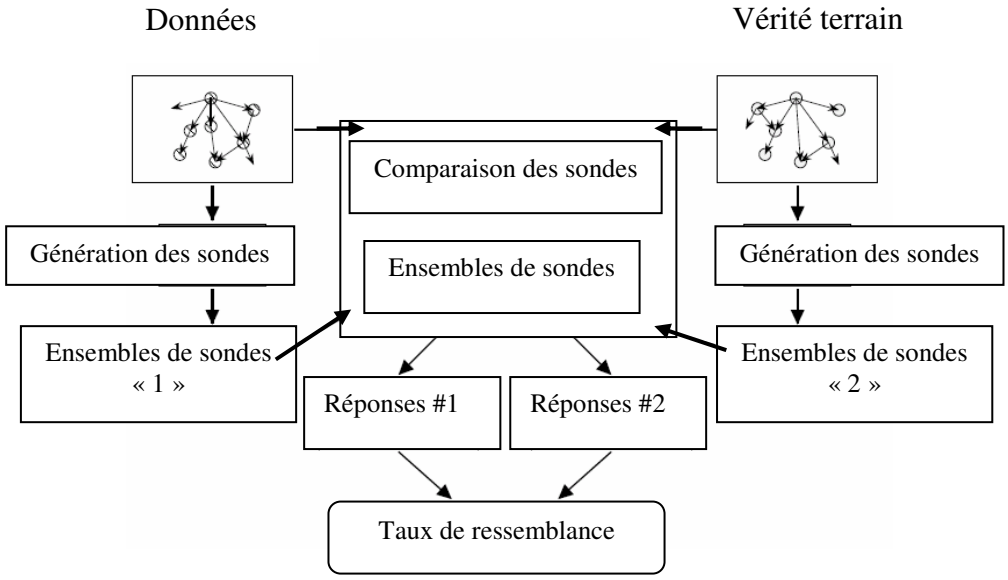


Fig. 1.44 : Architecture du « graph probing » [Lopresti, 2001].

Comme indiqué dans la figure.1.44, les vecteurs de sondes sont comparés pour calculer une mesure de similarité. Lopresti propose d'étudier le nombre de fois où les deux sondes sont similaires pour produire un degré de ressemblance. Le sondage de graphes est une bonne méthode pour obtenir un degré de ressemblance entre deux graphes mais il ne fournit pas l'appariement sommets à sommets correspondant à la meilleure mise en correspondance. De plus, la mesure de similarité proposée par Lopresti n'est pas symétrique : elle dépend du graphe source et des caractéristiques extraites.

- **Comparaison d'histogrammes**

Une approche similaire au sondage de graphe a été proposée par Papadopoulos [Papadopoulos, 1998]. Cette méthode propose d'utiliser les histogrammes des degrés de sommets pour comparer les graphes. L'idée est d'avoir une technique d'indexation efficace de graphes pour trouver rapidement des similarités simples dans une grande base de données de graphes. Malheureusement, cette méthode se focalise seulement sur le degré des sommets, elle utilise une caractéristique peu discriminante. Cette méthode, comme le graphe probing, ne fournit aucune information sur l'isomorphisme trouvé.

1.6.2 Mesure de similarité avec appariement

Nous avons choisi de séparer les méthodes appartenant à cette catégorie en deux sous-classes. La première classe regroupe des méthodes dites un-avec-un et la deuxième classe regroupe des méthodes dites un-avec-plusieurs.

- **Un avec Un**

Les méthodes un avec un sont celles qui nécessitent d'avoir exactement le même nombre de sommets dans les deux graphes à comparer. Dans [Umeyama, 1988], les auteurs effectuent une analyse spectrale qui utilise les valeurs propres et vecteurs propres des matrices d'adjacence de graphes pour chercher l'isomorphisme. Si l'on représente la matrice d'adjacence d'un graphe par A_D, les valeurs propres peuvent être calculées par l'équation $|A_D - \lambda I| = 0$. Le vecteur propre ϕ_i associé à la valeur propre λ_i peut être trouvée avec l'équation $A_D\phi_i = \lambda_i\phi_i$. Etant donné deux graphes $G_1(V_1, E_1)$ et $G_2(V_2, E_2)$ de même taille, $B_1^T = (\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{|v_1|})$ et $B_2^T = (\lambda_1, \lambda_2, \lambda_3, ..., \lambda_{|v_2|})$ les vecteurs propres correspondant aux matrices d'adjacence de graphes G_1 et G_2, les

auteurs proposent d'utiliser un simple calcul de distance euclidienne entre ces vecteurs propres pour fournir la mesure de similarité entre les deux graphes.

Malheureusement, les méthodes basées sur les approches par calcul de valeurs propres à partir de matrices d'adjacence ne sont pas très performantes puisque des graphes non isomorphiques peuvent avoir les mêmes valeurs propres. De plus, ces méthodes ne peuvent pas traiter les graphes des tailles différentes.

Les réseaux de neurones ont aussi été largement appliqués aux problèmes d'appariement un-avec-un. L'idée de base est qu'un neurone représente une mise en correspondance entre une paire de sommets et le poids d'une connexion représente une mesure de compatibilité entre ces deux nœuds. La fonction de minimisation de l'énergie du réseau est définie en termes de compatibilité entre les mises en correspondance. Ce type de technique a été utilisée, par exemple, pour la reconnaissance automatique de sillons corticaux du cerveau humain dans les images IRM [Riviere, 2002], pour la reconnaissance du langage des signes [Huang, 1998], pour l'authentification de surfaces frontales [Kotropoulos, 2000] ou pour la reconnaissance des caractères manuscrits chinois [Suganthan, 1998] [Merad, 2004]. D'autres types de méthodes comme les approches probabilistes [Wang, 2007] et les méthodes génétiques [Raveaux, 2007] permettent également d'obtenir des appariements un-avec-un et des mesures de similarité.

- **Un avec Plusieurs**

Champin [Champin, 2003] propose une méthode de comparaison de graphes étiquetés permettant d'associer un sommet à plusieurs autres dans le second graphe lors de l'appariement. De plus, afin de distinguer différents types de composants et de relations dans les graphes, des attributs peuvent être ajoutés aux sommets et aux arcs.

La mesure de similarité proposée par Champin utilise une comparaison des caractéristiques communes entre les deux objets (graphes) tenant compte du nombre total de caractéristiques. Lorsqu'un appariement est mis en place entre les deux graphes, l'intersection entre les caractéristiques correspondant à cet appariement peut être calculée. Elle correspond à l'ensemble des arcs et nœuds ayant les mêmes étiquettes dans G_1 et dans G_2. Cette méthode ne fonctionne donc que pour des attributs (étiquettes) symboliques. L'algorithme prend deux graphes étiquettes en entrée et retourne la meilleure mise en correspondance des sommets entre les deux graphes $best_m$. Etant

donné : L_v un ensemble d'étiquettes de sommets, L_E un ensemble d'étiquettes d'arcs, les graphes manipulés par l'algorithme sont définis par des triplets qui décrivent complètement toutes leur caractéristiques : $G = < V, r_{v,} r_E >$ où

- V est un ensemble fini de sommets,
- $r_v \subseteq V \times L_v$ est une relation associant une ou plusieurs étiquettes aux sommets
- $r_E \subseteq V \times V \times L_E$ est une relation associant une ou plusieurs étiquettes aux arcs

$descr(G) = r_v \cup r_E$ décrit complètement G

Etant donné $G_1 = < V_1, r_{v1,} r_{E1} >$ et $G_2 = < V_2, r_{v2,} r_{E2} >$ définis sur les mêmes ensembles L_v et L_E d'étiquettes de sommets et d'arcs et tel que $V_1 \cap V_2 = 0$, la similarité entre G_1 et G_2 est définie par rapport à un appariement multivoque Mp, i.e., une relation $Mp \subseteq V_1 \times V_2$.

De plus, la définition de la mesure de similarité entre graphes proposée permet de prendre en compte les appariements multiples d'un sommet (un sommet peut être éclaté, i.e., apparié à plusieurs sommets). Pour cela, l'auteur il introduit la fonction split(Sp) qui retourne l'ensemble des sommets éclatés avec l'ensemble des sommets auxquels ils sont associés dans un appariement Mp:

$$Sp(Mp) = \{(v, s_v) \mid v \in V_1 \cup V_2, s_v = Mp(v) \mid Mp(v)| \geq 2\}$$

Il devient alors possible d'évaluer, à partir des caractéristiques communes à G_1 et G_2, la similarité entre G_1 et G_2 qui peut être définie comme le rapport de leurs caractéristiques communes sur le nombre total de caractéristiques des deux graphes.

$$Sim_{Mp}(G_1, G_2) = \frac{f(decrip(G_1) \cap^{Mp} decrip(G_2)) - g(Sp(Mp))}{f(decrip(G_1) \cup decrip(G_2))}$$

où f et g sont deux fonctions introduites pour pondérer les caractéristiques et les splits en fonction de l'application considérée.

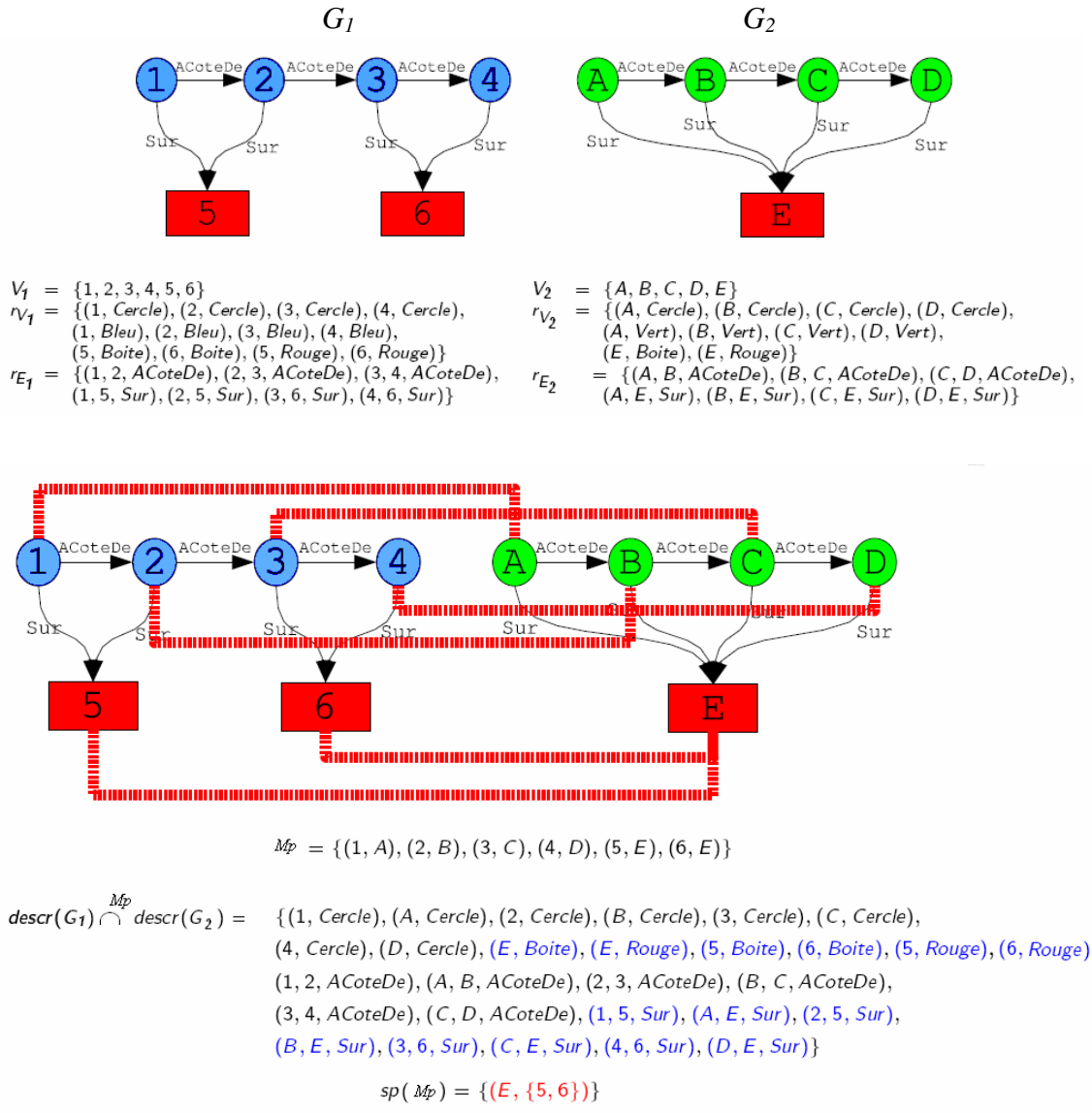


Fig. 1.45 : Description d'objets à l'aide de graphes étiquetés et mise en correspondance obtenue par un appariement *Mp* [Sorlin, 2005b].

Afin, d'éviter une recherche exhaustive des meilleurs appariements au sens de la mesure de similarité définie, un algorithme sous optimal de recherche du meilleur appariement a été proposé à l'aide d'un algorithme glouton (figure 1.46).

La recherche débute avec une carte des mises en correspondance vide *m* et, à chaque itération, un couple de sommets candidats (u_1, u_2) est autorisé à entrer dans la carte des mises en correspondance courantes à condition qu'il y ait augmentation maximale du

score de similarité. S'il y a plus d'un couple candidat qui fait augmenter la fonction de score de manière similaire, un score de second niveau est calculé sur la base de l'analyse des arcs associés aux sommets candidats. Cette condition aide au choix des sommets à faire rentrer dans l'appariement à l'aide d'une fonction appelé « look_ahead » qui teste l'efficacité d'un couple candidat avant de l'autoriser à rentrer dans l'appariement. Ce processus d'insertion itératif de couples de nœuds associés continue jusqu'à ce que le score ait cessé d'augmenter et qu'il n'y ait plus, ni de nœuds, ni d'arcs, à ajouter dans la carte des mises en correspondance courante.

fonction Glouton(G_1=$\langle V_1, r_{V_1}, r_{E_1}\rangle$, G_2=$\langle V_2, r_{V_2}, r_{E_2}\rangle$)
retourne un appariement $m \subseteq V_1 \times V_2$
  $m \leftarrow \emptyset$
  $best_m \leftarrow \emptyset$
  itérer
    $cand \leftarrow \{(u_1,u_2) \in V_1 \times V_2 - m \mid score(m \cup \{(u_1,u_2)\}) \text{ est maximal}\}$
    $cand' \leftarrow \{(u_1,u_2) \in cand \mid f(look\_ahead(u_1,u_2)) \text{ est maximal}\}$
      où $look\_ahead(u_1,u_2)$=$\{(u_1,v_1,l) \in r_{E_1} \mid \exists v_2 \in V_2, (u_2,v_2,l) \in r_{E_2}$
        $\cup \{(u_2,v_2,l) \in r_{E_2} \mid \exists v_1 \in V_1, (u_1,v_1,l) \in r_{E_1}$
        $\cup \{(v_1,u_1,l) \in r_{E_1} \mid \exists v_2 \in V_2, (v_2,u_2,l) \in r_{E_2}$
        $\cup \{(v_2,u_2,l) \in r_{E_2} \mid \exists v_1 \in V_1, (v_1,u_1,l) \in r_{E_1}$
        $-descr(G_1) \sqcap_{m\{(u_1,u_2)\}} descr(G_2)$
  sortir si $\forall (u_1,u_2) \in cand', score(m \cup \{(u_1,u_2)\}) \leq score(m)$
    et $look\_ahead(u_1,u_2) = \emptyset$
    choisir aléatoirement un couple (u_1,u_2) dans $cand'$
    $m \leftarrow m \cup \{(u_1,u_2)\}$
    si $score(m) > score(best_m)$ **alors** $best_m \leftarrow m$ **finsi**
  fin itérer
  retourner m

Fig. 1.46 : Recherche gloutonne d'un appariement.

Cependant, l'algorithme glouton peut retourner un appariement optimal localement puisque l'ajout ou la suppression d'un couple de sommets dans l'appariement n'est jamais remis en cause.

Cet algorithme peut être amélioré en ajoutant et/ou en supprimant plus qu'un couple de sommets à chaque itération. Sorlin et Solnon ont proposé un tel mécanisme via un algorithme Tabou [Sorlin, 2005a] (vois fig. 1.47). A partir d'un bon appariement initial, obtenu avec l'algorithme glouton, l'espace de recherche est exploré de proche en proche jusqu'à ce qu'une solution optimale soit trouvée (quand la valeur optimale est connue), ou jusqu'à effectuer un nombre maximum de mouvements. A chaque étape, une heuristique sélectionne les prochains voisins à explorer. Pour éviter de rester autour

d'appariements localement optimaux en effectuant toujours les mêmes mouvements, une liste taboue est utilisée. Cette liste de longueur *k* mémorise les *k* derniers mouvements (les *k* derniers couples ajoutés/supprimés) afin d'interdire les mouvements en arrière (ajouter/supprimer un couple récemment ajouté/supprimé). Une exception nommée aspiration est ajoutée : si un mouvement interdit atteint un appariement de meilleure qualité que le meilleur appariement connu, le mouvement est toujours effectué.

La longueur *k* de la liste taboue est un paramètre critique et difficile à choisir : si la liste tabou est trop longue, la diversification de la recherche devient tellement importante que l'algorithme converge très lentement. Si la liste est trop courte, l'algorithme pourrait rester bloqué autour d'un maxima local, et ainsi, échouer à améliorer la solution actuelle.

fonction Tabou$(G = \langle V, r_V, r_E\rangle, G' = \langle V', r_{V'}, r_{E'}\rangle, k, limiteQualite, maxMouv)$
retourne un appariement $m \subseteq V \times V'$
  $m \leftarrow Glouton(G, G')$; $best_m \leftarrow m$; $nbMouv \leftarrow 0$
  tant que $sim_{best_m}(G, G') < limiteQualite$ **et** $nbMouv < maxMouv$ **faire**
    $cand \leftarrow \{m' \in voisinage(m) / sim_{m'}(G, G') > sim_{best_m}(G, G')\}$
    si $cand = \emptyset$ **alors**/* *pas d'aspiration* */
      $cand \leftarrow \{m' \in voisinage(m) / pasTabou(m, m', k)\}$
    fin si
    $cand \leftarrow \{m' \in cand / m'$ est maximal par rapport au critère de glouton$\}$
    choisir aléatoirement $m' \in cand$
    $rendreTabou(m, m', k)$; $m \leftarrow m'$; $nbMouv \leftarrow nbMouv + 1$
    si $sim_m(G, G') > sim_{best_m}(G, G')$ **alors** $best_m \leftarrow m$ **fin si**
  fin tant que
  retourner $best_m$

Fig. 1.47 : Algorithme Tabou.

Afin de résoudre le problème d'optimisation du paramètre, [Battiti, 2001] introduit la recherche taboue réactive où la longueur de la liste taboue est dynamiquement adaptée durant la recherche. Pour que l'algorithme tabou soit réactif, il est nécessaire d'évaluer le besoin de diversifier la recherche. Quand le même appariement est exploré deux fois, la recherche doit être diversifiée. Afin de détecter de telles redondances, une clé de hachage est mémorisée pour chaque appariement exploré. Quand une collision arrive dans la table de hachage, la longueur de la liste est augmentée. A l'opposé, quand il n'y a pas de collision durant un nombre fixé de mouvements, ceci indique que la recherche est suffisamment diversifiée qu'on peut réduire la longueur de la liste taboue.

1.7 « *Graph-Mining* »

Les méthodes présentées jusqu'à présent nécessitent d'avoir segmenter correctement les formes, puis de les avoir transformées en graphes pour permettre leur comparaison. D'autres techniques visent plutôt à détecter des sous-graphes fréquents dans des graphes de grand taille, on parle alors de « *Graph Mining* » plutôt que de « *Graph Matching* ». Ces méthodes sont utilisées en recherche d'information et fouilles de données [Barbu, 2004] plutôt qu'en analyse d'images ou reconnaissance des formes. Mais lorsque l'on s'intéresse à la problématique de l'indexation d'images, ces approches restent pertinentes [Iváncsy, 2004].

Les algorithmes pour détecter les sous-graphes fréquents dans des graphes varient d'une méthode à l'autre mais la plupart des méthodes se base sur le principe proposé par les auteurs de [Agrawal, 1993]. L'idée de ces algorithmes est de traiter la base de données (souvent immense) selon différents niveaux (du plus simple aux plus complexes). Le procédé de recherche est donc incrémental et la fréquence des graphes est testée en ajoutant un sommet (ou un arc) dans les sous-graphes fréquents déjà trouvés au niveau précédent. Les algorithmes utilisent ensuite les approches classiques de comparaison de graphes comme par exemple « depth-first search » pour détecter les graphes similaires par exploration de l'espace de recherche.

Barbu [Barbu, 2006] propose d'utiliser un tel mécanisme pour rechercher des symboles (sous graphes fréquents) dans des graphes. Il se base sur l'hypothèse que les sous-graphes fréquents détectés doivent correspondre aux symboles dans l'image. Cette méthode utilise une technique de *clustering* pour la détection des sous-graphes fréquents. La détermination des seuils sur la fréquence d'apparition, nécessaire pour qu'une forme soit considérée comme symbole, reste difficile et des techniques de filtrage sophistiquées sont actuellement indispensables sans pour autant résoudre complètement les problèmes.

1.8 Bilan

La littérature abordant le sujet de la comparaison ou de la recherche de graphes montre l'importance de l'utilisation des graphes pour décrire et représenter les formes, en particulier les images. Malgré l'élégance de cet outil, la majorité des algorithmes existants doivent faire face à un problème de complexité. L'isomorphisme de sous-graphe est un problème NP-complet [Garey, 1979] et la question de complexité de l'isomorphisme de graphe n'a pas encore de bonne réponse. La plupart du temps, la complexité dans le pire des cas augmente exponentiellement avec le nombre de sommets, ce qui limite l'application des méthodes aux petits graphes d'une dizaine de sommets.

Quelques algorithmes proposent de réduire la complexité en fixant des contraintes sur le type des graphes, comme les graphes planaires [Hopcroft, 1974] ou les arbres [Aho, 1974]. La détection de cliques maximales [Falkenhainer, 1989] nécessite un espace mémoire encore trop important. L'arbre de décision [Messmer, 1995] permet de résoudre le problème du temps mais au prix de prétraitement très couteux en temps et en mémoire. L'algorithme VF et VF2 [Cordella, 2001] traitent que les graphes orientés et ne s'intéressent qu'à la structure du graphe alors que les propriétés associées aux sommets et des arcs sont indispensables pour la reconnaissance d'objets.

Actuellement, la tendance est d'explorer les méthodes de comparaison et recherche inexacte de graphes et plusieurs auteurs ont contribué significativement à faire avancer cet axe. Bien que, les algorithmes proposés autorisent une tolérance relative et réduisent la complexité (temps polynomial), la plupart des solutions ne renvoient pas une solution idéale (non optimale ou incomplète).

Dans le cadre de l'analyse d'images de documents, il nous semble évident que seules les méthodes de recherche/comparaison de graphes inexactes méritent d'être utilisées si on veut rester suffisamment résistant aux bruits et distorsions.

De plus, nous nous intéressons, non seulement à la reconnaissance, mais aussi à la localisation de formes dans leur contexte (images complètes), les informations sur les mises en correspondance des sommets constituent donc également une connaissance indispensable pour nous afin de produire un système capable d'interpréter complètement les documents.

Compte tenu de ces objectifs, et après avoir analysé les différentes méthodes de la littérature, c'est le travail proposé par Champin & Solnon [Champin, 2003] qui nous semble le plus intéressant et le plus générique ; nous avons donc choisi de nous inspirer de ces travaux pour mettre au point de nouvelles méthodes d'analyse d'images de documents graphiques.

Chapitre 2

Principales méthodes de localisation et de reconnaissance de symboles graphiques

Les deux domaines les plus étudiés de l'analyse d'images de documents se distinguent par le type d'éléments à analyser : si les entités à analyser sont principalement textuelles (document composite), les recherches vont se situer dans la problématique de la reconnaissance optique de caractères (OCR), si les entités à analyser sont à composantes majoritairement graphiques (schéma électronique, diagramme d'architecture, musique, tableau, logo, formules ou formes complexes) les travaux concernent alors le domaine de reconnaissance de dessins et symboles graphiques. Des systèmes commerciaux d'OCR sont déjà disponibles aujourd'hui et procurent de très bons taux de reconnaissance de l'ordre de 99% sur les documents contemporains. Par contre, il existe très peu de systèmes suffisamment génériques et capables de reconnaître des symboles graphiques en contexte avec des taux de reconnaissance suffisants. La reconnaissance de symboles graphiques semble donc plus complexe que la reconnaissance de caractères notamment car la segmentation est complexe et car le nombre et la variété de symboles à reconnaître sont beaucoup plus élevés. Les recherches dans ce domaine ont débuté dans les années 1980 lors de la démocratisation des campagnes de numérisation des documents. L'objectif est de convertir les plans techniques du format papier au format électronique. Progressivement, des méthodes pour la reconnaissance de tableaux, de

diagrammes, et de partitions musicales ont été présentées. Depuis une dizaine d'années, le sujet est traité encore plus sérieusement et des concours ont été organisés afin de mieux évaluer les performances des différentes méthodes mélangeant des techniques de reconnaissance des formes et d'intelligence artificielle.

Dans cette section, nous décrivons les différentes techniques développées au cours de ces dernières années dans le cadre de la reconnaissance de symboles graphiques sans a priori. Les méthodes utilisant des connaissances a priori fortes sur les documents à analyser et donc dédiées uniquement à un type de document précis ne sont quant à elles pas présentées dans ce manuscrit.

2.1 Documents graphiques et types de symboles

Différents types de symboles et de documents peuvent être construits avec différentes propriétés visuelles. Les formes binaires simples composées de segments et d'arcs se retrouvent dans les schémas électroniques et les plans d'architecture (figure.2.1). Une combinaison de lignes et de formes pleines peut correspondre à des partitions musicales (figure 2.2). Les symboles en niveaux de gris ou en couleur apparaissent fréquemment dans les logos (figure 2.3) [Lladós, 2003]. Tous ces symboles ont une signification particulière qui dépend du domaine spécifique d'application. Ils peuvent aussi être associés les uns aux autres selon des règles pouvant aider à les identifier.

A l'intérieur des travaux scientifiques du domaine, il est important de distinguer les recherches qui se focalisent sur les méthodes de reconnaissance de symboles pré-segmentés (majorité des travaux), des méthodes fonctionnant sur des images de documents complets (travaux marginaux actuellement). Lorsque les symboles sont déjà segmentés, le problème consiste à identifier la classe de l'image parmi les n modèles possibles pour le symbole contenu dans l'image. Dans le $2^{ème}$ cas, un double processus est nécessaire : tout d'abord, il faut localiser et segmenter les symboles, puis lancer des procédures pour reconnaître les différentes entités extraites. La segmentation des symboles (appelée aussi localisation des symboles) est un problème beaucoup plus difficile que les problèmes de séparation texte/graphique souvent discutés dans la littérature scientifique et correctement résolu aujourd'hui : l'algorithme de Fletcher et Kasturi utilise une analyse des composantes connexes et une transformation de Hough [Fletcher, 1988] pour localiser le texte dans les documents graphiques mais ne permet pas de segmenter des symboles dans une image.

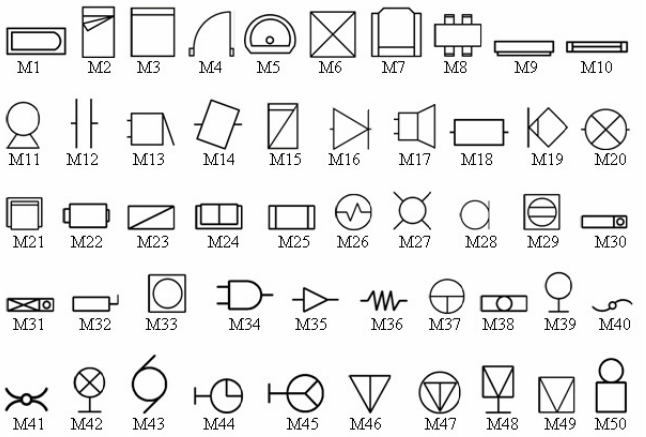


Fig. 2.1 : Symboles linéaires composés de lignes et d'arcs.

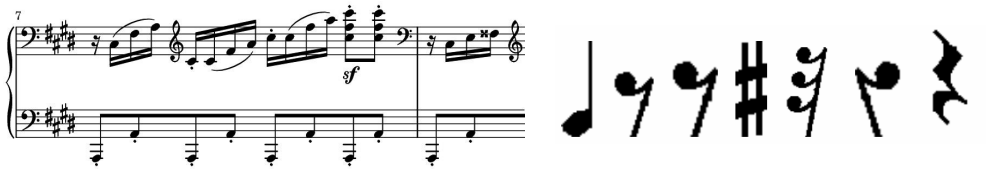


Fig. 2.2 : Symboles musicaux composés de lignes et de formes pleines interconnectées.

Fig. 2.3 : Quelques exemples de logos en niveaux de gris ou couleur.

En fonction du type de symboles rencontrés dans les documents, nous distinguons trois familles principales de documents graphiques.

1) Les schémas techniques qui incluent les plans électriques et électroniques, les diagrammes logiques, les organigrammes, les dessins mécaniques, les schémas architecturaux et les plans de service (cartes d'énergie électrique, plans de gaz et de téléphone, etc.)

2) Les documents correspondant aux cartes géographiques, topographiques, cadastrales et hydrographiques.

3) Les autres documents contenant des symboles de types spéciaux, tels que les symboles musicaux, des symboles mathématiques, des formules chimiques structurées …

Dans cette thèse, nous nous intéresserons principalement aux familles 1 et 2 pour ce qui concerne aussi bien la reconnaissance de symboles isolés que leur localisation dans des documents complets.

D'une manière générale, le traitement menant à l'interprétation d'une image est constitué par trois phases principales. Pendant la première, que nous nommerons phase de **construction de la représentation**, l'objectif est de mettre en place une description plus appropriée que l'image pour les traitements suivants (pour localiser et pour décrire les zones d'intérêt présentes dans les documents). La deuxième étape est la phase de **localisation/segmentation** durant laquelle les parties significatives (symboles) sont extraites de la représentation. Enfin, la troisième phase a pour objectif de classifier les parties significatives afin d'obtenir **une interprétation du contenu**.

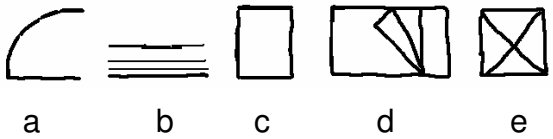
Ce découpage correspond plutôt à l'usage d'une stratégie ascendante (Bottom-up). Comme son nom l'indique, celle-ci débute avec une représentation bas niveau (matrice des pixels) et évolue vers des représentations de plus haut niveau jusqu'à des comparaisons finales entre les parties significatives de l'image et un ensemble de prototypes décrivant les symboles à reconnaitre. A part dans les niveaux élevés d'abstraction, les méthodes ascendantes n'emploient pas de modèle de document. Par opposition, les stratégies descendantes se basent sur un modèle du document (et des objets à trouver) pour essayer d'adapter les données au modèle. Elles fonctionnent par itérations successives, l'interprétation du contenu est raffinée par la détection de caractéristiques de plus en plus détaillée sur les objets décrits dans le modèle [Cordella, 2000c].

Les sections suivantes de ce chapitre décrivent les principales contributions concernant les trois phases menant à l'interprétation de l'image du document graphique. Nous ne nous attardons cependant pas sur les méthodes de vectorisations déjà présentées et comparées de nombreuse fois dans la littérature. Nous avons plutôt choisi de privilégier les techniques récentes de localisation et reconnaissance de symboles.

2.2 Localisation de symboles à l'aide de signatures vectorielles

P. Dosch a proposé d'utiliser des signatures vectorielles [Dosch, 2004] pour distinguer les différents symboles plutôt que pour leur reconnaissance. La technique est basée sur l'idée proposée par Etemadi [Etemadi**,** 1991] visant à regrouper des segments en utilisant leurs relations topologiques. La méthode fait une étude de cinq relations topologiques de base pouvant apparaître entre paires de lignes (segments) afin de distinguer les symboles. Les relations concernent les colinéarités, le parallélisme avec ou sans recouvrement, les intersections (L-Jonction et V-Jonction). Les signatures vectorielles de quelques symboles graphiques sont fournies tableaux 2.1. Les symboles de référence (modèles) sauvegardés dans la bibliothèque de symboles, sont analysés afin de calculer leur signature. Ensuite, l'image est décomposée en plusieurs petites fenêtres. La taille de la fenêtre est définie de manière relative par rapport à la taille du plus grand symbole modèle dans la base de référence. Cette méthode est donc très sensible à la résolution de l'image et suppose que les formes à localiser sont de taille strictement identique aux modèles. La signature de chaque fenêtre est comparée avec les signatures des symboles de référence pour classer le contenu de la fenêtre.

Compte tenu des signatures calculées, l'approche est invariante aux transformations affines mais ne peut être considérée que comme une méthode basique pour localiser les symboles. L'auteur indique clairement que la signature vectorielle n'est pas développée pour résoudre le problème difficile de localisation ou de la reconnaissance de symboles. Son but principal est la discrimination rapide et la définition des régions d'intérêt dans une image. De plus, les tests réalisés sur les plans d'architecture montrent que beaucoup de fausses alarmes sont générées, particulièrement avec des symboles non présents dans la bibliothèque et avec les coins qui, après la vectorisation, ont une représentation proche de la représentation des portes. Cette méthode est également sensible au positionnement de la grille servant à découper l'image en fenêtres locales comme le montre la figure 2.4.



Tab. 2.1 : Les signatures vectorielles de quelques symboles graphiques.

Symboles	Parallèles avec recouvrement	Parallèles sans recouvrement	Colinéaire	L Jonction	V Jonction
(a)	1	0	0	0	4
(b)	6	0	0	0	0
(c)	2	0	0	4	0
(d)	6	4	2	10	15
(e)	2	3	0	4	12

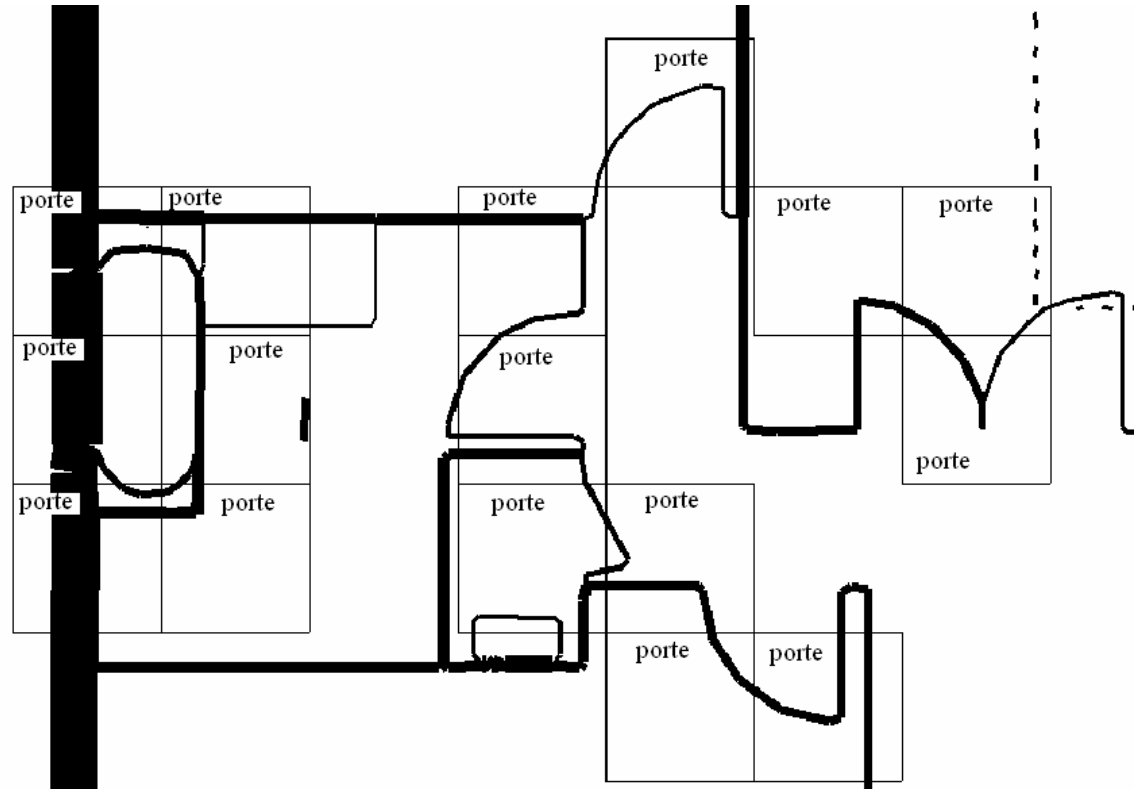


Fig. 2.4 : Quelques résultats obtenus par signature vectorielle.

2.3 Localisation interactive de symboles graphiques

Une approche interactive pour identifier des objets graphiques dans des schémas de technologie est proposée dans [Luo, 2003] [Wenyin, 2007]. L'utilisateur fournit un exemple d'un type d'objets graphiques en le détourant dans un schéma de technologie. Le système est alors capable d'apprendre les « connaissances » nécessaires pour reconnaître ou pour rechercher d'autres objets graphiques semblables. La « connaissance générique » utilisée pour apprendre correspond à quatre types de contraintes géométriques prédéfinis que sont l'intersection entre les lignes, le parallélisme, la perpendicularité et d'autres contraintes pour les arcs et les cercles.

Le système extrait toutes les contraintes géométriques et construit un graphe *G* (*V*, *E*). Dans ce graphe, les nœuds représentent les primitives vecteurs et les arcs représentent les contraintes géométriques existantes entre les vecteurs. Si le graphe n'est pas entièrement relié (il est divisé en sous-graphes), le système essaye d'employer le parallélisme ou les contraintes géométriques de perpendicularité pour relier les sous-graphes. Quand le graphe *G* (*V*, *E*) est connexe, l'arbre couvrant minimum est extrait à partir de *G* (*V*, *E*) (Figure. 2.5). Le système utilise alors l'algorithme « Bredth First Search » (BFS) lancé nécessairement sur chaque vecteur, pour trouver le vecteur semblable parmi les vecteurs des modèles. De ce fait, l'analyse continue jusqu'à ce que tous les vecteurs aient été traités.

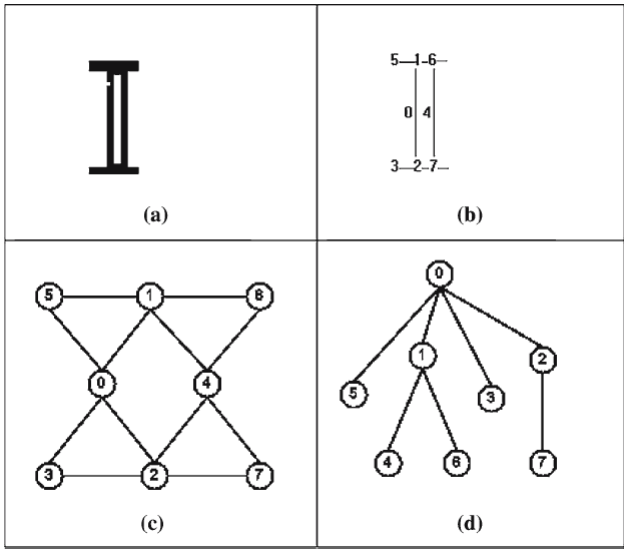


Fig. 2.5 : Extraction de l'arbre couvrant minimum d'un symbole.

Dans ce système, la construction de la représentation vectorielle (qui inclue une vectorisation de l'image) est réalisée en utilisant un logiciel libre appelé Ras2Vec [WEB-1] qui reste à améliorer. Nous avons vu que la reconnaissance est basée sur un graphe connexe, par conséquent, si le système n'arrive pas à construire un graphe connexe : les connaissances associées à l'objet graphique ne peuvent pas être apprises (extraction de contraintes géométriques).

La transformation du graphe en *un arbre couvrant minimum* est également une étape risquée car il y a plus qu'*un arbre couvrant minimum* possible pour un même graphe. Par exemple, dans la figure 2.5(d), la relation (contrainte géométrique) entre les nœuds 4 et 7, ou 4 et 6 est perdue à la réduction. En outre, les résultats d'identification restent très sensibles aux variations de formes ou d'environnement pour un même symbole : la connaissance graphique est apprise sur un seul exemple simple, difficile ainsi d'identifier les caractéristiques stables et celles liées à l'environnement de l'exemple. Rappelons enfin que ce système ne fonctionne que par l'intermédiaire de nombreuses interactions homme-machine sur chacune des images à analyser.

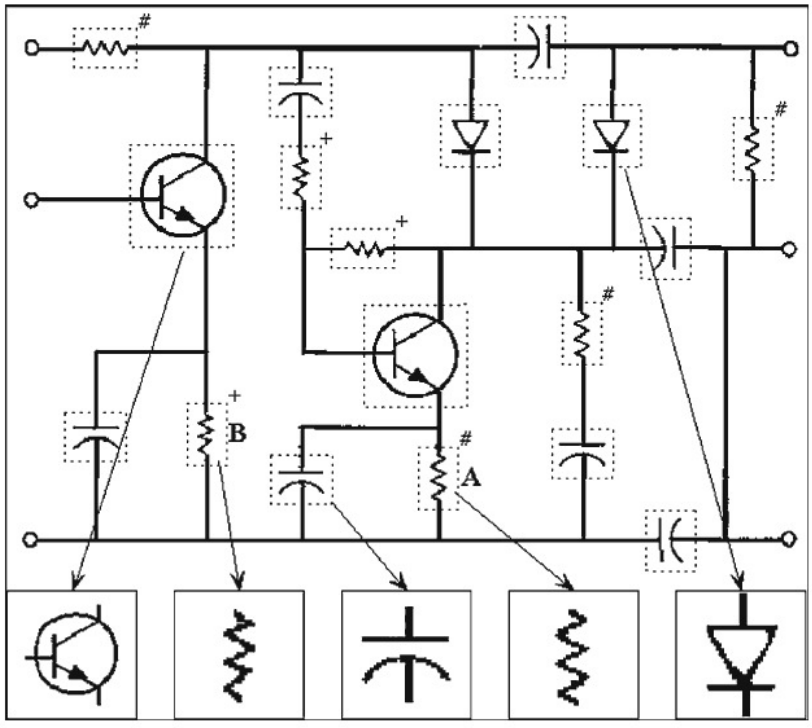


Fig. 2.6 : un exemple d'analyse interactive de schéma.

2.4 Localisation de symboles graphiques par découpage hiérarchique des traits

Daniel Zuwala a présenté une nouvelle méthode pour segmenter des symboles dans des documents graphiques par une approche de découpage du document en chaînes de points clés [Zuwala, 2006]. Le système se base sur une méthode de description structurelle qui va permettre de mettre en avant un certain nombre de régions pouvant contenir un symbole.

Le document graphique est binarisé (fig. 2.7(a)), puis squelettisé (fig. 2.7(b)). Ensuite, les points de jonctions (c'est-à-dire les points du squelette ayant plus de deux voisins), et les points terminaux (les points n'ayant qu'un seul voisin) sont marqués dans le squelette (fig. 2.7(c)). Les chaînes de points correspondent aux ensembles de points connectés ne possédant que deux voisins, et dont les extrémités sont, soient des points terminaux, soit des points de jonctions (fig. 2.7(d)).

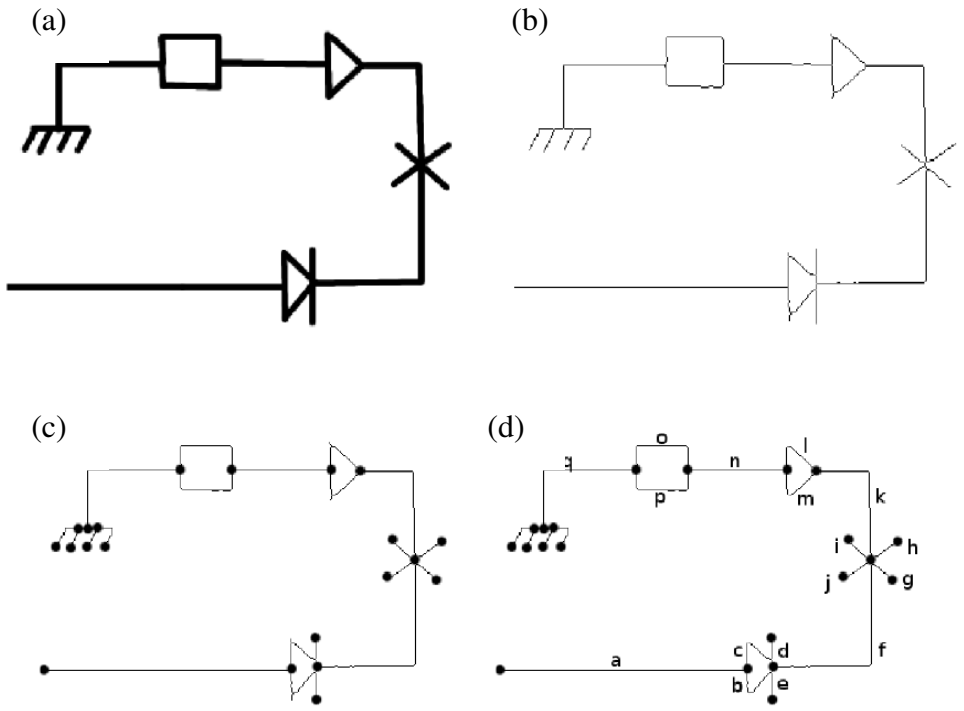


Fig. 2.7 : Pré-traitement du document (a) Document binarisé (b) Document squelettisé (c) Localisation des points de jonctions (d) Décomposition en chaînes de points [Zuwala, 2006].

Un graphe de jonctions est alors construit, dans lequel, les nœuds représentent les chaînes de points, tandis que deux chaînes de points connectées sont reliées par un arc. Une fusion successive des nœuds de ce graphe de jonction est réalisée pour construire un dendrogramme (fig. 2.8). Pour guider les fusions dans le dendrogramme, un critère d'agrégation est basé sur les deux hypothèses suivantes :

– un symbole est un ensemble compact de chaînes de points connectées
– les chaînes de points d'un symbole ont tendance à être convexes

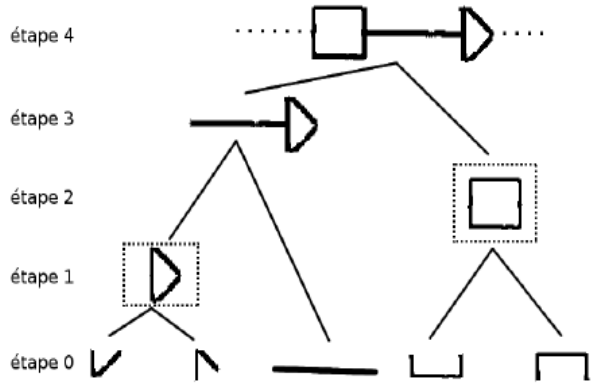


Fig. 2.8 : Une partie du dendrogramme pour un simple document composé de deux symboles simples (un carré et un triangle) reliés par une ligne [Zuwala, 2006].

Chaque sommet de dendrogramme est considéré comme un symbole probable. L'utilisateur peut alors soumettre une requête directement issue d'un des documents analysés. La requête (un symbole) correspond à un dendrogramme ou à un morceau de dendrogramme. Les symboles candidats dans le dendrogramme appris précédemment sont filtrés en utilisant des critères géométriques simples (largeur et hauteur) par rapport à la requête. Ensuite, un descripteur pixel (l'ART - Angular Radial Transform [Kim, 1999]) est calculé pour tous les symboles restants. Les *n* meilleurs symboles obtenus en calculant une distance euclidienne sont classés par ordre croissant de distance par rapport à la requête. Ils sont ensuite présentés à l'utilisateur qui peut choisir les symboles les plus pertinents et les moins pertinents afin de produire un bouclage de pertinence permettant à l'utilisateur d'obtenir satisfaction.

Les hypothèses utilisées considèrent les symboles comme des ensembles de chaînes de points compactes et convexes. De nombreux problèmes restent donc à résoudre, par exemple, les symboles qui sont composés de chaînes de points non connectées (comme

par exemple les capacités) ou les symboles ne présentant pas les points de jonctions (comme les résistances) ne sont pas traités. De plus, le nombre de symboles candidats retournés par le système est énorme (200 000 symboles pour 100 documents), alors la méthode d'indexation proposée doit être améliorée afin de présenter une possibilité de recherche approximative.

Peu d'autres travaux proposent des méthodes suffisamment génériques de localisation de symboles dans les documents graphiques. La plupart des systèmes proposés jusqu'à présent sont dédiés à des applications précises et utilisent des connaissances a priori (heuristiques) pour déterminer la position des symboles. Nous ne détaillons pas ces approches ici. Concernant l'étape de reconnaissance, dans le chapitre 1, nous avons étudié les méthodes classiques de comparaison de graphes, les sections qui suivent, reviennent sur certaines de ces méthodes afin d'expliciter comment elles ont été adaptées et appliquées à la reconnaissance de symboles graphiques.

2.5 Reconnaissance de symboles à l'aide de graphes de contraintes

La méthode de Messmer [Messmer, 1995] et ses dérivées (Messmer et Bunke [Messmer, 1996]) combine reconnaissance de formes structurelles et apprentissage pour résoudre le problème de reconnaissance des symboles graphiques. L'approche par décomposition est utilisée et chaque symbole modèle est découpé en ses sous-parties récursivement jusqu'à aboutir à des primitives de base (vecteurs). L'ensemble de symboles peut ainsi être représenté par une hiérarchie (un arbre), dans lequel chaque nœud correspond à une sous-partie d'un symbole. La figure. 2.9 montre l'arbre proposé pour la représentation des symboles 0, 1, 2 et 3.

Chaque nœud de l'arbre (partie de symbole) est un graphe relationnel attribué (GRA). Les nœuds du GRA sont les vecteurs et leur longueur constitue un attribut. Les arcs représentent les relations entre les vecteurs. L'attribut associé aux arcs est l'angle entre les vecteurs. Un symbole (ou partie de symbole) est alors structurellement décrit comme un sous-graphe particulier du graphe entier.

Pendant la phase de reconnaissance, les isomorphismes de sous-graphe sont progressivement examinés à partir des nœuds élémentaires de l'arbre. Pour tenir compte des déformations qui peuvent affecter la forme des symboles réels, des opérations d'édition sont tolérées pendant la mise en correspondance comme l'insertion de nœuds

et d'arcs, la fusion de nœuds et une tolérance d'erreur sur des valeurs d'attribut est gérée. Un coût est associé à chaque opération d'édition. Une mise en correspondance est acceptée si son coût est inférieur à une valeur seuil. Cependant, la structure du système autorise uniquement le traitement des symboles composés de lignes connectées. Les formes pleines, les symboles comportant des traits plus épais et les symboles comportant plusieurs parties doivent être traités séparément.

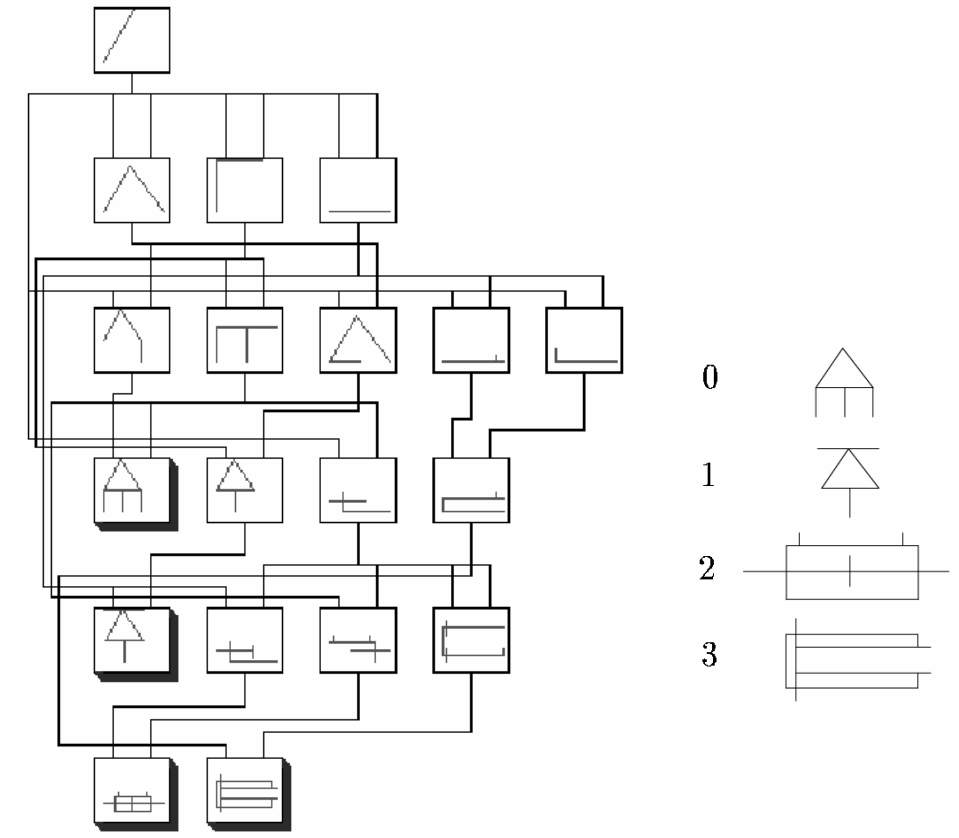
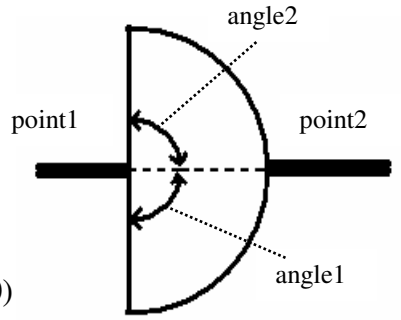


Fig. 2.9 : Un exemple d'arbre de symboles graphiques [Messmer, 1995].

Christian Ah-Soon [Soon, 1997] a apporté quelques améliorations à la méthode de Messmer pour détecter des symboles dans les plans architecturaux. Les symboles sont représentés comme des ensembles de contraintes sur les segments et les arcs. Un langage de description a été défini afin de décrire ces contraintes. Les deux types de contraintes étudiées sont les contraintes de connexion qui expriment des relations entre les extrémités de deux segments ou arcs, et les contraintes simples qui expriment toutes autres contraintes pouvant être examinées (direction, longueur…). Toutes les

descriptions sont stockées dans des fichiers textes composés d'un ensemble d'entrées. Chaque entrée correspond à une description d'une partie de symbole et contient principalement un ensemble de contraintes sur les lignes qui constituent le symbole. Ces entrées sont écrites dans une grammaire qui a été spécifiquement développée pour cet usage. Par exemple, la description d'une porte est donnée comme :

```
   #porte SEGMENT 1 ARC 1
arc1.angle ( ) <= 90;
segment1. taille ( ) <=100;
arc1.rayon ( ) == segment1.taille ( );
arc1.point1 ( ) == segment1.point1 ( ) ;
arc1.centre ( ) ==segment1.point2 ( ) ;
porte (arc1.centre ( ), acr1.point2 ( ), arc1.rayon ( ), 0)
```



Dans cet exemple, la construction du réseau de contraintes commence par la création des deux nœuds de racine : NNSegment et NNArc. Après initialisation, la construction est un processus itératif et les contraintes sont créées séquentiellement pour chaque partie du symbole. Afin de mesurer la déviation entre un modèle et le symbole à reconnaître, une erreur est calculée sur les contraintes. Sur les nœuds par exemple, si la contrainte concerne l'égalité entre deux longueurs, l'erreur est la différence entre ces deux longueurs. Pour éviter ou réduire les effets négatifs des traitements de bas niveau, le réseau est utilisé seulement sur les lignes fines de l'image. La détection de symboles est effectuée en propageant les contraintes sur les nœuds et les arcs dans le réseau de contraintes.

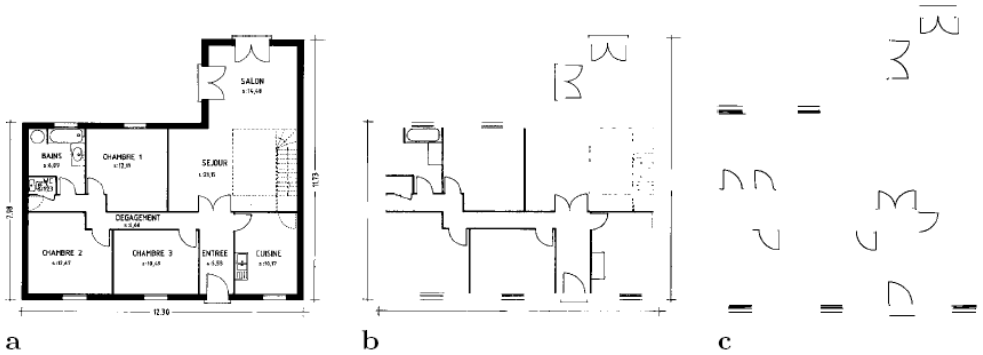


Fig. 2.10 : a) Schéma d'architecture, b) Lignes fines, c) Symboles détectés [Soon, 1997]

2.6 Reconnaissance à base de graphes d'adjacences

– Méthode de Lladós

L'algorithme d'isomorphisme de graphe/sous-graphe tolérant aux erreurs formulé en termes de graphe d'adjacence de régions proposé dans [Lladós, 2001a] (voir chapitre 1) à été utilisé pour reconnaître des symboles dans des diagrammes dessinés à main levée. Dans cette approche, lors d'une première étape, le document est vectorisé et converti en graphe planaire. Les nœuds du graphe représentent les points caractéristiques du squelette comme les points de jonction, les points extrémités ou les coins. Les arcs du graphe correspondent aux segments entre les points caractéristiques. À partir de cette structure initiale, un second graphe (d'adjacence de régions) est ensuite développé. Les nœuds représentent les régions et les arcs les relations de voisinage ou relations d'adjacence (fig. 2.11). Pour ce faire, l'algorithme de Jiang [Jiang, 1993] est utilisé pour fournir les boucles fermées minimales du graphe planaire construit précédemment. Ensuite, pour la mise en correspondance des graphes, une distance d'édition classique est utilisée pour calculer la similarité entre les chaînes décrivant les frontières des régions contenues dans les graphes (voir chapitre 1). Actuellement, dans les chaines, chaque terme représente la longueur et l'orientation du segment formant la frontière de la région (fig. 2.12). La distance considérée est une somme pondérée du coût des opérations d'édition (la suppression, l'insertion et la substitution des nœuds et des arcs pour transformer un graphe en un autre graphe). Puisqu'un nœud du second graphe représente une région, la suppression de nœud correspond à la suppression de la région entière et par conséquent cette opération d'édition n'est pas considérée. La méthode est robuste aux distorsions vectorielles. Des difficultés surviennent lorsqu'un graphe modèle est sous-graphe d'un autre graphe modèle (figure 2.13). Cette approche ne peut également pas détecter les symboles non-fermés (comme le symbole d'une porte dans un plan d'architecture, ou une résistance, dans un schéma électronique).

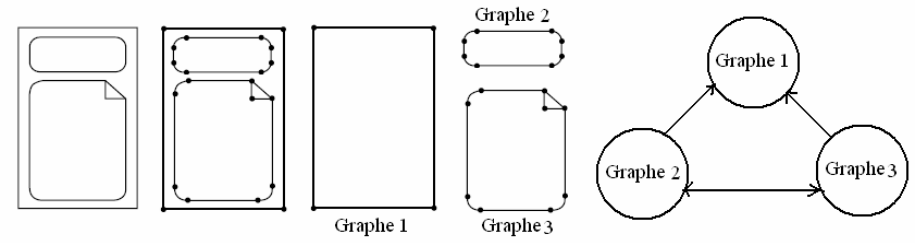
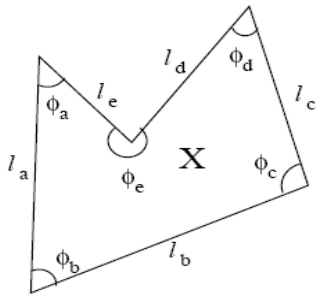


Fig. 2.11 : Un symbole de lit représenté par un graphe de régions.



$$X = \begin{array}{|c|c|c|c|c|} \hline a & b & c & d & e \\ \hline l_a & l_b & l_c & l_d & l_e \\ \hline \phi_a & \phi_b & \phi_c & \phi_d & \phi_e \\ \hline \end{array}$$

Fig. 2.12 : Les longueurs et les orientations des segments d'une frontière.

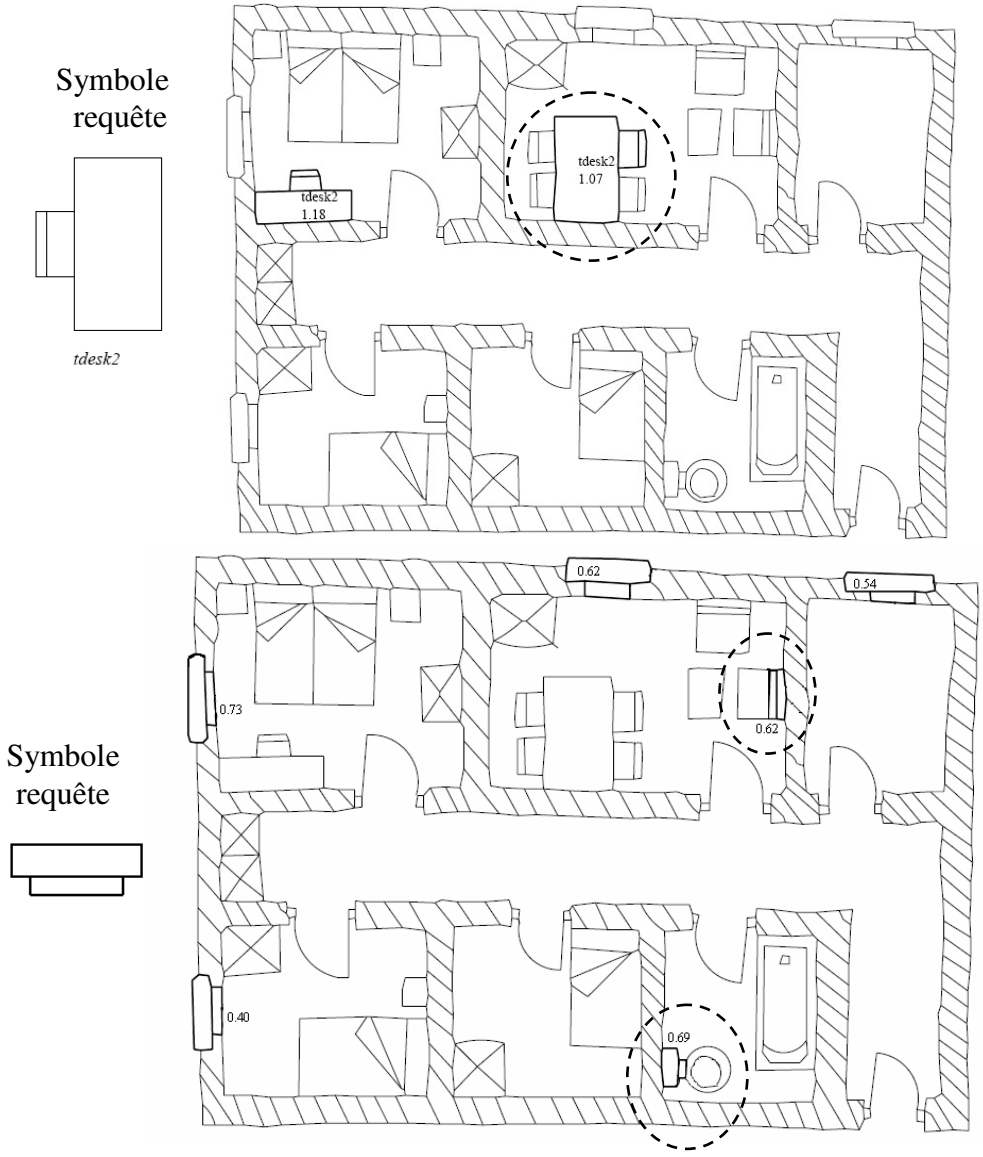


Fig. 2.13 : Erreur à cause de graphes modèles correspondant à un sous-graphe d'un autre graphe modèle.

• Méthode de Locteau

Mathieu Delalandre [Delalandre, 2005] effectue une analyse détaillée des différentes méthodes d'extraction de primitives et de représentation du contenu de documents graphiques. Cette étude démontre l'engouement pour les méthodes structurelles lorsqu'il s'agit de traiter des documents complets. Dans son étude, Matthieu Delalandre parle de multi-représentations mais l'implémentation qu'il propose reste très complexe car elle exploite, en fait, plusieurs représentations sous forme de graphe plutôt qu'une seule fédérant l'ensemble des informations statistiques et structurelles [Delalandre, 2004].

Afin de simplifier ce système, Hervé Locteau [Locteau, 2006] introduit une modélisation des symboles faisant intervenir en parallèle l'approche structurelle et statistique. Un graphe d'adjacence de régions est construit à partir des occlusions contenues dans l'image et est associé à un second graphe basé sur les primitives vectorielles (fig. 2.14 a). Les deux graphes sont ensuite fusionnés en un seul afin d'englober à la fois :

– une définition des contours de l'occlusion à base de primitives vectorielles,
– une définition de la forme de l'occlusion sous forme d'invariants (pseudo-invariants de Zernike).

Les arcs du graphe mettent en évidence les frontières communes à deux régions adjacentes sous forme de liste de primitives vectorielles. Des propriétés telles que la position des centres de gravité, le rapport de surfaces et l'orientation sont également spécifiées [Locteau, 2006]. Un algorithme glouton est ensuite utilisé pour la mise en correspondance de nœuds du graphe candidats avec les nœuds des graphes modèles pour réaliser la recherche d'isomorphismes de sous-graphes.

L'auteur commente que, comme leur système exploite les occlusions et leurs contours vectorisés, les autres éléments du dessin sont négligés et le système ne peut donc pas distinguer certains types de symboles (fig. 2.14b). Lors de la reconnaissance, l'algorithme glouton, est guidée par la modélisation « région » et les relations d'adjacence.

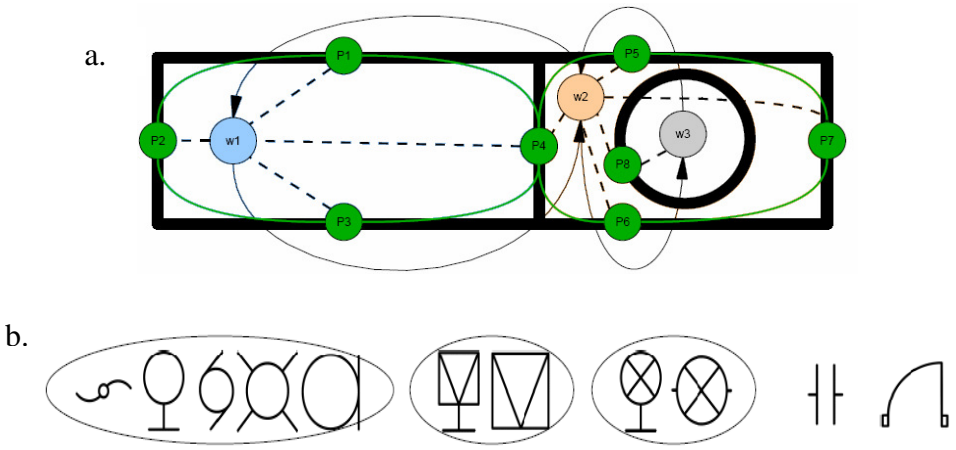


Fig. 2.14 : a) La représentation à base de deux graphes b) les symboles non distingués [Locteau, 2006].

- **Méthode de Barbu**

Récemment, Barbu [Barbu, 2006] a proposé un système capable de classifier des graphes représentant des symboles en faisant intervenir un algorithme génétique pour l'apprentissage de prototypes de graphes. L'approche proposée se décompose en 3 étapes. Dans un premier temps, un algorithme de génération de bruits synthétiques est appliqué sur l'image de chaque symbole modèle afin d'obtenir un ensemble de *M* images de symboles par classe. Ensuite, les composantes connexes (composantes noires) et les occlusions (composantes blanches) sont extraites des images binaires et sont automatiquement étiquetées à l'aide d'un algorithme de classification non supervisée [Kaufman, 1990] en utilisant les moments de Zernike comme caractéristiques [Khotazad, 1990]. Un graphe est ensuite construit, dont les nœuds correspondent aux composantes blanches (occlusions) et noires (composantes connexes) pré-étiquetées. La figure 2.15 montre les graphes construits pour deux images de symboles. Les *M*x*N* images obtenues à partir de *N* classes de symboles sont fournies à un algorithme d'apprentissage dont l'objectif est de générer un ensemble de *K* graphes prototypes par classe. L'algorithme de construction des prototypes de classe choisi est un algorithme génétique qui cherche à maximiser le taux de bonne reconnaissance sur une base de graphes de référence en effectuant des opérations génétiques (mutation, croisement …) entre les graphes modèles.

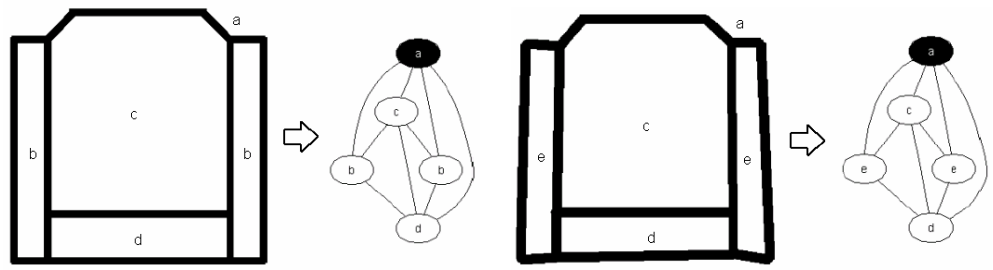


Fig. 2.15 : Construction des graphes [Barbu06].

Le classificateur préconisé par les auteurs pour la phase de création des prototypes (apprentissage) et pour la phase de reconnaissance est un 1-PPV sur la base d'une description des graphes par graph probing [Lopresti, 2001].

2.7 Reconnaissance à l'aide de treillis de Galois

Le treillis de Galois est un graphe particulier dont la structure est proche de celle de l'arbre de décision. Dans l'arbre de décision, un seul chemin, de la racine vers une feuille, permet d'atteindre une classe, alors que dans le treillis tous les chemins possibles pour atteindre une classe sont définis et représentés. De même, plusieurs caractéristiques des formes à reconnaître peuvent être examinées en même temps pour progresser d'un nœud à l'autre contrairement à l'arbre de décision où une seule caractéristique est traitée à la fois. Ils intègrent néanmoins tous les deux la possibilité intéressante de pouvoir traiter des données numériques et symboliques. Avec les treillis de Galois, l'apprentissage se décompose en deux étapes :

– une étape de discrétisation des données numériques : les données sont réparties dans des intervalles disjoints. Cette étape est essentielle à la construction du treillis de Galois et se paramètre par un critère de définition des intervalles.

– une étape de construction du treillis à partir des données discrétisées qui ne nécessitant aucun paramétrage.

La classification consiste ensuite à déterminer la classe de nouveaux objets, plus ou moins détériorés, par navigation dans le treillis. Le treillis de Galois peut être vu comme un espace de recherche dans lequel on évolue par validation de caractéristiques. La navigation débute à partir du concept minimal (O, f (O)) où toutes les classes sont candidates à la reconnaissance et aucun intervalle n'est validé. Il s'agit ensuite de progresser concept par concept (nœud par nœud) au sein du treillis de Galois par validation de nouveaux intervalles et par conséquent, réduction de l'ensemble d'objets possibles. La principale limite de l'utilisation du treillis de Galois est due à son coût à la fois en temps et en espace. Les expérimentations pratiques réalisées dans le domaine de la reconnaissance de symboles graphiques montrent que le treillis de Galois offre un cadre intéressant en classification, malgré une complexité théorique exponentielle dans le pire des cas, mais polynomiale en pratique. L'aptitude de sélection de caractéristiques du treillis de Galois le rend difficile à comparer avec les autres classifieurs. Néanmoins, S. Guillas indique que les treillis de Galois donnent des taux de reconnaissance proches de ceux des autres classifieurs (comme le classificateur bayésien ou le k-ppv) [Guillas, 2006] lorsque l'on l'utilise pour reconnaître des symboles graphiques.

2.8 Reconnaissance à l'aide de descripteurs de formes

Un descripteur de formes (statistique) est un ensemble de caractéristiques extraites à partir de l'image ou de primitives de plus haut niveau qui peut être utilisé pour la tâche de reconnaissance. On parle, la plupart du temps, soit de descripteurs ‘‘pixels’’, soit de descripteurs ‘‘structurels’’. La différence entre les deux se situe dans la façon de représenter le descripteur et implicitement dans la méthode de mise en correspondance entre deux formes. Dans le premier cas, la forme est décrite par un vecteur de caractéristiques à n dimensions. L'autre cas correspond à l'usage de graphes ou de grammaires pour décrire les formes. Les signatures correspondent alors à une collection de primitives extraites à partir de la description vectorielle de la couche graphique (après une vectorisation de l'image). L'ensemble de primitives peut inclure, par exemple, le nombre de points d'intersection entre segments, les différences d'angles entre segments, la longueur ou l'épaisseur des segments et les orientations principales peuvent également être utilisées. Nous ne revenons pas sur les descripteurs structurels dans cette section.

Il est important de noter que, dans tous les cas, la forme doit d'abord avoir été segmentée pour qu'il soit possible de calculer ses descripteurs.

Lorsqu'il s'agit uniquement de reconnaître des symboles pré-segmentés (situation irréaliste dans la pratique), l'usage des descripteurs statistiques s'avère possible et les performances obtenues sont alors souvent supérieures aux résultats obtenus par les méthodes structurelles.

Les techniques décrites dans cette partie ne peuvent donc pas être appliquées directement sur une image de document technique complet (aussi simple soit-elle). Les descripteurs peuvent être définis sur les contours internes et/ou externes ou construit à partir de la forme complète. Les exemples de descripteurs statistiques classiques basés sur les contours sont l'aire, le périmètre, la compacité, la convexité… Ces descripteurs sont en général très rapides à calculer mais sont aussi peu discriminants. Ils ne sont donc utilisés que pour reconnaître des formes simples et donc rarement pour reconnaître des symboles graphiques. Pour traiter les formes complexes, il est nécessaire d'avoir recours à des descripteurs plus évolués comme ceux décrits dans la suite.

- **Moments géométriques**

Certains descripteurs sont réversibles et permettent de reconstruire la forme initiale. Parmi ceux-ci, on peut citer les moments géométriques et autres moments (Zernike, Hu …) [Adam, 2000a] [Adam, 2000b]. Il s'agit probablement de la technique de description de symboles graphiques la plus utilisée en pratique. Sebastien Adam [Adam, 2001] compare, dans sa thèse, ces différents descripteurs et propose finalement d'utiliser le descripteur Fourier-Melin pour reconnaître les composantes textuelles contenues dans les documents techniques. Le principal problème de ce type de descripteur réside dans le choix de la dimension du vecteur (ordre des moments) et dans le choix du mode de calcul des moments : mode local (calcul sur des parties de l'image) ou mode global (calcul des moments sur l'image dans sa globalité).

Une méthode basée sur les moments de Zernike pour la reconnaissance des symboles dessinés à la main est présentée dans [Hse, 2004]. Les moments de Zernike ne sont pas directement invariants au changement d'échelle et translation. Une normalisation est nécessaire de sorte que tous les symboles aient la même dimension et que leur centre soit positionné à l'origine du repère. Ils sont robustes au bruit et aux légères variations des formes ; il n'y a pas de redondance d'information car ils constituent une base orthogonale [Ansary, 2005].

- ***F et R Signatures***

Plus récemment, dans le domaine de la reconnaissance de symboles, l'usage de F-signature et R-Signature a été proposé par l'équipe QGAR [Tabbone, 2003]. Ces deux signatures particulières sont construites à partir d'histogrammes dont l'interprétation permet d'obtenir une information sur les relations spatiales entre des objets ou parties d'objets. Le calcul d'une F-Signature d'un symbole graphique binaire prend en considération une droite dont on fait varier l'orientation (angle θ). La méthode est ensuite fondée sur des calculs de distance et longueurs entre segments (noirs sur chacune de ces droites), ce qui peut être vu comme, le calcul de la force d'attraction d'un segment par rapport à un autre (fig. 2.16).

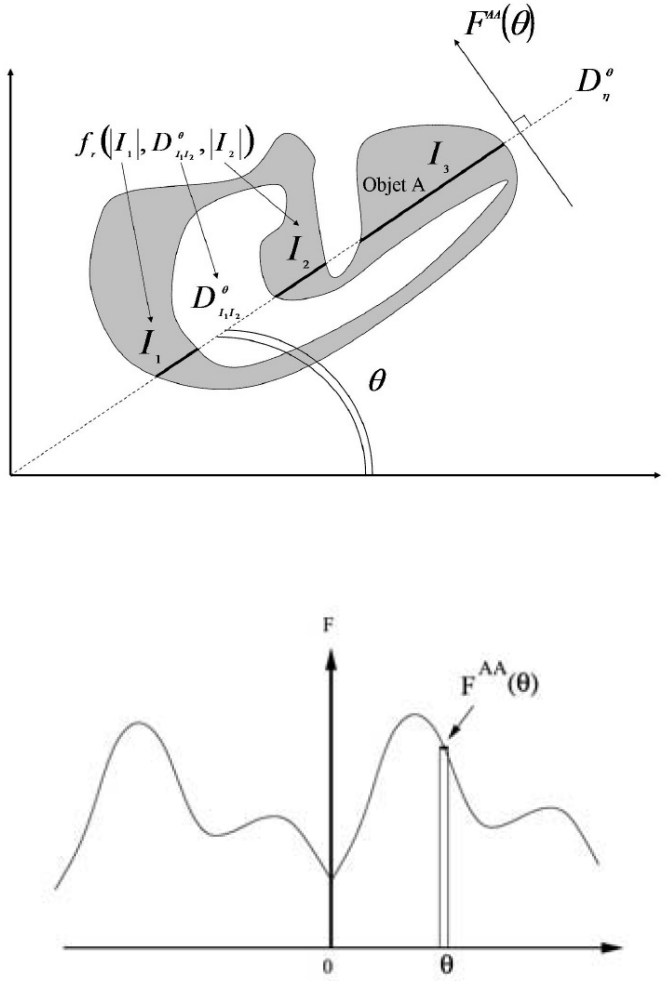


Fig. 2.16 : Le calcul d'une F-Signature d'une forme [Tabbone, 2003].

Soit I_1 et I_2 deux segments portés par une même droite, la force d'attraction liant les deux segments I_1 et I_2 est donnée par la relation suivante :

$$fr\left(|I_1|, D^{\theta}_{I_1 I_2}, |I_2|\right) = \int_{D^{\theta}_{I_1 I_2}+|I_2|}^{|I_1|+D^{\theta}_{I_1 I_2}+|I_2|} \left(\int_0^{|I_2|} \varphi_r(u-v)dv \right) du$$

où $D^{\theta}_{I_1 I_2}$ est la distance entre I_1 et I_2.

Si l'on considère un ensemble de lignes droites parallèles (faisceau), également défini par l'angle theta (θ) (par rapport à une ligne horizontale) couvrant entièrement l'objet ***A***, la F-signature effectue alors l'analyse totale de l'objet ***A***. Soient $\overline{F}^A, \overline{F}^B$ les signatures normalisées des objets A et B, respectivement. Une mesure de similarité a été définie qui dépend du facteur de changement d'échelle et de l'angle de rotation entre les deux signatures. Pour plus d'information, on peut se référer à [Tabbone, 2003].

S. Barrat [Barrat, 2006] utilise une approche d'analyse discriminante conditionnelle couplée à la *R*-Signature pour mettre en place un apprentissage progressif adapté à la reconnaissance de symboles. La *R*-signature [Tabbone, 2002] se base sur la transformée de Radon pour représenter une forme (toujours à l'aide d'histogrammes). La transformée de Radon est la projection d'une image dans un plan particulier. Cette projection possède des propriétés géométriques intéressantes qui font d'elle un bon descripteur de forme. A partir de ces propriétés géométriques, une signature de cette transformée a été créée. Cette signature possède les propriétés d'invariance à certaines transformations géométriques, telles que la translation et le changement d'échelle (après normalisation). Par contre, l'invariance à la rotation est obtenue par permutation cyclique de la signature ou directement à partir de sa transformée de Fourier. La figure. 2.17 montre des exemples de R-signature pour un symbole à différentes orientations.

Il existe bien d'autres descripteurs de formes pouvant être utilisés : comme par exemple, les descripteurs de Fourier calculés à partir des contours des formes, les descripteurs ART « *Angular Radial Transform* » [Kim, 1999], les lecteurs désirant plus d'informations sur ce type de méthodes auxquelles nous nous sommes moins intéressés pourront se référer à [Tabbone, 2005].

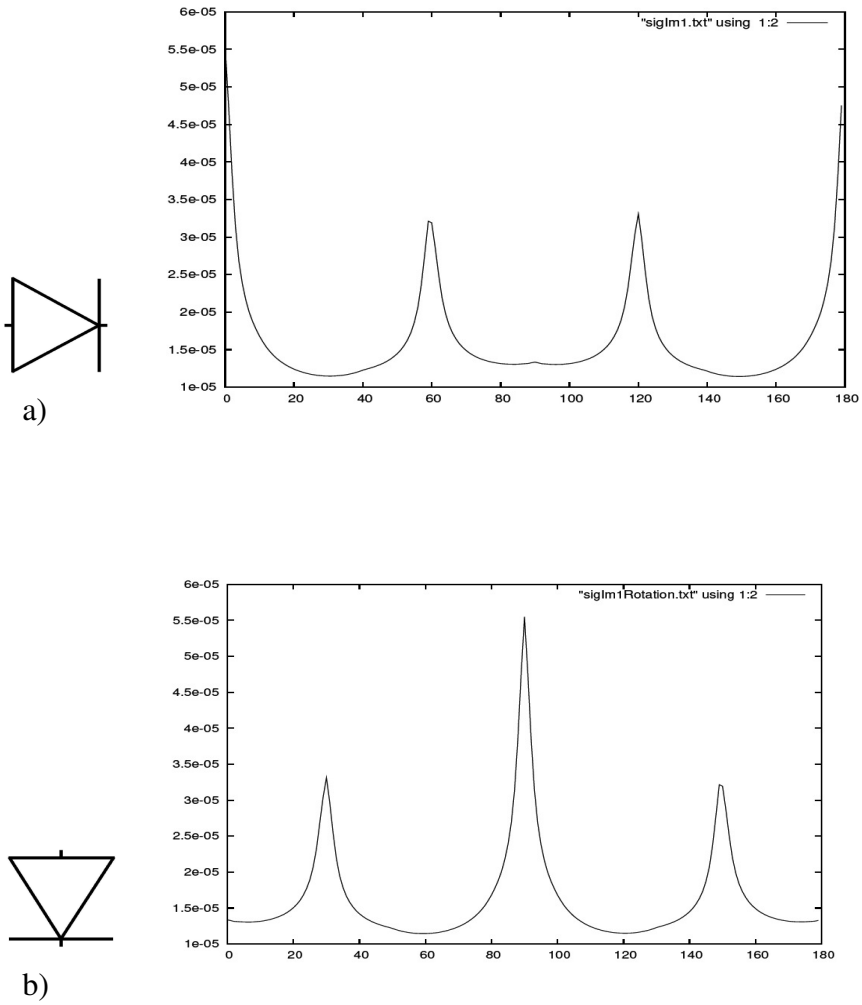


Fig. 2.17 : a) R-signature d'un symbole parfait, b) le même symbole tourné de 90° [Barrat, 2006].

2.9 Campagnes d'évaluation de performances

- ***Présentation***

Ces dix dernières années, beaucoup de méthodes de reconnaissance de symboles ont été développées en utilisant différentes approches, ce qui a fait apparaitre un besoin de protocoles standardisés pour comparer et évaluer la performance des systèmes proposés sur des ensembles de données uniformes. Les taux de reconnaissance constituent, la plupart du temps, les critères principaux d'évaluation. Cependant, le temps de calcul et le nombre de classes de symboles sont liés et ont également été couplés avec d'autres mesures afin d'avoir une analyse plus précise de la pertinence des méthodes de reconnaissance existantes. En général, dans beaucoup d'applications d'analyse d'images, un bon nombre de seuils ad hoc doivent être déterminés. Dans la plupart des cas, ces seuils sont fixés d'une manière très empirique. Ainsi, l'organisation de compétitions permet de mieux caractériser le comportement des algorithmes et d'analyser l'influence du paramétrage [Tombre, 1998]. Notons qu'en reconnaissance de graphiques, les problèmes de sélection de valeurs seuils et autres paramètres qui reviennent fréquemment sont : différence entre lignes épaisses et lignes fines, taille maximum des composantes textuelles, tolérance angulaire pour considérer deux segments comme alignés ou parallèles, différence entre segments ou arcs de cercle et ainsi de suite…

L'objectif du concours de GREC'95 [GREC, 1995] était d'évaluer les systèmes capables de reconnaître les lignes continues et les lignes pointillées dans des images de dessins (line drawings). Le deuxième concours international sur la reconnaissance de graphiques [GREC, 1997] qui s'est tenu à Nancy (France), a concerné la segmentation du texte dans les documents techniques. Un autre objectif était de localiser et reconnaitre les entités graphiques simples telles que les lignes, les arcs, et les cercles dans les schémas [Chhabra, 1998]. Deux autres événements d'évaluation ont concerné la reconnaissance de symboles pré-segmentés. Le premier a eu lieu pendant la 15ème conférence internationale sur la reconnaissance des formes (ICPR 2000) et utilisait une bibliothèque de symboles pré-segmentés composée de 25 symboles électriques, qui ont été dégradés avec un peu de bruit binaire et des changements d'échelle [Aksoy, 2000]. Après cet événement, un autre concours a été organisé durant [GREC, 2003], connu sous le nom du premier concours international sur la reconnaissance de symboles, le nombre de symboles était beaucoup plus conséquent et ces dernier étaient obtenus en utilisant différents niveaux de dégradation et de bruit.

Grâce à ces concours, trois éléments principaux ont été identifiés comme fondamentaux pour l'évaluation des performances des algorithmes de reconnaissance. Il s'agit du type des données de test utilisées, des métriques d'évaluation et des protocoles d'évaluation. L'ensemble de données utilisées s'accroit à chaque concours et inclue toutes sortes de variabilités qui peuvent partiellement être retrouvées dans de vraies données. Néanmoins, la reconnaissance de symboles concerne une grande variété de domaines (architecture, électronique, mécanique, organigrammes, géographique, des cartes, musique, etc.), on peut donc regretter que pendant GREC-2003 et GREC-2005, seule une collection de 50 symboles principalement, architecturaux et électroniques ait été employée (figure 2.1). Tous ces symboles sont composés uniquement des deux primitives graphiques : lignes et arcs. Ils sont rarement composés de plusieurs parties et ne comportent jamais de parties pleines. Concernant la qualité des images, cinq catégories d'images ont été produites incluant : les données idéales, les images après une transformation (rotation, changement d'échelle), les images avec du bruit binaire, des images avec des distorsions vectorielles, et des images bruitées et avec des distorsions vectorielles. Les modèles de dégradation employés sont ceux définis par Kanungo [Kanungo, 1994] permettant de produire neuf modèles différents de bruits sur des images binaires (figure. 2.18). Pour la déformation des formes, un autre modèle basé sur le modèle actif de Cootes [Cootes, 1995] a été employé pour simuler des images dessinées à main levée (figure. 2.19).

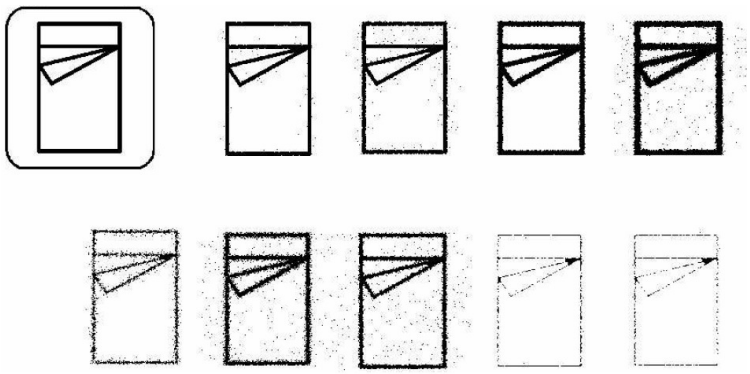
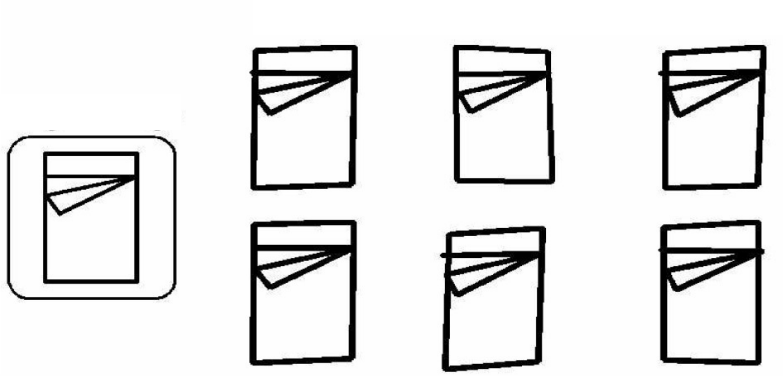


Fig. 2.18 : Les neuf modèles de la dégradation utilisés lors du concours GREC-03 [Dosch, 2006].

On peut noter que les dégradations apportées aux images lors de ces concours étaient parfois exagérées, ne reflétant plus vraiment ce que l'on peut trouver dans des images réelles. Cette constatation reste triviale si les résultats tiennent compte de cette information. On peut par contre regretter qu'aucune image réelle n'est jamais été

utilisées lors de ces concours jusqu'à aujourd'hui, aussi bien pour les images de symboles présegmentés que pour des documents complets. D'ailleurs, le premier concours permettant d'évaluer le comportement des algorithmes sur des documents complets devrait avoir lieu sous peu durant la compagne d'évaluation prévue dans le projet techno-vision EPEIRES. Il s'agira pour l'instant uniquement d'images synthétiques ayant des structures bien particulières et encore bien différentes de ce que l'on peut trouver dans des images réelles. Ces événements témoignent néanmoins d'une volonté forte de se rapprocher toujours plus de la vérité terrain.



(a) (b) (c)

Fig. 2.19 : Les trois modèles de déformation utilisés pour le concours de GREC-03 [Dosch, 2006].

- **Mesures de performances et protocoles retenus**

Concernant les concours de reconnaissance de symboles pré-segmentés, les mesures utilisées pour évaluer les performances sont les classiques *taux de bien classés* et *taux de mal classés*. Le rejet n'était pas prévu dans les compagnes organisées jusqu'à aujourd'hui. On peut également noter qu'un seul modèle était disponible par classe pour l'apprentissage. Aucune influence de la taille de l'ensemble d'apprentissage n'a donc été étudiée jusqu'à présent puisque aucune distinction n'est faite entre base de test, base d'apprentissage et base de validation.

Même si, comme on le voit, des améliorations restent à apporter, ces compagnes d'évaluation ont eu d'énormes conséquences sur les recherches dans le domaine et on ne peut que féliciter les chercheurs ayant œuvrés pour leur organisation.

D'autres métriques, plus liées aux aspects localisation ont été proposées récemment dans le cadre du projet EPEIRES. Elles sont semblables à celles utilisées pendant les conférences ICDAR-03 dans le cadre de la reconnaissance de textes imprimés. La métrique proposée est basée sur les notions de *précision* et de *rappel*. Pour une donnée test, soient *T* le nombre de cibles appartenant à la vérité-terrain, et *R* l'ensemble de résultats fournis par une application. Le nombre de résultats exacts parmi les *T* fournis est noté *E*. La *précision* p est alors définie comme le nombre de résultats exacts divisé par le nombre de résultats fournis.

$$p = \frac{e}{|R|}$$

Le *rappel r* est défini comme le nombre de résultats exacts divisé par le nombre de cibles :

$$r = \frac{e}{|T|}$$

La *précision* et le *rappel* peuvent alors être combinés, pour déterminer un score global *s*, exprimant le taux d'identification [VAL 07] :

$$s = \frac{2}{(1/p) + (1/r)}$$

Notons, enfin qu'une représentation vectorielle simple des images (le format VEC) a été proposée par Chhabra et Phillips [Chhabra, 1998] pour être employée par les participants de ces différents concours. Ce format vectoriel est donc connu et utilisé principalement par la communauté de recherche en reconnaissance de symboles. Il simplifie fortement la mise en place de nouvelles compagnes d'évaluation.

- **Principaux résultats des campagnes d'évaluation**

Cinq méthodes de reconnaissance ont participé au concours GREC-03. Toutes n'ont pas pu participer à tous les tests à cause des particularités de chaque méthode. Les cinq participants étaient : Université de Rouen-La Rochelle, la National University d'Irlande-Maynooth, City University de Hong Kong, Université de Nottingham et Fudan University.

Le système de reconnaissance proposé par les Universités de Rouen et La Rochelle était une approche mixte, structurelle et statistique. Un graphe d'inclusion dans lequel les nœuds correspondaient aux occlusions et les arcs correspondaient aux relations d'inclusion était utilisé.

La méthode de la National University d'Irlande-Maynooth était aussi basée sur une combinaison de descripteurs statistiques et structurels. Les descripteurs étaient utilisés pour générer un numéro unique qui identifiait chaque symbole. Pour cela, les valeurs de descripteurs numériques (comme le nombre de segments, des informations sur les angles, sur des ratios de tailles de segment, etc.) étaient fusionnés. La reconnaissance était réalisée en comparant la valeur obtenue pour le symbole requête avec l'ensemble des valeurs calculées pour tous les symboles présents dans la bibliothèque. Dans cette approche structurelle, trois types de formes (le segment, le cercle et l'arc) étaient cherchés dans l'image et ensuite une étude de leurs caractéristiques était utilisée. La méthode a obtenu de bons résultats pour les transformations affines, mais pour les images dégradées, les performances diminuaient rapidement.

Le système proposé par la City University de Hong Kong est similaire à celui de Yan et Wenyin [Luo, 2003] [Wenyin, 2007] décrit en section 2.3.

L'algorithme proposé par l'Université de Nottingham utilisait la description d'un symbole en termes d'ensemble de segments et arcs. Pour reconnaître un symbole, tous les segments et les arcs de longueur supérieure à un seuil étaient comparés avec les segments et les arcs des symboles modèles, en appliquant différentes transformations géométriques. L'inconvénient de cette approche était le temps de calcul qui augmente proportionnellement au nombre de primitives présentes dans les symboles.

L'université Fudan a proposé d'utiliser un nouveau descripteur basé « pixels » construit après : prétraitement et par extraction des caractéristiques simples. La classificateur utilisé était les K-plus proches voisins. Il semble que les bonnes performances obtenues par cette méthode soient principalement dues aux filtres de prétraitement utilisés (basés sur des connaissances a priori sur le type de bruit présent dans les images).

La figure. 2.20(a) montre les résultats obtenus sur les images idéales de symboles pour des ensembles comportant 5, 20 et 50 classes. Ces résultats témoignent du bon pouvoir discriminant des méthodes. Dans la figure. 2.20 (b, c, d), nous pouvons trouver

les résultats pour des images avec rotations et changements d'échelle. L'algorithme proposé par l'Université de Fudan est plus robuste aux dégradations que la méthode de le City University de Hong Kong qui reste cependant plus rapide. Comme prévu, les temps de calcul (figure 2.21) augmentent quand le nombre de symboles modèles (classes) augmente [Valveny, 2007].

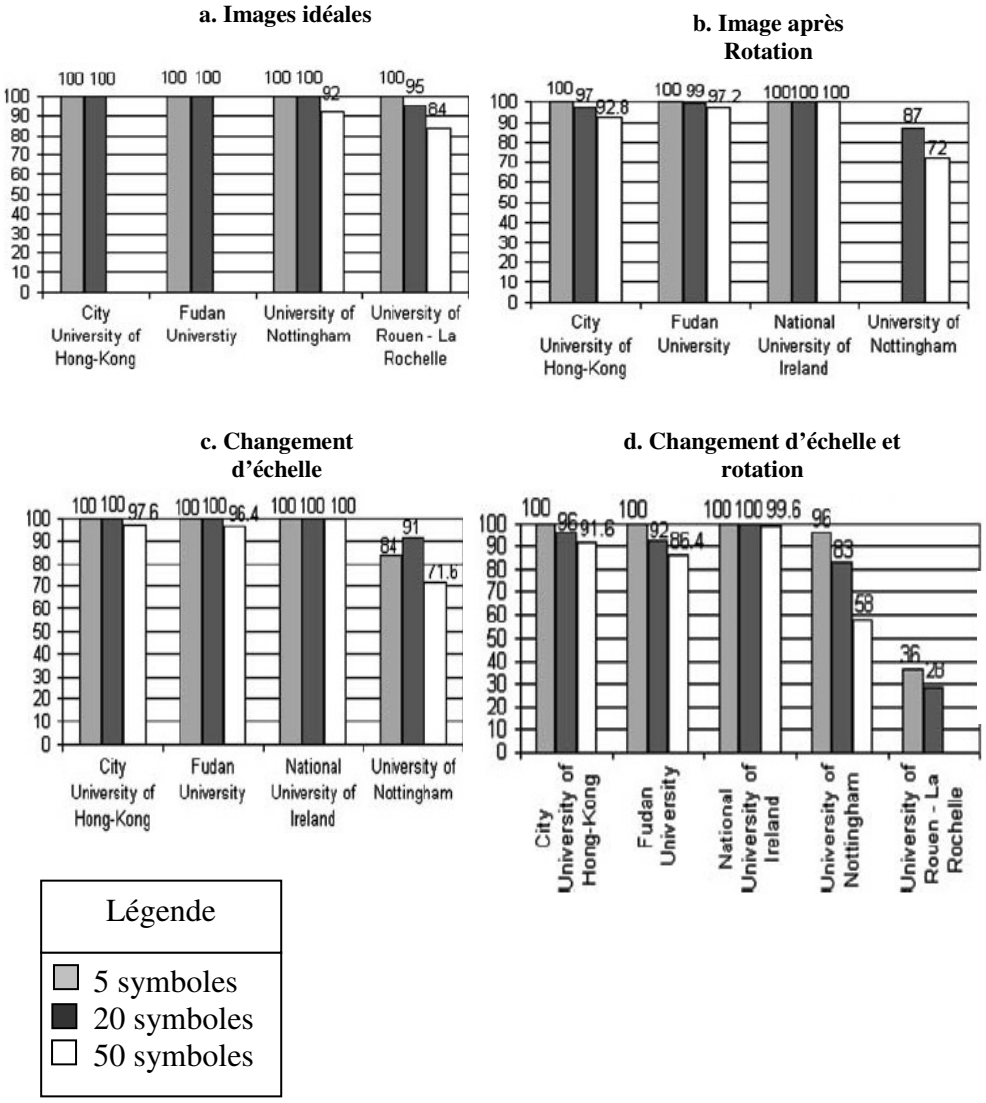
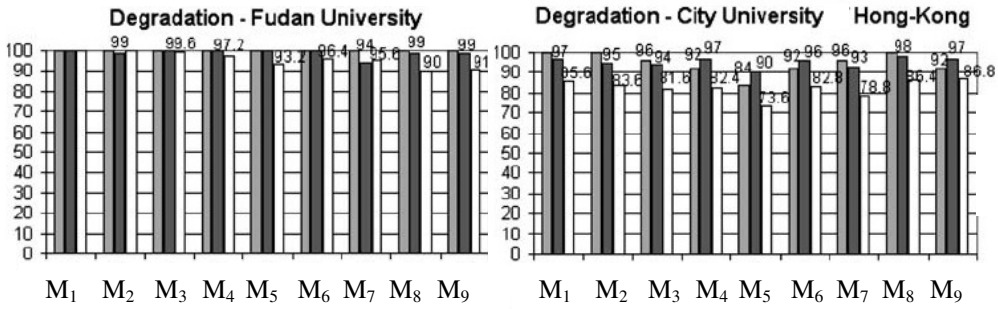
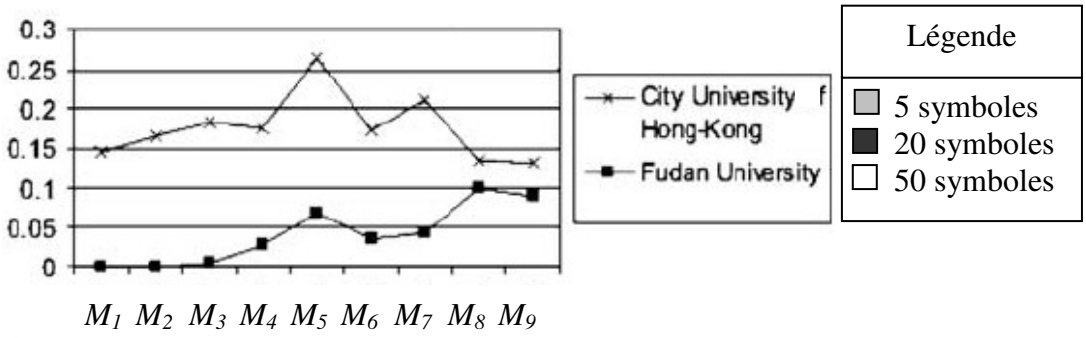


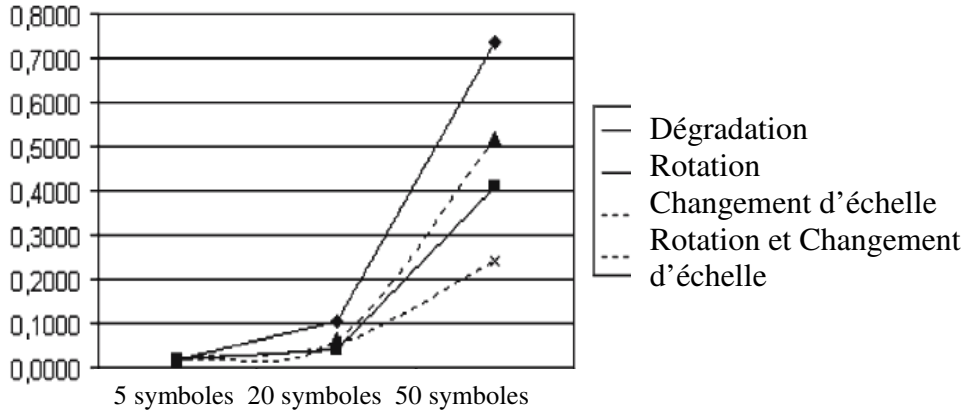
Fig. 2.20 : Résultats obtenus par les différentes méthodes durant GREC-03 sur les symboles idéaux.



a) Résultats pour des symboles dégradés selon le modèle de la [Fig. 2.18].



(b) Robustesse à la dégradation



(c) Temps de calcul

Fig. 2.21: Résultats sur les images dégradées et comparaison des temps de calcul (repris de [Valveny, 2007]).

La plupart des méthodes proposées ont obtenu de très bons taux de reconnaissance compris entre 95% et 100% ce qui rend difficile leur comparaisons. Généralement, les performances diminuent significativement lorsque le nombre de classes augmente. Toutes les méthodes ne sont pas entièrement invariantes à la rotation et au changement d'échelle.

Les performances se dégradent de manière significative lorsque le bruit devient important ou quand la connectivité des lignes est perdue. Les performances diminuent moins rapidement avec les déformations vectorielles qu'avec ces dégradations.

Durant l'édition 2005 du concours, plus de symboles et plus de modèles de bruit avaient été ajoutés (figure 3.22). De même, le nombre de classes est passé de 50 à 150 symboles différents, permettant de tester le passage à l'échelle « *scalability* » des méthodes de reconnaissance. De plus, quatre nouveaux modèles de dégradation ont été ajoutés. Les tests ont seulement inclus des images en format bitmap pour cette édition.

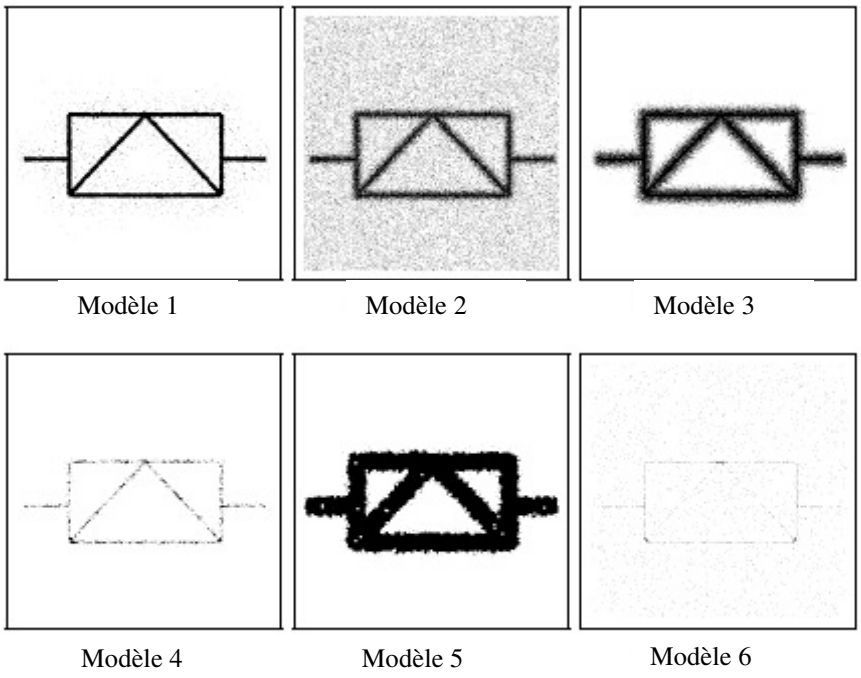


Fig. 2.22: Les six nouveaux modèles de dégradation utilisés pour GREC-05 [Dosch, 2006].

Les quatre participants ont obtenu à nouveau de très bons résultats même si le taux de reconnaissance diminue toujours quand le nombre de symboles augmente. Le taux moyen de reconnaissance reste égal à 98% en traitant 150 symboles non bruités différents. Le taux de reconnaissance diminue presque de 5% en traitant 150 symboles ayant subi des rotations. Une réduction de 9% et de 16% a également été remarquée pour les tests de changement d'échelle et la combinaison de deux transformations.

Une seule méthode conserve un taux d'identification supérieure à 50% lors de fortes dégradations des images. C'est celle de Min Feng avec un taux d'identification égal à 59.81%. Min Feng est également le gagnant global du concours avec un taux moyen de reconnaissance de 83.33% pour tous les tests proposés (voit tableaux. 2.2). Malheureusement, nous n'avons pu trouver aucun descriptif des différentes méthodes testées lors de ce concours.

Tab. 2.2 : Résumé des résultats de chaque participant lors de GREC-05.

Participant	**Taux de reconnaissance moyens**
Andyardja Weliamto	70.28%
Min Feng	83.33%
Jing Zhang	67.65%
Wan Zhang	82.82%

2.10 Pour conclure

Voici le bilan que nous dressons de cet état de l'art non exhaustif sur les travaux menés en analyse de documents graphiques jusqu'à aujourd'hui.

Représentation des images de documents

Nous avons détaillé, dans ce chapitre, les méthodes classiques de représentation d'images de documents graphiques non dédiées à des applications spécifiques. Lorsqu'il s'agit de documents graphiques bien que le vecteur soit approprié pour la représentation des lignes, pour des structures telles que les courbes et les régions pleines, une vectorisation basique n'est pas le traitement idéal et des traitements supplémentaires sont nécessaires. Selon nous, le vecteur constitue une primitive efficace et facile à utiliser pour identifier d'autres primitives graphiques plus complexes mais nous pensons indispensable d'ajouter une dimension contextuelle à cette primitive afin de constituer une bonne base pour l'extraction de caractéristiques de plus haut niveau indispensable pour l'interprétation complète de différents types de documents.

Localisation/Segmentation

Nous avons noté que peu de travaux ont abordé la problématique de la localisation/segmentation de symboles dans les documents (en dehors d'applications dédiées à un type d'images bien précis (partitions musicales, schémas électroniques, plans d'architectures). Tout au moins aucune méthode ne semble se dégager du lot. Il est à noter que, parmi le faible nombre de travaux sur cette problématique, les références à des méthodes de segmentation exploitant des représentations structurelles (graphes) sont quasi inexistantes ce qui peut paraître surprenant quand on connaît le pouvoir de représentation, la généricité de ce type d'outils et des méthodes de traitement qui leur sont associées. Peut être est-ce dû à la complexité algorithmique de ce type de méthodes ? Nous pensons pourtant qu'il s'agit d'une voie à explorer si l'on veut gagner en généricité.

- **Reconnaissance**

Concernant les techniques de reconnaissance, nous avons décrit rapidement les méthodes statistiques pour nous étendre plus largement sur les principales méthodes structurelles basées sur les graphes. Concernant les méthodes statistiques, nous avons noté que de plus en plus de méthodes basées sur un calcul de signatures assez évaluées sont proposées pour identifier des formes extraites d'images de documents [Ventura, 1994], [Dosch, 2004], [Wang, 2006]. Ces signatures sont constituées d'une collection de caractéristiques numériques extraites sur des formes pré-segmentées (le plus souvent les composantes connexes pour les symboles graphiques. Parmi les signatures les plus souvent utilisées, on peut citer :

1 Les profils horizontaux et verticaux des formes, les histogrammes des projections, les histogrammes des forces [Wendling, 2002] …
2 Les moments géométriques (Zernike, Hu ...) [Khotazad, 1990],
3 Les transformées (Fourier [Zhang, 2005], Fourier-Mellin [Adam, 2000a], Radon [Tabbone, 2002]…),
4 Les signatures vectorielles [Dosch, 2004].

Malheureusement, si ces méthodes marchent bien sur des formes pré-segmentées, leur utilisation est quasiment impossible lorsqu'il devient nécessaire de travailler sur des images complètes contenant un ensemble de symboles interconnectés les uns avec les autres.

Les méthodes structurelles utilisent principalement des représentations à base de graphe. On distingue différents types de modélisation des relations entre primitives graphiques [Delalandre, 2005]. La première solution consiste à exploiter les relations d'inclusion et/ou de voisinage entre composantes connexes, occlusion, ou régions fermées (boucles). Ce type de modèle (à base de graphe d'adjacence de régions) est utilisé dans différents systèmes utilisant une approche région. [Seong, 1993], [Lladós, 2001a], [Weindorf, 2002]. Le second type de méthodes décrit les relations géométriques et topologiques entre primitives vectorielles (segments et/ou arcs) avec des graphes et utilise ensuite des méthodes de comparaisons de graphes plus ou moins évaluées pour reconnaitre les symboles graphiques.

Leurs limites concernent principalement la complexité (temps de calcul) et leur manque de tolérance aux variabilités. Pour l'instant, aucune méthode complètement structurelle n'a été évalué dans le cadre d'un concours, probablement car ces méthodes sont moins adaptées (moins performantes que les méthodes statistiques) à la reconnaissance de formes pré-segmentées globalement similaires et sur lesquelles des connaissances à priori peuvent être exploitées facilement. Nous sommes pourtant convaincus qu'elles montreront leur intérêt lorsque des images réelles, diversifiées et complètes devront être traitées !

Chapitre 3

Représentation structurelle d'images de documents graphiques et localisation de symboles

3.1 Introduction

Nous avons vu, dans les chapitres précédents, les lacunes des systèmes d'analyse d'images de documents graphiques tant du point de vue de la "reconnaissance de symboles" que du point de vue de la "segmentation" lorsqu'il s'agit de traiter des documents diversifiés et complets. Nous pensons que la mise en place d'une représentation unique et performante du contenu des images permettra d'obtenir de nouveaux résultats sur ce type d'images. Compte tenu du type de primitives pouvant se trouver dans les images, la construction de cette représentation implique selon nous inévitablement une phase de vectorisation. Cependant, cette dernière nécessite d'être améliorée afin de mieux tenir compte des divers contenus devant être représentés.

Afin d'avoir une représentation complète et compacte de toutes les formes susceptibles d'apparaître dans les documents graphiques, nous présentons une nouvelle représentation qui se base sur un graphe multi-primitive et multi-attribut. Notre proposition vise à pallier les lacunes des anciens travaux, ainsi un procédé de vectorisation original produit, tout d'abord, une liste de vecteurs représentant les

contours des formes. Ensuite, une procédure d'appariement des vecteurs de contours permet la localisation des formes fines tandis que les vecteurs restants représentent les frontières des formes pleines. Finalement, les formes linéaires qui se composent de lignes et d'arcs sont représentées avec une primitive graphique appelé quadrilatère et obtenue par appariement de deux vecteurs de contours. Les quadrilatères comme les vecteurs constitueront les nœuds d'un graphe tandis que les arcs décriront les relations spatiales existant entre ces deux types de primitives de description.

3.2 Quelle méthode de vectorisation : Squelette ou Contour ?

Le but principal de la phase de construction d'une représentation est de réduire la quantité de données nécessaire pour pouvoir analyser le contenu de l'image mais a également comme objectif de supprimer les parties non significatives pour mieux mettre en évidence les zones d'intérêt dans le document. Si l'on considère que l'étape de vectorisation répond à ces objectifs, deux approches principales peuvent être utilisées : la première se base sur le calcul d'un squelette des formes en utilisant, soit une technique d'amincissement, soit une carte de distances aux bords. Le second type de méthodes utilise les contours opposés des formes linéaires à les vectoriser. Les deux approches ont des avantages et des inconvénients et une comparaison de ces approches a d'ailleurs été présentée par Tombre dans [Tombre, 2000a]. Voici quelques informations complémentaires sur ces deux techniques qui permettent de mieux comprendre nos choix futurs.

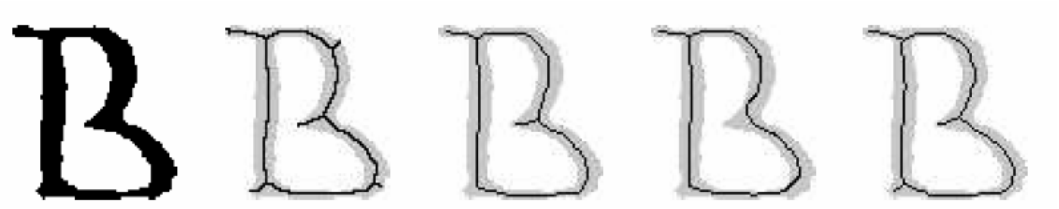
3.2.1 Les approches "Squelette"

Les techniques à base de squelettes cherchent à représenter les formes par la représentation la plus mince possible restant significative de la forme originale et préservant sa topologie. Les méthodes d'amincissement topologique [Lam, 1992] et les méthodes par cartes de distances sont souvent utilisées.

- **Calcul du squelette**

Ces méthodes réduisent significativement l'espace mémoire nécessaire, puisque les formes sont simplifiées par rapport aux formes originales tout en gardant les informations structurelles. C'est pourquoi elles sont souvent privilégiées en

reconnaissance de caractères [Arrivault, 2006]. Mais, toutes les techniques d'amincissement connues, comme par exemple [Gonzalez, 1992] [Deutsch, 1972] [Zhang, 1984] [Zhang, 1994], génèrent généralement des branches approximatives aux croisements et à la jonction des lignes (figure 3.1). En regardant la figure 3.2, il est évident qu'une irrégularité de frontière peut induire une modification complète du squelette.



[Gonzalez, 1992] [Deutsch, 1972] [Zhang, 1984] [Zhang, 1994]

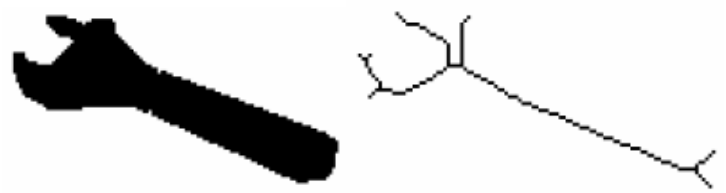


Fig. 3.1 : Branches approximatives dans les squelettes aux croisements et à la jonction des lignes.

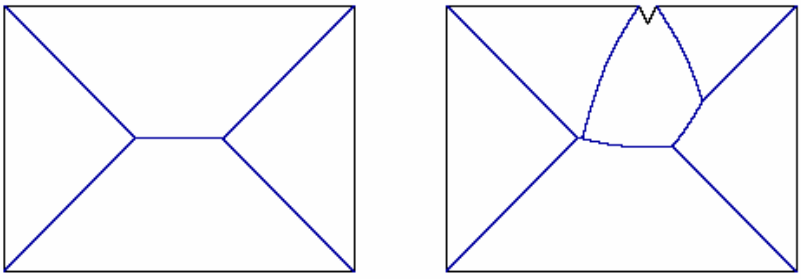


Fig. 3.2 : Sensibilité au bruit : de petits changements sur la frontière peuvent induire des changements cruciaux dans le squelette.

Comme de nombreuses fois auparavant, une amélioration de la représentation de formes par squelettes est présentée dans [Ruberto, 2002] ; les auteurs appliquent des

transformations morphologiques paramétriques pour éliminer les branches courtes du squelette appelées barbules (figure 3.3). Ce type de traitements nécessite malheureusement de nombreux seuils choisis de manière empirique et n'est donc pas toujours efficace [Cordella, 2000c]. Le temps de calcul est aussi souvent très élevé.

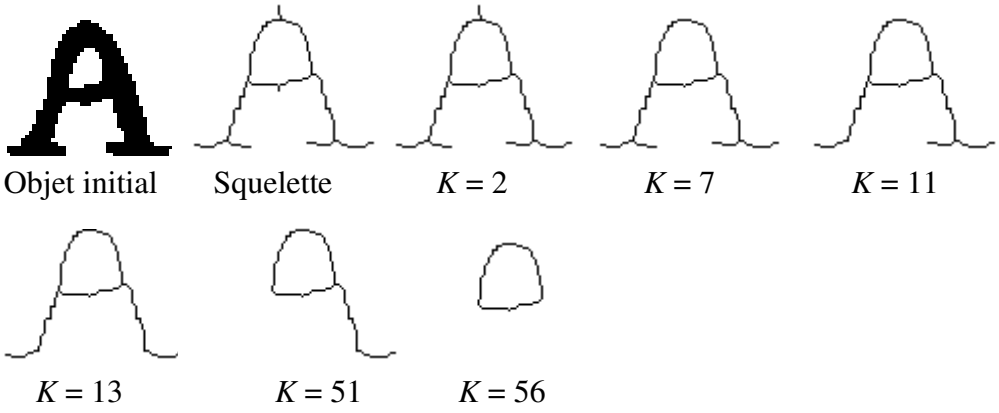


Fig. 3.3 : Nettoyage de squelette pour différentes valeurs de K (nombre minimum de pixels dans les branches du squelette).

Rosenfeld et Pfaltz [Rosenfeld, 1968] définissent la carte de distance « Distance transform » d'une image binaire en remplaçant chaque pixel objet par une valeur numérique représentant la distance minimale jusqu'à un pixel du fond. La distance entre les deux pixels est définie comme le nombre de pixels présent dans la plus courte chaine de 4-connexité entre eux. Le calcul des distances nécessite un double parcourt des pixels de l'image (de gauche à droite et de droite à gauche). Ensuite, l'axe médian est trouvé par une opération de recherche des maxima locaux. Cette approche est plus rapide que l'approche par amincissements mais le squelette obtenu ne garantit pas toujours la conservation de la connexité (particulièrement aux jonctions). De plus, il est difficile de calculer la carte de distance « Distance transform » sans stocker l'image complète en mémoire.

- **Du squelette aux vecteurs**

Une fois les squelettes extraits, les méthodes comme le « line fitting » [Hori, 1993] et les méthodes basées sur la transformée de Hough [Risse, 1989] peuvent être utilisées pour convertir le squelette en un ensemble de petits segments (vecteurs). Mais, ces approches posent d'autres types de problèmes : la technique de « line fitting » est basée

sur une analyse ordonnée des segments adjacents. Ainsi, des lignes ne seront pas récupérées correctement si quelques parties sont absentes ou possèdent des déformations sérieuses (figure 3.4).

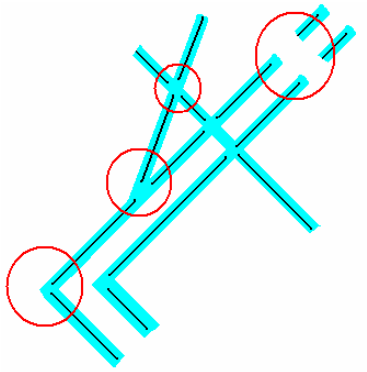


Fig. 3.4 : Vectorisation par une analyse ordonnée des segments adjacents.

La transformée de Hough n'est pas sensible aux environnements bruités et peut détecter les lignes droites, les cercles et les ellipses dans les images. Elle a été utilisée comme première technique pour la segmentation des arcs par [Hallard, 1981] [Hunt, 1988] [Dori, 1997]. Ses inconvénients sont des calculs lourds et la grande capacité mémoire nécessaire. Malgré de nombreux travaux sur la réduction de complexité comme les efforts de [Kimme, 1975] [Gorman, 1984] [Haralick, 1992] [Illingworth, 1987] [Conker, 1988], la méthode de la transformée de Hough n'est pas très efficace et est habituellement appliquée qu'aux images de petite taille.

L'approximation polygonale peut également être appliquée pour transformer le squelette en un ensemble de segments. Une approche de cette catégorie est celle de Rosin & West [Rosin, 1989] qui ne nécessite aucun paramètre ou seuil choisi par l'utilisateur. L'algorithme fait un coupage récursif des courbes dans les plus petits segments possible aux points où la déviation est maximale. La méthode de Wall [Wall, 1984] est basée sur un calcul de surface algébrique itérativement entre la courbe et le segment. La méthode est très rapide et ne nécessite qu'un seul seuil (le ratio entre surface algébrique et longueur de segment). Cependant, elle est moins précise que l'algorithme de Rosin au regard du nombre de segments (figure 3.5). Selon Karl Tombre, la reconnaissance de symboles est plus difficile lorsque le symbole est décomposé en trop de segments. Néanmoins, la méthode de Wall ne produit pas des petits segments non désirés aux jonctions [Tombre, 2000b] comme le montre la figure 3.5.

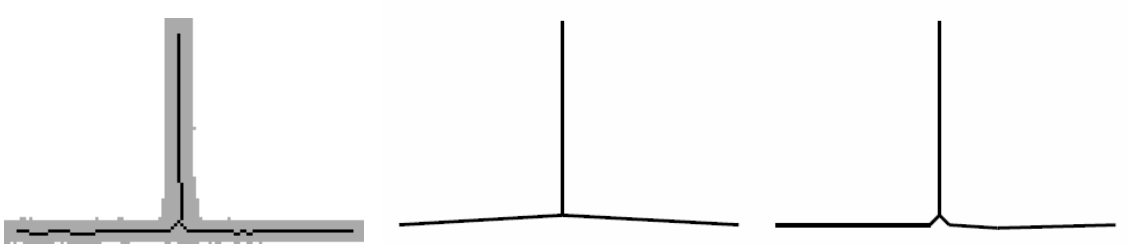


Fig. 3.5 : a) Image initiale et squelette, b) Approximation par l'algorithme de Wall, c) Approximation par l'algorithme de Rosin [Tombre, 2000b].

3.2.2 Les approches "Contour"

Pour localiser (et amincir) les formes fines présentes dans un document, certains travaux proposent de se déplacer de point en point en restant, soit à l'extérieur de l'objet (suivi de contours), soit à l'intérieur de l'objet (suivi de traits).

Les approches contours sont moins sensibles au bruit et produisent de meilleuers réprésentations des jonctions que les approches squelette. Parmi les diverses méthodes pour la détection des contours, il y a deux approches très répandues qui fonctionnent de manière plus ou moins similaire : il s'agit de la détection morphologique des contours [Rosenfeld, 1982] [Labelle 1998] [Zhang, 1998] [Hassan, 2000] et le suivi des contours [Braun, 1995] [Turner, 1996] [Jordan 1997] [Han, 1994] [Jimenez, 1982]. Les méthodes de vectorisation par appariement de contours comme [Han, 1994] [Jimenez, 1982] et [Ramel, 1996] sont souvent plus robustes au bruit et plus rapide du fait que les contours opposés constituent une information fiable sur la disposition et la forme des entités contenues dans les images. Néanmoins, il faut être capable de suivre les deux contours opposés simultanément pour ensuite, utiliser les points milieu des 2 frontières sont utilisés afin d'obtenir les axes médians.

Une ligne peut aussi être vue comme un ensemble de pixels noirs entre deux régions blanches. Partant d'un point initial, il s'agit alors de progresser à l'intérieur de la ligne jusqu'à l'extrémité opposée. Pour effectuer cette progression, Paquet [Paquet, 1990, Paquet, 1991] définit comme direction locale de la ligne (figure 3.6), la direction de Freeman permettant le plus grand déplacement à l'intérieur de la ligne.

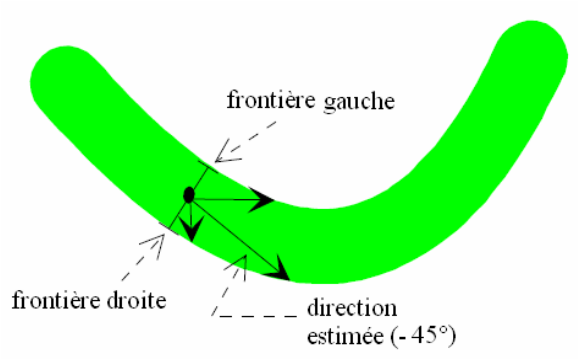


Fig. 3.6 : Suivi de trait, extrait de [Ramel, 2006].

L'algorithme de suivi estime, à chaque étape, les directions successives de la ligne et fournit la liste des points particuliers : jonctions (épaisseur anormale) ou extrémités. Paquet propose d'utiliser un pas de progression proportionnel à l'épaisseur moyenne de la ligne suivie. Cette méthode est utilisée par Ogier [Ogier, 1994] pour l'analyse de plans cadastraux.

L'algorithme « *Orthoganl Zig-Zag (OZZ)* » de Dov Dori [Dori, 1997] est un algorithme exploitant une idée similaire et très rapide. L'auteur a montré que la complexité de temps et d'espace mémoire nécessaire pour son algorithme est 20 fois plus petite que celle de la transformée de Hough. L'algorithme OZZ exploite la même idée que les méthodes précédentes (parcours de faisceaux lumineux d'une fibre optique). L'algorithme se base sur le déplacement dans un trait selon une direction qui change de 90° chaque fois qu'un pixel fond est rencontré (pixel juste extérieur au contour). L'image est scannée de gauche à droite et le premier pixel noir rencontré est noté comme point de départ. Une projection à partir de ce point, parallèle à l'un des axes est effectuée, jusqu'à la frontière opposée au contour. Le suivi se poursuit itérativement selon une direction perpendiculaire à la précédente (figure 3.7a). Ce processus de déplacements orthogonaux continue jusqu'à ce qu'une extrémité de segment soit rencontrée. L'algorithme est alors relancé une nouvelle fois depuis le point de départ mais, cette fois-ci, dans le sens opposé. De plus, pendant ces déplacements orthogonaux, différentes statistiques comme les extrémités, les points milieu et les longueurs (codage des plages pour ces pixels noirs) des traits sont stockés. Ces informations sont utilisées pour effectuer des vérifications : par exemple, si la longueur est plus grande que les longueurs précédentes, l'algorithme cherche l'existence d'un point de jonction ou le début d'un nouveau segment (figure 3.7b). Cette approche

fonctionne pour les segments avec un certain angle qui ne sont pas horizontaux ou verticaux (non parallèles à un des deux axes). Pour les segments parallèles aux axes, l'algorithme OZZ donne des traits très longs dans une direction et très petits dans l'autre (direction perpendiculaire) et on ne peut donc pas utiliser directement cette méthode. Pour ces types de segments, Dori propose de tracer l'axe médian (figure 3.7c) en utilisant le même critère de la longueur de projection avec vérification de la largeur du trait pour rester dans le même segment.

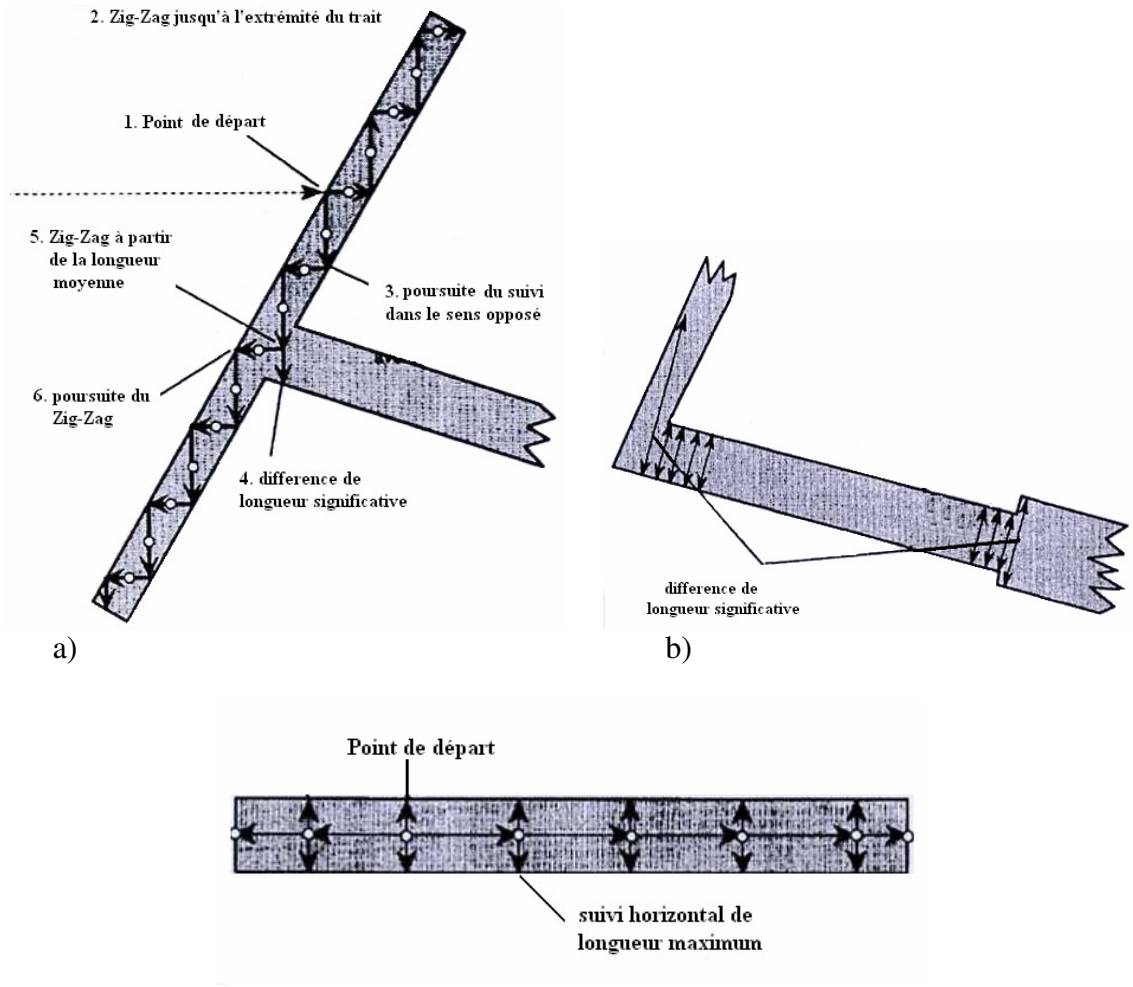


Fig. 3.7 : Déplacements orthogonaux par Zig-Zag de Dori [Dori, 1997].

Han et Fan [Han, 1994] proposent quatre étapes pour l'extraction d'ensembles de lignes idéalisées appelées également axe médian ou squelette. Ils incluent l'extraction des contours, la vectorisation des contours, l'extraction de l'axe médian par mise en correspondance des contours opposés et l'analyse des connexions entre axes médians en

cherchant les points d'intersection (figure 3.8). Pour la mise en correspondance des contours trois conditions principales sont testées. Premièrement, les deux vecteurs représentant les contours, opposés doivent avoir la même pente (différence d'angles proche de 0° ou 180°), deuxièmement, les deux vecteurs doivent avoir un recouvrement non nul et le ratio entre la partie commune (recouvrement) et le plus petit vecteur doit dépasser un certain seuil(α).

$$\frac{L_1 \cap L_2}{\min(L_1, L_2)} > \alpha$$

Finalement, un critère de distance maximum entre extrémités utilisant un seuil prédéfini évite la mise en relation de vecteurs d'axes trop éloignés. Pour l'appariement des parties courbes des contours, les cirières à vérifier sont : les centres des deux arcs doivent être proches, les rayons doivent être similaires, et les deux arcs doivent se recouvrir.

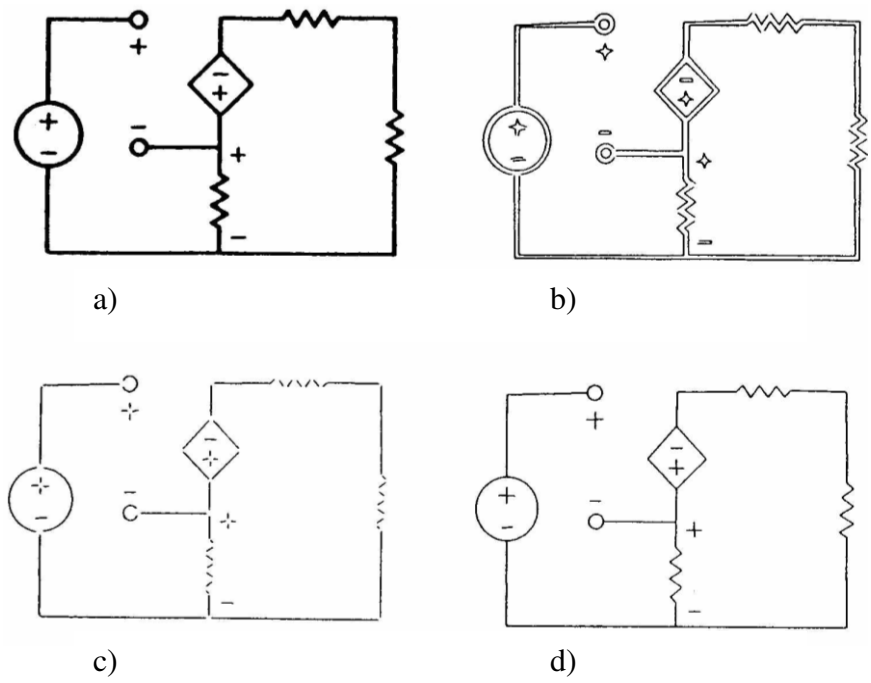


Fig. 3.8 : a) Image initiale b) contours c) axes médians d) résultat après connexion des axes médians voisins [Han, 1994].

Une seconde méthode de cette catégorie est celle de Jimenez et Navalon [Jimenez, 1982] qui utilise plusieurs projections orthogonales d'un contour sur l'autre pour générer les points de l'axe médian et déterminer le milieu du segment reliant les 2

frontières (figure 3.9a, 3.9b). Cette approche a des performances limitées en ce qui concerne la représentation des jonctions où des intersections difficiles à détecter (figure 3.9c, 3.9d).

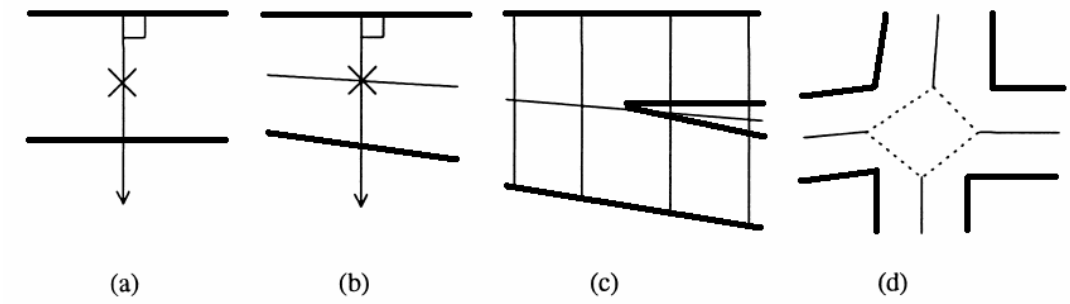


Fig. 3.9: Utilisation des contours pour extraire les axes médians [Wenyin, 1999].

3.2.3 Bilan

La qualité de la vectorisation dépend de la méthode choisie, du nombre de paramètres et de seuils utilisés. Pour obtenir la robustesse, il est important de réduire le nombre de paramètres et de seuils nécessaires pour la vectorisation. Il peut être mieux de combiner différentes méthodes simples par rapport à l'implémentation une méthode avec beaucoup de paramètres. Les méthodes de vectorisation basées sur les contours nous semblent plus résistantes aux bruits et distorsions. Elles semblent également fournir une meilleure représentation des extrémités et jonctions. De plus, les approches squelette ou axe médian ne fournissent pas d'information sur l'épaisseur locale qui peut pourtant jouer un rôle essentiel pour distinguer les formes pleines et les formes linéaires. La technique *Vect + Quad* proposée dans [Ramel, 2000] propose de représenter les images de documents graphiques à l'aide d'un ensemble de quadrilatères et de vecteurs. Cette méthode utilise une approximation polygonale des contours des formes. Cette méthode permet d'obtenir une représentation des formes fines et pleines contenues dans l'image. Nous avons choisi de repartir de ces travaux pour apporter des améliorations visant à obtenir une représentation structurelle fédérant l'ensemble des informations disponibles sur le contenu des images.

3.3 Description rapide de la technique *Vect + Quad*

■ *Les vecteurs (Vect)*

La méthode *Vect + Quad* utilise l'algorithme d'approximation polygonale proposée par Wall [Wall, 1984] pour obtenir une liste de vecteurs représentatifs des contours des formes binaires contenues dans l'image à analyser. Cet algorithme effectue l'approximation polygonale durant le suivi des contours. Il génère en sortie une liste de vecteurs décrivant l'ensemble des contours des formes binaires constituant le document.

■ *Les quadrilatères (Quad)*

La représentation de l'image sous forme de chaînes de vecteurs correspondant aux contours des formes permet de limiter la perte d'information et donne une approximation très fidèle du dessin, mais elle n'est pas facilement exploitable et ne fournit pas suffisamment d'information sur la structure du document. Une étape d'appariement des vecteurs permet d'obtenir des quadrilatères. C'est l'association de deux vecteurs constituant les frontières opposées d'un trait qui produit un quadrilatère.

La procédure d'obtention des quadrilatères consiste à apparier certains vecteurs. L'algorithme utilisé effectue une recherche du vecteur V_1 de longueur maximale dans la liste des vecteurs puis essaie de trouver le vecteur V_2 correspondant à la frontière opposée. Pour cela, on cherche dans la liste des vecteurs, le vecteur le plus proche suivant la distance euclidienne entre extrémités des vecteurs.

Cependant, cela n'est pas suffisant puisque deux vecteurs très proches ne sont pas nécessairement associables (vecteurs orthogonaux par exemple). Il est donc nécessaire de vérifier la validité de certains critères. Dans la méthode *Vect + Quad* les critères suivants sont vérifiés :

- Un trait est une forme fine : la distance entre les extrémités des deux vecteurs doit être minimale.

- Les frontières du trait sont parallèles : il faut donc que les vecteurs aient des pentes proches.

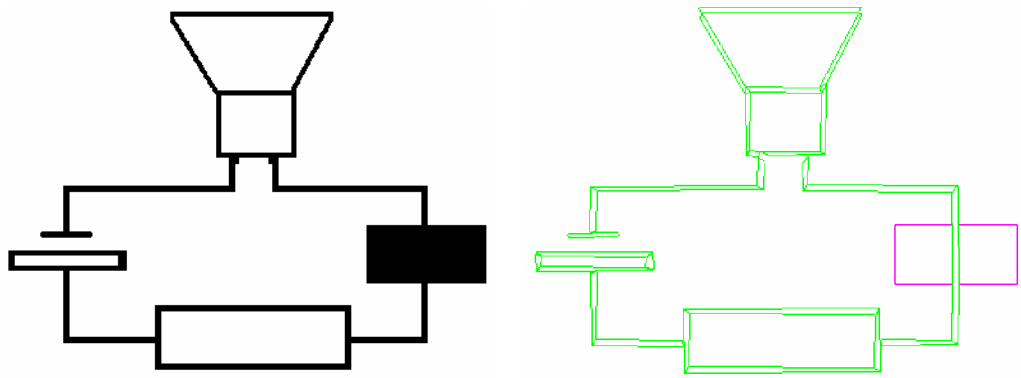
- Un trait est une forme pleine: Il doit donc impérativement y avoir des pixels noirs entre les deux vecteurs.

- Un trait est une forme allongée: La longueur des vecteurs doit donc être supérieure à la distance les séparant.

Si l'un de ces critères n'est pas vérifié, alors on cherche un autre vecteur proche, et ainsi de suite. Si aucun vecteur V_2 n'a pu être trouvé, le vecteur V_1 est laissé dans la liste des vecteurs.

Pour améliorer l'appariement, il peut parfois être nécessaire de décomposer certains vecteurs en deux. Le point de coupure est déterminé par projection d'une extrémité d'un vecteur sur le support de l'autre. Ceci permet d'avoir des vecteurs de longueur plus semblable. Le vecteur restant est alors placé dans la liste des vecteurs à étudier.

A la fin de l'opération, nous avons donc une liste de quadrilatères modélisant les traits du dessin. Mais nous disposons également d'une liste de vecteurs n'ayant pas pu être appariés. Ces vecteurs sont généralement représentatifs des contours de formes pleines.



(a) image (b) Résultat

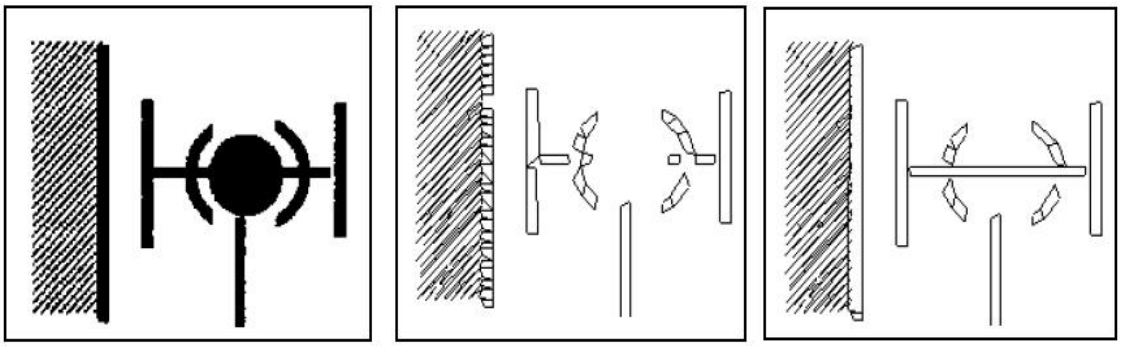
Fig. 3.10 : Un exemple de résultat fournit par la technique *Vect + Quad* (Quadrilatères en vert et Vecteur en rouge).

■ *Fusion des quadrilatères*

Compte tenu du mode de création des quadrilatères, un trait ne devant normalement être représenté que par un quadrilatère se trouvera parfois représenté par un ensemble de quadrilatères. C'est pourquoi il est nécessaire d'introduire une phase de fusion des quadrilatères pour obtenir des primitives de description se rapprochant le plus des traits de l'image.

Avant de procéder à la fusion, il est nécessaire de trier les quadrilatères selon leur proximité. Si les quadrilatères sont relativement alignés alors ils sont susceptibles d'être fusionnés. Pour cela, on vérifie que la différence de pente de leurs axes médians est relativement faible. De plus, pour qu'ils soient fusionnés, il faut qu'ils appartiennent au même trait. Il faut donc vérifier dans le document d'origine la présence de pixels noirs entre les quadrilatères.

Le premier quadrilatère étudié est celui ayant le voisin le plus éloigné. L'algorithme d'approximation polygonale de Wall est ensuite appliqué aux extrémités des axes médians des quadrilatères mais en tenant compte cette fois-ci de leurs attributs. En effet, deux quadrilatères ne peuvent être fusionnés s'ils ont des épaisseurs trop différentes ou s'ils sont séparés par une zone blanche dans l'image d'origine. Enfin, le processus est réitéré jusqu'à stabilisation, c'est-à-dire lorsque plus aucune fusion n'est possible.



(a) Image initiale (b) Quadrilatères avant fusion (c) Quadrilatères après fusion

Fig. 3.11 : Fusion des quadrilatères.

3.4 Améliorations de la méthode Vect + Quad

3.4.1 Amélioration de l'approximation polygonale

La méthode *Vect + Quad* cherche à approximer les contours à l'aide de vecteurs. On cherche donc à décomposer les contours en un ensemble de vecteurs se succédant en un circuit fermé. Ce problème d'approximation polygonale peut être formulé de deux manières différentes:

Problème min - ϵ : Étant donné une courbe polygonale *P* de *N* sommets, on cherche à l'approximer par une autre courbe polygonale *Q* avec un nombre donné M de segments de telle manière que l'erreur d'approximation soit minimale.

Problème min - # : Etant donné une courbe polygonale *P* de *N* sommets, on cherche à l'approximer par une autre courbe polygonale *Q* avec un nombre minimum de segments de telle manière que l'erreur ne dépasse pas une valeur de tolérance maximale Δ.

Le problème qui nous intéresse ici est le problème min- #. D'autre part, il existe bon nombre de méthodes permettant de résoudre ces deux types de problèmes. Pour les problèmes min-ϵ, il existe des méthodes optimales basées sur la programmation dynamique [Perez, 1994] [Horng, 2002] ou la théorie des graphes [Chan, 1996] [Chen, 1998]. Plus récemment, des méthodes basées sur la reconnaissance de segments flous [Nguyen, 2005] ont également été proposées. Les deux types de problèmes étant proches, on peut parfois étendre une méthode à l'autre. Cependant, ces méthodes sont très lentes, pas forcément adaptées à notre problème et ne donnent pas toujours de très bons résultats. En ce qui nous concerne, il existe deux grandes familles de solutions [Garnesson, 1991] :

- Méthodes de fusion

Le principe est de parcourir la chaîne du premier point au dernier point. Au fur et à mesure du parcours, les points rencontrés sont fusionnés en un même segment tant que l'ensemble de ces points vérifie un certain critère. Le dernier point vérifiant le critère correspond à l'extrémité du segment. On recommence alors ce processus à partir de ce point. Le principal défaut de ces méthodes (méthode de Wall notamment) est que les

points de coupure se situent généralement au-delà des points de coupure souhaités (figure. 3.12).

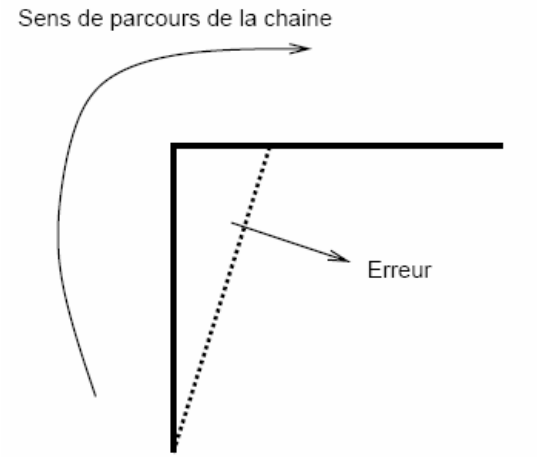


Fig. 3.12 : Les méthodes de fusion génèrent une erreur sur la position du point de coupure.

- Méthodes par découpage

Le principe est de commencer à approximer grossièrement la courbe puis de l'affiner par découpages récursifs tant qu'un certain critère est vérifié. Généralement, la courbe est d'abord approximée à une droite entre deux points extrémités de la courbe. On cherche alors le point de la courbe se situant le plus loin de la droite suivant un certain critère. Si cette distance dépasse un certain seuil, on coupe la droite en ce point et on recommence sur les deux segments obtenus sinon on arrête. Le processus contrairement à la précédente, cette méthode est plus précise quant aux choix des points de coupure. Cependant, elle a l'énorme désavantage d'être très sensible au bruit (figure 3.13 et figure 3.14).

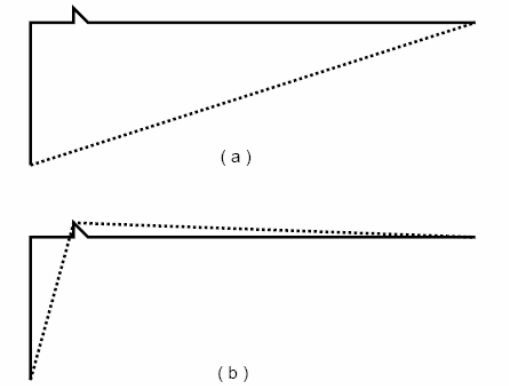


Fig. 3.13 : L'influence du bruit : le coin est "cassé ".

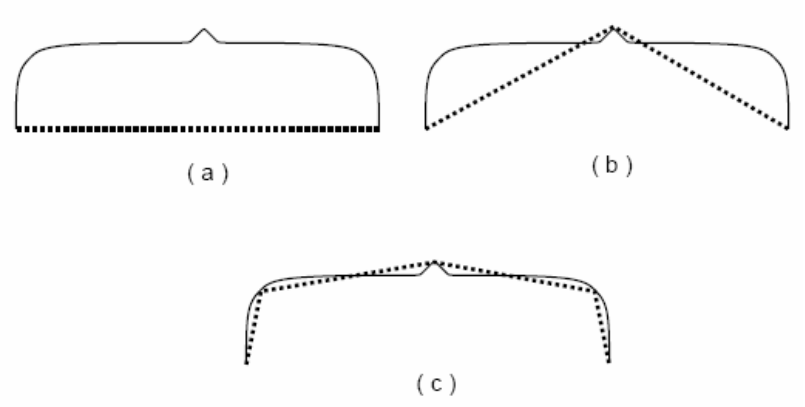


Fig. 3.14 : L'influence du bruit : (a) résultat à la première étape, (b) résultat à la deuxième étape, (c) résultat final, une droite a été coupée à cause du bruit [Garnesson, 1991].

On voit déjà que quelle que soit la méthode utilisée, il va falloir introduire un paramètre modulant la précision avec laquelle le contour va être défini. Si l'approximation est trop précise, on risque de ne conserver que des vecteurs de taille très petite et en grand nombre n'apportant rien de plus que les pixels. Par contre si l'approximation n'est pas assez précise, on aura de grands vecteurs en nombre réduit et donc une perte importante d'information. Ainsi, la méthode a pour but de fournir une approximation des contours en diminuant considérablement les données de manière à ne conserver que ce qui est nécessaire sans pour autant entraîner trop de perte d'information.

La méthode de Wall (utilisé dans *Vect + Quad*) est une méthode de fusion qui calcule une approximation de la surface séparant la courbe réelle de son approximation afin de juger de la qualité de la représentation. Si l'erreur totale (surface) dépasse un seuil dépendant de la longueur du segment alors un vecteur supplémentaire est créé. Le schéma suivant précise la méthode de calcul de l'aire considérée à chaque étape.

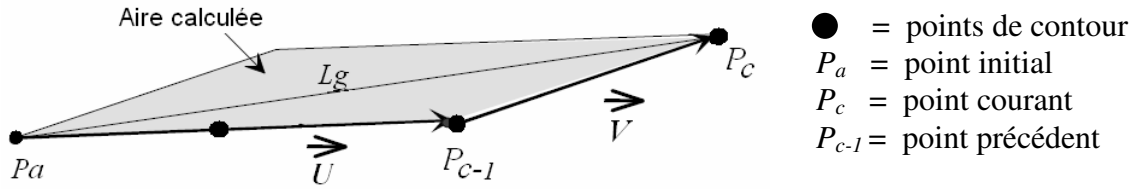


Fig. 3.15 : Critère utilisé lors de l'approximation.

Avec ces notations, l'erreur vérifie :

$$Erreur = Erreur + \left\| \vec{U} \wedge \vec{V} \right\| \quad (1)$$

$$Erreur = Erreur + \left| (x_{c-1} - x_a)(y_c - y_{c-1}) - (y_{c-1} - y_a)(x_c - x_{c-1}) \right| \quad (2)$$

La condition d'arrêt est alors :

$$Erreur > Lg \times \varepsilon \quad (3)$$

$$Erreur > \sqrt{(x_c - x_a)^2 + (y_c - y_a)^2} \times \varepsilon \quad (4)$$

On s'aperçoit alors que la qualité de l'approximation va dépendre du paramètre ϵ. Plus celui-ci est grand, plus on obtiendra de grands vecteurs et une erreur importante sur la position des points de coupure. Ceux-ci étant généralement situés au-delà des points d'inflexion effectifs. Il est alors nécessaire de tenter d'appliquer une sorte de retour en arrière de manière à bien positionner le point de coupure.

Nous avons vu que le seuil choisi dans l'algorithme de Wall est pondéré par la longueur du segment considéré. Sa valeur est :

$$Seuil = \varepsilon \times Lg$$

Ceci peut être vu comme la surface définie par le rectangle englobant les segments de longueur *Lg* et de largeur ϵ. Le problème peut être représenté avec le schéma suivant :

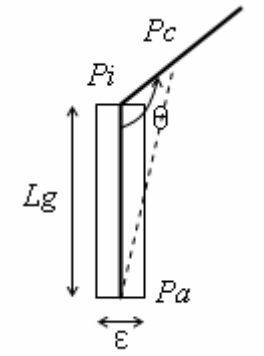


Fig. 3.16 : Retour avec un angle quelconque.

Ici, *Pa* désigne le point de départ du segment, *Pc* le point de coupure trouvé par l'algorithme de Wall et *Pi* le point d'inflexion réel. La quantité de retour cherchée est alors la longueur du segment [*Pc*, *Pi*]. La surface calculée par l'algorithme est alors :

$$S = Lg \times |[P_i, P_c]| \times \sin(\theta)$$

On a donc :

$$\varepsilon \times Lg = Lg \times |[P_i, P_c]| \times \sin(\theta)$$

Et ainsi :

$$|[P_i, P_c]| = \frac{\varepsilon}{\sin(\theta)}$$

On vérifie alors que si l'angle est nul, il n'y a pas de point d'inflexion et donc de point de coupure ce qui correspond à une valeur de retour infinie. Par contre, si l'angle est de 90° le retour est égal à ϵ comme on peut le vérifier sur le schéma suivant :

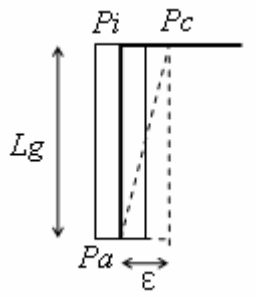


Fig. 3.17 : Retour avec un angle à 90°.

La valeur de l'angle peut osciller autour de 90°. De plus, il n'est pas aisé de déterminer cet angle. Nous avons alors décidé, dans un premier temps, de le fixer à 90° ce qui correspond à un retour valant ϵ. L'implémentation de ce procédé nous a permis d'améliorer la méthode "*Vect + Quad*" dans notre système. De par ce choix et la nature des calculs, ceci reste une approximation par rapport à l'optimale mais donne d'assez bons résultats tout en ouvrant la voie à une réflexion (pour trouver peut être une solution encore plus précise) que nous n'avons pas pris le temps de poursuivre pour l'instant (car en dehors des objectifs de la thèse).

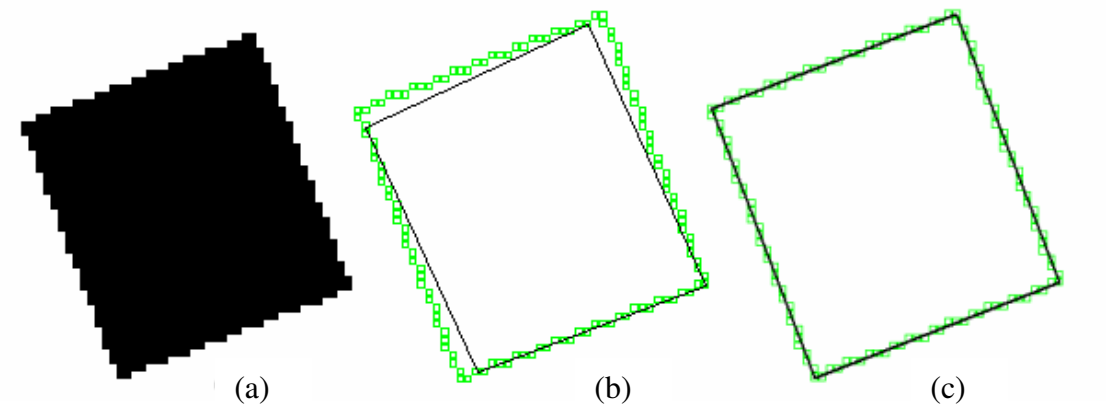


Fig. 3.18 : (a) image d'origine, (b) approximation sans retour, (c) approximation avec retour, en clair : les pixels du contour détecté, en foncé : les segments déterminés par approximation polygonale.

Enfin, on peut également imaginer une méthode supplémentaire tirant partie des deux types de méthodes que nous avons vus, à savoir fusion et découpage. On pourrait, par exemple, débuter par une approximation du contour à l'aide d'une méthode de découpage afin d'obtenir de bons points de coupure puis d'essayer d'´eliminer les erreurs liées au bruit à l'aide d'une méthode de fusion. Ce type de méthode reste un autre champ d'investigation à explorer.

3.4.2 Amélioration de la représentation des formes fines

La méthode *Vect + Quad* produit une représentation des formes fines à l'aide de quadrilatères qui peut, selon nous, être améliorées sur quelques points. Par exemple, le quadrilatère vertical (Q_1) aurait dû être découpé en deux parties pour donner une représentation plus idéale de la forme initiale.

Les critères retenus pour autoriser la fusion de deux quadrilatères étaient la similarité des épaisseurs et la présence de pixels noirs entre les deux quadrilatères. Nous avons observé qu'une étude de la distance entre les extrémités des deux quadrilatères peut éviter certaines fusions inutiles, on définit une distance inter-quadrilatères δ calculée entre les extrémités des segments médians des deux quadrilatères. Les vérifications effectuées avant la fusion dans la nouvelle version sont :

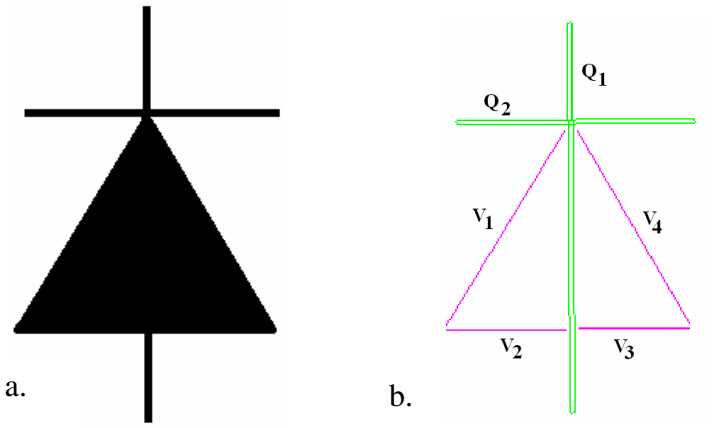


Fig. 3.19: a) Image initiale b) vecteurs et quadrilatères produits par Vect + Quad.

Si | angleQ_1 – angleQ_2| < seuil_Angle **ET** Noir_entre(Q_1, Q_2) **ET** (δ < seuil_distance) **alors**

Autoriser la fusion de quadrilatères

Sinon

Refuser la fusion de quadrilatères

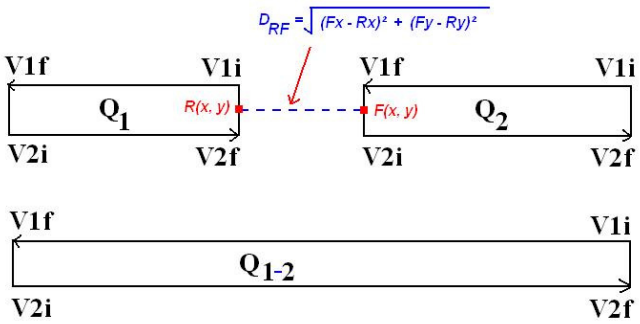


Fig. 3.20 : Distance entre quadrilatères.

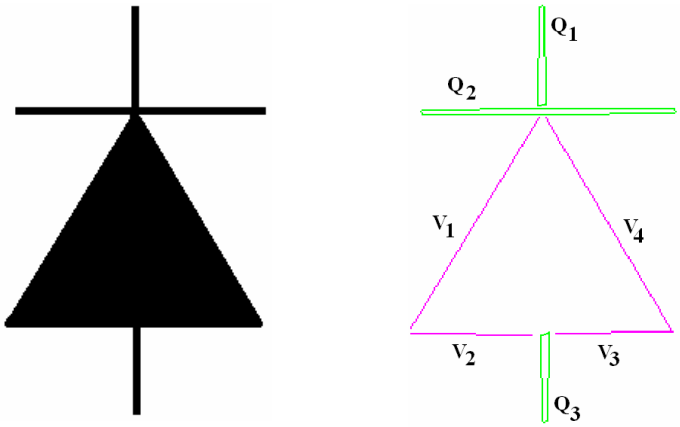


Fig. 3.21 : Fusion des quadrilatères intégrant un critère de distance.

3.4.3 Amélioration de la représentation des formes pleines

La figure 3.24b montre un problème de représentation des frontières des formes pleines à l'aide de vecteurs liés aux intersections possibles entre formes pleines et forme fines et au choix non optimal du point de départ par l'algorithme de suivi de contour (multiplication de vecteurs successifs). Afin de corriger ce problème, nous ajoutons une étape de fusion des vecteurs pour obtenir une liste minimale de vecteurs et pour augmenter encore la robustesse de notre représentation de l'image initiale. Cette étape de fusion était déjà présente pour les quadrilatères mais n'avait pas été mise en place pour les vecteurs. En effet, l'idée est de réappliquer l'algorithme d'approximation polygonale sur les points extrémités de vecteur de manière à fusionner chaque vecteur avec un vecteur présent dans le rayon défini et avec qui la différence de pente est inférieure à un certain seuil.

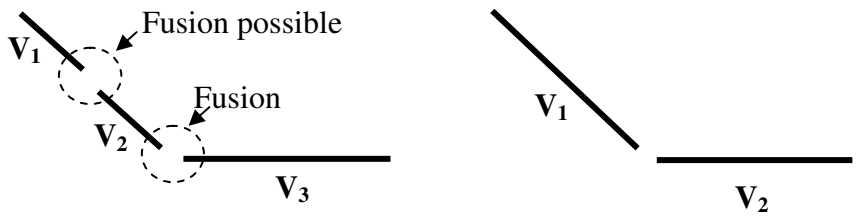
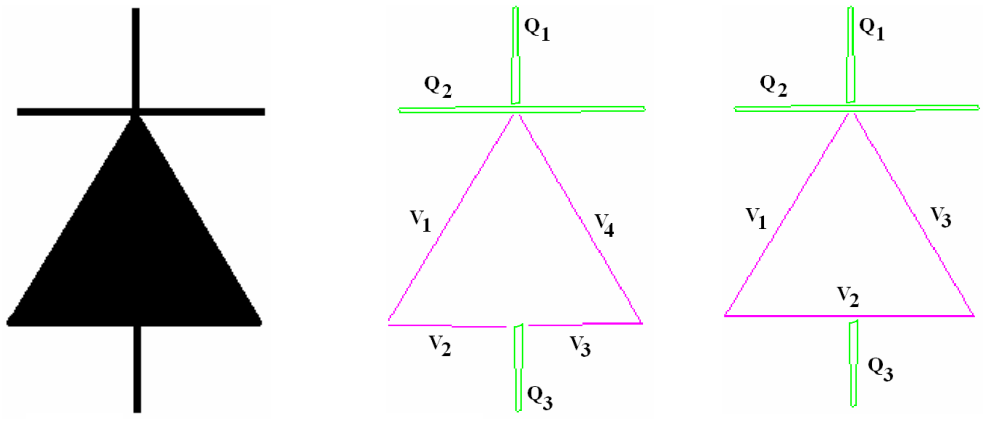


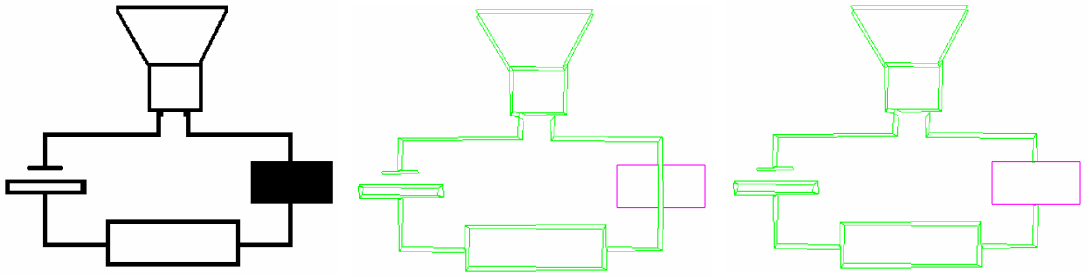
Fig. 3.22 : Fusion de vecteurs.

Ce procédé est effectué itérativement jusqu'à stabilisation, c'est-à-dire lorsque l'algorithme ne parvient plus à fusionner de vecteurs.



a. b. c.

Fig. 3.23 : a) image initiale b) vecteurs produits par *Vect + Quad* c) Vecteurs après modifications de la méthode (fusion ajoutée).



a) image initiale b) Résultat *Vect* + *Quad* c) Résultat avec les améliorations apportées

Fig. 3.24: Exemples de résultats avant et après améliorations de la méthode *Vect* + *Quad*.

3.4.4 Améliorations apportées au graphe structurel

Nous avons vu dans le chapitre 1, la puissance de la description des graphes de primitives mais aussi la difficulté de la construction d'une représentation unique et suffisamment générique. Le graphe structurel proposé dans [Ramel, 1996] ne représente justement qu'une partie du contenu des images de documents graphiques : les formes fines à l'aide de la primitive quadrilatère (nœuds) tandis que les arcs représentent le type de relation (*T*, *P*, *X*, *S*, ou *L*) entre primitives quad (figure 3.25).

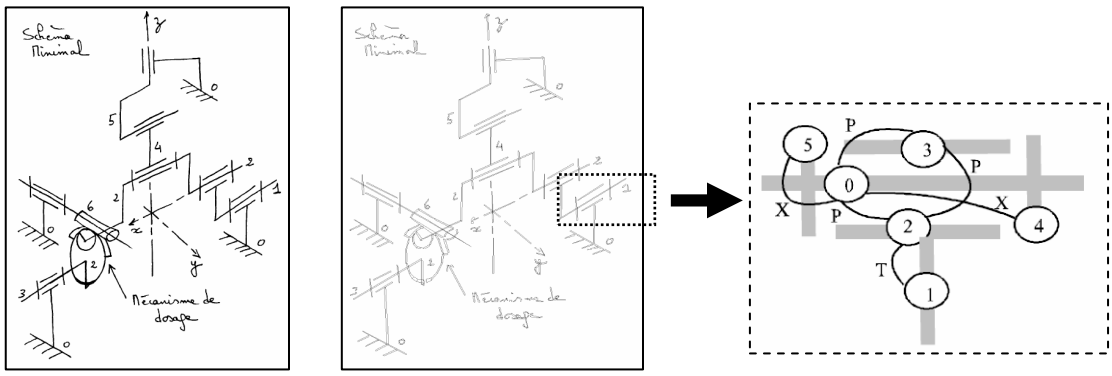
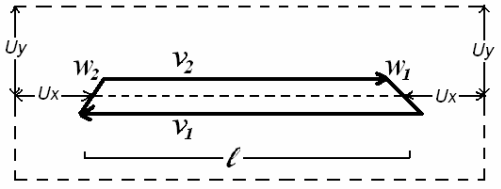


Fig. 3.25 : Le graphe de quadrilatères [Ramel, 1996].

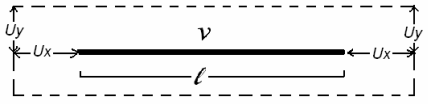
Les formes pleines (représentées par les vecteurs de contours) ne sont quant à elles pas décrites par le graphe structurel. D'ailleurs, à notre connaissance aujourd'hui, aucune représentation structurelle n'intègre simultanément une description des formes fines et formes pleines. Nous avons donc désiré intégrer les primitives vecteurs représentant les formes pleines et les quadrilatères représentant les formes fines dans un unique graphe afin d'obtenir une représentation unique et complète du contenu de documents diversifiés.

■ *Analyse des relations entre primitives*

Reprenant le procédé utilisé pour construire le graphe des quadrilatères, notre graphe est construit en définissant autour de chaque primitive une zone d'influence. La taille de cette zone est fonction de l'épaisseur et de la longueur de la primitive traitée (figure 3.26). Pour chaque primitive, on cherche à déterminer les primitives se trouvant dans la zone d'influence. On analyse alors précisément les relations existantes entre les primitives appartenant à la zone et on les sauvegarde sous forme d'attributs associés aux arcs.



a)



b)

Fig. 3.26: a) Un quadrilatère et sa zone d'influence, $Ux = l/4$ et $Uy = (w_1 + w_2)$
b) Un vecteur et sa zone d'influence, $Ux = l/4$ et $Uy = 10$.

La figure 3.27 montre comment les primitives proches spatialement sont associées lors de la construction du graphe. Chaque sommet du graphe représente une primitive graphique (soit un quadrilatère, soit une vecteur) et ses caractéristiques sont enregistrées comme attributs (longueur, angle, épaisseur…). De même, les attributs de chaque arc fournissent des informations précises sur la relation existante entre les deux primitives considérées.

L'analyse successive de la zone d'influence de chaque primitive extraite de l'image initiale permet d'obtenir, au final, un graphe complet représentant l'ensemble du contenu de l'image. Cette méthode permet d'obtenir une représentation du contenu d'une image à l'aide d'un graphe non orienté muni d'attributs numériques et symboliques sur les sommets et les arcs (figure 3.27).

Dans la suite de ce manuscrit, chaque image sera représentée par un graphe attribué G défini par un 4-tuple $G = (V,E,\alpha,\beta)$ où : V est l'ensemble fini des sommets, $E \subseteq V \times V$ est l'ensemble des arcs, $\alpha : V \rightarrow A_V^i$ est une fonction qui assigne les attributs aux sommets, $\beta : E \rightarrow A_E^j$ est une fonction qui assigne les attributs aux arcs. Soit A_v et A_E désignant respectivement les attributs des sommets et des arcs, i varie de 1 à δ et j varie de 1 à Ω, où δ et Ω représentent respectivement le nombre d'attributs associés aux sommets et aux arcs. La section suivante rappelle les attributs disponibles sur les nœuds et arcs de notre graphe.

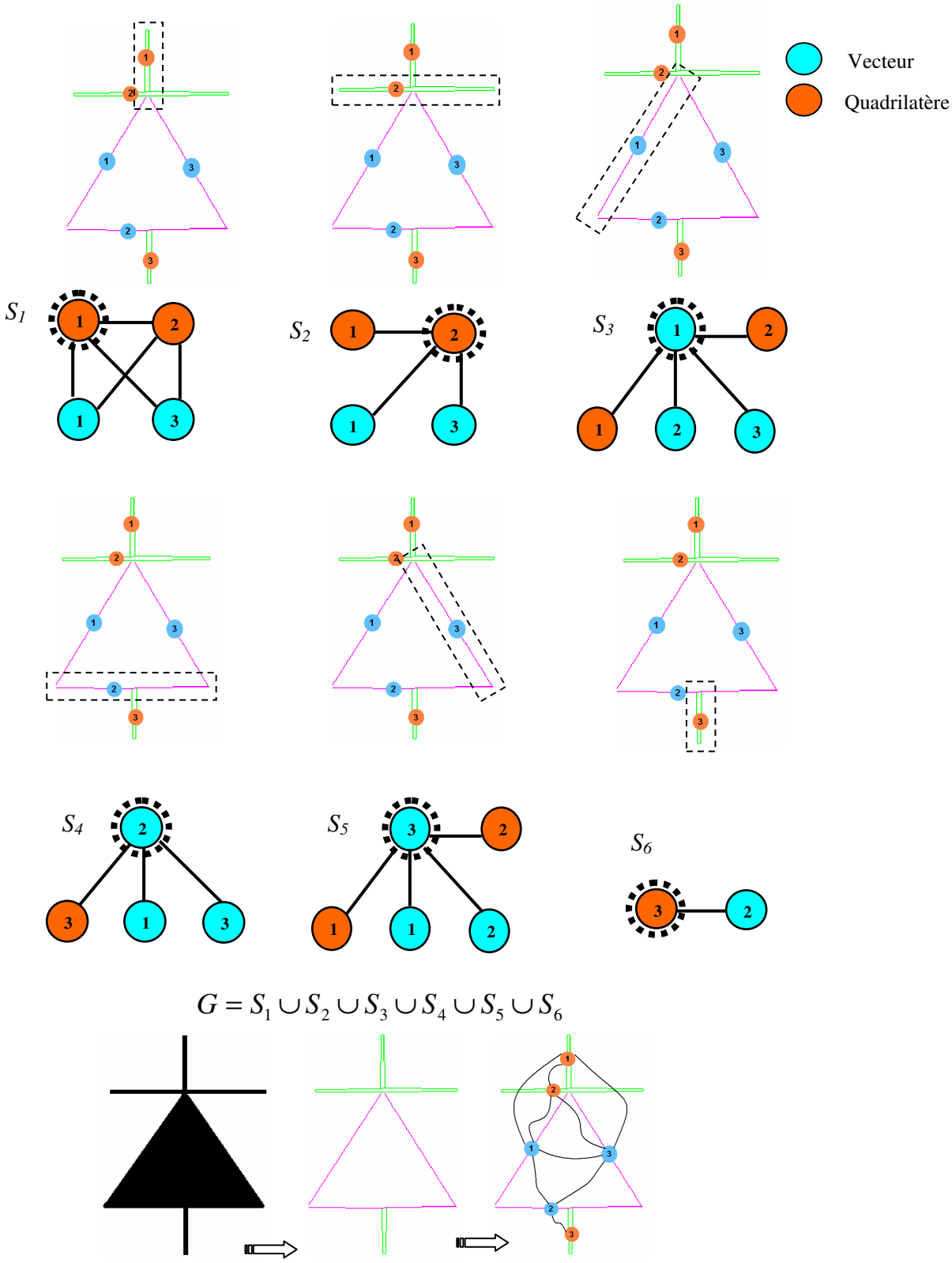


Fig. 3.27 : Construction du graphe associé à une image.

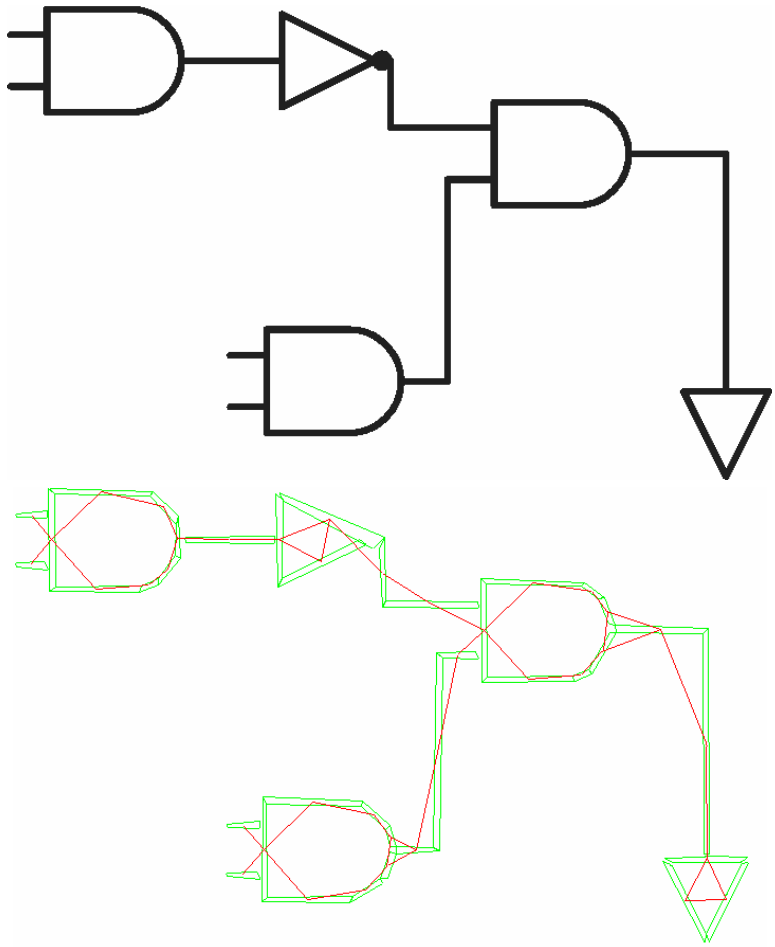


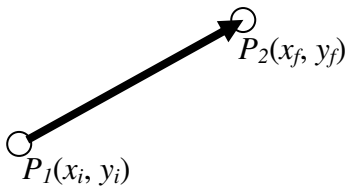
Fig. 3.28 : Un diagramme logique et son graphe associé.

■ Attributs associés aux nœuds et arcs du graphe

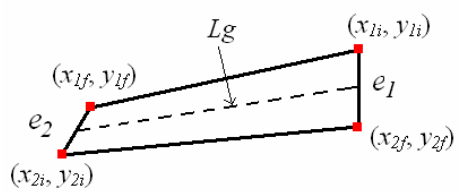
■ *Pour les nœuds*

Les attributs associés aux nœuds varient selon le type de primitives considérées : chaque représentant de la classe Vecteurs possède les attributs suivants :

- Pixel initial (x_i, y_i) : P_1
- Pixel final (x_f, y_f) : P_2
- Longueur (exprimée en pixels)
- Angle θ (par rapport à l'horizontale)
- Epaisseur (un pixel)



Chaque représentant de la classe Quad (construit à partir de deux Vecteurs) possède les attributs suivants :



- Pixel initial du Vecteur-1 (x_{1i}, y_{1i}) : P_{1i}
- Pixel final du Vecteur-1 (x_{1f}, y_{1f}) : P_{1f}
- Pixel initial du Vecteur-2 (x_{2i}, y_{2i}) : P_{2i}
- Pixel final du Vecteur-2 (x_{2f}, y_{2f}) : P_{2f}
- Angle du Vecteur-1 θ_1 (par rapport à l'horizontale)
- Angle du Vecteur-2 θ_2 (par rapport à l'horizontale)
- Epaisseur-1 : e_1
- Epaisseur-2 : e_2
- Longueur (exprimée en pixels) : Lg

■ *Pour les arcs*

Les arcs décrivent les relations que l'on peut trouver entre primitives. Nous avons choisi de caractériser ces relations grâce aux attributs suivants :

1. Le type des relations topologiques : Il s'agit d'un attribut symbolique qui peut prendre les valeurs suivantes (voir figure. 1.16, chapitre-1):

   – T (liaison en T)
   – X (intersection)
   – P (parallèles)
   – L (liaison en L)
   – S (successifs)

2. L'angle relatif : Il s'agit d'une valeur numérique exprimée en degré comprise dans l'intervalle [0 ; 90°] calculée à partir de la différence entre les angles des axes médians des deux primitives mises en relation (figure. 3.29).

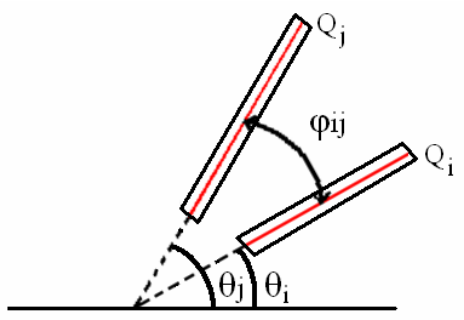


Fig. 3.29 : Angle entre des axes médians des primitives.

■ *Format de stockage et de diffusion des graphes*

Nous avons choisi d'utiliser le format GXL [Web-2] pour stocker et diffuser les graphes produits par notre logiciel. GXL est un format standard d'échange pour ces graphes. Le format GXL respecte le standard XML, sa syntaxe est donnée par une DTD (Document Type Définition) XML. Ce format d'échange offre un moyen flexible et adaptable pour gérer l'interopérabilité entre les outils basés sur des graphes. L'annexe-1 montre un exemple de représentation d'un graphe sous le format GXL avec également, un schéma pour décrire la structure des données.

3.5 Localisation de symboles graphiques à partir de cette représentation structurelle

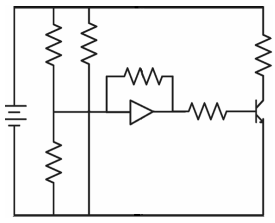
Beaucoup de méthodes de reconnaissance graphique ont été développées tout au long de ces dernières années pour la reconnaissance des symboles graphiques pré-segmentés mais très peu de techniques ont atteint l'objectif de localisation et de reconnaissance de symboles sans connaissance a priori. Un état de l'art des méthodes existantes a été présenté dans le chapitre 2. La plupart du temps les méthodes proposées utilisent une des deux approches classiques suivantes : La première catégorie correspond à l'approche directe basée sur la recherche des configurations particulières de pixels, d'ensembles de lignes, de texture, de boucles, de composantes connexes, etc. [Song, 2002a] [Song, 2002b]. La deuxième catégorie inclut les méthodes à base de signatures (une collection de caractéristiques) calculées sur différentes parties des images [Dosch, 2004] [Wang, 2006] [Wenyin, 2007] et ensuite comparées avec les signatures des modèles pour un type de documents stockés dans la bibliothèque de symboles. Selon nous, aucune méthode suffisamment générique n'existe. De plus,

vouloir produire un système complètement automatique semble illusoire et une interaction entre l'utilisateur et le système est toujours nécessaire.

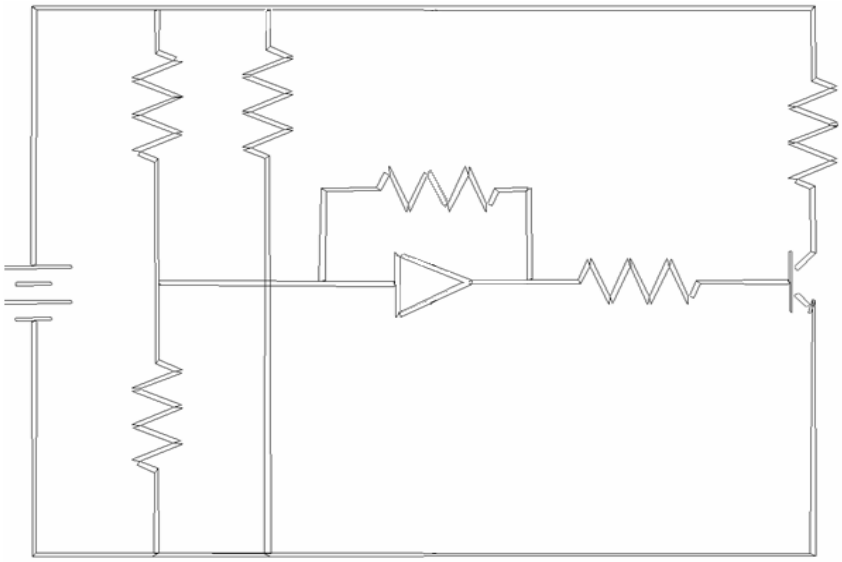
Pour avancer sur cette problématique, nous présentons une solution originale pour la localisation de symboles en utilisant notre représentation structurelle des documents. La stratégie proposée comporte donc deux étapes principales. Dans la première étape, l'image de document est représentée par un graphe de primitives. Dans la deuxième étape, le graphe est employé pour localiser les parties intéressantes de l'image qui correspondent potentiellement aux symboles sans aucune connaissance à priori sur le type de documents graphiques analysés. L'approche est parallèle et est potentiellement capable de localiser tous les symboles dans un schéma en un seul passage. Les sous-graphes associés aux zones d'intérêt choisies peuvent être soumis à un algorithme de mise en correspondance de graphes afin de prendre la décision finale et d'identifier la classe du symbole. Dans les sections suivantes, nous expliquons seulement les étapes de sélection et extraction des zones d'intérêt correspondant potentiellement aux symboles « Bounding Boxes ». Par contre, la mise en correspondance de graphe/sous-graphe est détaillée dans le chapitre 4. Les résultats expérimentaux obtenus sur différents types de documents démontrent que le système peut manipuler différents types d'images sans modification.

3.5.1 Exploitation de notre représentation sous forme de graphe du contenu des images

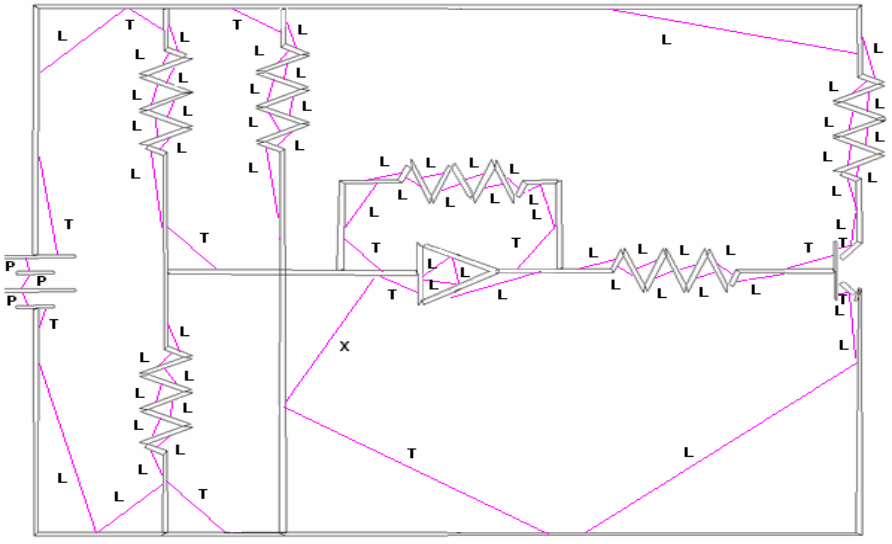
Dans notre graphe structurel, les nœuds représentent des primitives de description vectorielles (Vecteurs, Quadrilatères) (figure 3.30b) et les arcs représentent les relations spatiales et géométriques entre les nœuds voisins. La longueur de la primitive est associée à chaque nœud comme attribut, sur les arcs, l'angle relatif décrit l'angle entre les nœuds voisins. Tous les arcs possèdent également une étiquette représentant le type des relations topologiques (*L*-jonction, *S*-jonction, *T*-jonction, *X*-intersection ou *P*-parallélisme) qui existe entre les deux primitives (figure 3.30c). Nous avons fourni une description détaillée de cette représentation dans la section 3.4 de ce chapitre.



(a)



(b)



(c)

Fig. 3.30: (a) Image initiale, (b) résultat de la vectorisation, (c) représentation obtenue sous forme de graphe.

3.5.2 Classification des nœuds et arcs du graphe

L'idée principale de notre méthode est de détecter les parties du graphe qui peuvent correspondre à des symboles sans connaissance a priori au sujet du type de document. Ces nœuds et arcs constitueront des parties de symboles. Ces éléments détectés et étiquetés comme "morceau de symbole" seront analysés et groupés pour générer les sous-graphes qui correspondent potentiellement aux symboles dans l'image de document. Dans un premier temps, nous avons choisi de ne pas mettre en place un véritable système d'apprentissage et classification à même de séparer les éléments des graphes en 2 classes distinctes (symboles et non symboles) mais d'essayer de dégager des heuristiques générales sur la structure des schémas. Nous avons observé de l'analyse de structures de schémas, de circuits, de cartes et de diagrammes, que des symboles se composent de primitives avec des caractéristiques spécifiques facilement discernables dans la représentation sous forme de graphe. Disposant d'un grand nombre de documents de types variés (figures 3.31, 3.32, 3.33), une étude a été menée (voir tab. 3.1) afin d'analyser les caractéristiques des éléments (nœuds et arcs) correspondant aux symboles dans les graphes obtenus à partir de ces images. Ces images ne comportent pas de formes pleines. Nous nous sommes intéressés, pour l'instant, qu'aux primitives de type QUAD. La longueur relative $\boldsymbol{R}_{Len}$ est obtenue en normalisant toutes les longueurs des primitives par rapport à la longueur maximum détectée dans l'image. Il nous semble que cet attribut associé aux nœuds du graphe constitue l'un des critères les plus pertinents pour distinguer les symboles des autres parties du dessin.

Tab. 3.1 : Etude des longueurs relatives(R_{Len}) des QUAD dans les circuits électroniques, circuits logiques et les plans.

Fichiers	circuits électroniques										
	R_{Len}	0-0,1	0,1-0,2	0,2-0,3	0,3-0,4	0,4-0,5	0,5-0,6	0,6-0,7	0,7-0,8	0,8-0,9	0,9-1,0
20	Nb. de Quad	606	223	103	54	28	27	18	10	4	4
	% Quad	0,562	0,207	0,095	0,050	0,025	0,025	0,016	0,009	0,003	0,003
circuits logiques											
	R_{Len}	0-0,1	0,1-0,2	0,2-0,3	0,3-0,4	0,4-0,5	0,5-0,6	0,6-0,7	0,7-0,8	0,8-0,9	0,9-1,0
20	Nb. de Quad	951	410	323	93	52	41	37	29	44	16
	% Quad	0,48	0,205	0,161	0,046	0,026	0,020	0,018	0,014	0,022	0,01
plans de maison											
	R_{Len}	0-0,1	0,1-0,2	0,2-0,3	0,3-0,4	0,4-0,5	0,5-0,6	0,6-0,7	0,7-0,8	0,8-0,9	0,9-1,0
11	Nb. de Quad	3411	383	169	41	26	22	13	10	12	7
	% Quad	0,833	0,093	0,041	0,010	0,006	0,005	0,003	0,002	0,002	0,001

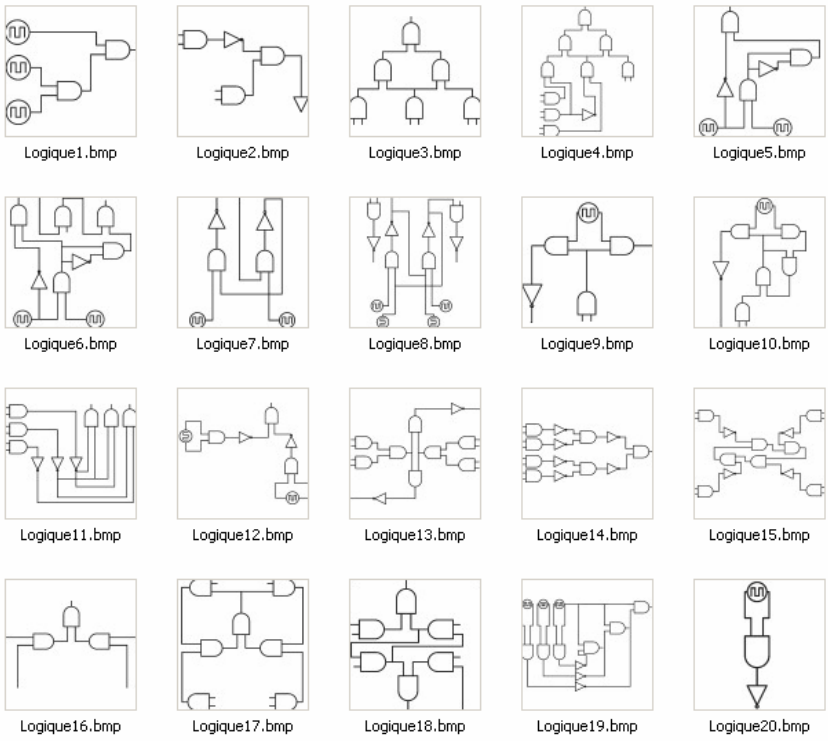


Fig. 3.31 : Base d'images test - Logique (BD_1).

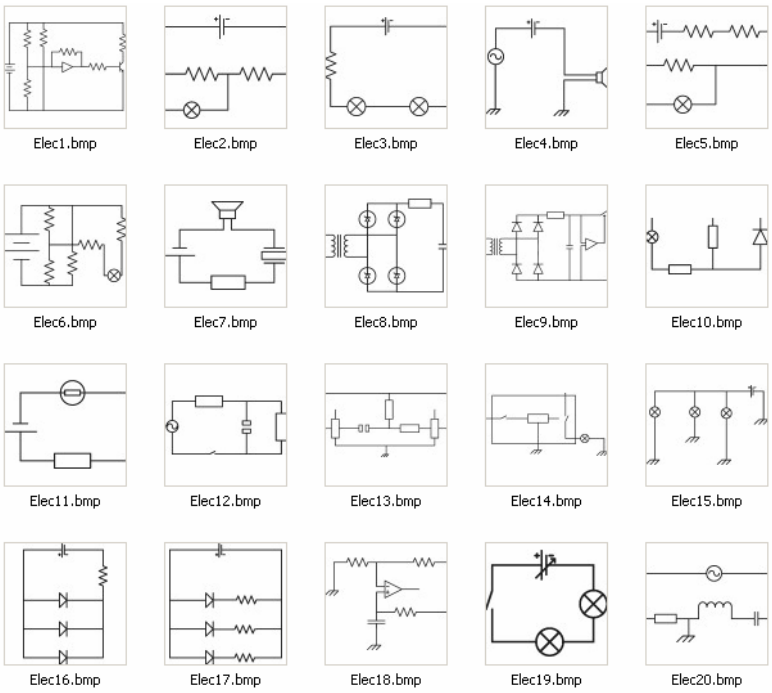


Fig. 3.32 : Base d'images test - Electronique (BD_2).

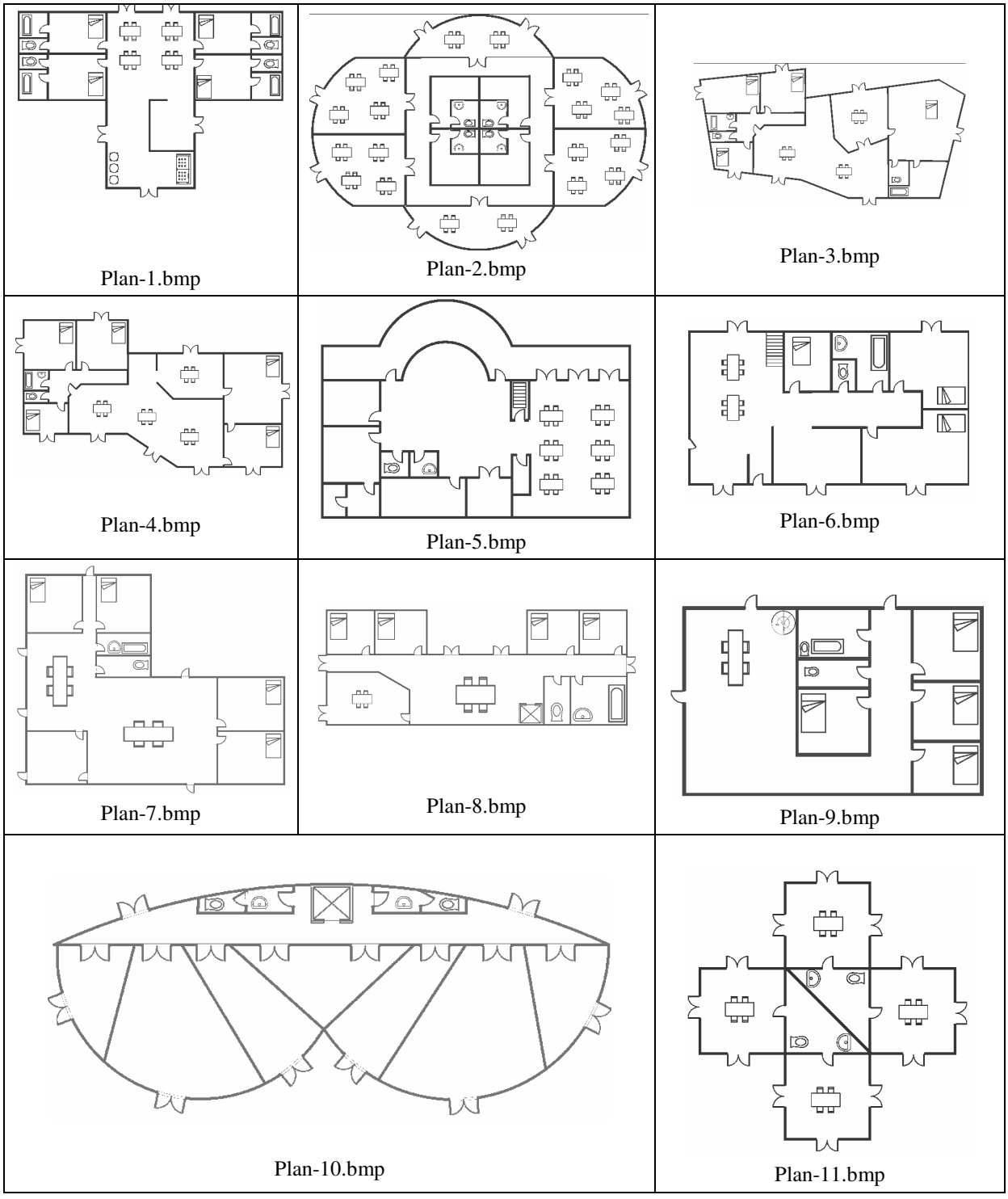


Fig. 3.33 : Base d'images test – Plans architecturaux (BD_3).

Dans ce tableau, deux valeurs sont remarquables : 0< R_{Len}<0,1 et 0,1< R_{Len}<0,2. montrant que les symboles sont généralement constitués d'au moins un petit QUAD quelque soit le type de documents. Les images de plan de maison possèdent un grand nombre de QUAD de petite taille. En effet, 84% des QUAD constituant les images de plan sont des QUAD de longueur relative comprise entre 0 et 0,1. Cela s'explique par le fait que dans ce type de schéma, les murs sont de taille très grande par rapport à la taille des plus grands QUAD composant les symboles. Les camemberts suivants représentent d'une manière plus simplifiée la répartition des longueurs de primitives dans nos graphes.

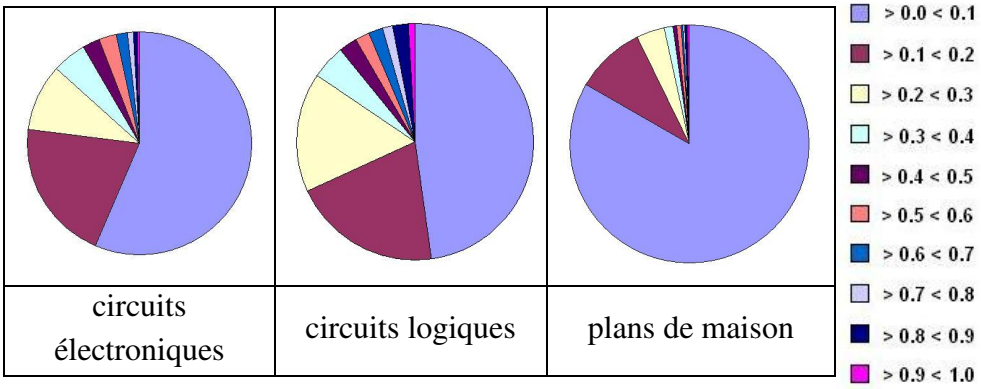


Fig. 3.34 : Répartition des longueurs relatives.

Le nombre de voisins pour un nœud semble également un critère discriminant. Le nombre de QUAD ayant 2 voisins est très important dans les circuits électroniques. En effet, la proportion des QUAD ayant deux voisins (2V) dans les différents types d'image est sensiblement égale. On compte 48% pour les circuits logiques, 37% pour les circuits électroniques et 34% pour les plans de maison.

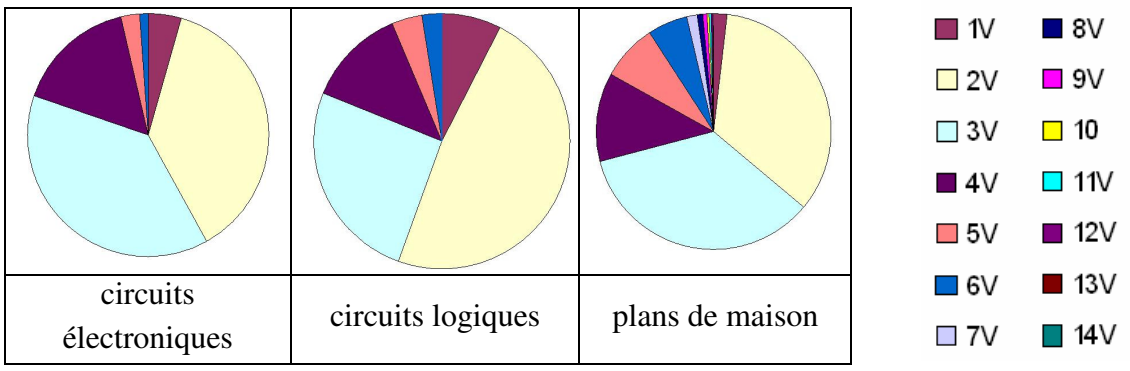


Fig. 3.35 : Influence du nombre de voisins des QUAD.

De telles analyses ont été produites pour l'ensemble des informations disponibles dans nos graphes. Cette étude nous a permis de dégager des hypothèses sur les caractéristiques des nœuds et arcs des graphes correspondant à la classe symbole. Nous avons choisi d'utiliser les conclusions de cette étude pour mettre en place des fonctions associant des scores aux arcs et aux nœuds des graphes selon leurs caractéristiques. Ces scores (compris entre 0 et 1) peuvent être assimilés à la probabilité, pour un élément du graphe, d'appartenir à la classe "symbole".

Les 6 hypothèses employées pour calculer les scores des nœuds et des arcs est :

- H_1 – Les symboles se composent de petits segments comparés à d'autres éléments des schémas (fonction P_{N3} *( ) dans éq. 2)*
- H_2 – Les segments constituant un symbole ont des longueurs similaires (fonction P_{E3} *( ) dans éq. 1)*
- H_3 – Deux segments voisins avec un angle relatif loin de 90° correspondent souvent à une partie de symbole (fonction P_{E2} *( ) dans éq. 1)*
- H_4 – Des symboles se composent souvent de segments parallèles (fonction P_{E1}*( ) dans éq. 1)*
- H_5 – Un segment de symbole est rarement relié à plus de 3 autres segments (fonction P_{N2} *( ) dans éq. 2)*
- H_6 – Les symboles comportent souvent des boucles fermées « *loops* » (Propagation de score dans le graphe)

A partir des attributs des nœuds et des arcs des graphes, des scores entre 0 et 1 sont calculés pour chaque nœud et arc en utilisant les hypothèses ci-dessus. Le score d'un arc E_i est une fonction des deux longueurs et angles relatifs des nœuds (quadrilatères) mise en relation et du type de relation topologique représentée (éq.1). Le score d'un nœud N_i (éq. 2) est calculé en employant les scores accumulés de ses arcs incidents, du nombre d'arcs le reliant et de la longueur de la primitive correspondante (quadrilatère).

$$Score(E_i) = \alpha.\ P_{E1}(E_i \rightarrow Type) + \beta.\ P_{E2}(E_i \rightarrow Angle) + \gamma.\ P_{E3}(E_i \rightarrow N_1 \rightarrow Length,\ E_i \rightarrow N_2 \rightarrow Length) \quad (1)$$

$$Score(N_i) = \lambda\ \left(\frac{\sum_{j=1}^{\Omega(Ni)} Score(Ej)}{\Omega(Ni)}\right) + \mu\ .\ P_{N2}(\Omega(N_i)) + \omega\ .P_{N3}(N_i \rightarrow Length) \quad (2)$$

Dans éq.1, E_i est un arc du graphe, α, β et γ sont des poids et P_{Ei} sont des fonctions renvoyant des scores entre 0 et 1 selon les valeurs passées en paramètres. Dans nos expérimentations, nous avons choisi de fixer empiriquement α, β et γ à 1/3.

Dans éq.2, N_i est un nœud du graphe, $\Omega(N_i)$ est le degré de N_i, ω, μ et λ sont des poids et les P_{Ni} sont des fonctions renvoyant des scores entre 0 et 1 selon les valeurs passées en paramètres. Afin de donner plus de poids à la moyenne des scores associés aux arcs qu'aux autres caractéristiques du nœud, nous avons choisi de fixer $\lambda = 1/2$ et $\mu = \omega = ¼$).

3.5.3 Propagation des scores dans le graphe

Afin de tenir compte de l'hypothèses H_6, nous avons choisi de mettre en place une procédure de recherche de boucles (plus court chemin) dans les graphes et d'effectuer une propagation du plus grand score le long de ces boucles. Les scores de tous les nœuds appartenant à une boucle détectée sont ainsi homogénéisés (figure 3.36). Durant cette phase seuls les arcs avec des attributs de types *L* et *S* sont considérés lors de la construction des chemins. Cette étape modifie seulement les scores des symboles composés de boucles fermées mais n'a aucune influence sur d'autres parties du graphe.

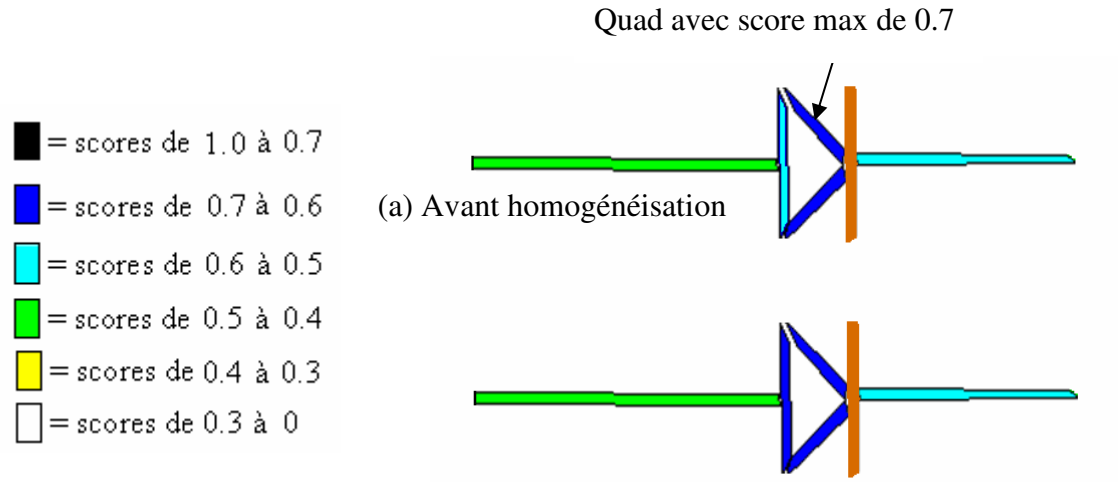


Fig. 3.36 : Propagation du score maximum à tous les nœuds dans la boucle.

3.5.4 Extraction des sous-graphes et génération des Bounding Boxes

Après le calcul des scores des nœuds et des arcs, l'extraction des sous-graphes du graphe complet doit être réalisée dans le but de fournir au module de reconnaissance de symboles les zones correspondant potentiellement à un symbole dans le document. Pour obtenir ces sous-graphes et les boîtes englobantes correspondantes (figure 3.38), un simple seuillage sur les scores des nœuds est réalisé. Notre méthode cherche dans le graphe, tous les ensembles de nœuds interconnectés ayant un score suffisant. Il est possible de faire varier le seuil (*Ts*) et d'analyser le nombre de symboles générés pour chaque valeur testée. Ce seuil (*Ts*) peut alors être choisi automatiquement en conservant la valeur qui maximise le nombre de symboles détectés en utilisant l'algorithme décrit dans la figure 3.37. Les expériences ont montré que les meilleurs résultats sont obtenus pour le seuil fixé à 0.6 qui correspond bien souvent au nombre maximum de symboles dans l'image (figure 3.38e).

Algorithme-SymbSpotting

Entrée : Un graphe représentant d'un document graphique
Sortie : Nombre de symboles localisés + liste de Bounding boxes(*BB*) +
liste de sous-graphes(*SG*) + seuil optimal choisi(*Ts)*

Début
pour $Ts = 0$ à 1 avec step = 0.1
$Max\_BB \leftarrow 0$ // Nombre maximum du bounding boxes
$Best\_Ts \leftarrow 0$ // Meilleure valeur du *Ts*
pour $j = 1$ à $| E |$
calcule le score(E_j)
pour $i = 1$ to $| N |$
calcule le score(N_i)

$Graines \leftarrow \forall (N_i) \mid \text{score}(N_i) \geq Ts$

Propagation des scores des nœuds dans les boucles arcs avec *L* et/ou *S* uniquement
$SG \leftarrow$ Sous-graphes (Graines)
$BB \leftarrow$ Bounding Boxes (Graines)

si (Cardinal(*BB*) > *Max_BB*)
alors
$Max\_BB \leftarrow$ Cardinal(*BB*)
$Best\_Ts \leftarrow Ts$
fin si
fin pour
Retourner (*Max_BB*, *BB*, *SG*, *Best_Ts*))
fin

Fig. 3.37: Algorithme de recherche de symboles (SymbSpotting).

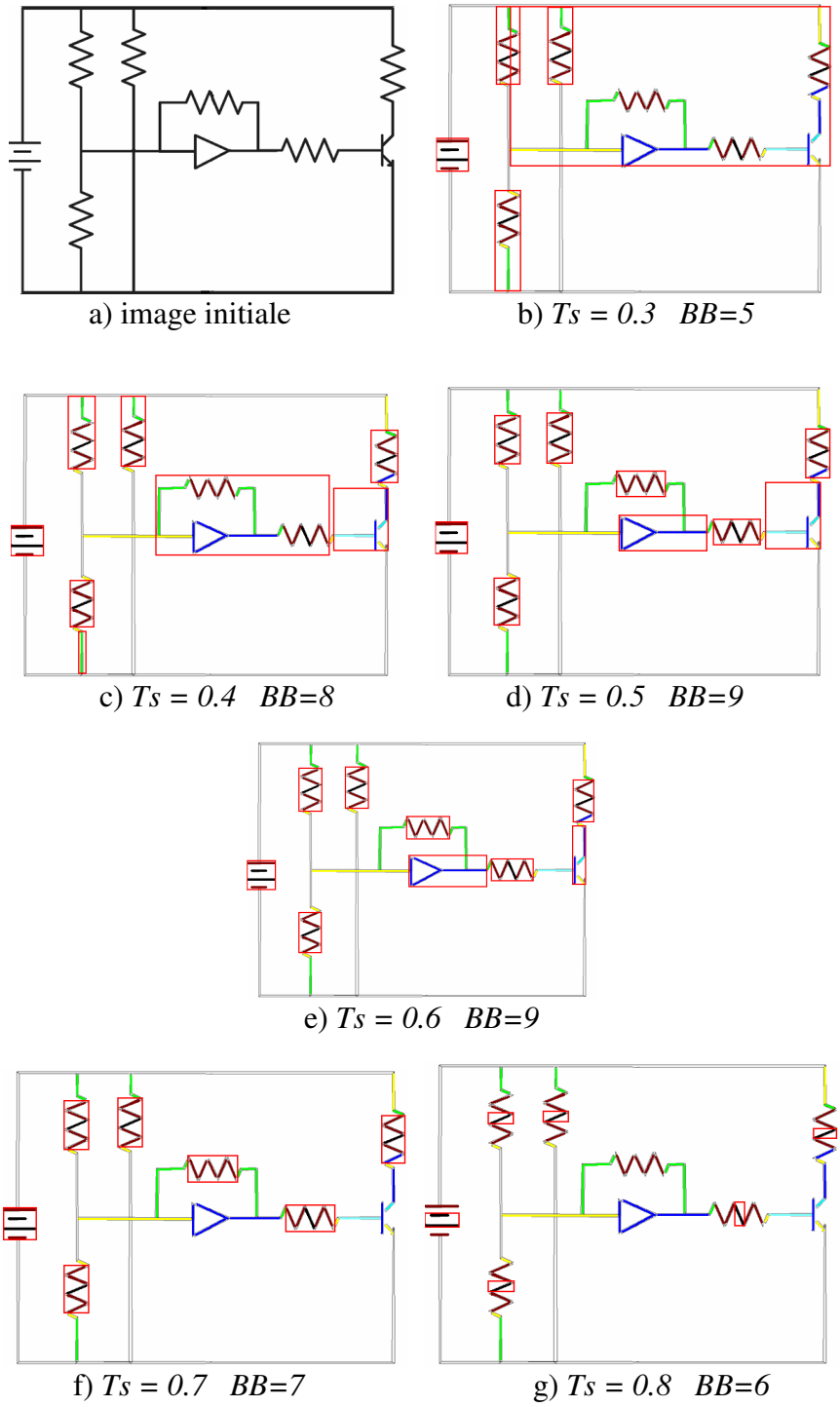


Fig. 3.38 : Influence du seuil (*Ts*) sur la localisation de symboles (Bounding Box) dans une image.

3.5.5 Validation ou rejet de la localisation par reconnaissance du symbole

Lorsque l'ensemble des symboles à localiser est connu a priori, il est possible de soumettre chaque sous-graphe considéré comme symbole potentiel à un modèle de comparaison de graphe afin de valider ou rejeter la zone candidate en fonction d'un critère de similarité entre le sous-graphe à tester et les graphes des symboles modèles.

Par conséquent, nous avons employé notre méthode [Qureshi, 2006a] de mise en correspondance de graphes. Cet algorithme, décrit dans le chapitre suivant, produit des scores de similarités. L'algorithme est tolérant aux erreurs et fonctionne bien en cas de sous ou sur-segmentation des symboles. Les scores de similarité produits peuvent assez facilement être employés (en utilisant un seuil) pour décider automatiquement si le sous-graphe extrait correspond à un symbole ou pas (rejet de la zone). Ainsi, les différentes étapes de la stratégie de localisation proposée sont résumées dans la figure. 3.39.

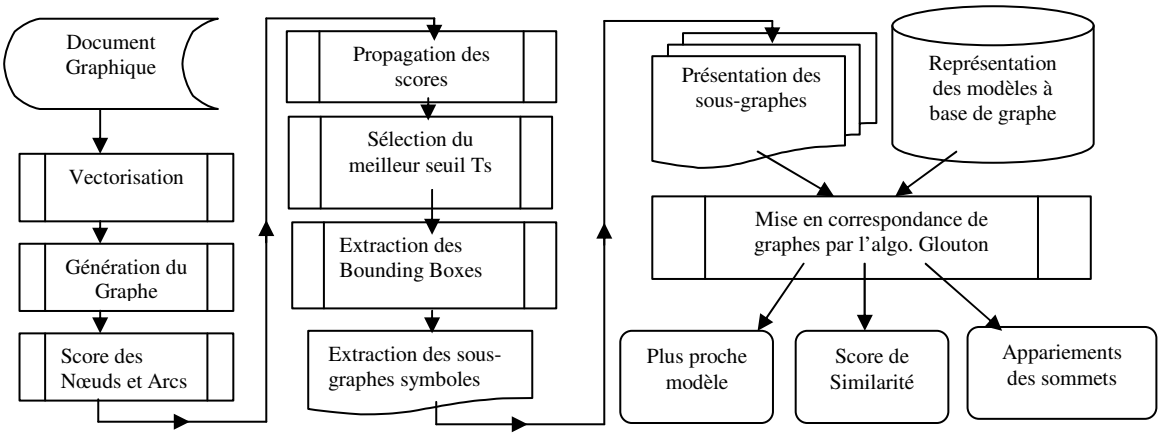


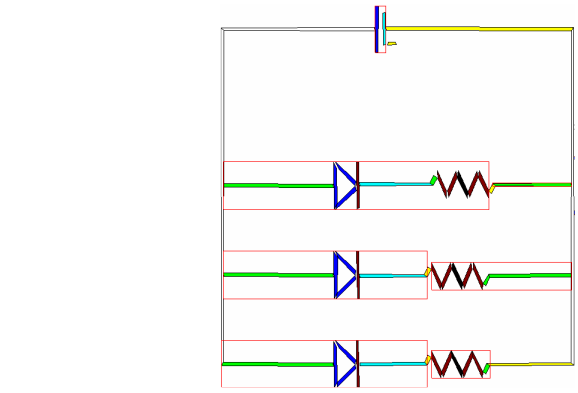
Fig. 3.39 : Architecture du système proposé.

3.5.6 Résultats et bilan

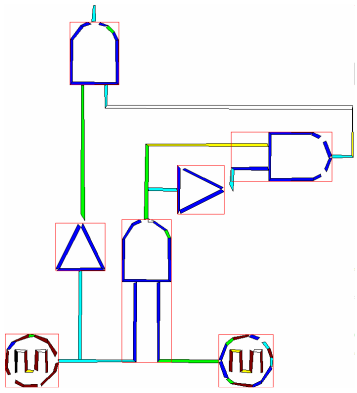
Pour évaluer les performances du système de localisation proposé, nous avons suivi le cadre général utilisant les notions de précision et de rappel présentées dans [Valveny, 2007]. (Voir section 2.9)

Rappelons qu'un système de localisation qui surestime le nombre de résultats est pénalisé par un petit score de précision et qu'un système qui sous-estime le nombre de résultats est pénalisé par un petit score de rappel.

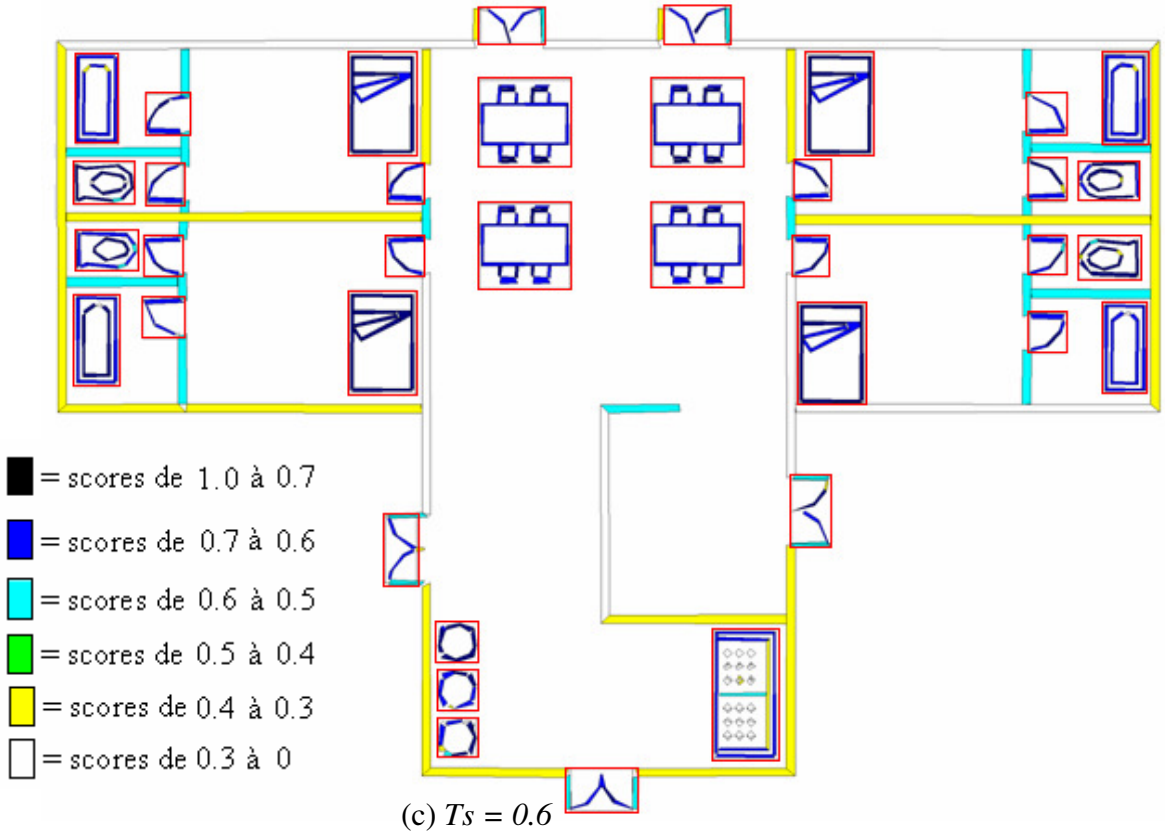
Les évaluations de notre système ont été effectuées sur trois types des documents graphiques : des circuits électroniques, des diagrammes de logique et de cartes architecturales. Le taux de localisation (sans considérer l'étape de reconnaissance) dépend du seuil *Ts* utilisé pour sélectionner certains éléments du graphe (arcs et nœuds). La précision, le taux de rappel et le score global trouvés avec différentes valeurs de seuil sont donnés dans la figure 3.42.



(a) *Ts = 0.4*



(b) *Ts = 0.6*



(c) *Ts = 0.6*

Fig. 3.40 : Symboles localisés sur trois types des documents graphiques.

Dans nos évaluations, nous avons seulement considéré une boite englobante « *bounding boxe* » comme correcte s'il y a un seul symbole complètement inclue dans cette boîte. Les cas de sur-segmentation (*split*) (fig. 3.41a) et de sous-segmentation (*merge*) (figure. 3.41b) sont comptés comme des erreurs.

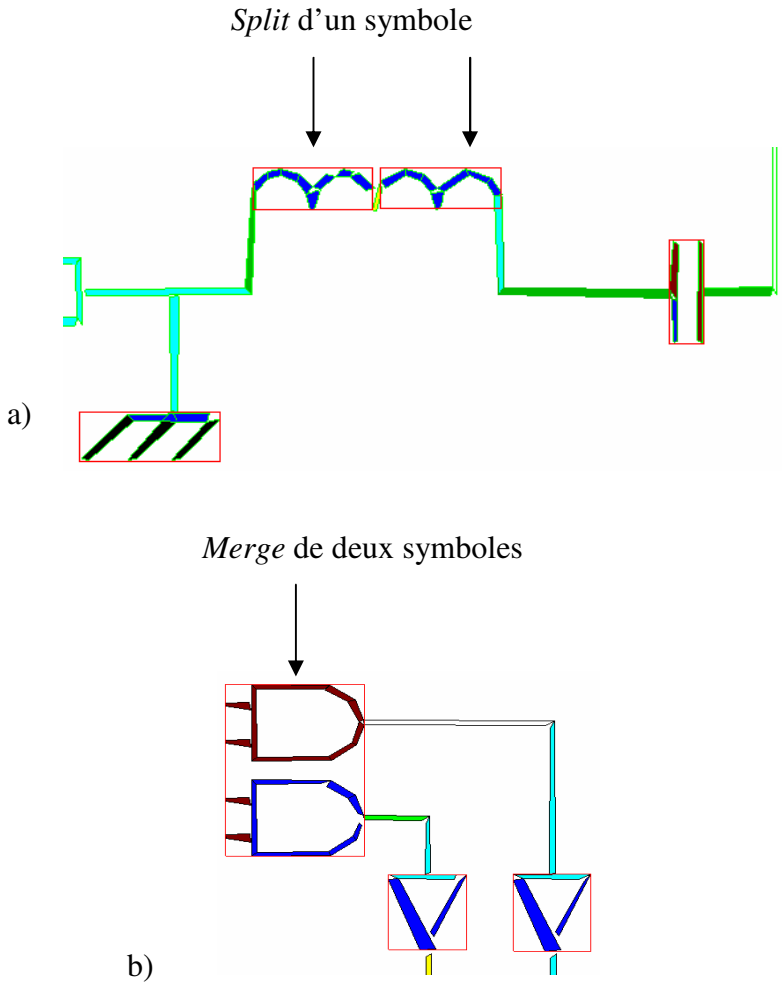
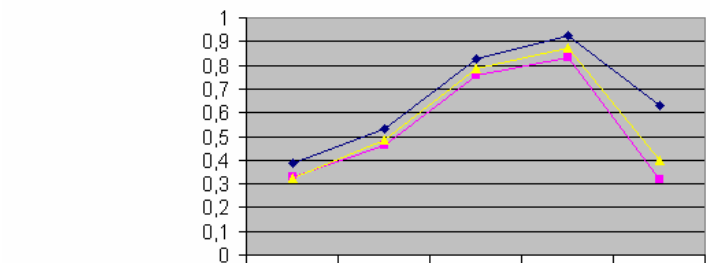


Fig. 3.41 : Exemples de symboles mal localisés (avec *split* et *merge*).

Le taux de localisation correcte est meilleur dans le cas des circuits électroniques et dans le cas des diagrammes de logique dans lesquels les symboles sont faciles à segmenter. Cependant, les taux de bonnes localisations diminuent pour les plans architecturaux contenant des symboles connectés aux lignes représentant les murs. Dans ce cas, il est préférable de choisir une petite valeur de seuil pour la sélection des nœuds du graphe afin de ne pas trop louper de symboles. Ensuite, la reconnaissance peut être utilisée pour valider ou rejeter les « *bounding boxes* » trouvées.

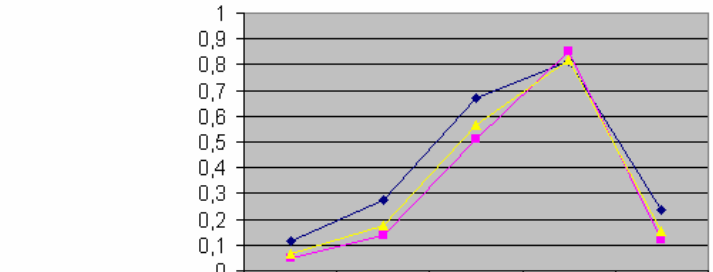
3.6 Conclusion

Nos conclusions de cette étude sur notre proposition est que le développement d'une méthode flexible fonctionnant dans un cadre général pour la localisation de symboles est toujours un challenge en raison des grandes variations dans les éléments de base constituant les documents graphiques. L'approche proposée a montré de bons résultats sur les trois différents types de documents graphiques testés jusqu'ici (sans aucune modification des paramètres utilisés par le système). La méthode est rapide, et est capable de localiser simplement et efficacement des zones d'intérêt dans des images de documents graphiques en une seule passe grâce à l'utilisation de 6 heuristiques indépendantes du domaine d'application. La mise en place d'un mécanisme d'apprentissage et classification des éléments constituent les graphes en remplacement des heuristiques est envisageable de même que l'utilisation d'un mécanisme de rejet grâce au couplage d'un système de reconnaissance de symboles. Ceci n'est possible que lorsque le module de comparaison de graphes est tolérant aux erreurs même en cas de sous ou de sur-segmentation des symboles.



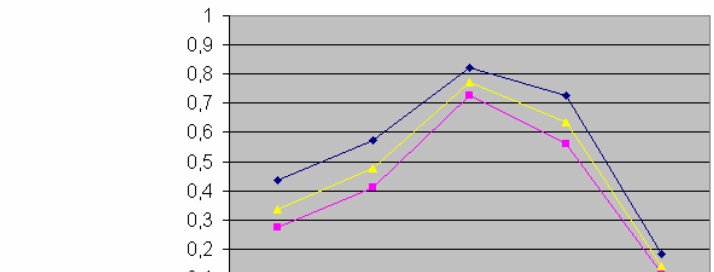
	0,3	0,4	0,5	0,6	0,7
Précision(Moyenne)	0,384524	0,534578	0,826766	0,926964	0,629167
Rapel(Moyenne)	0,331349	0,461647	0,754901	0,831151	0,316964
Score(Moyenne)	0,324281	0,48526	0,783361	0,872918	0,398561

a) Circuits électroniques



	0,3	0,4	0,5	0,6	0,7
Précision(Moyenne)	0,116667	0,2725	0,668095	0,81358	0,234286
Rapel(Moyenne)	0,046786	0,135238	0,51221	0,85071	0,122421
Score(Moyenne)	0,066753	0,177257	0,56701	0,819414	0,156145

b) Diagrammes de logique



	0,3	0,4	0,5	0,6	0,7
Précision (Moyenne)	0,435251	0,571437	0,822454	0,726542	0,182342
Rapel(Moyenne)	0,275816	0,412587	0,725512	0,562141	0,115245
Score(Moyenne)	0,337659	0,479191	0,770948	0,633855	0,141229

c) Plans architecturaux

Fig. 3.42 : Résultats du module de localisation de symboles sur les bases (BD_1, BD_2, BD_3 – figure 3.31, 3.32, 3.33 respectivement).

Chapitre 4

De l'appariement de graphes symboliques à l'appariement de graphes numériques

4.1 Introduction

Ce chapitre décrit les différentes approches de mise en correspondance inexacte de graphes que nous avons utilisées pour résoudre le problème de reconnaissance de formes complexes et symboles graphiques en contexte. Normalement, la mise en correspondance de graphe est considérée inexacte quand le nombre de sommets et le nombre d'arcs dans les deux graphes ne sont pas les mêmes. Cependant, dans le cas de graphe relationnel attribué, la présence d'un même nombre de sommets et d'arcs ne donne pas forcément une indication de l'existence d'isomorphisme entre les deux graphes. Il est, dans ce cas, nécessaire de verifier la similarité entre les attributs des sommets et des arcs. Le terme "inexacte" est alors utilisé pour exprimer la mise en correspondance de graphes faite en associant des éléments des graphes qui sont proches en terme d'attributs mais pas nécessairement exactement les mêmes. Quand les attributs sont symboliques, la fonction de similarité est la plupart du temps une fonction binaire qui donne une réponse oui/non. Nous verrons que lorsque les attributs sont numeriques une fonction de similarité utilisant les valeurs numériques des attributs peut souvent être définie assez simplement.

Dans notre cas (documents graphiques), le graphe de données est souvent bruité, l'appariement recherché ne doit donc pas être forcément complet : un nœud d'un graphe n'est pas forcément associé à un nœud du 2^{e} graphe mais peut être associé à aucun nœud ou au contraire à plusieurs nœuds. De plus, les attributs sur les sommets et les arcs sont numériques et/ou symboliques, ainsi trouver l'appariement de l'ensemble des sommets dans un graphe à l'ensemble des sommets dans l'autre graphe revient à définir une certaine mesure de similarité globale (qui dépendra autant de la structure d'adjacence entre sommets que des attributs des deux graphes) qui devra être maximisée.

Nous avons fait l'hypothèse que ce type d'approches serait efficace pour résoudre le problème de reconnaissance de symboles graphiques en contexte. Ce chapitre présente donc trois nouvelles stratégies que nous proposons pour de comparer des graphes attribués extraits de documents graphiques.

4.2 Rappel des informations disponibles dans le graphe

Nous avons choisi de travailler avec une approche structurelle utilisant principalement un graphe relationnel attribué. Comme nous l'avons vu dans le chapitre précédent, dans l'approche proposée, les symboles sont d'abord décomposés en primitives structurelles et un graphe attribué est alors généré pour décrire chaque image(document complet ou symbole). Les nœuds du graphe représentent les primitives structurelles (vecteurs ou quadrilatères) tandis que les arcs décrivent les relations topologiques entre les primitives. Dans notre représentation, chaque image et symbole est représenté par un graphe étiqueté dont les nœuds et les arcs possèdent plusieurs attributs décrits figure 4.1 et figure 4.2. Nous disposons donc d'informations très précises et de nature très différente pour décrire les primitives et leurs relations. Certains attributs sont numériques comme les longueurs, les angles ou les épaisseurs et il sera possible d'effectuer des calculs de distance, de moyennes, de différences sur ces attributs. Pour les attributs symboliques tels que le type de formes ou de relations, il sera plus difficile de mettre en place des mesures comparatives autres que booléennes.

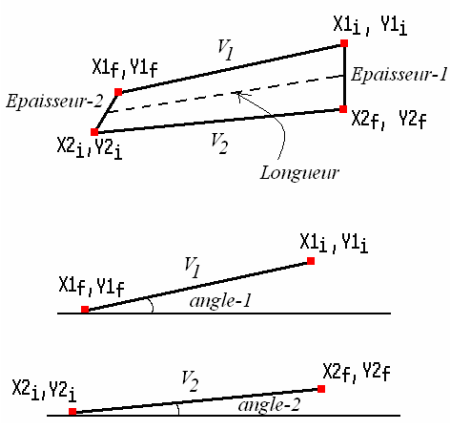
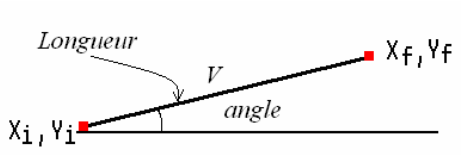
<table>
<tr><th>Information sur les Quadrilatères</th><th>Information sur les Vecteurs</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td><node id="node0">
<attr name="forme"><string>Quad</string></attr>
<attr name="x1i"><string>232</string></attr>
<attr name="y1i"><string>497</string></attr>
<attr name="x1f"><string>231</string></attr>
<attr name="y1f"><string>417</string></attr>
<attr name="x2i"><string>227</string></attr>
<attr name="y2i"><string>418</string></attr>
<attr name="x2f"><string>229</string></attr>
<attr name="y2f"><string>498</string></attr>
<attr name="angle1"><string>29</string></attr>
<attr name="angle2"><string>25</string></attr>
<attr name="Epaisseur1"><string>8</string></attr>
<attr name="Epaisseur2"><string>7</string></attr>
<attr name="Longueur"><string>81</string></attr>
<attr name="score"><string>0.703</string></attr>
</node></td><td><node id="node0">
<attr name="forme"><string>Vecteur</string></attr>
<attr name="xi"><string>232</string></attr>
<attr name="yi"><string>497</string></attr>
<attr name="xf"><string>231</string></attr>
<attr name="yf"><string>417</string></attr>
<attr name="angle"><string>30</string></attr>
<attr name=" Longueur "><string>81</string></attr>
<attr name="score"><string>0.703</string></attr>
</node></td></tr>
</table>

Fig. 4.1: Attributs associés aux primitives Quad et Vecteur.

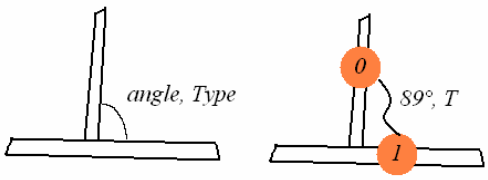
<table>
<tr><th>Information sur les relations entre primitives</th></tr>
<tr><td><edge id="edge1" from="node0" to="node1">
<attr name="angle"><string>89</string></attr>
<attr name="type"><string>T</string></attr>
<attr name="score"><string>0.409</string></attr>
</edge></td><td> 
</td></tr>
</table>

Fig. 4.2: Attributs associés aux arcs du graphe.

Les attributs scores ne décrivent pas les primitives ou relations entre primitives. Ils ne sont utilisés que pendant la phase de localisation des symboles (spotting).

4.3 Méthode de comparaison de graphes choisie

Nous avons déjà indiqué que notre choix s'était porté vers le développement d'une méthode de comparaison de graphes permettant la mise en place d'appariements inexacts entre graphes. Pour cela, nous avons choisi d'adapter la méthode proposée par Champin [Champin, 2003](présentée section 1.6.2) à la reconnaissance de formes complexes (symboles graphiques).

Comme expliqué en section 1.6.2., cette méthode propose d'évaluer la similarité de deux graphes en tenant compte des étiquettes associées aux éléments des graphes, i.e., la similarité de deux sommets ou arcs doit être fonction des étiquettes qu'ils partagent. De plus, la similarité entre les deux graphes est évaluée par rapport à un appariement donné et est quantifiée en comparant les étiquettes qu'ils ont en commun par rapport au nombre total d'étiquettes.

De plus, pour introduire une certaine "souplesse" dans la mesure de similarité, pour être tolérante aux différences dans les graphes comparés, un sommet d'un graphe doit pouvoir être associé à aucun, à un, ou à un ensemble de sommets de l'autre graphe.

4.3.1 Mesure de similarité

La similarité est une mesure qui reflète la ressemblance entre deux objets ou deux caractéristiques. Cette quantité est habituellement incluse dans l'intervalle [-1 ; +1] ou est normalisée dans [0 ; 1]. Pour mesurer la similarité entre nos graphes étiquetés représentant des symboles graphiques, des informations de différentes natures doivent être comparées et combinées. La méthode de Champin ne fonctionne qu'avec des attributs symboliques sur les arcs et les nœuds. Elle ne pourra donc être utilisée que dans des cas très particuliers.

4.3.2 Mise en correspondance entre graphes

Pour déterminer le meilleur appariement, la manière la plus simple de procéder est de tester toutes les mises en correspondance possibles pour les deux graphes et de garder celle qui produit le meilleur score de similarité. Comme l'espace de

recherche s'accroit de manière exponentielle quand la taille des graphes augmente, la demande en ressources de calcul devient rapidement excessive. C'est pourquoi des algorithmes d'une complexité moindre comme les procédures de séparation et évaluation ou les algorithmes gloutons sont souvent utilisés pour éviter les recherches exhaustives, même avec le risque de trouver une solution sous-optimale. Dans notre système, pour trouver le meilleur appariement entre les deux graphes, nous utilisons un algorithme qui s'inspire de l'algorithme glouton proposé par Champin [Champin, 2003] et présenté en section 1.6.2. SimGraph est le nom de la routine que nous avons mise en place pour trouver le meilleur appariement entre deux graphes. Elle prend deux graphes en entrée et retourne la meilleure mise en correspondance (M_{best}) entre les sommets des deux graphes. SimGraph a été utilisé pour tous les tests présentés dans les sections qui suivent.

4.4 Appariement de graphes symboliques

4.4.1 Introduction

Nous avons tous d'abord choisi d'appliquer la méthode proposée par Champin sans modification sur la problématique qui nous intéresse (reconnaissance des symboles graphiques). Il n'est alors possible d'utiliser que les attributs symboliques disponibles dans nos graphes.

4.4.2 Choix des attributs

Dans nos graphes, un seul attribut symbolique est disponible pour les nœuds (figure 4.3). Il s'agit de l'attribut *forme* qui permet de distinguer les primitives *Vect* des primitives *Quad*. Nous utiliserons cet attribut bien que les bases de symboles graphiques disponibles actuellement ne comportent jamais de symboles mixtes (forme pleine + forme mixte). Cet attribut ne sera donc pas discriminant et la mise en correspondance dépendra de la structure des graphes plutôt que des attributs associés aux nœuds puisque tous les sommets ont la même étiquette *Quad*. Les arcs, quant à eux, peuvent avoir des étiquettes différentes traduisant le type de relation topologique liant les deux quadrilatères comme "*X*", "*T*", "*L*", "*P*" ou "*S*". Ainsi, l'ensemble des étiquettes possibles pour les nœuds et les arcs sont :

$$L_v = \{quad\},\ L_E = \{X, T, L, S, P\}$$

Ainsi, les descriptions des deux graphes donnés dans la figure 4.3, sont :

descrip(G_1) = $r_{V1} \cup r_{E1}$
$G_1 = \langle V_1 = \{a, b, c, d, e, f\}$
$r_{v1} = \{(a, quad), (b, quad), (c, quad), (d, quad), (e, quad) (f, quad)\}$
$r_{E1} = \{(a, b, L), (b, c, L), (c, d, L), (d, e, L), (e, a, L), (f, a, T), (f, d, T), (f, e, P)\} \rangle$

descrip(G_2) = $r_{V2} \cup r_{E2}$
$G_2 = \langle V_2 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$rv_2 = \{(1, quad), (2, quad), (3, quad), (4, quad), (5, quad), (6, quad), (7, quad),$
$(8, quad)\}$
$r_{E2} = \{(1, 2, L), (1, 5, L), (1, 6, P), (2, 3, L), (3, 4, L), (4, 8, P), (4, 5, L), (5, 6, T),$
$(5, 8, T), (6, 7, L), (7, 8, L)\} \rangle$

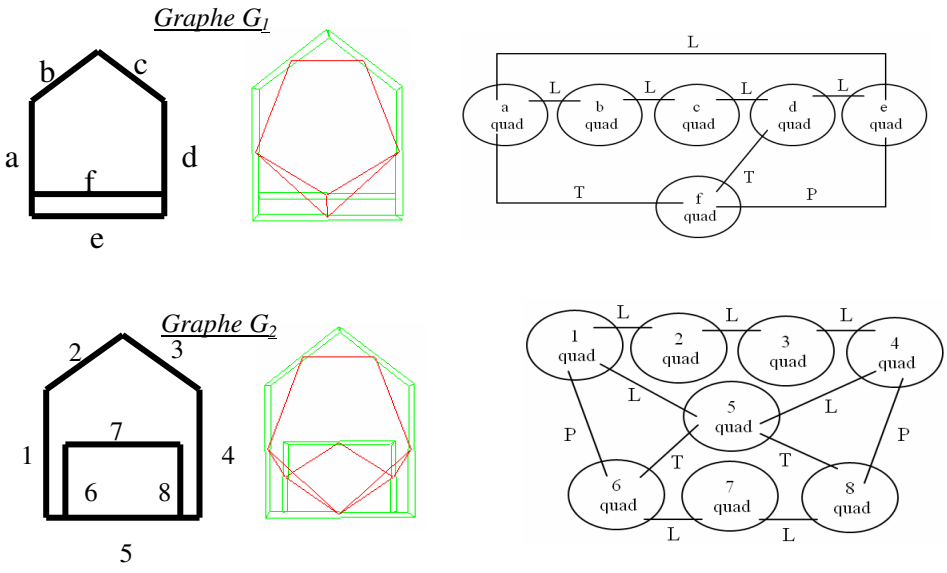


Fig. 4.3: Représentations de deux symboles et graphes correspondants

Pour les deux graphes étiquetés G_1 et G_2 donnés dans la figure 4.3, voici parmi d'autres, deux appariements possibles :

$$Mp_1 = \{(a, 6), (b, 7), (c, 8), (d, 8), (e, 5)\}$$

$$Mp_2 = \{(a, 1), (b, 2), (c, 3), (d, 4), (e, 7)\}$$

Dans le premier appariement Mp_1 *le* sommet "*8*" de G_2 a été associé à deux sommets ("*c*", "*d*") du graphe G_1 illustrant la possibilité d'appariements multiples (split) entre sommets. De plus, il y a certains sommets qui n'ont pas été considérés dans l'appariement proposé, par exemple, le sommet "*f*" de G_1 illustrant la possibilité de laisser un sommet sans correspondant dans l'autre graphe.

$$M_{P1}(8) = \{c, d\},\ M_{P1}(f) = \phi$$

Les caractéristiques communes descrip(G_1) $\cap^{Mp}$ descrip(G_2) aux deux graphes G_1 et G_2 sont alors définies par rapport à un appariement Mp de la façon suivante :

$$\begin{aligned}\text{descrip}(G_1) \cap^{Mp} \text{descrip}(G_2) = &\{(v, l) \in r_{V1} \mid \exists\, v' \in Mp(v),\ (v', l) \in r_{V2}\} \\ &\cup \{(v, l) \in r_{V2} \mid \exists\, v' \in Mp(v),\ (v', l) \in r_{V1}\} \\ &\cup \{(v_i, v_j, l) \in r_{E1} \mid \exists\, v'_i \in Mp(v_i), \exists\, v'_j \in Mp(v_j)\ (v'_i, v'_j, l) \in r_{E2}\} \\ &\cup \{(v_i, v_j, l) \in r_{E2} \mid \exists\, v'_i \in Mp(v_i), \exists\, v'_j \in Mp(v_j)\ (v'_i, v'_j, l) \in r_{E1}\}\end{aligned}$$

Ainsi, descrip(G_1) $\cap^{Mp_1}$ descrip(G_2) =
{(*a*, *quad*), (*b*, *quad*), (*c*, *quad*), (*d*, *quad*), (*e*, *quad*), (*5*, *quad*), (*6*, *quad*), (*7*, *quad*), (*8*, *quad*), (*a*, *b*, *L*), (*b*, *c*, *L*), (*6*, *7*, *L*), (*7*, *8*, *L*)}

descrip(G_1) $\cap^{Mp_2}$ descrip(G_2) =
{(*a*, *quad*), (*b*, *quad*), (*c*, *quad*), (*d*, *quad*), (*e*, *quad*), (*1*, *quad*), (*2*, *quad*), (*3*, *quad*), (*4*, *quad*), (*7*, *quad*), (*a*, *b*, L), (*b*, *c*, L), (*c*, *d*, L), (*1*, *2*, *L*), (*2*, *3*, *L*), (*3*, *4*, *L*)}

Nous utilisons ensuite la mesure de similarité précisée par Champin en considérant les fonctions f et g comme des fonctions de cardinalité. $S_p(Mp)$ renvoie le nombre d'associations multiples (split) ou de non-association pour les sommets. Les scores pour les deux appariements Mp_1 et Mp_2 sont décrits ci-dessous :

$$\text{Sim}_{Mp_1}(G_1, G_2) = \frac{f(\text{descrip}(G_1) \cap^{Mp_1} \text{descrip}(G_2)) - g(Sp(Mp))}{f(\text{descrip}(G_1) \cup \text{descrip}(G_2))} = \frac{13-1}{14+19} = \frac{12}{33} = 0.36$$

$$\mathrm{Sim}_{Mp_2}(\mathrm{G}_1,\mathrm{G}_2)=\frac{f(\mathrm{decrip}(\mathrm{G}_1)\cap^{Mp_2}\mathrm{decrip}(\mathrm{G}_2))-g(\mathrm{S}p(Mp))}{f(\mathrm{decrip}(\mathrm{G}_1)\cup\mathrm{decrip}(\mathrm{G}_2))}=\frac{16-0}{14+19}=\frac{16}{33}=0.48$$

On voit donc que l'appariement Mp_2 est meilleur que l'appariement Mp_1 selon cette mesure de similarité. L'algorithme teste les appariements et vérifie leur score de similarité afin d'obtenir le meilleur appariement entre les sommets des deux graphes sur la base du meilleur score produit.

4.4.3 Résulats et conclusion

Pour tester l'efficacité du couplage de cette représentation symbolique et de cette mesure de similarité entre graphes, nous avons utilisé la base de symboles de GREC-2003. Nous avons réalisé une expérimentation avec les symboles non dégradés (test-idéal). Ce cas est le plus simple et pourtant les taux de reconnaissance obtenus sont faibles (diminuant jusqu'à 80% figure 4.5) pour des bases comprenant de 5 à 50 symboles. Les scores de similarité sont très proches les uns des autres et donc peu discriminants. Les scores de similarité de 10 symboles non dégradés architecturaux sont présentés en tableau 4.1. Nous avons noté que le gros inconvénient de cette méthode vient du fait qu'un symbole peut être confondu avec un autre symbole parce qu'il possède le même nombre de nœuds et d'arcs, avec les mêmes étiquettes qu'un autre symbole bien qu'ils aient des formes différentes. Par exemple, les deux symboles de la figure 4.4 partagent le même graphe.

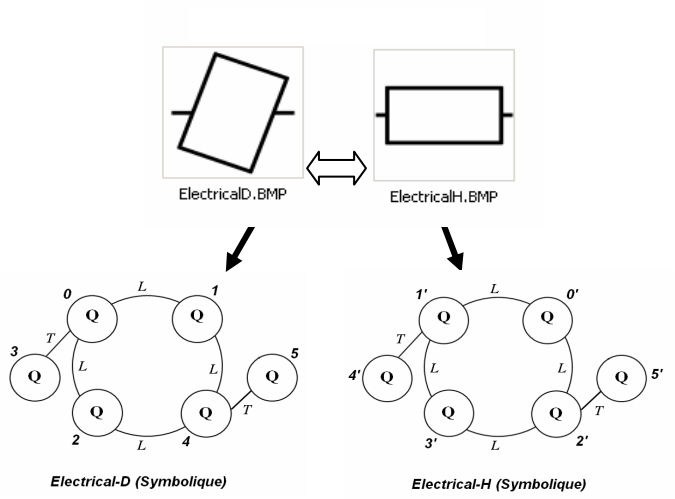
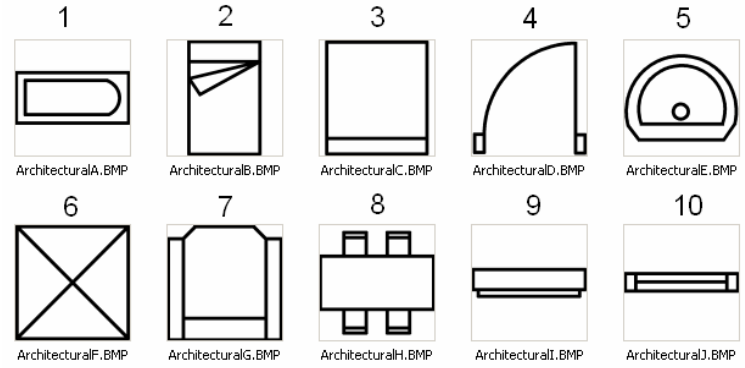


Fig. 4.4: Les deux symboles partagent le même graphe symbolique.

Tab. 4.1: Scores de similarité obtenus par mise en correspondance des graphes avec des attributs symboliques.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**1,00**	0,96	0,95	0,97	0,99	0,88	0,93	0,93	0,92	0,93
2		**1,00**	0,92	0,97	0,94	0,88	0,96	0,97	0,96	0,92
3			**1,00**	0,83	0,77	0,91	0,95	0,97	0,97	0,95
4				**1,00**	0,98	0,89	0,81	0,90	0,83	0,88
5					**1,00**	0,68	0,78	0,72	0,76	0,76
6						**1,00**	0,92	0,92	0,93	0,85
7							**1,00**	0,99	0,98	0,98
8								**1,00**	0,99	0,96
9									**1,00**	0,96
10										**1,00**



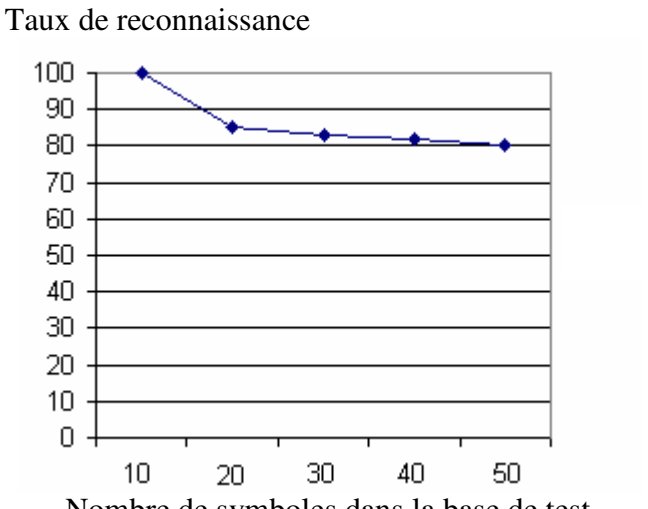


Fig. 4.5 : Taux de reconnaissance sur les symboles idéaux de GREC-2003.

Un autre test utilisant une base d'images de symboles avec 15 classes (modèles) et 300 symboles dégradés par distorsions vectorielles de 3 différents niveaux (voir figure 4.6) a été mené afin d'évaluer les performances de la technique proposée sur des symboles dégradés. Le tableau 4.2 montre les résultats obtenus.

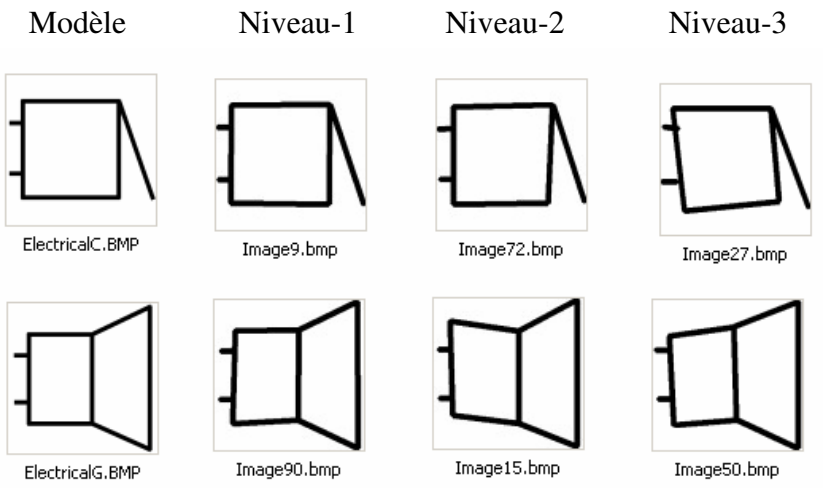


Fig. 4.6 : Symboles dégradés par distorsions vectorielles de 3 différents niveaux.

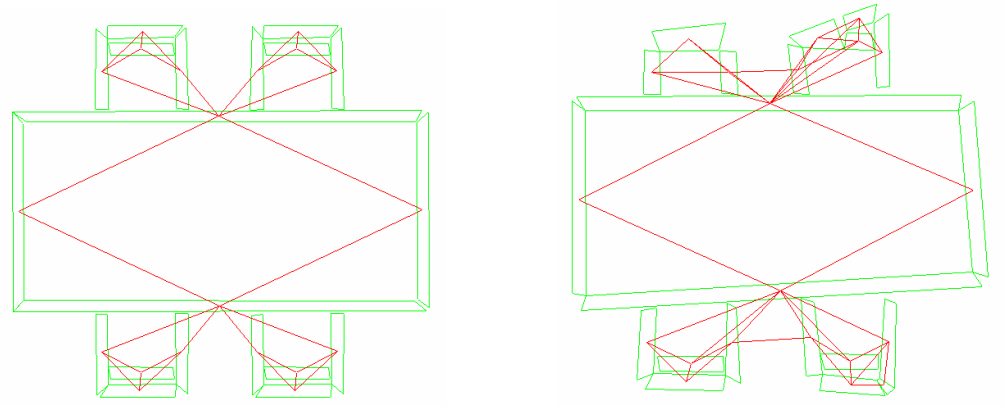
Tab. 4.2: Performances avec utilisation d'attributs symboliques sur des symboles dégradés.

Distorsion	Nombre de classes	Nombre d'images test	Bien reconnus	Taux de reconnaissance
Niveau-1	15	100	33	33%
Niveau-2	15	100	28	28%
Niveau-3	15	100	25	25%

Ces tests prouvent que les attributs symboliques ne sont pas efficaces pour reconnaître des symboles déformés. Bien que les images comportent des traits anormalement gros, dans cette base, provoquant des disparitions de traits (figure 4.7a) cela ne suffit pas à justifier les modifications sur les graphes (figure 4.7b) et n'explique donc pas complètement les très faibles taux de reconnaissance. La mesure de similarité utilisée reste assez simple et les attributs symboliques ne sont pas suffisants. Nous proposons d'ajouter les attributs numériques et d'améliorer la mesure de similarité pour tenir compte de nombreux autres types d'informations

moins sensibles aux distorsions ou transformations afin d'obtenir les scores de similarité beaucoup plus précis.

a) Image parfaite d'une table et sa version dégradée.



b) Impact des distorsions sur le graphe.

Fig. 4.7: Sensibilité des graphes aux distorsions vectorielles.

4.5 Appariement inexact de graphes numériques

4.5.1 Introduction

Les résultats précédents montrant qu'il est insuffisant de se contenter d'utiliser des attributs symboliques lors de la comparaison de graphes par mesure de similarité. Nous avons donc choisi d'apporter des modifications à la méthode précédente afin de permettre l'utilisation d'attributs numériques aussi bien que symboliques lors de comparaison de deux graphes. L'objectif est de rendre la méthode plus souple, plus robuste et de trouver le meilleur appariement possible entre les graphes.

4.5.2 Adaptation de la mesure de similarité

Nous avons désiré produire un système générique et robuste de reconnaissance de symboles graphiques et formes complexes représentés par des graphes multi-attributs numériques et symboliques. Pour cela, la formulation d'une nouvelle mesure de similarité entre représentations sous forme de graphes étiquetés est proposée. Soit A_v et A_E respectivement les ensembles des étiquettes des sommets et des arcs d'un graphe. On peut alors caractériser un graphe multi-étiqueté G par un 4 - tuple $G = (V, E, \alpha, \beta)$ où : V est l'ensemble fini des sommets, $E \subseteq V \times V$ est l'ensemble des arcs, $\alpha : V \to A_V^i$ est une fonction qui assigne les étiquettes aux sommets, $\beta : E \to A_E^j$ est une fonction qui assigne les étiquettes aux arcs. Les étiquettes peuvent correspondre aussi bien à des attributs numériques que symboliques ; ici, i varie de 1 à δ et j varie de 1 à Ω, où δ et Ω représentent respectivement le nombre d'étiquettes associées aux sommets et aux arcs.

Pour comparer deux graphes avec des attributs numériques et symboliques sur les sommets et sur les arcs, nous proposons une mesure basée sur un calcul de distance entre les attributs associés aux sommets et un calcul de distance entre les attributs associés aux arcs. Ainsi, nous proposons de calculer la similarité entre deux graphes attribués pour une mise en correspondance Mp donnée à l'aide de la formule suivante :

$$Sc_{Mp} = \left[\sum_{i=1}^{m} (1 - \Delta V_i) + \sum_{j=1}^{n} (1 - \Delta E_j) - \left(\sum_{i=1}^{k} \omega_i + \sum_{j=1}^{\ell} \omega'_j \right) \right] \quad (1)$$

où m est le nombre total de sommets appariés dans M_p et n le nombre total d'arcs entre ces nœuds. ω_i et ω'_j sont des pénalités choisies qui sont appliquées lorsqu'un nœud i ou un arc j est associé à plusieurs autres ou n'est pas associé dans l'appariement courant. Grâce à ce mécanisme de pénalité, nous autorisons l'association d'un sommet (ou d'un arc) dans un graphe à zéro, un ou plusieurs sommets (ou arcs) dans l'autre graphe. Ces pénalités doivent dépendre du nombre d'attributs utilisés lors des calculs de distance entre sommets (et entre arcs).

$$\Delta V_i = \frac{\sum_{k=1}^{\delta} f_k\left(A_V^k, A_{V'}^k\right)}{\delta} \quad (2) \qquad \text{et} \qquad \Delta E_j = \frac{\sum_{k=1}^{\Omega} g_k\left(A_E^k, A_{E'}^k\right)}{\Omega} \quad (3)$$

Dans la formule (1), ΔV_i correspond à la distance entre deux sommets appariés, normalisée entre 0 et 1 (formule(2)).

La fonction f_k compare la valeur du $k^{ième}$ attribut des deux sommets appariés et retourne une valeur de dissimilarité entre 0 et 1.

De même, ΔE_j correspond à la distance entre deux arcs normalisée entre 0 et 1 (formule(3)). La fonction g_k est utilisée pour comparer les deux valeurs du $k^{ième}$ attribut des deux arcs, elle retourne une dissimilarité entre 0 et 1.

Il est simple d'introduire de nouveaux attributs dans le calcul de similarité entre deux sommets ou deux arcs. Il suffit pour cela de définir les fonctions f_k ou g_k à utiliser lors de la comparaison. Enfin, pour normaliser la mesure de similarité des graphes entre 0 et 1, on utilise :

$$Sim(G, G') = \frac{2 \times Sc_{Mp}}{\left[\left(C(V) + C(V')\right) + \left(C(E) + C(E')\right)\right]}$$

où C est une fonction de cardinalité qui retourne le nombre de sommets ou d'arcs dans un graphe et Sc_{Mp} est le score pour un appariement Mp.

4.5.3 Choix des attributs

Afin de démontrer l'intérêt d'utiliser des attributs numériques en plus des attributs symboliques, nous allons montrer que les angles et les longueurs des quadrilatères peuvent être rajoutés pour lever de nombreuses ambiguïtés entre formes. Grâce à la nouvelle formule de similarité, il devient possible, lors de la phase de reconnaissance, d'utiliser un attribut numérique parmi tous ceux disponibles pour caractériser les primitives. Nous avons choisi d'utiliser, dans un premier temps, l'attribut longueur associé aux primitives de description (nœuds dans le graphe) lors du calcul de notre mesure de similarité. Afin d'obtenir l'invariance au changement d'échelle, la longueur absolue est remplacée par la longueur relative. La longueur relative λ_i de la $i^{ème}$ primitive d'un symbole peut être calculée comme le rapport entre sa longueur absolue l_i et la longueur l_{max} de la plus longue primitive extraite du sous-graphe représentant le symbole, c'est-à-dire :

$\lambda_i = l_i / l_{max}$

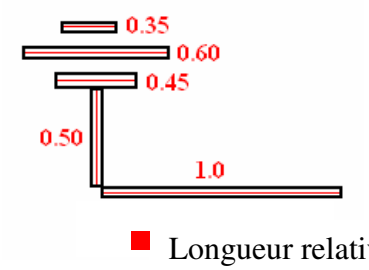


Fig. 4.8: Longueur relative des primitives pour un sous-graphe.

L'usage de la longueur relative à la place de la longueur exprimée en pixels est obtenu en modifiant l'attribut longueur des primitives avant de lancer le calcul de similarité. Cette procédure est utilisée aussi bien pour les Vecteurs que pour les Quadrilatères. Ensuite, pour la comparaison des attributs, on utilise simplement la formule (4).

$$f_1\left(A_V^1 , A_{V'}^1\right) = \left|\lambda_i - \lambda_{i'}\right| \qquad (4)$$

De même, chaque arc posséde un attribut numérique correspondant à l'angle relatif (φ_{ij} [0, 90°]) entre les deux primitives mises en relation. Pour deux quadrilatères (Q_i, Q_j) ou deux vecteurs (V_i, V_j) ou un quadrilatère et un vecteur (Q_i, V_j), il est possible de calculer cette valeur numérique. Cette valeur est calculée à partir des angles de leur axe d'inertie. Si θ_i et θ_j sont les angles des axes d'inertie de Q_i et Q_j par rapport à l'axe horizontal (figure 3.29), l'angle relatif entre Q_i et Q_j est exprimé par

$$\varphi_{ij} = \left|\theta_i - \theta_j\right|$$

L'usage de l'angle relatif permet au graphe d'être invariant par rotation. Pour comparer les valeurs prises par cet attribut numérique associé aux arcs du graphe, nous avons défini la fonction g_1 comme indiqué formule (5).

$$g_1\left(A_E^1 , A_{E'}^1\right) = \frac{\left|\varphi_{ij} - \varphi_{i'j'}\right|}{90} \qquad (5)$$

4.5.4 Résulats et discussions

Pour tester nos propositions, nous avons effectué des tests en utilisant à nouveau la base GREC2003 [GREC, 2003]. En travaillant avec 50 symboles différents (Base BD_1/ figure 4.9), nous avons généré un ensemble de 950 exemples de symboles avec différents niveaux de distorsion, des transformations et des ajouts de bruits classiques. En raison de l'usage d'attributs invariants à la rotation et au changement d'échelle, un taux de reconnaissance de 100% a été obtenu dans le cas des images de test ayant subi uniquement des rotations et des changements d'échelle.

La table 1 montre en détail les scores de similarité obtenus en utilisant notre méthode d'appariement de graphes entre des modèles de symboles et des symboles identiques globalement transformés respectivement par rotation et changement d'échelle. Pour des raisons de concision, seuls les scores de quelques prototypes sont fournis dans les tableaux.

Les tableaux 2 et 3 montrent les scores de similarité obtenus à l'aide de notre algorithme d'appariement de graphes entre des modèles de symboles et les symboles déformés localement, respectivement par du bruit et par une distorsion vectorielle (figure 4.6). Nous pouvons voir que le score de similarité n'est pas toujours égal à 1, mais que la plupart du temps le score de similarité maximal est présent sur la diagonale du tableau ce qui montre l'aptitude au rappel de notre système. Un résumé des taux de reconnaissance obtenus est présenté dans le tableau 4.6. Il existe de nombreux facteurs qui peuvent influencer la performance d'une méthode de reconnaissance de symboles. Notre méthode montre un très grand pouvoir de discrimination dans le cas d'images non bruitées avec un taux de reconnaissance de 100% pour un ensemble de 50 modèles de symboles. Sur une telle base, les méthodes des autres participants affichaient un taux de reconnaissance de 84% et 92% (figure 4.21). Ainsi, dans le cas de transformations affines, les résultats de la méthode proposée sont meilleurs que ceux de la majorité des participants à GREC'03.

Par contre, les taux de reconnaissance obtenus par notre méthode sur les images bruitées restent plus faibles. Ce résultat s'explique probablement par le fait que nous n'avons pas utilisé de prétraitement comme les filtrages ou la suppression

des petits quadrilatères qui introduisent des sommets supplémentaires dans les graphes. Le nombre de classes de symboles pour les tests de distorsion vectorielle est plus faible (seulement 15 symboles – Base BD_2/ figure 4.10) parce que la méthode utilisée dans la compétition GREC03 pour produire des distorsions vectorielles fonctionne uniquement avec des symboles composés exclusivement de lignes droites (pas d'arcs de cercle, voir figure 4.10). Notre taux de reconnaissance pour ces symboles déformés (table 4.6), même s'il n'est pas supérieur, est comparable à celui des meilleures méthodes utilisées par les autres participants.

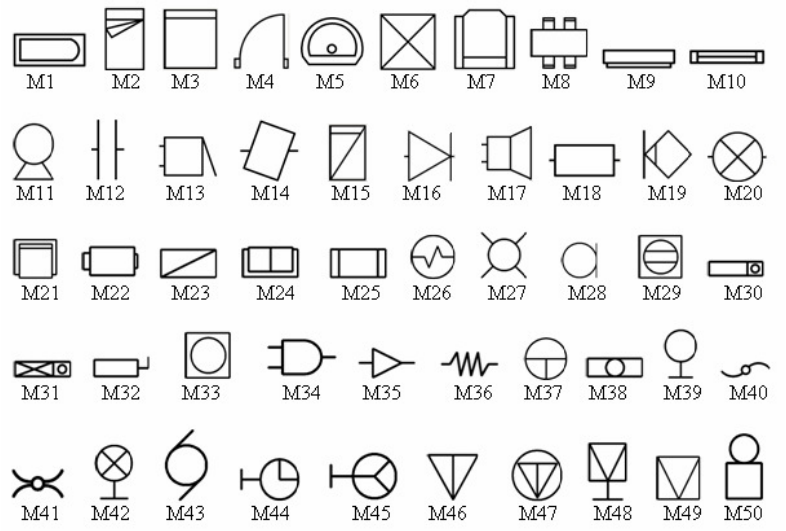


Fig. 4.9 : Première base d'images test comportant 50 modèles (*BD_1*).

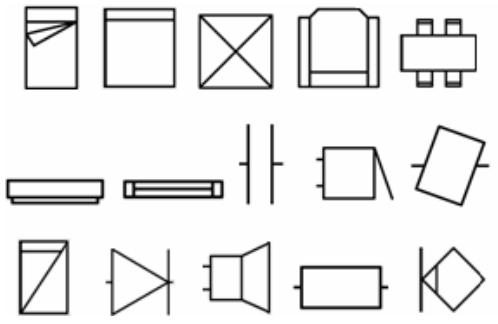


Fig. 4.10: Deuxième base d'images test comportant 15 modèles (*BD_2*).

Tab. 4.3: Scores de similarité obtenus entre symboles modèles M_i et symboles après rotation et changements d'échelle (RS_i).

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	...
RS_1	1.00	0.48	0.51	0.80	0.49	0.42	0.56	0.62	...
RS_2		1.00	0.58	0.83	0.60	0.67	0.53	0.58	...
RS_3			1.00	0.65	0.77	0.45	0.62	0.63	...
RS_4				1.00	0.76	0.52	0.68	0.56	...
RS_5					1.00	0.42	0.61	0.69	...
RS_6						1.00	0.54	0.65	...
RS_7							1.00	0.61	...
RS_8								1.00	...

Tab. 4.4: Scores de similarité entre symboles modèles M_i et symboles bruités N_i.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	...
N_1	0.92	0.51	0.55	0.80	0.54	0.49	0.51	0.64	...
N_2		0.91	0.53	0.80	0.61	0.61	0.54	0.55	...
N_3			0.92	0.62	0.72	0.37	0.63	0.61	...
N_4				0.90	0.73	0.56	0.56	0.58	...
N_5					0.90	0.44	0.67	0.63	...
N_6						0.93	0.59	0.61	...
N_7							0.91	0.59	...
N_8								0.91	...

Tab. 4.5: Scores de similarité entre symboles modèles M_i et symboles déformés D_i

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	...
D_1	0.90	0.68	0.46	0.70	0.64	0.67	0.46	0.65	...
D_2		0.93	0.83	0.78	0.68	0.70	0.68	0.45	...
D_3			0.91	0.72	0.66	0.64	0.64	0.56	...
D_4				0.91	0.63	0.76	0.66	0.75	...
D_5					0.92	0.63	0.69	0.56	...
D_6						0.91	0.71	0.49	...
D_7							0.93	0.60	...
D_8								0.92	...

Tab. 4.6: Résumé des performances de SimGraph.

		Nombre de Modèles	Nombre de tests	Bien reconnus	Taux de reconnaissance
Rotation		50	150	150	100%
Changement d'échelle		50	100	100	100%
Bruit	Niveau-1	50	200	181	90.5%
	Niveau-2	50	200	167	83.5%
	Niveau-3	50	200	159	79.5%
Distorsions		15	100	91	91.0%

Pour examiner la robustesse et la généralité de la méthode proposée, deux autres jeux de données ont été employés contenant des formes autres que des symboles architecturaux ou électroniques (notamment des formes pleines). D'abord, une grande base de données de formes binaires non linéaires (BD_3) rassemblées par le « *LEMS Vision Group* » de Brown Université[1], et ensuite une encyclopédie de « *Western signs and ideograms*[2] » contenant 2500 symboles (BD_4). Les essais ont été lancés sur différents sous-ensembles d'images. Pour la concision, seulement les résultats de quelques prototypes sont montrés dans les tables 4.7 et 4.8. Les petits scores de similarité entre les différentes images démontrent le fort pouvoir discriminant de l'approche proposée. Les tables 4.9 et 4.10 montrent qu'il est possible de mettre en place un système de recherche de symboles à partir d'images requêtes basé sur notre mesure de similarité. Seules les images ayant un taux de similarité supérieur à environ 70% sont affichées.

Tab. 4.7: Exemples de scores de similarité entre différentes formes de la base (BD_3).

							
	1.000	0.535	0.445	0.566	0.517	0.645	0.529
		1.000	0.630	0.563	0.527	0.554	0.630
			1.000	0.492	0.398	0.502	0.579
				1.000	0.652	0.673	0.532
					1.000	0.615	0.522
						1.000	0.605
							1.000

Tab. 4.8: Exemples de score de similarité entre différentes images d'un même group (BD_3).

							
	1.00	0.62	0.75	0.83	0.67	0.79	0.65
		1.00	0.58	0.63	0.72	0.66	0.57
			1.00	0.70	0.64	0.75	0.63
				1.00	0.67	0.74	0.79
					1.00	0.69	0.62
						1.00	0.74
							1.00

[1]http://www.lems.brown.edu/vision/
[2]http://www.symbols.com/

Tab. 4.9: Score de similarité et formes retrouvées à partir d'une forme requête (*BD_3*).

Forme raquette	Nombre d'images dans la base	Rang - 1	Rang - 2	Rang - 3	Rang - 4	Rang - 5
 Q_1	43	1.000  bd-007	0.925  bd-030	0.888  bd-019	0.784  bd-024	0.744  bd-042
 Q_2	100	1.000  bf-001	0.872  bf-003	0.826  bf-026	0.792  bf-009	0.786  bf-011
 Q_3	100	1.000  f-026	0.801  f-075	0.793  f-044	0.774  f-065	0.733  f-046
 Q_4	45	1.000  v-004	0.849  v-018	0.814  V-017	0.754  v-038	0.678  v-044

Tab. 4.10: Score de similarité et formes retrouvées à partir d'une forme requête (*BD_4*).

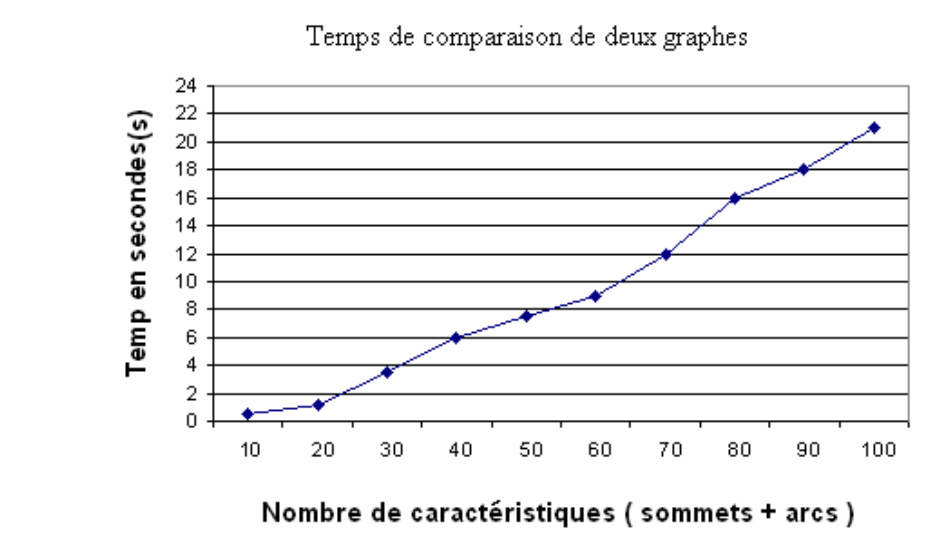
Forme raquette	*Rang*-1	*Rang*-2	*Rang*-3	*Rang*-4	*Rang*-5
 Q_1	1.000 C-1106	0.903  C-1112	0.859 C-1113	0.831  C-1111	0.789  C-1107
 Q_2	1.000 C-0304	0.860  C-0305	0.819 C-0306	0.781 C-0308	0.709 C-0307
 Q_3	1.000 C-0206a	1.000  C-0206b	1.000  C-0206c	1.000 C-0206d	0.534 C-0202a
Q_4	1.000 C-430b	0.909 C-0419	0.641 C-0431b	0.588 C-0431a	0.562 C-0421

Nous avons également voulu tester les performances de notre méthode sur des formes mixtes (avec des parties pleines et des parties linéaires). Pour cela nous avons utilisé la base d'images présentée en Annexe 2. Les scores de similarité obtenus sont encore une fois de bonne qualité. Nous nous permettons, de plus, de rappeler qu'à notre connaissance aucune autre méthode structurelle permettant de traiter des formes mixtes n'a été proposée jusqu'à aujourd'hui.

Tab. 4.11: Score de similarités obtenues sur les formes mixte (BD_4).

										
	1.000	0.674	0.545	0.339	0.510	0.580	0.475	0.380	0.536	0.434
		1.000	0.527	0.300	0.457	0.542	0.491	0.374	0.487	0.422
			1.000	0.262	0.444	0.537	0.566	0.322	0.536	0.386
				1.000	0.423	0.337	0.251	0.499	0.333	0.527
					1.000	0.491	0.338	0.505	0.548	0.635
						1.000	0.426	0.428	0.557	0.426
							1.000	0.355	0.456	0.331
								1.000	0.405	0.552
									1.000	0.506
										1.000

La figure. 4.11 fournit des informations sur les temps de calcul nécessaires à notre méthode pour produire les résultats. Les résultats montrent clairement le lien existant entre le temps de calcul et la taille des graphes. Le temps de calcul augmente avec le nombre de caractéristiques (i.e. le nombre de sommets et le nombre d'arcs dans les deux graphes à comparer). Bien qu'il y ait seulement quelques graphes représentant des symboles graphiques qui aient des tailles importantes (symboles avec beaucoup d'arcs, leur présence influence beaucoup les temps de calcul). Ces résultats nous ont poussé à comparer notre méthode de mise en correspondance de graphes avec une technique beaucoup plus rapide mais aussi beaucoup plus rudimentaire ainsi qu'à chercher des solutions pour optimiser le temps nécessaire à la comparaison des graphes avec SimGraph. Ces études et solutions sont présentées dans les sections suivantes.



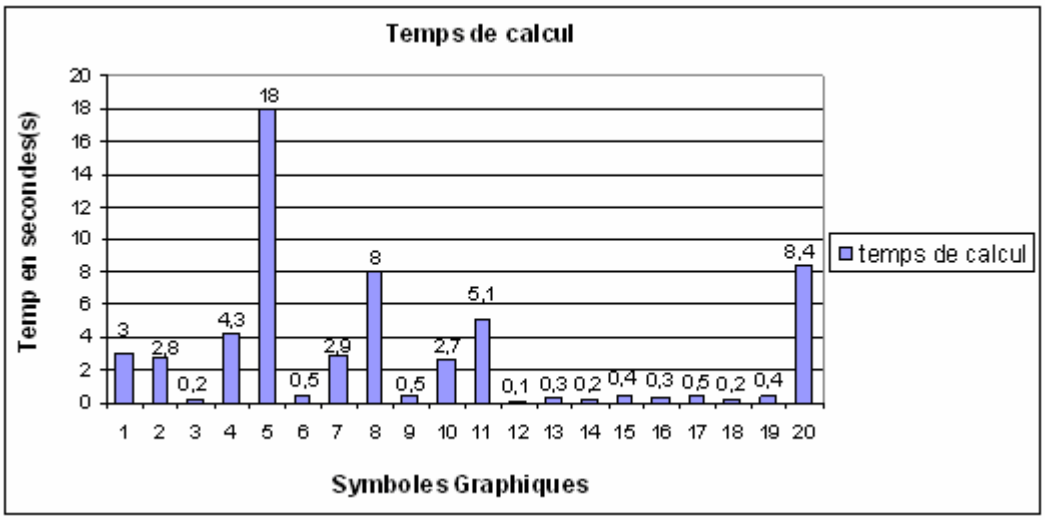


Fig. 4.11: Les temps de calcul nécessaires pour comparer le symbole-1 avec 20 autres symboles de la Base BD_1, le temps dépend fortement dela taille des graphes

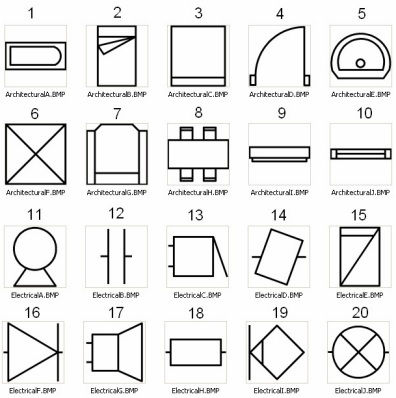


Fig. 4.12 : Les 20 symboles extraits de BD_1.

4.6 Comparaison de SimGraph avec un algorithme de mise en correspondance plus rudimentaire

SimGraph n'étant pas rapide, nous avons cherché à voir s'il était possible de trouver une bonne mise en correspondance de graphes plus rapide. Dans le but d'accélérer le processus de recherche, nous avons développé un algorithme plus simpliste qui réduit l'espace de recherche d'une manière considérable et permet de trouver une solution approximative, rapidement.

Le processus de recherche est itératif et guidé par la mise en correspondance des sommets en premier lieu (similarité entre primitives), suivi d'une mise en correspondance entre les arcs (similarité spatiale entre ces primitives) pour produire un score à chaque étape de la mise en correspondance.

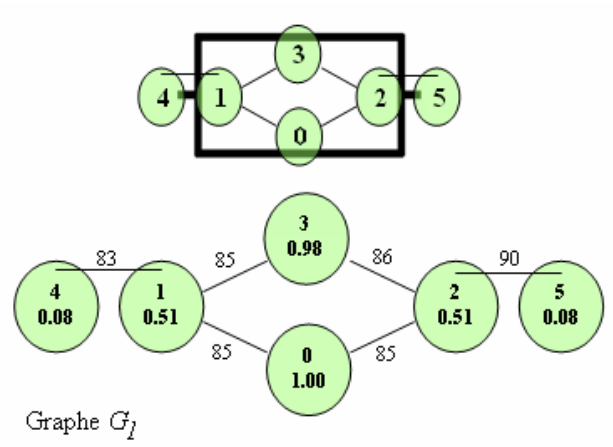
Formellement, l'algorithme peut être décrit de la manière suivante :

Etant donné deux graphe $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ et $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, V_1 et V_2 sont les ensembles finis des sommets, $E_1 \subseteq V_1 \times V_1$ et $E_2 \subseteq V_2 \times V_2$ sont les ensembles des arcs, α_1, α_2 sont des fonctions qui assignent les étiquettes aux sommets ($\alpha : V \rightarrow A_V^i$) et β_1, β_2 sont des fonctions qui assignent les étiquettes aux arcs ($\beta : E \rightarrow A_E^j$)

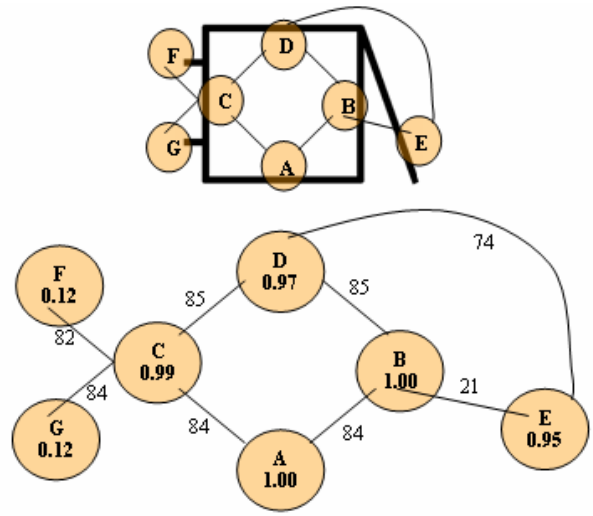
Une matrice $N = (n_{ij})$ de dimension $|V_1| \times |V_2|$ est produite dans laquelle chaque élément n_{ij} représente la similarité entre le sommet i du premier graphe et le sommet j du deuxième graphe. Une deuxième matrice $P = (p_{lm})$ de dimension $|E_1| \times |E_2|$ est introduite également, dans laquelle chaque élément p_{ij} représente la similarité entre l'arc l du premier graphe et l'arc j du deuxième graphe.

L'algorithme construit les appariements qui comportent les meilleures mises en correspondance en associant d'abord les sommets produisant les scores les plus hauts parmi l'ensemble des mises en correspondances possibles entre deux graphes. Le couple (v_i, v_j) où $v_i \in V_1$ et $v_j \in V_2$ est choisi sur la base du plus haut score de similarité dans la matrice de similarité entre les sommets. Il devient un couple candidat. On calcule alors le nouveau score en ajoutant le score de similarité du couple candidat et le score généré à partir

des mises en correspondance entre les arcs relatifs aux mises en correspondance entre sommets existant à l'étape *t*. Vu que cet algorithme de mise en correspondance n'autorise que les appariements univoques (pas d'associations multiples possibles). A chaque étape le couple de sommets produisant la plus forte augmentation du score (en considérant les mises en correspondance sommet-à-sommet et arc-à-arc) est sélectionné. Les lignes et colonnes associées à ces sommets dans matrice de similarité entre sommets sont alors éliminées (non utilisable par la suite). Cette procédure continue jusqu'à n'avoir plus aucune association possible entre les sommets des deux graphes. L'espace de recherche exploré est ainsi très limité et l'algorithme peut déterminer la solution après quelques itérations. La figure. 4.13 résume le fonctionnement et décrit le résultat obtenu avec cet algorithme (nommé Auction) sur un exemple simple.



Graphe G_1

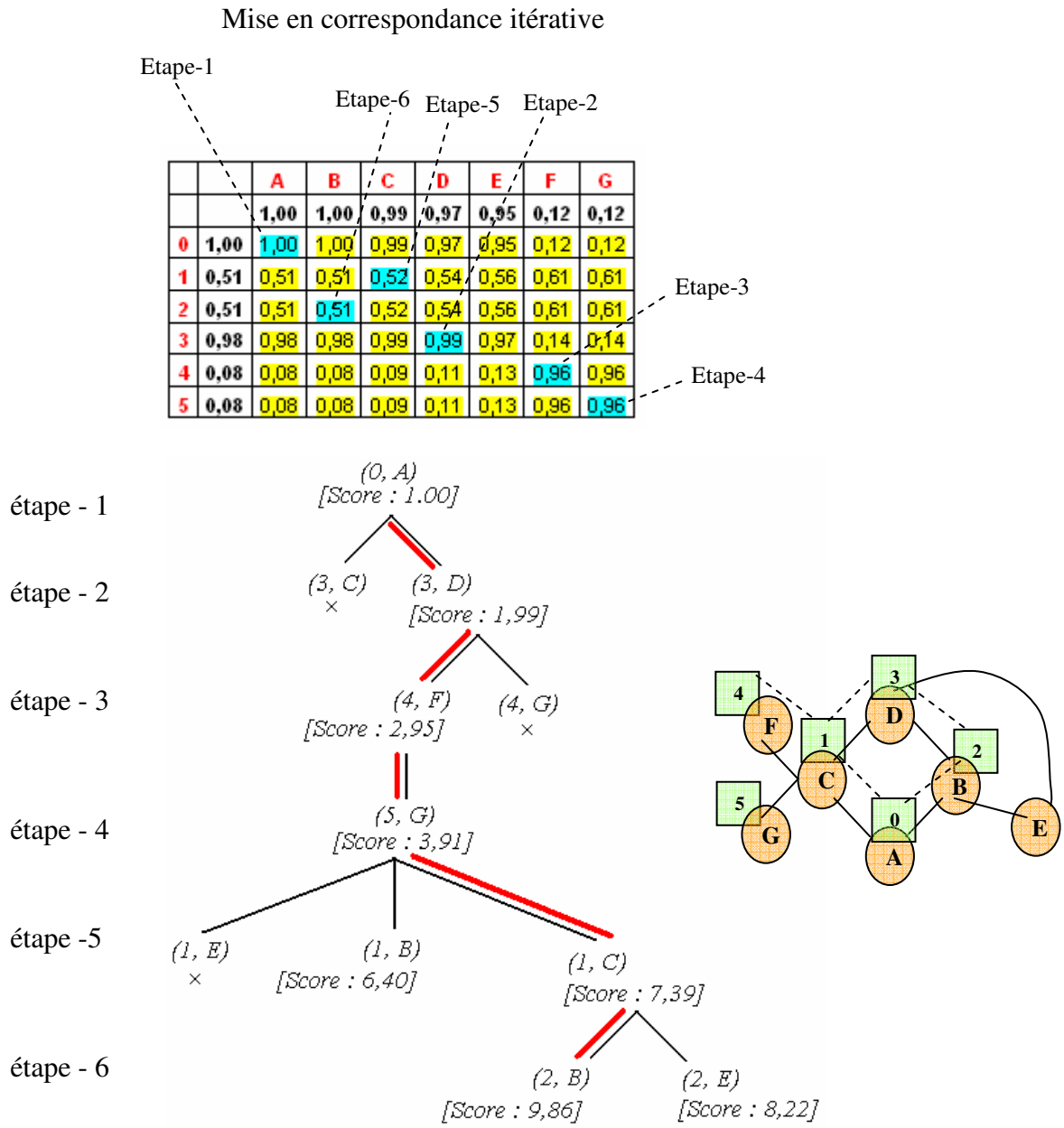


Graphe G_2

Matrice de similarité sommets et matrice de similarité entre arcs des graphes G_1 et G_2

λ_1 \ λ_2		A	B	C	D	E	F	G
		1,00	**1,00**	**0,99**	**0,97**	**0,95**	**0,12**	**0,12**
0	**1,00**	1,00	1,00	0,99	0,97	0,95	0,12	0,12
1	**0,51**	0,51	0,51	0,52	0,54	0,56	0,61	0,61
2	**0,51**	0,51	0,51	0,52	0,54	0,56	0,61	0,61
3	**0,98**	0,98	0,98	0,99	0,99	0,97	0,14	0,14
4	**0,08**	0,08	0,08	0,09	0,11	0,13	0,96	0,96
5	**0,08**	0,08	0,08	0,09	0,11	0,13	0,96	0,96

Φ_1 \ Φ_2		A-B	A-C	B-D	B-E	C-D	C-F	C-G	D-E
		84	**84**	**85**	**21**	**85**	**82**	**84**	**74**
0-1	**85**	0,98	0,98	1,00	0,28	1,00	0,96	0,98	0,87
0-2	**85**	0,98	0,98	1,00	0,28	1,00	0,96	0,98	0,87
1-3	**85**	0,98	0,98	1,00	0,28	1,00	0,96	0,98	0,87
1-4	**83**	0,98	0,98	0,97	0,31	0,97	0,98	0,98	0,90
2-3	**86**	0,97	0,97	0,98	0,27	0,98	0,95	0,97	0,86
2-5	**90**	0,93	0,93	0,94	0,23	0,94	0,91	0,93	0,82



		A	B	C	D	E	F	G
		1,00	1,00	0,99	0,97	0,95	0,12	0,12
0	1,00	1,00	1,00	0,99	0,97	0,95	0,12	0,12
1	0,51	0,51	0,51	0,52	0,54	0,56	0,61	0,61
2	0,51	0,51	0,51	0,52	0,54	0,56	0,61	0,61
3	0,98	0,98	0,98	0,99	0,99	0,97	0,14	0,14
4	0,08	0,08	0,08	0,09	0,11	0,13	0,96	0,96
5	0,08	0,08	0,08	0,09	0,11	0,13	0,96	0,96

Fig. 4.13: L'arbre de scores et appariement des sommets fournissent par le meilleur score.

Le score de similarité final entre les deux graphes peut être calculé à l'aide de la formule suivante :

$$Sim(G_1, G_2) = \frac{2 \times Sc_{Mp}}{\left[\left(C(V_1) + C(E_1)\right) + \left(C(V_2) + C(E_2)\right)\right]} = \frac{2 \times 9{,}86}{12 + 15} = 0{,}73$$

Nous avons construit une base d'images test avec des distorsions vectorielles à partir de 20 modèles de la base GREC-03 (10 architecturaux et 10 électroniques), afin, de comparer SimGraph avec cet algorithme. La table suivante présente les résultats sur chaque symbole et les taux de reconnaissance.

Tab. 4.12: Comparaison d'Auction et SimGraph.

Modèles	**Nombre de Tests**	**Bien reconnus**	
		Auction	**SimGraph**
1	30	20	30
2	30	21	28
3	30	22	23
4	30	25	28
5	30	26	28
6	30	23	25
7	30	24	26
8	30	29	30
9	30	24	29
10	30	27	28
11	30	28	30
12	30	29	30
13	30	22	25
14	30	17	26
15	30	10	28
16	30	21	24
17	30	20	24
18	30	24	30
19	30	27	27
20	30	28	30
	600	467	549
Taux de Reconnaissance		**77,83%**	**91,5%**

Les résultats fournis par SimGraph restent largement supérieurs et montrent l'interêt de parcourir correctement l'espace des mises en correspondance possibles pour trouver le meilleur appariement. Cette conclusion nous a poussés à chercher à optimiser SimGraph plutôt qu'à chercher à le remplacer.

4.7 Optimisations possibles de SimGraph

4.7.1 Introduction

Les résultats obtenus avec SimGraph (utilisant un algorithme Glouton) sont de bonne qualité. Notre méthode permet de traiter aussi bien les formes pleines, les formes mixtes que les formes uniquement linéaires. Notre représentation structurelle est également invariante aux transformations affines de base. SimGraph fournit en sortie, non seulement un score de similarité, mais également l'appariement sommet à sommet associé à ce score. Ceci permet de traiter des documents complets (via le mécanisme de spotting présenté dans ce manuscrit par exemple) et de remettre dans leur contexte (environnement) les symboles reconnus ainsi que leurs différentes parties. Les lacunes de SimGraph concernent donc principalement la résistance aux bruits et les temps de calcul. Cette partie donne des pistes pour pallier ces inconvénients.

4.7.2 Accélération à l'aide d'une pré-sélection des modèles significatifs

L'idée que nous avons retenu pour accélérer le processus de reconnaissance consiste à limiter le nombre de comparaisons de graphes (utilisation de SimGraph). Pour cela, il est nécessaire d'utiliser un mécanisme (très rapide) de filtrage/sélection des graphes modèles pertinents.

Pour cela, nous avons choisi d'utiliser un mécanisme de comparaison de graphes basé sur le calcul de signatures de graphes (graph probing).

- **Présentation de notre méthode de signatures de graphes**

L'idée fondamentale de la signature de graphe ou G-signature est de transformer les graphes en vecteurs de dimension *l* contenant un ensemble de caractéristiques numériques discriminantes calculées sur les graphes mais plus faciles à stocker et à manipuler. Nous avons déterminé un ensemble de 23 caractéristiques de trois types différents pour construire nos signatures de graphes.

- **Caractéristiques quantitatives**

Ce premier ensemble de caractéristiques se compose du nombre de sommets dans le graphe, du nombre d'arcs dans le graphe, du nombre de sommets reliés à 1, 2, 3, 4 ou à plus de 4 sommets. Cet ensemble de caractéristiques a été choisi de manière à rapprocher les graphes de tailles similaires.

- **Caractéristiques symboliques**

Le deuxième ensemble de signatures résume les informations symboliques contenues dans nos graphes (sur les arcs). Ces attributs jouent également un rôle important pour rassembler les formes similaires. Cinq attributs indiqueront le nombre d'arcs ayant chaque étiquette L, P, T, X, ou S

- **Caractéristiques numériques**

La valeur relative de la longueur d'une primitive (nœuds du graphe) est comprise entre 0 et 1. Nous avons choisi de discrétiser cet intervalle en 5 classes par pas de 0.2. De même, l'angle relatif entre les primitives stockées dans un attribut associé aux arcs du graphe et dont la valeur est toujours comprise entre 0° et 90° est divisé en 3 classes par pas de 30°. Les signatures contiennent alors le nombre de sommets ayant une longueur relative (*RL*) dans un intervalle spécifique et le nombre d'arcs ayant l'angle relatif (*RA*) dans un intervalle spécifique (voir figure. 4.14).

Quantitatifs	***Symboliques***	***Numériques***
f_1 Nb de Nœuds	f_{11} Nb d'arcs avec L	f_{16} Nb de Nœuds ont λ (0.0 – 0.2)
f_2 Nb d'arcs	f_{12} Nb d'arcs avec P	f_{17} Nb de Nœuds ont λ (0.2 – 0.4)
f_3 Nb de Nœuds connecté à 1 Nœud	f_{13} Nb d'arcs avec T	f_{18} Nb de Nœuds ont λ (0.4 – 0.6)
f_4 Nb de Nœuds connecté à 2 Nœud	f_{14} Nb d'arcs avec X	f_{19} Nb de Nœuds ont λ (0.6 – 0.8)
f_5 Nb de Nœuds connecté à 3 Nœud	f_{15} Nb d'arcs avec S	f_{20} Nb de Nœuds ont λ (0.8 – 1.0)
f_6 Nb de Nœuds connecté à 4 Nœud		f_{21} Nb d'Arcs ont angle (0° – 30°)
f_7 Nb de Nœuds connecté à 5 Nœud		f_{22} Nb d'Arcs ont angle (30° – 60°)
f_8 Nb de Nœuds connecté à 6+ Nœud		f_{23} Nb d'Arcs ont angle (60° – 90°)
f_9 Nb de Nœud de forme 1(Quad)		
f_{10} Nb de Nœud de forme 2 (Vect)		

Fig. 4.14: Signatures de graphes choisies.

Nous obtenons donc, au final, un vecteur composé de 23 valeurs numériques à même de résumer les caractéristiques de nos graphes décrivant les formes. Ces caractéristiques ont été choisies principalement pour leur simplicité et leur rapidité d'obtention. Nous n'avons pas eu suffisamment de temps pour effectuer une sélection moins empiriques et plus rigoureuse des caractéristiques discriminantes. Néanmoins, la figure 4.15 donne une idée du pouvoir discriminant des différentes signatures extraites des graphes.

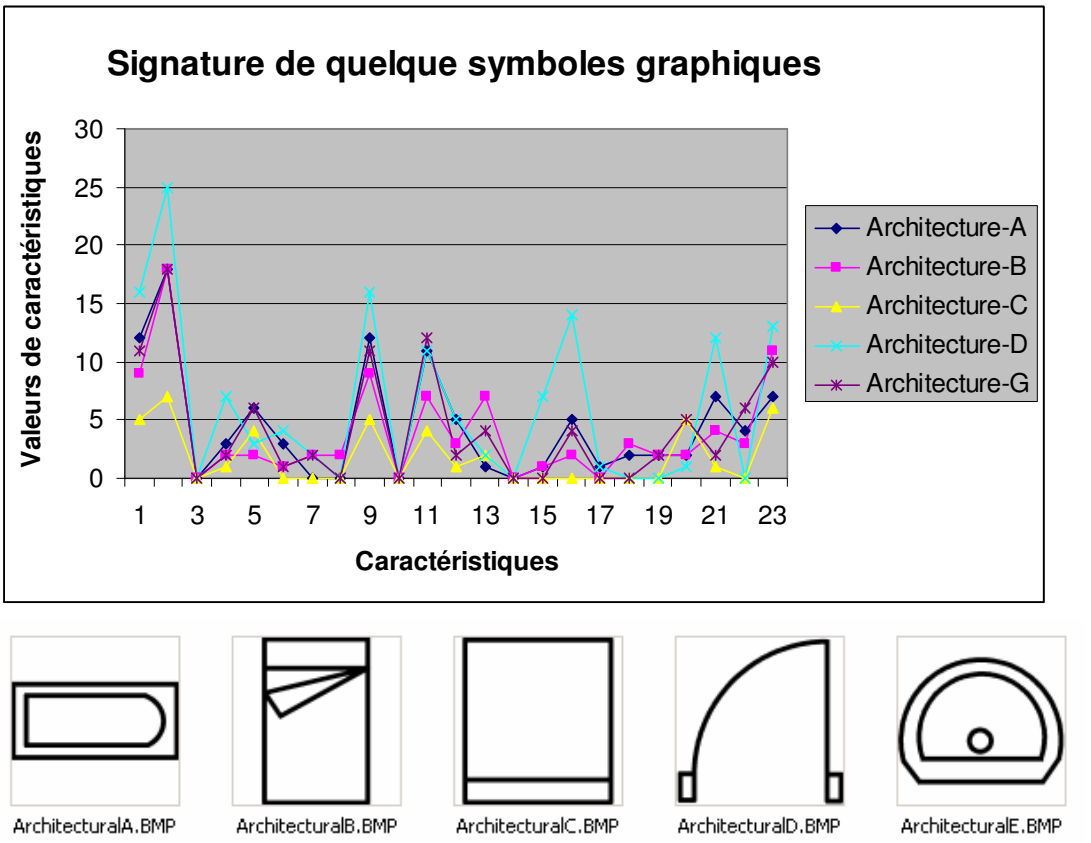


Fig. 4.15 : Comparaison de G-signatures associées à différents symboles graphiques.

Pour avoir un score de similarité entre deux graphes à partir de leur signature (vecteur), nous avons utilisé la distance de Bray Curtis[3], parfois appelée aussi, distance de Sorensen[1]. La distance est normalisée entre 0 et 1 en utilisant la différence absolue des éléments de deux vecteurs divisée par leur somme. Le distance d_{ij} entre graphes G_i et G_j est décrite par :

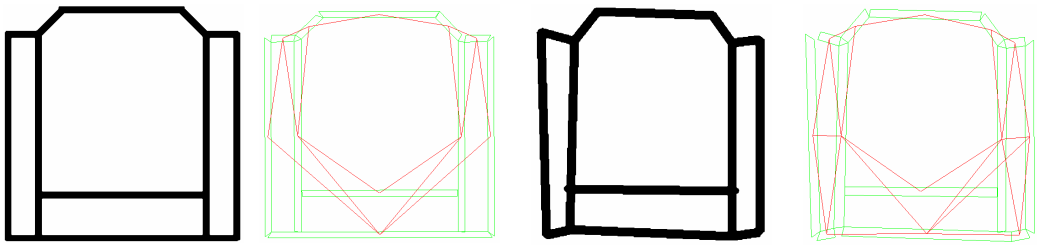
[3]http://people.revoledu.com/kardi/tutorial/Similarity/BrayCurtisDistance.html

$$d_{ij} = \frac{\sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|}{\sum_{k=1}^{n} \left(x_{ik} + x_{jk} \right)} \quad \text{où} \quad d_{ij} = \frac{\sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|}{\sum_{k=1}^{n} x_{ik} + \sum_{k=1}^{n} x_{jk}}$$

Si G_i et G_j sont deux graphes, et x_{ik} et x_{jk} les $k^{ième}$ éléments des vecteurs de caractéristiques extraites à partir de deux graphes ; les vecteurs de caractéristiques étant de longueur k.

$$G_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad \ldots \quad x_{ik}]$$
$$G_j = [x_{j1} \quad x_{j2} \quad x_{j3} \quad \ldots \quad x_{jk}]$$

$$\text{Score}(G_i, G_j) = 1 - d_{ij}(G_i, G_j)$$



a) Architecture – G b) Architecture – G(D)

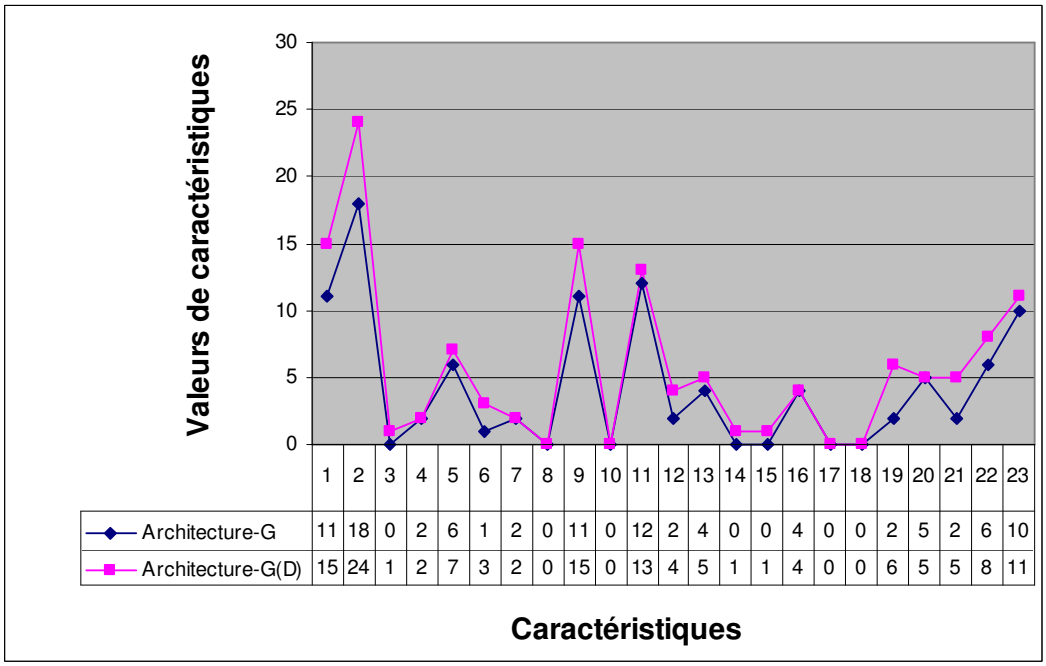


Fig. 4.16 : Le G-signature d'un symbole graphique contre sa version dégradée.

La figure 4.16 montre les différences et points communs entre la signature d'un symbole graphique et sa version dégradée par distorsion vectorielle. On voit bien que les deux signatures sont légèrement différentes à cause des distorsions.

Afin d'évaluer les performances de ce classificateur seul, nous avons choisi de calculer son pouvoir discriminant avec la formule suivante :

$$\text{Discrimination} = [S_k - max\ (S_1, ..., S_f, \ldots, S_n)] \times 100$$

S_k est le score de similarité entre le prototype et le modèle associé (classe identique), S_f représente les scores de similarité entre le prototype et les autres modèles (autres classes).

Les tableaux 4.13 à 4.15 fournissent les pouvoirs discriminants obtenus sur quelques prototypes de symboles architecturaux et électroniques.

Tab. 4.13 : Reconnaissance de symboles architecturaux à partir de leur G-Signature.

						Pouvoir discriminant
	**1.000000**	0.765550	0.549020	0.759690	0.379447	23%
	0.765550	**1.000000**	0.541667	0.666667	0.309859	23%
	0.549020	0.541667	**1.000000**	0.414508	0.158730	45%
	0.759690	0.666667	0.414508	**1.000000**	0.483516	24%
	0.379447	0.309859	0.158730	0.483516	**1.000000**	52%

Les chiffres en rouges dans les tableaux 4.13 à 4.15 représente les scores de similarités entre les prototypes et les plus proches modèles voisins.

Tab. 4.14: Reconnaissance de symboles électroniques à partir de leur G-Signature

						Pouvoir discriminant
	**1.000000**	0.789474	0.748201	0.679245	0.832000	17%
	0.789474	**1.000000**	0.692913	0.808511	0.778761	20%
	0.748201	0.692913	**1.000000**	0.672269	0.753623	25%
	0.679245	0.808511	0.672269	**1.000000**	0.704762	20%
	0.832000	0.778761	0.753623	0.704762	**1.000000**	17%

Tab. 4.15: Reconnaissance de symboles avec distorsion
à partir de leur *G*-Signature.

						Pouvoir discriminant
	**0.918367**	0.359649	0.822222	0.573643	0.555556	10%
	0.373984	**0.808511**	0.352941	0.512635	0.657534	15%
	0.765957	0.366071	**0.953488**	0.528000	0.528571	19%
	0.684211	0.426230	0.622642	**0.758621**	0.700000	5%
	0.661654	0.547529	0.592000	0.756098	**0.905028**	15%

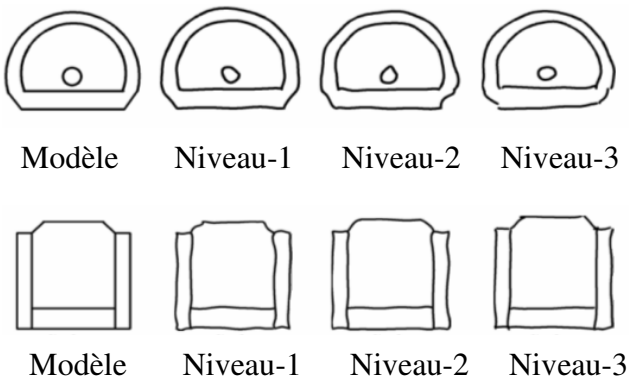


Fig. 4.17 : Les trois modèles de dessin à main levée utilisés.

Un résumé des résultats de reconnaissance de symboles avec la G-Signature est présenté dans la table 4.16. Les taux de reconnaissance que nous avons obtenus dans le cas d'images non bruitées et d'images après rotations ou changements d'échelle sont de 100%. Cependant, pour les symboles avec des distorsions vectorielles (figure 4.17), les taux de reconnaissance diminuent significativement à cause de changements significatifs sur les angles relatifs pour les arcs, de l'augmentation du nombre de sommets et de changements importants de leurs valeurs numériques (Tableaux 4.16).

Tab. 4.16: Résumé de résultats de reconnaissance de symboles avec G-Signature.

		Nombre de Modèles	Nombre de Tests	Bien Reconnus	Taux de Reconnaissance
Test idéal		50	50	50	100%
Rotation		50	150	150	100%
Changement d'échelle		50	100	100	100%
Bruit	Niveau-1	50	200	150	75%
	Niveau-2	50	200	133	66.5%
	Niveau-3	50	200	123	61,5%
Distorsion		15	100	85	85%
Dessin à main levée	Niveau-1	15	450	402	89,33%
	Niveau-2	15	450	379	84,22%
	Niveau-3	15	450	352	78,22%

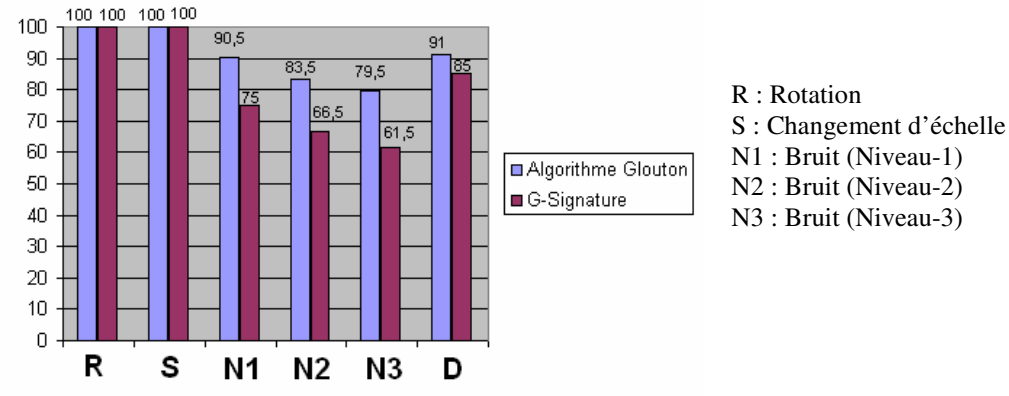


Fig. 4.18: Comparaison des performances obtenues avec l'algorithme Glouton et par G-Signature.

La signature de graphe est un outil rapide mais pas très efficace lorsque pour les symboles sont bruités et/ou déformés. De plus, rappelons que notre algorithme Glouton fournit un appariement sommet à sommet en plus du score de similarité ce qui n'est pas le cas avec la G-signature. Ainsi, son couplage avec SimGraph(Algorithme Glouton) nous parait intéressant. Cela permettrait d'obtenir le meilleur appariement, de diminuer les temps de calcul et de garder la robustesse et la précision en même temps. Nous avons donc testé cette proposition.

- **Mise en cascade de G-signature suivi de SimGraph**

Nous proposons d'utiliser la signature de graphe pour trouver les *K* plus proches voisins d'un graphe parmi les graphes modèles et ensuite de lancer SimGraph pour déterminer le meilleur modèle parmi les *K* voisins sélectionnés. En fait, le nombre de voisins n'est pas fixé d'avance mais déterminé à partir d'un seuil sur les scores de similarité obtenus sur les voisins proches (méthodes des ϵ voisins). Par exemple, tous les modèles voisins pour quelques symboles sont montrés dans le Tableau 4.17. Ici, le seuil était fixé à 0,75. Nous voyons bien que le modèle correct d'un symbole n'est pas toujours placé en première position mais est toujours présent dans la liste des voisins. Comme nous avons réduit l'espace de recherche pour SimGraph, les temps de calcul sont réduits significativement.

Tab. 4.17: Symboles et leurs plus proches voisins trouvés par G-Signature.
(m-n représente l'exemple de distorsion *n* d'un symbole *m*.).

Symbole	1er – N N	2ème – N N	3ème - N N	4ème - N N	5ème – N N
m-n					
1-1	**1** (0.908257)	7 (0.790698)			
1-3	**1** (0.895928)	7 (0.798165)	4 (0.758621)		
1-5	**1** (0.887931)	7 (0.812227)	4 (0.794118)		
…					
2-1	**2** (0.826667)	4 (0.824818)	1 (0.820513)	10 (0.792793)	7 (0.779221)
2-3	4 (0.802768)	7 (0.756098)	**2** (0.785714)		
2-4	**2** (0.864322)	10 (0.785714)	7 (0.780488)	1 (0.778846)	17 (0.765714)
…					
3-2	**3** (0.857143)	14 (0.844444)	13 (0.831683)	18 (0.822222)	16 (0.795918)
3-4	**3** (0.880952)	14 (0.867470)	18 (0.843373)	13 (0.808511)	16 (0.791209)
3-5	14 (0.866667)	**3** (0.857143)	13 (0.831683)	18 (0.800000)	16 (0.795918)
…					
4-1	**4** (0.833333)	1 (0.785714)	11 (0.752688)		
…					
5-4	**5** (0.846676)				
…					
6-4	7 (0.830357)	1 (0.810573)	**6** (0.779817)		
6-5	**6** (0.846561)	7 (0.816901)	1 (0.796296)	2 (0.782609)	
…					
7-1	**7** (0.890995)	1 (0.859813)	6 (0.770053)	9 (0.758621)	

Même si les taux de reconnaissance restent les mêmes avec ou sans SimGraph sur des images non dégradées, nous conservons SimGraph afin d'obtenir l'appariement et les temps de calcul sont réduits significativement comme le montre la figure suivante.

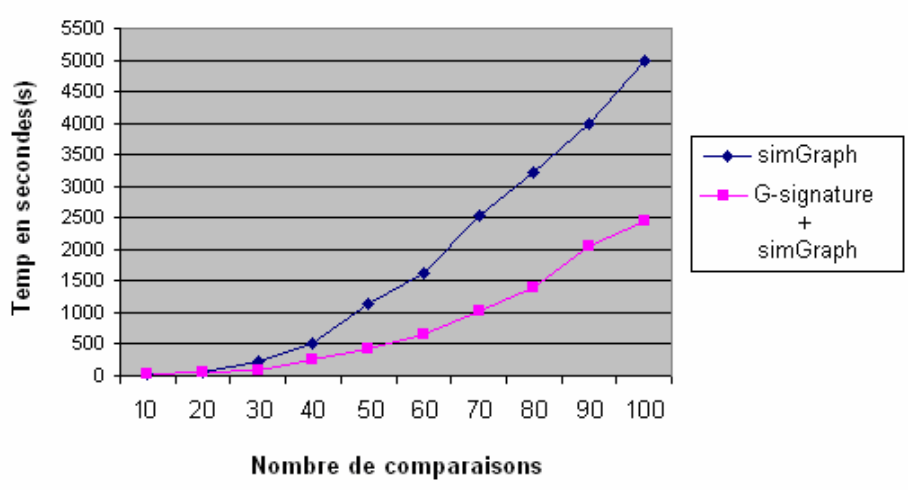


Fig. 4.19: Comparaison des temps de calcul avec SimGraph seul et couplé avec la G-signature.

4.8 Amélioration de la résistance au bruit

Nous avons noté que l'approche proposée donne des résultats moins bons lorsque les images sont bruitées. Des petits quadrilatères apparaissent (figure 4.20b) à cause du bruit présent dans l'image. Une solution pour éviter les quadrilatères parasites est d'appliquer des filtre de pré-traitements comme les dilatations, erosions etc. Il n'est malheureusement pas toujours facile de déterminer quels filtres appliqués sur l'image. La solution que nous avons choisie est de supprimer toutes les primitives qui satisfont les deux conditions suivantes: longueur relative inférieure à un certain seuil pré-défini, et primitive isolée (nombre de voisins = 0).

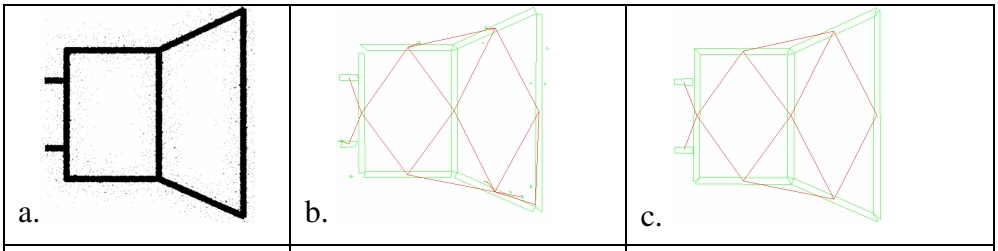
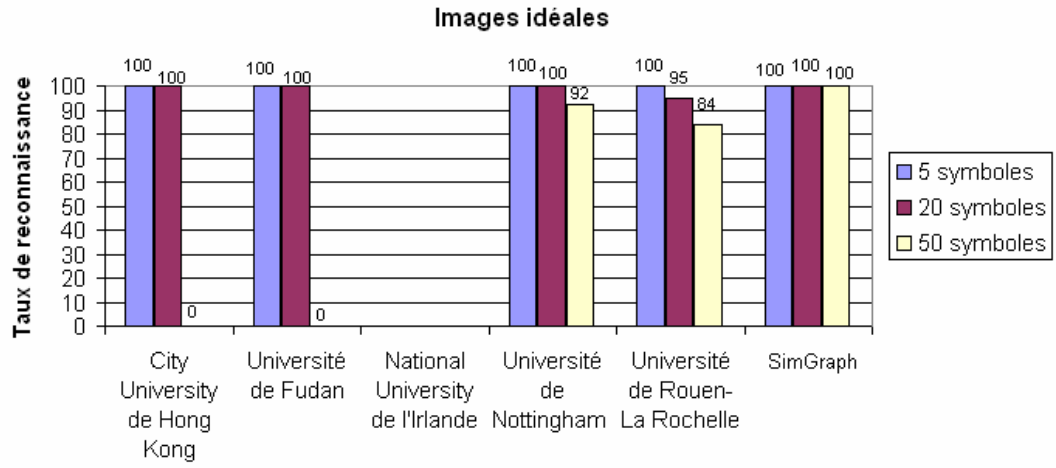


Fig. 4.20: Influence du bruit sur le graphe représentant l'image.

Avec ces prétraitements les résultats sont améliorés lorsque le niveau de bruit reste faible mais cela ne permet pas de traiter les modèles de bruits de 5 à 9 (figure. 2.18).

Dans la partie suivante, nous présentons les performances de notre méthode sur la base de données de GREC-03 afin de comparer ses performances avec les autres méthodes proposées. Un problème commun à l'ensemble des méthodes présentées à la compétition GREC 2003 est le passage à l'échelle. Les méthodes présentées fonctionnent bien pour un nombre limité de classes de symboles (de 5 à 20) mais leurs performances sont significativement dégradées pour une base de plus de 50 symboles différents. Notre méthode donne de très bons taux de reconnaissance sur les images non bruitées, dans le cas de transformations affines et distorsions vectorielles. Cependant, les taux de reconnaissance obtenus par notre méthode sur les images bruitées restent plus faibles que ceux des autres méthodes ayant participé au concours.


Images idéales
Taux de reconnaissance
100 100 0
100 100 0
100 100 92
100 95 84
100 100 100
5 symboles
20 symboles
50 symboles
City University de Hong Kong
Université de Fudan
National University de l'Irlande
Université de Nottingham
Université de Rouen-La Rochelle
SimGraph

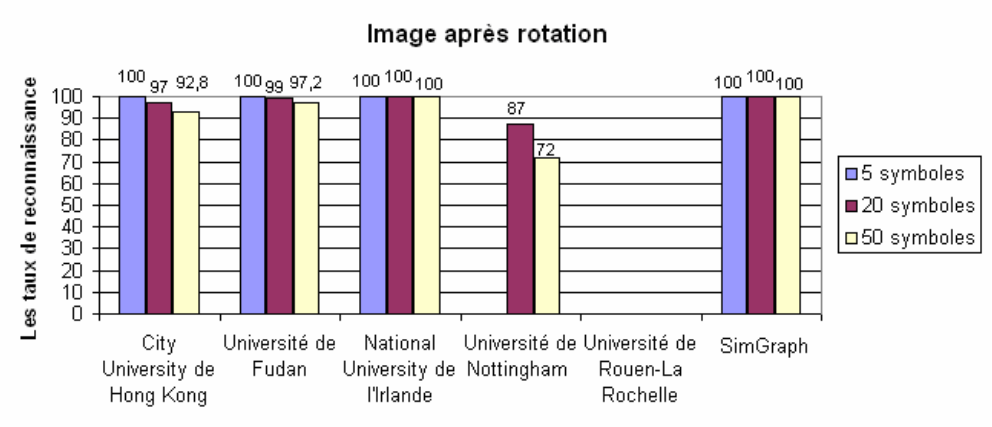
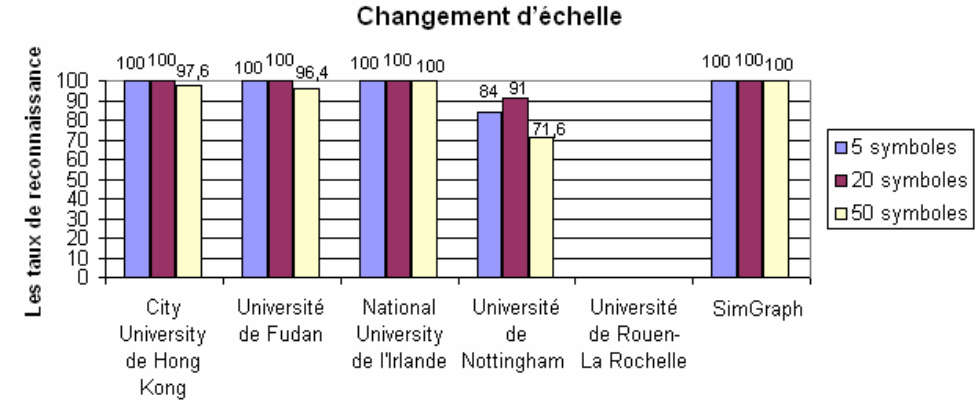


Image après rotation
Les taux de reconnaissance
100 97 92,8
100 99 97,2
100 100 100
87
72
100 100 100
5 symboles
20 symboles
50 symboles
City University de Hong Kong
Université de Fudan
National University de l'Irlande
Université de Nottingham
Université de Rouen-La Rochelle
SimGraph


Changement d'échelle
Les taux de reconnaissance
100 100 97,6
100 100 96,4
100 100 100
84 91 71,6
100 100 100
5 symboles
20 symboles
50 symboles
City University de Hong Kong
Université de Fudan
National University de l'Irlande
Université de Nottingham
Université de Rouen-La Rochelle
SimGraph

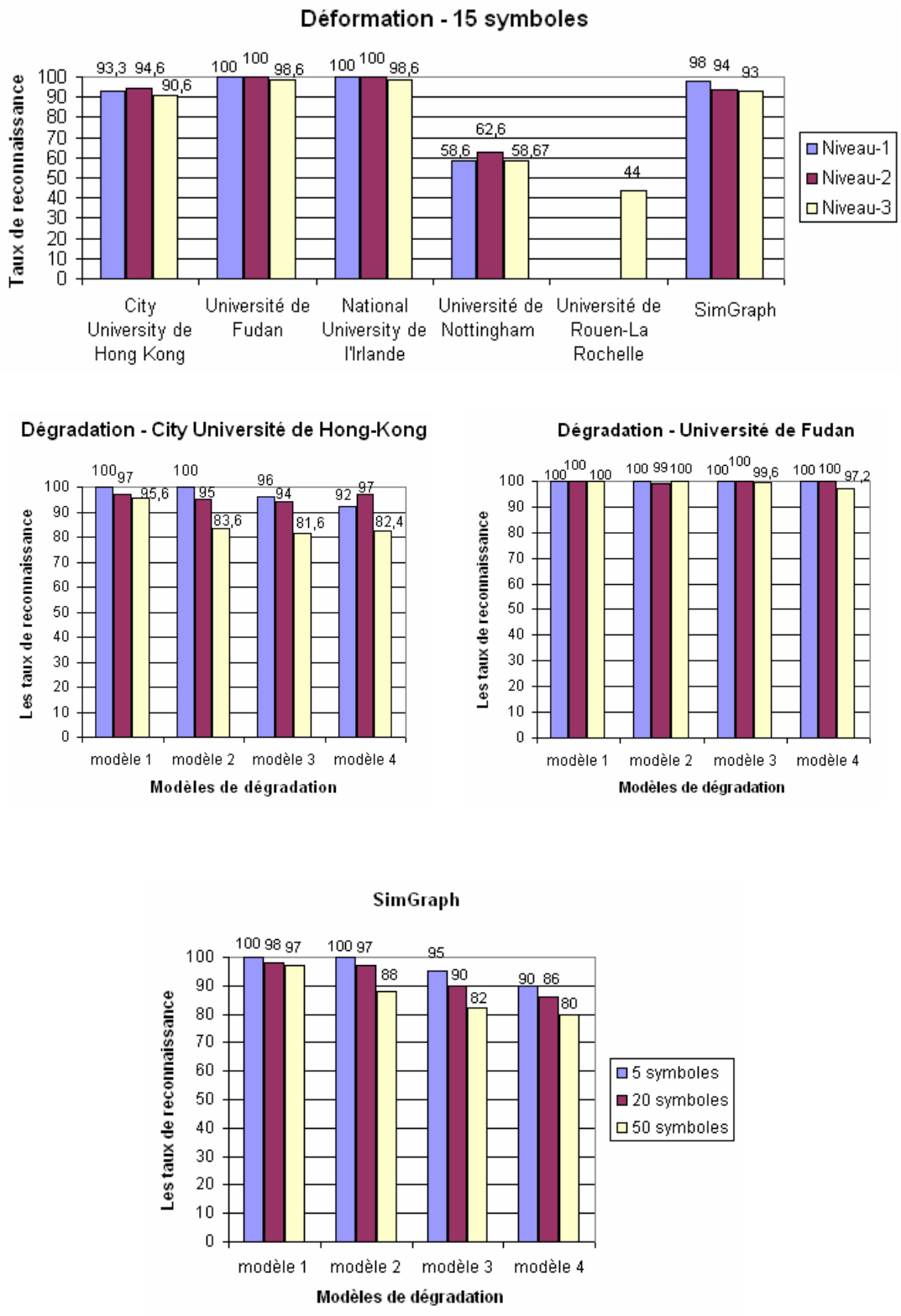


Fig. 4.21: Performances de SimGraph sur la base de données GREC-03.

4.9 Conclusion

De nombreuses études ont été faites sur la reconnaissance de symboles graphiques, mais la plupart de ces études ne prenaient en considération que les formes linèaires et négligeaient les symboles plus complexes ou les formes pleines. Pour tenir compte de la grande variabilité des symboles et de la difficulté de les segmenter, nous avons essayé de développer des méthodes de reconnaissance capable de traiter différents types d'images.

Les résultats produits par les mises en correspondance des graphes avec des attributs symboliques seulement sont insuffisants. Nous avons donc choisi d'apporter des modifications à ce type de méthodes afin de permettre l'utilisation d'attributs numériques aussi bien que symboliques lors de la comparaison de deux graphes.

Notre méthode, SimGraph montre un fort pouvoir discriminant dans le cas d'images non bruitées avec un taux de reconnaissance de 100% pour un ensemble de test comportant jusqu'à 50 modèles de symboles différents. Notre représentation structurelle étant également invariante aux transformations affines de base, notre méthode est donc robuste aux rotations et changements d'échelle. SimGraph fournit en sortie non seulement un score de similarité, mais également le meilleur appariement sommet à sommet associé à ce score. Ceci permet de traiter des documents complets (via le mécanisme de spotting présenté dans ce manuscrit par exemple) et de remettre dans leur contexte (environnement) les symboles reconnus ainsi que leurs différentes parties. Les lacunes de SimGraph sont les problèmes hérités des méthodes structurelles c'est-à-dire le manque de résistance aux bruits et les temps de calcul trop importants.

Dans le but d'étudier l'influence de la réduction de l'espace de recherche (du meilleur appariement) afin d'accélérer le processus de reconnaissance, nous avons développé « Auction Algorithm », un algorithme plus simpliste qui réduit l'espace de recherche d'une manière considérable et permet de trouver une solution approximative rapidement. Mais, les résultats fournis par SimGraph restent largement supérieurs grâce à son exploration correcte de l'espace des mises en correspondance possibles pour trouver le meilleur appariement avec possibilité d'appariements multivoques. Cette conclusion nous a poussés à chercher à

optimiser SimGraph plutôt que de chercher à le remplacer. Nous proposons finalement d'utiliser la signature de graphe pour trouver rapidement les plus proches voisins d'un graphe parmi les graphes modèles et ensuite de lancer SimGraph pour déterminer le meilleur modèle parmi les K voisins sélectionnés. La signature de graphe est un outil rapide et son couplage avec SimGraph a prouvé son efficacité. Nous avons réussi à garder la précision et à diminuer les temps de calcul en même temps.

SimGraph fournit des résultats moins bons lorsque les images sont bruitées. Mais nous pensons que le problème ne provient pas de la méthode de recherche des mises en correspondance par SimGraph. Ces erreurs viennent plutôt de l'étape de générations des graphes (vectorisation et construction des primitives). C'est pour cela que les solutions proposant des prétraitements ont amélioré les résultats en présence de bruit.

Conclusion

Globalement, nous pouvons répertorier nos principales contributions en reprenant les trois grandes étapes menant à l'interprétation de documents.

Représentation des documents graphiques. Durant ce travail, nous avons complété l'idée proposée par JY Ramel, visant à obtenir une représentation structurelle des formes fines et des formes pleines d'une image de document à partir d'un mécanisme de vectorisation originale. Nous avons proposé une représentation unique utilisant des graphes de primitives structurelles (*Vecteur* + *Quad*) avec des attributs numériques et symboliques. Nous avons ensuite utilisé cet outil de façons très diverses pour explorer sa pertinence dans différents axes de recherche.

L'amélioration produite au niveau de l'approximation polygonale des contours augmente beaucoup la qualité de la vectorisation particulièrement dans les parties d'images avec des arcs ou courbes. Les critères ajoutés pour autoriser ou rejeter les opérations de fusions des vecteurs et de découpages des quadrilatères aident à mieux modéliser les formes complexes avec nos primitives graphiques. Les deux primitives Vecteurs et Quadrilatères choisies pour représenter les formes fines et les formes pleines contenues dans les images constituent une combinaison parfaite pour pallier les défauts des méthodes d'amincissement et les imperfections des méthodes d'extraction des axes médians. D'ailleurs aujourd'hui, à notre connaissance, aucune autre représentation structurelle n'intègre simultanément une description des formes fines et formes pleines. Le graphe représentant l'ensemble du contenu de l'image est invariant aux transformations affines. Les attributs associés aux sommets et aux arcs décrivent bien les différentes formes en calculant différentes statistiques. Le stockage des graphes au format GXL offre un moyen flexible et adaptable pour gérer l'interopérabilité entre les outils basés sur des graphes et permet de les échanger et de les diffuser.

Reconnaissance de symboles graphiques. La difficulté principale à laquelle nous avons dû faire face est de trouver un équilibre entre rapidité et précision. Dans cette thèse, nous avons proposé trois nouvelles techniques de mise en correspondance de

graphes pour résoudre le problème du temps de calcul et de tolérance aux imperfections. Nous avons tout d'abord choisi d'appliquer la méthode proposée par Champin sans modification avec les attributs symboliques disponibles dans nos graphes. Nous avons réalisé que la mesure de similarité utilisée restait trop simple et que l'usage d'attributs symboliques n'était pas suffisant. Nous avons alors ajouté la possibilité d'utiliser des attributs numériques comme par exemple les longueurs relatives (associées aux sommets) et les angles relatifs (associées aux arcs) afin d'obtenir la précision nécessaire pour comparer deux formes graphiques complexes en utilisant en plus les relations topologiques entre les primitives les constituant. L'utilisation d'attributs numériques pour caractériser les primitives et leurs relations permet d'allier précision et invariance à la rotation et au changement d'échelle. Nous avons proposé une fonction de similarité qui permet de coupler les valeurs numériques et symboliques pour produire un score de similarité. En utilisant l'algorithme glouton, notre système "SimGraph" teste différents appariements possibles, calcule leur score de similarité et garde l'appariement correspondant au plus haut score afin d'obtenir le meilleur appariement entre les sommets des deux graphes. Cette mesure de similarité a de nombreuses propriétés intéressantes comme un fort pouvoir discriminant et une invariance aux transformations affines. Les lacunes de SimGraph que nous avons notées concernent principalement la résistance aux bruits et les temps de calcul. L'approche proposée donne des résultats perfaits en absence de bruit. Si l'image est bruitée ou s'il y a des distorsions vectorielles, notre représentation basée sur les vecteurs et les quadrilatères peut rencontrer des problèmes se traduisant par l'apparition de petits quadrilatères et/ou de ruptures de connexité. Une solution pour éviter les quadrilatères parasites que nous avons choisi d'appliquer est de supprimer toutes les primitives isolées et/ou ayant une longueur relative inférieure à un certain seuil pré-defini. D'autres solutions à base de filtrage (lissage, dilatation, érosion) peuvent aussi être envisagées comme prétraitement avant de lancer la vectorisation. Le problème de rupture de quadrilatères est automatiquement traité par notre système 'SimGraph' en autorisant les appariements multivoques pendant la phase de mise en correspondance des graphes. Dans le but d'accélérer le processus de recherche et pour réduire les temps de calcul pour les comparaisons en cas de base de données de taille importante, nous avons envisagé deux autres algorithmes. Le premier à base d'arbres de recherche (Auction Algorithm) ne parcourt que très partiellement l'ensemble des appariements possibles entre deux graphes. Le deuxième algorithme utilise le concept de signature de graphes (G-signature). Les résultats montrent que ces approches sont rapides mais ne sont pas précises comparées à SimGraph. L'idée que nous avons retenue pour accélérer le processus et garder la précision en même temps consiste à coupler deux classificateurs.

La G-Signature semble plus précise par rapport à l'algorithme Auction, nous avons donc testé la mise en cascade de G-signature avec SimGraph. Les taux de reconnaissance obtenus par l'association de ces deux classificateurs restent la plupart du temps similaires à ceux de SimGraph mais les temps de calcul sont réduits significativement comme prévu.

Localisation de symboles graphique. Beaucoup de méthodes ont été développées tout au long de ces dernières années pour la reconnaissance des symboles graphiques pré-segmentés mais très peu de techniques ont atteint l'objectif de localisation et de reconnaissance de symboles sans connaissance a priori. Les acteurs principaux de ce domaine de recherche semblent ne vouloir s'intéresser qu'aux cartes architecturales.

D'autres méthodes ont besoin de nombreuses interactions Homme-Machine sur chacune des images à analyser. Concernant la localisation des symboles, il est indéniable qu'étant donné la densité des documents et la variabilité des symboles pouvant être présents dans les documents graphiques, l'objectif de mettre en place une méthode la plus générique possible reste difficile à atteindre. Nous avons fait le choix d'essayer de produire un score pour chaque élément des graphes (nœuds et arcs) représentant la probabilité d'appartenir à un symbole. A notre connaissance, il s'agit de la seule approche qui n'utilise pas de connaissance a priori sur le domaine, ni d'apprentissage à partir de modèles. Notre approche est automatique et nécessite peu d'interactions utilisateur. L'ensemble des régions pertinentes est extrait et les sous-graphes et « *Bounding Boxes* » sont soumis à SimGraph afin de trouver le modèle correspondant avec un objectif de reconnaissance et localisation au sein d'un seul système. Notre approche est générique du fait de sa capacité à localiser des symboles dans différents types d'images. Néanmoins, pour l'instant nous avons testé notre technique que sur les trois types d'images de documents (électronique, architecture et logique) et démontré la capacité de cette méthode à extraire des régions d'intérêt plutôt que directement les symboles. Il parait donc intéressant d'ajouter d'autres types d'hypothèses, ainsi que plus d'interactions utilisateur et de tester sur d'autres types de documents.

Perspectives

Nous avons plusieurs études en cours ayant comme objectifs d'améliorer la reconnaissance des formes complexes représentées à l'aide de graphes. Face à la complexité de certains symboles (particulièrement les symboles avec des arcs), il serait intéressant de définir et d'intégrer dans les graphes, des primitives de haut niveau comme les cercles et les arcs. La difficulté réside alors dans la définition d'attributs de description communs et utilisables pour caractériser des nœuds et des arcs de types variés (vecteurs, quadrilatères, cercles, triangles…). En effet, certains symboles peuvent être composés d'un ensemble très important de primitives graphiques de bas niveau. Une fusion de plusieurs primitives en quelques primitives de haut niveau pourrait donc engendrer une réduction significative de la taille des graphes représentant un symbole. Dans la représentation proposée, cette opération est tout à fait envisageable en utilisant des critères simples comme une fusion des petites primitives avec la relation S (en successive). Le même principe permet d'ores et déjà de localiser les arcs dans les images « arc segmentation ».

Ce type de procédé fait intervenir des mécanismes de reconnaissance incrémentale de formes de plus en plus complexes et la difficulté réside alors dans la mise en place du processus d'analyse incrémentale des images. Pour résoudre ce type de problème, l'équipe de recherche RFAI du LI de Tours travaille actuellement sur des systèmes d'analyse interactive d'images de documents à l'aide de scénarios. Ces deux approches pourraient donc être couplées prochainement. Le nombre d'attributs exploités par SimGraph, dans nos expérimentations est resté faible afin de garantir l'invariance aux rotations et changements d'échelle. Il parait intéressant de faire d'autres investigations en multipliant le nombre d'attributs sur les nœuds et les arcs afin d'étudier l'influence de la dimensionnalité (nombre d'attributs) sur ce type de méthodes (comparaison de graphes).

La liste des attributs actuels peut paraître limitée surtout si nous voulons explorer SimGraph dans d'autres axes de la reconnaissance graphiques comme la reconnaissance de logos. Dans ce cas, nous avons besoin de segmenter le logo en régions sur la base des couleurs (niveau de gris) et les relations entre ces régions doivent être redéfinies. Les attributs pour les sommets représentant les régions peuvent être la surface relative, la couleur, et sur les arcs les distances, la position relative (gauche, droite …). Le modèle

de comparaison de graphes proposé est tout à fait capable de gérer ce genre de changement, il suffit de définir les fonctions de distance *f* et *g* pour calculer les score normalisés entre 0 et 1.

Toujours pour la reconnaissance, intégrer un mécanisme d'apprentissage (à partir de plusieurs modèles d'un même symbole) est une voie de recherche qui nous parait à explorer. L'objectif est de reproduire ce qui se fait habituellement lors de l'usage de méthodes de reconnaissance statistique dans lesquelles on ne se contente pas d'un seul représentant (modèle) par classe ! Ce type de procédé reste un champ à explorer dans les années à venir pour ce qui concerne les approches structurelles.

Concernant, l'analyse d'images complètes (pour la rétro conversion ou l'indexation) le challenge reste selon nous la généricité (ou tout au moins la flexibilité) des méthodes proposées. Il parait réalisable de produire des systèmes de localisation efficaces en utilisant des heuristiques spécifiques à un type de documents (modèle de document). Nos expérimentations ont montré qu'il était beaucoup plus difficile de déterminer des heuristiques pertinentes quelque soit le domaine applicatif. Nous sommes, pour l'instant, uniquement parvenu à mettre en place un système capable de déterminer automatiquement des régions d'intérêt dans les graphes (correspondant potentiellement à des symboles) sur la base d'heuristiques construites de manière empirique. Ces résultats sont néanmoins encourageants et, là aussi, nous pensons qu'introduire un mécanisme d'apprentissage et de classification plus sophistiqué (que nos heuristiques) permettrait sûrement une avancée importante dans ce domaine.

Bibliographie

A

[Adam, 2000a] Adam, S., Ogier, J. M., Cariou, C., Mullot, R., Gardes, J., Lecourtier, Y. 2000. Fourier Mellin Based Invariants for the Recognition of Multi-Oriented and Multi-Scaled Shapes: Application to Engineering Drawing Analysis. *Invariants for Pattern Recognition and Classification.* Dans la série Machine Perception & Artificial Intelligence, World Scientific Publishing.

[Adam, 2000b] Adam, S., Ogier, J. M., Cariou, C., Mullot, R., Labiche, J., Gardes, J. 2000. Symbol and Character recognition: Application to Engineering Drawings. *International Journal of Document Analysis and Recognition,* **3(2)**, pp. 89 -101.

[Adam, 2001] Adam, S. Interprétation de Documents Techniques : des Outils à Leur Intégration dans un Système à Base de Connaissances. Thèse soutenue à Rouen le 11/12/2001, 206p.

[Agrawal, 1993] Agrawal, R., Imielinski, T., et Swami, A. 1993. Mining Associations between Sets of Items in Large Databases, *In Proc. of the 1993 ACM SIGMOD International Conference on Management of Data,* Washington DC, pp. 207-216.

[Aho, 1974] Aho, A.V., Hopcroft, J.E., Ullman, J.D. 1974. *The design and analysis of computer algorithms*, Addison Wesley.

[Aksoy, 2000] Aksoy, S., Ye, M., Schauf, M., Song, M., Wang, Y., Haralick, R., Parker, J., Pivovarov, J., Royko, D., Sun, C., Farneboock, G. 2000. Algorithm Performance Contest. In: *Proceedings of 15th International Conference on Pattern Recognition,* **4**, pp. 870 – 876. Barcelona, Spain.

[Alt, 1984] Alt, H., Mehlhorn, K., et Munro, J. 1984. On the Complexity of Partial Match Retrieval. *Information Processing Letters*, **19(2)**, pp. 61- 65.

[Ansary, 2005] Ansary, T. F., Vandeborre, J-P., Daoudi, M. 2005. Recherche de Modèles 3D de Piéces Mécaniques Basée sur les Moments de Zernike. Dans *Compression et Représentation des Signaux Audiovisuels (CORESA'2005).* Rennes, France.

[Arrivault, 2006] Arrivault, D., Richard, N., Bouyer, P. 2006. Proposition d'un Système de Reconnaissance Structurelle Probabiliste de Caractères. *Actes du 9ème Colloque International Francophone sur l'Ecrit et le Document,* pp. 73-78. Fribourg-Switzerland.

B

[Balas, 1986] Balas, E., Yu, C.S. 1986. Finding a Maximum Clique in an Arbitrary Graph. *SIAM Journal on Computing,* **15(4),** pp. 1054 – 1068.

[Barrat, 2006] Barrat, S., Tabbone, S. 2006. Apprentissage Progressif pour la Reconnaissance de Symboles dans les Documents Graphiques. Dans *15eme Congrès Francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle – RFIA*. Tours, France.

[Battiti, 2001] Battiti, R. et Protasi, M. 2001. Reactive Local Search for the Maximum Clique Problem. Dans *Springer-Verlag, Algorithmica*, **29**, pp. 610 – 637.

[Barbu, 2004] Barbu, E., Héroux, P., Adam, S., Trupin, E. 2004. Frequent Graph Discovery: Application to Line Drawing Document Images. *Electronic Letters on Computer Vision and Image Analysis*, **5**(2), pp. 47-57.

[Barbu, 2006] Barbu, E., Raveaux, R., Locteau, H., Adam, S., Héroux, P., Trupin, E. 2006. Classification de Graphes par Algorithmes Génétiques et Signatures de Graphes, Application à la Reconnaissance de Symboles. *CIFED, 9ème colloque international francophone sur l'écrit et le document,* pp. 91 - 96. Fribourg-Switzerland.

[Bomze, 1999] Bomze, I. M., Budinich, M., Pardalos, P. M., et Pelillo, M. 1999. The Maximum Clique Problem. Dans *Handbook of Combinatorial Optimization*, **4**, Kluwer Academic Publisher, Boston Pattern MA.

[Braun, 1995] Braun, A., Caesar, T., Gloger, J.M. et Mandler, E. 1995. Preprocessing Raw Binary Images by Means of Contours. *In International Conference on Document Analysis and Recognition (ICDAR)*, pp. 640–643.

[Brunner, 2004] Brunner, D. et Brunnett, G. 2004. Mesh Segmentation Using the Object Skeleton Graph. Dans *International Conference on Computer Graphics and Imaging*, pp. 48 – 55, Kauai, Hawaii, USA.

[Bunke, 1998] Bunke, H. et Shearer, K. 1998. A Graph Distance Metric Based on Maximal Common Subgraph. *Pattern Recognition Letters*, **19**, pp. 255 - 259.

[Bunke, 1997b] Bunke, H. 1997. On a Relation between Graph Edit Distance and Maximum Common Subgraph, *Pattern Recognition Letters*. pp. 689 - 694.

[Bunke, 2000a] Bunke, H. 2000. Recent Development in Graph Matching, *IEEE 15th International Conference on Pattern Recognition,* **2**, pp. 117 - 124.

[Bunke, 2000b] Bunke, H. 2000. Graph matching: Theoretical Foundations, Algorithms, and Applications, Dans Proc. *Vision Interface 2000*, pp. 82 - 88. Montreal.

[Bunke, 1997a] Bunke, H. et Messmer, B.T. 1997. Recent Advances in Graph Matching. *International Journal of Pattern Recognition and Artificial Intelligence*, **11**(1), pp.169-203.

[Bunke, 1999] Bunke, H. 1999. Error Correcting Graph Matching. On the Influence of the Underlying Cost Function, *IEEE Transactions on pattern analysis and machine intelligence*, **21**(9), pp. 917- 922.

C

[Chan, 1996] Chan, W.S., Chin, F. 1996. On Approximation of Polygonal Curves with Minimum Number of Line Segments or Minimum Error. *Int. J. on Comput. Geometry and Applications*, **6**, pp. 59-77.

[Champin, 2003] Champin, P. A., Solnon, C. 2003. Measuring the Similarity of Labelled Graphs, *Proceedings of 5th Int. Conf. on Case-Based Reasoning* (ICCBR 2003), LNAI, **2689**, Springer Verlag, pp. 80 – 95.

[Chen, 1998] Chen, D.Z., Daescu, O. 1998. Space Efficient Algorithm for Polygonal Curves in Two Dimensional Space. Dans le *Proc. 4th Int. Conf. Computing and Combinatorics*, pp. 55 - 64.

[Chhabra, 1998] Chhabra, A., Phillips, I.T. The 2nd International Graphics Recognition Contest— Raster to Vector Conversion: A Report. 1998. *Graphics Recognition— Algorithms and Systems, LNCS-1389*, pp. 390 – 410. Springer, Berlin Heidelberg New York.

[Conker, 1988] Conker, R.S. 1988. A Dual Plane Variation of the Hough Transform for Detecting Non-concentric Circles of Different Radii. *Computer Vision, Graphics and Image Processing*, **43**, pp. 115-132.

[Conte, 2004] Conte, D., Foggia, P., Sansone, C., et Vento, M. 2004. Thirty Years of Graph Matching in Pattern Recognition *International Journal of Pattern Recognition and Artificial Intelligence,* **18**(3), pp. 265 – 298.

[Conte, 2003] Conte, D., Guidobaldi, C., Sansone, C. 2003. A Comparison of Three Maximum Common Subgraph Algorithms on a Large Database of Labelled Graphs. Dans le *4th IAPR International Workshop on Graph Based Representation in Pattern Recognition*, LNCS-2726, pp. 130-141. York, UK.

[Cootes, 1995] Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J. 1995. Active Shape Models: Their Training and Application. *Computer Vision and Image Understanding*, **61**(1), pp. 38 – 59.

[Corneil, 1970] Corneil, D.G., Gotlie, C.C. 1970. An Efficient Algorithm for Graph Isomorphism, *Journal of the Association for Computing Machinery*, **17**, pp. 51- 64.

[Cordella, 1998] Cordella, L.P., Foggia, P., Sansone, C. et Tortorella, F. 1998. Graph Matching: A Fast Algorithm and its Evaluation. Dans le proceeding de *14th International Conference on Pattern Recognition,* Brisbane, Australia, pp. 1582-1584.

[Cordella, 2001] Cordella, L.P., Foggia, P., Sansone, C. et Vento, M. 2001. An Improved Algorithm for Matching Large Graphs, *Proc. 3rd IAPR-TC15 Workshop Graph-Based Representations in Pattern Recognition,* pp. 149 -159.

[Cordella, 2000] Cordella, L.P., et Vento, M. 2000. Symbol Recognition in Documents : A Collection of Techniques, *IJDAR*, **3**, pp. 73 - 88.

D

[Delalandre, 2005] Delalandre, M. 2005. Analyse des Documents Graphiques: Une Approche par Reconstruction d'Objets. *Thèse de Doctorat*, Mention Informatique, Université de Rouen, France.

[Delalandre, 2004] Delalandre, M., Trupin, E. et Ogier, J. M. 2004. Système de Reconnaissance Structurelle de Symboles, Basé sur une Multi Représentation en Graphes de Régions, et Exploitant une Représentation XML des Données. *Colloque International Francophone sur l'Ecrit et le Document (CIFED),* pages 177-182.

[Deutsch, 1972] Deutsch, E. 1972. Thinning Algorithms on Rectangular, Hexagonal, and Triangular Arrays, *Communications of the ACM*, **15**(9), pp. 827-837.

[Dickinson, 2001] Dickinson, S., Pelillo, M. et Zabih, R. 2001. Introduction to the Special Section on Graph Algorithms in Computer Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**(10), pp.1049 – 1052.

[Dijkstra, 1959] Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Matematik*, **1**, pp. 269 - 271.

[Dosch, 2004] Dosch, P., Lladós, J. 2004. Vectorial Signatures for Symbol Discrimination. *Selected papers from GREC'03, LNCS 3088, Springer Verlag,* pp. 154 -165.

[Dosch, 2006] Dosch, P., Valveny, E. 2006. Report on the Second Symbol Recognition Contest Dans *Sixth IAPR International Workshop on Graphics Recognition (GREC'05), LNCS 3926*, pp. 381-397.

[Dosch, 2000] Dosch, P., Tombre, K., Ah-Soon, C., Masini, G. 2000. A Complete System for the Analysis of Architectural Drawings. *International Journal on Document Analysis and Recognition*, **3**(2), pp.102 – 116.

[Dori, 1997] Dori, D. 1997. Orthogonal Zig-Zag: An Algorithm for Vectorizing Engineering Drawings Compared with Hough Transform. *Advances in Engineering Softwares*, **28**(1), pp. 11-24.

[Durand, 1999] Durand, P.J., Pasari, R., Baker, J. W., et Tsai, C. C. 1999. An Efficient Algorithm for Similarity Analysis of Molecules. *Internet Journal of Chemistry*, **2**, 1999.

E

[Eshera, 1986] Eshera, M.A. et Fu, K-S. 1986. An Image Understanding System using Attributed Symbolic Representation and Inexact Graph Matching. *PAMI*, **8**, pp. 604 - 618.

[Etemadi, 1991] Etemadi, A., Schmidt, J.P., Matas, G., Illingworth, J., Kittler, J. 1991. Low-level Grouping of Straight Line Segments. Dans: *Proceeding of Second British Machine Vision Conference,* pp.118 - 126. Glassgow, Scotland.

F

[Falkenhainer, 1989] Falkenhainer, B., Forus, K. D., et Gentner, D. 1989. The Structure-Mapping Engine: Algorithms and Examples. *Artificial Intelligence*, **41**(1), pp. 1 - 63.

[Fletcher, 1988] Fletcher, L.A. et Kasturi, R. 1988. A Robust Algorithm for Textbox String Separation from Mixed Text/Graphics Images. *IEEE Trans. PAMI*, **10**(6), pp. 900-918.

[Foggia, 2001] Foggia, P., Sansone, C., et Vento, M. 2001. A Performance Comparison of Five Algorithms for Graph Isomorphism. In *3rd IAPR-TC15 Workshop on Graph-Based Representations in Pattern Recognition*, pp. 188 – 199. Cuen.

[Foggia, 1999] Foggia P., Sansone C., Tortorella F. et Vento, M. 1999. Definition and Validation of a Distance Measure Between Structural Primitives. *Pattern Analysis and Applications*, **2**(3), pp. 215-227.

[Fonseca, 2004] Fonseca, M. J., Barroso, B., Ribeiro, P. et Jorge, J. A. 2004. Retrieving Vector Graphics Using Sketches. *In Symposium on Smart Graphics (SG), LNCS 3031,* pp. 66–76.

[Franco, 2003] Franco, P., Ogier, J-M., Loonis, P., et Mullot, R. 2003. A Topological Measure for Image Object Recognition, *GREC-2003, LNCS3088*, pp. 279 – 290.

G

[Garnesson, 1991] Garnesson, P., et Giraudon, G. 1991. Polygonal Approximation - Overview and Perspectives. *Technical report, Institut National de Recherche en Informatique et en Automatique.*

[Garey, 1979] Garey, M.R. et Johnson, D.S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, page. 194, San Francisco, Freeman and Company.

[Geusebroek, 1999] Geusebroek, J., Smeulders, A., Cornelissen, F. et Geerts, H. 1999. Segmentation of Tissue Architecture by Distance Graph Matching, *Cytometry*, **35**(1), pp. 11 - 22.

[Gorman, 1984] Gorman, L. O., Sanderson, A.C. 1984. The Converging Squares Algorithm: An Efficient Method for Locating Peaks in Multi-dimensions. *IEEE Trans. On PAMI*, **6**, pp. 280-288.

[Gonzalez, 1992] Gonzalez, R., Woods, R. 1992. *Digital Image Processing*, Addison Wesley.

[GREC 03] GREC-2003, 2003. *Fifth IAPR International Workshop on Graphics Recognition* Computer Vision Center, Barcelona, Catalonia, Spain, July 30-31.

[GREC 97] Proceedings of Second IAPR Workshop on Graphics Recognition, Nancy, France, August 1997.

[GREC 95] Kasturi, R. et Tombre, K. editors. *Graphics Recognition: Methods and Applications, First International Workshop, University Park, PA, USA, August 1995, Selected Papers*, *Springer LNCS,* **1072**, 1996. Berlin.

[Guillas 06] Guillas, S., Bertet, K., Ogier, J.-M. 2006. Reconnaissance de Symboles Bruités à l'Aide d'un Treillis de Galois. *Actes du 9ème Colloque International Francophone sur l'Ecrit et le Document,* pp. 85 – 90. Fribourg-Switzerland.

H

[Hallard, 1981] Hallard, D.H. 1981. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, **13**(2), pp. 111-122.

[Han, 94] Han, C., Fan. K.1994. Skeleton Generation of Engineering Drawings via Contour Matching. *Pattern Recognition*, **27**(2), pp. 261- 275.

[Haralick, 1992] Haralick, R.M., Shapiro, L. 1992. Computer and Robot Vision. Addison Wesley, Reading, MA

[Haris, 1998] Haris, K., Efstratiadis, S. N., Maglaveras, N., et Katsaggelos, A. K. 1998. Hybrid Image Segmentation Using Watersheds and Fast Region Merging. *IEEE Transactions on Image Processing*, **7**(12), pp.1684 -1699.

[Hasan, 2000] Hasan, M.Y., et Karan, L.J. 2000. Morphological Reversible Contour Representation. *Pattern Analysis and Machine Intelligence (PAMI)*, **22**(3), pp. 227–239.

[Hlaoui, 2003] Hlaoui, A. et Wang, S. 2003. Graph Representation in Content-Based Image Retrieval. *In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2003).* Tunisia, Tunisia.

[Hopcroft, 1974] Hopcroft, J. E. et Wong, J. K. 1974. Linear Time Algorithm for Isomorphism of Planar Graphs. *In Proceedings 6th ACM Symposium on Theory of Computing,* pp. 172 -184.

[Hori, 1993] Hori, O., Hori, O., Tanigawa, S. 1993. Raster-to-vector Conversion by Line Fitting Based on Contours and Skeletons. Dans *proceeding: of ICDAR'93*, pp. 353 – 358. Tsukuba, Japan.

[Horng, 2002] Horng, J.-H., Li, J.T. 2002. An Automatic and Efficient Dynamic Programming Algorithm for Polygonal Approximation of Digital Curves. *Pattern Recognition Letters*, **23**, pp. 171-182.

[Hse, 2004] Hse, H., Newton, A.R. 2004. Sketched Symbol Recognition Using Zernike Moments. Dans *Proceedings: of the 17th International Conference on Pattern Recognition (ICPR'04)*, **1,** pp. 367 – 370.

[Hunt, 1988] Hunt, D.J., Nolte, L.W. 1988. Performance of the Hough Transform and its Relationship to Statistical Signal Detection Theory. *Computer Vision, Graphics, and Image Processing*, **43**, pp. 221-238.

[Huang, 1998] Huang, C. et Huang, W. 1998. Sign Language Recognition Using Model-Based Tracking and a 3D Hopfield Neural Network. *Machine Vision and Applications,* **10,** pp. 292-307.

I

[Illingworth, 1987] Illingworth, J., Kittler, J., 1987. The Adaptive Hough Transform. *IEEE Trans. on PAMI*, **9**(5), pp. 690-697.

[Iváncsy, 2004] Iváncsy, G., Iváncsy, R., et Vajk. I. 2004. Graph mining-based Image Indexing. *5th International Symposium of Hungarian Researchers on Computational Intelligence*, Budapest.

J

[Jiang, 1993] Jiang, X.Y. et Bunke, H. 1993. An Optimal Algorithm for Extracting the Regions of a Plane Graph. *Pattern Recognition Letters*, **14** (7), pp. 553-558.

[Jiang, 1999] Jiang, X. Bunke, H. 1999. Optimal Quadratic-Time Isomorphism of Ordered Graphs, *Pattern Recognition*, **32**, pp. 1273-1283.

[Jimenez, 1982] Jimenez, J., Navalon, J. L. 1982. Some Experiments in Image Vectorization. *IBM Journal of research and developments*, **26**, pp. 724-734.

[Jordan, 1997] Le Buhan Jordan, C. et Ebrahimi, T. 1997. Progressive Polygon Encoding of Shape Contours. *In Conference on Image Processing and its Applications (IPA),* pp. 17–21.

K

[Kanungo, 1994] Kanungo, T., Haralick, R.M., Baird, H.S., Stuetzle, W., Madigan, D. 1994. Document Degradation Models: Parameter Estimation and Model Validation. Dans: *Proceedings of IAPR Workshop on Machine Vision Applications*, pp. 552 - 557. Kawasaki, Japan.

[Kaufman, 1990] Kaufman, L., et Rousseeuw P.J. 1990. Finding Groups in Data. *John Wiley & Sons, Inc.* New York, USA.

[Kelly, 2002] Kelly, A. R. et Hancock, E. R. 2002. String Edit Distance, Random Walks and Graph Matching. Dans *Structural, Syntactic and Statistical Pattern Recognition, LNCS 2396*, pp.104 -112.

[Kelly, 2003] Kelly, A. R., Hancock, E. R. 2003. Graph Matching Using Spectral Seriation and String Edit Distance. *In the 4th IAPR International Workshop on Graph Based Representation in Pattern Recognition, LNCS 2726,* pp.154 -165. York, UK.

[Khotazad, 1990] Khotazad, A., et Hong, Y. H. 1990. Invariant Image Recognition by Zernike Moments, *IEEE PAMI*, **12**(5), pp. 489 - 497.

[Kimme, 1975] Kimme, C., Ballard, D.H., Sklansky, J. 1975. Finding Circles by an array of accumulators. *Communications of the ACM*, **18**, pp.120-122.

[Kim, 1999] Kim, W.-Y., Kim, Y.-S. 1999. A New Region-Based Shape Descriptor, *TR 15-01*, Pisa, Italy.

[Koch, 2001] Koch, I. 2001. Enumerating All Connected Maximal Common Sub-graphs in Two Graphs, *Theoret. Comput. Sci.* pp. 1-30.

[Kotropoulos, 2000] Kotropoulos, C., Tefas, A., et Pitas, I. 2000. Frontal Face Authentication using Morphological Elastic Graph Matching. *IEEE Transactions on Image Processing,* **9**(4), pp. 555 - 560.

L

[Labelle, 1998] Labelle, L., Lauzon, D., Konrad, J. et Dubois, E. 1998. Arithmetic Coding of a Lossless Contour-based Representation of Label Images. *In InternationalConference on Image Processing (ICIP)*, pp. 261–265.

[Lam, 1992] Lam, L., Lee, S. W., Suen, C. Y. 1992. Thinning Methodologies-A Comprehensive Survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence,* **14**, pp**.** 869 – 885.

[Liu, 1998] Liu, W. et Dori, D. 1998. A Generic Integrated Line Detection Algorithm and its Object-Process Specification. *Computer Vision and Image Understanding,* **70**(3), pp. 420 – 437.

[Liu, 2004] Liu, C.-L., Jaeger, S., et Nakagawa, M. 2004. Online Recognition of Chinese Characters: The state-of-the-Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(2), pp.198 - 213.

[Lladós, 1997] Lladós, J. 1997. Combining Graph Matching and Hough Transform for Hand-Drawn Graphical Document Analysis. Application to Architectural Drawings. *PhD thesis.*

[Lladós, 2001a] Lladós, J., Martí, E., Villanueva, J. J. 2001. Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **23**(10), pp.1137 – 1143.

[Lladós, 2001b] Lladós, J., Valveny, E., Sánchez, G., Martí, E. 2001. Symbol Recognition: Current Advances and Perspectives. The *4th International Workshop, GREC 2001*, *Selected Papers, LNCS,* **2390**, pp. 104-127, Ontario-Canada.

[Lladós, 2003] Lladós, J., Sánchez, G. 2003. Graph Matching Vs. Graph Parsing in Graphics Recognition. A Combined Approach. *International Journal of Pattern Recognition and Artificial Intelligence*, **18**(3), pp.455 – 473.

[Locteau, 2006] Locteau, H., Adam, S., Trupin, E., Labiche, J., and Héroux, P. 2006. Reconnaissance de Symbole Guidée par une Modélisation Basée sur les Graphes de Régions Adjacentes. Dans *Colloque International Francophone sur l'Écrit et le Document*, pp. 151-156. Fribourg, Suisse.

[Lopresti, 2001] Lopresti, D., Wilfong, G. 2001. Applications of Graph Probing to Web Document Analysis. Dans *proceedings of the First International Workshop on Web Document Analysis,* pp. 51– 54. Seattle, WA.

[Luo, 2003] Luo, Y., et Liu, W. 2003. Interactive Recognition of Graphic Objects in Engineering Drawings, Graphics Recognition – Recent Advances and Perspectives, *Springer-LNCS,* **3088**, pp.128-141.

M

[McGregor, 1982] McGregor, J.J. 1982. Backtrack Search Algorithms and the Maximal Common Subgraph Problem. *Software Practice and Experience*, **12**, pp. 23 – 34.

[McKay, 1981] McKay, B.D. 1981. Practical Graph Isomorphism. Congressus Numerantium, **30***,* pp. 45 – 87.

[McKay, 1978] McKay, B.D. 1978. Computing Automorphisms and Canonical Labellings of Graphs. *Combinatorial Mathematics,* **686***, Springer Lecture Notes in Mathematics*, pp. 223-232.

[Merad, 2004] Merad, D. 2004. Reconnaissance 2D/2D et 2D/3D d'objets à partir de leurs squelettes, *Thèse*, 2004.

[Messmer, 1999] Messmer, B.T., et Bunke, H. 1999. A Decision Tree Approach to Graph and Subgraph Isomorphism Detection, *Pattern Recognition*, **32**, pp. 1979 – 1998.

[Messmer, 1995] Messmer, B.T. 1995. Efficient Graph Matching Algorithms, *PhD thesis*, Univ. of Bern, Switzerland.

[Messmer, 1998b] Messmer, B.T., et Bunke, H. 1998. Error-Correcting Graph Isomorphism using Decision Trees. *Int. Journal of Pattern Recognition and Art. Intelligence,* **12**, pp. 721–742.

[Messmer, 1996] Messmer, B.T., Bunke, H. 1996. Automatic Learning and Recognition of Graphical Symbols in Engineering Drawings. *Graphics Recognition: Methods And Applications (Selected Papers from the Proceedings of GREC'95), LNCS,* **1072***,* pp. 123-134.

N

[Nguyen, 2005] Nguyen, T., Gagalowicz, A., Philips, W. 2005. A Linear Algorithm for Polygonal Approximations of Thick Curves, *CAIP 2005 (Computer Analysis Of Images And Patterns),* **3691**, pp. 17-25. Versailles, France.

O

[Ogier, 1994] Ogier, J.M., Mullot, R., Labiche, J., Lecourtier, Y. 1994. *Intégration d'outils bas niveau dans une stratégie d'interprétation de documents. 9e Congrès Reconnaissance de Formes et Intelligence Artificielle*, **1**, pp. 533-544. Paris, France.

P

[Papadopoulos, 1998] Papadopoulos, A.N., et Manolopoulos, Y. 1998. Structure-based Similarity Search with Graph Histograms. Dans *Proceedings of the 10th International Workshop on Database & Expert System Applications. IEEE Computer Society Press,* pp. 174-178.

[Paquet, 1990] Paquet, T., Vallee, T., Lecourtier, Y. 1990. Extraction de primitives par suivi de traits dans l'image binarisée d'un mot manuscrit. *Colloque National sur l'Ecrit et le Document,* Bigre, **68**, pp. 179-188.

[Paquet, 1991] Paquet, T., Lecourtier, Y. 1991. Reconnaissance de l'écriture manuscrite sur des chèques. *In : Proceedings du 8e Congrès Reconnaissance des Formes et Intelligence Artificielle*, Lyon-Villeurbanne (France), 25-29 novembre, **2**, pp. 687-693.

[Pavliedis, 1972] Pavliedis, T. Representation of Figures by Labelled Graphs. 1972. *Pattern Recognition,* **4**, pp. 5 – 17.

[Pelillo, 1999] Pelillo, M., Siddiqi, K., et Zucker, S. W. 1999. Matching Heirarchical Structures using Associate Graphs. *IEEE Trans. Pattern Analysis and Machine Intelligence,* **21**(11), pp.1105 - 1120.

[Perez, 1994] Perez, J.C., Vidal, E. 1994. Optimum Polygonal Approximation of Digitized Curves. *Pattern Recognition Letters*, **15**, pp. 743-750.

[Pham, 2005] Pham, T.-T. 2005. Méthode de Mise en Correspondance Hiérarchique en Reconnaissance d'Objets, *Rapport Master 2 recherche.* Grenoble, France.

Q

[Qureshi, 2007] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2007. Symbol Spotting in Graphical Documents Using Graph Representations, *Seventh IAPR International Workshop on Graphics Recognition - GREC 2007.* Curitiba, Brazil. September 20-21, 2007.

[Qureshi, 2007] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2007. Graph Based Shapes Representation and Recognition, Dans *le 6th IAPR Workshop on Graph-based Representations in Pattern Recognition, GbRPR 2007*, *LNCS*, **4538**, pp.49-60. Alicante, Spain. June 11-13, 2007

[Qureshi, 2007] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2007. Graph Signature: A Simple Approach for Clustering Similar Graph, *Pascal workshop: learning from and with graph.,* Alicante, Spain. June 14th, 2007

[Qureshi, 2007] Qureshi, R.J., Ramel, J-Y., Mukherji, P., Cardot, H. 2007. Combination of Symbolic and Statistical Features for Symbols Recognition, *IEEE International Conference on Signal Processing, Communications and Networking*, pp. 477-482. Chennai, India February 22-24, 2007.

[Qureshi, 2006a] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2006. Graphic Symbol recognition using flexible matching of attributed relational graphs, Dans le proceeding du *6th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP),* pp. 383-388. Palma de Mallorca, Spain. August 28-30, 2006.

[Qureshi, 2006b] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2006. De l'appariement de graphes symboliques à l'appariement de graphes numériques: Application à la reconnaissance de symboles, *Neuvième Colloque International Francophone sur l'écrit et le Document,* pp. 31-36. *Fribourg, Suisse*, Septembre 18-21, 2006. **(Prix du meilleur article)**

[Qureshi, 2006c] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2006. A novel similarity measure for the recognition of degraded graphic symbols, *10th Annual Meeting on Health, Science & Technology,* page-39*, l'U.F.R. Sciences et Techniques* – Site de Grandmont, Tours-France. mai 23-24, 2006.

[Qureshi, 2006d] Qureshi, R.J., Ramel, J-Y., Cardot, H. 2006. Reconnaissance de symboles graphiques à partir de graphes étiquetés, *MajecStic (MAnifestation des Jeunes Chercheurs STIC)*, à Palais des Congrès, Lorient – France. Novembre 22-24, 2006

[Qureshi, 2005] Qureshi, R.J., Ramel, J-Y., Ali, U., Cardot, H. 2005. Graph Based Recognition of Isolated Graphic Symbols, *IEEE International Conference on Emerging Technologies (ICET'05),* pp. 109-114, Islamabad, Pakistan.17-18 September, 2005.

R

[Ramel, 1992] Ramel, J.Y. 1992. Lecture Automatique de Partitions Musicales. *Mémoire de DEA Ingénierie Informatique.*

[Ramel, 1996] Ramel, J.Y. 1996. Une Nouvelle Méthode pour l'Indexation Automatique de Dessins : Application aux Plans Cinématiques. Thèse de Doctorat, Institut National Polytechnique de Lyon, France.

[Ramel, 2000] Ramel, J.Y., Vincent, N., Emptoz, H. 2000. A Structural Representation for Understanding Line-Drawing Images. *International Journal on Document Analysis and Recognition*, **3**(2), pp. 58 – 66.

[Raveaux, 2007] Raveaux, R., Eugen, B., Locteau, H., Adam, S., Héroux, P., et Trupin, E. : A Graph Classification Approach Using a Multi-objective Genetic Algorithm Application to Symbol Recognition. *Dans le 6th International Workshop on Graph-Based Representations in Pattern Recognition, LNCS*, **4538**, pp. 361-370. Alicante, Spain.

[Raymond, 2002] Raymond, J. W., et Willett, P. 2002. Maximum Common Subgraph Isomorphism Algorithms for the Matching of Chemical Structures, *Journal of Computer-Aided Molecular Design*, **16**, pp. 521–533.

[Risse, 1989] Risse, T. 1989. Hough Transformation for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection. *CVGIP*, **46**(3), pp. 327-345.

[Riviere, 2002] Riviere, D., Mangin, J., Papadopoulos, D., Martinez, J., Frouin, V. et Regis, J. 2002. Automatic Recognition of Cortical Sulci of the Human Brain Using a Congregation of Neural Networks, *Medical Image Analysis,* **6**(2), pp. 77 – 92.

[Rosenfeld, 1982] Rosenfeld, A. et Kak. A.C. 1982. Digital picture processing. Academic Press, **2** edition, ISBN : 0125973020.

[Rosenfeld, 1968] Rosenfeld, A., et Pfaltz, J. L. 1968. Distance functions in digital pictures. *Pattern Recognition,* **1**, pp.33 – 61.

[Rosin, 1989] Rosin, P. L., et West, G. A. 1989. Segmentation of Edges into Lines and Arcs. *Image and Vision Computing*, **7**(2), pp. 109–114.

[Ruberto, 2002] Ruberto, C. D., Rodriguez, G., Casta, L. Recognition of Shapes by Morphological Attributed Relational Graphs, URL = citeseer.ist.psu.edu/535355.html

S

[Seong, 1993] Seong, D.S., Kim, H.S. Park, K.H. 1993. Incremental Clustering of Attributed Graphs. *Transactions on Systems, Man and Cybernetics (TSMC)*, **23**(5), pp.1399-1411.

[Shinano, 1998] Shinano, Y., Fujie, T., Ikebe, Y. et Hirabayashi, R. 1998. Solving the Maximum Clique Problem using PUBB, *in Proc. First Merged Int. Parallel Processing Symposium,* pp. 326 -332.

[Song, 2002a] Song, J., Su, F., Tai, M., Cai, S. 2002. An Object-Oriented Progressive-Simplification-Based Vectorization System for Engineering Drawings: Model, Algorithm, and Performance. In *IEEE Transaction on Pattern Analysis and Machine Intelligence,* **24**(8), pp.1048-1060.

[Song, 2002b] Song, J., Cai, M., Lyu, M.-R., Cai, Shijie. 2002. Graphics Recognition from Binary Images: One Step or Two Steps, *16th International Conference on Pattern Recognition (ICPR'02)*, **3**, pp.135-138.

[Sonbaty, 1998] Sonbaty, Y.-El., et Ismail, M.A. 1998. A New Algorithm for Subgraph Optimal Isomorphism, *Pattern Recognition,* **31**(2), pp. 205-218.

[Soon, 1997] Soon, C.-Ah. 1997. A Constraint Network for Symbol Detection in Architectural Drawings. *Graphics Recognition: Algorithms and Systems, GREC'97 LNCS 1389,* pp. 80- 90. Nancy, France.

[Sorlin, 2004] Sorlin, S., et Solnon, C. 2004. A Global Constraint for Graph Isomorphism Problems. In *the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR 2004). LNCS,* **3011**, pp. 287-302.

[Sorlin, 2005a] Sorlin, S., Solnon, C. 2005. Reactive Tabu Search for Measuring Graph Similarity. *5th IAPR Workshop on Graph-based Representations in Pattern Recognition*, *LNCS – 3434, Springer Verlag*, pp.172-182.

[Sorlin, 2005b] Sorlin, S., Solnon, C. 2005. Similarité de Graphes : Une Mesure Générique et un Algorithme Tabou Réactif. Dans *7ème Rencontres Jeunes Chercheurs en intelligence artificielle.*

[Stéphane, 2006] Stéphane, Z., Deville, Y. et Dupont, P. 2006. Elimination des Symétries pour l'appariement de Graphes. *JFPC'2006, Deuxièmes Journées Francophones de Programmation par Contraintes*, pp. 357 – 367. Nîmes, France.

[Suganthan, 1998] Suganthan, P. et Yan, H. 1998. Recognition of Hand Printed Chinese Characters by Constrained Graph Matching. *Image and Vision Computing*, **16**(3), pp.191 – 201.

T

[Tabbone, 2003] Tabbone, S., Wendling, L., Tombre, K. 2003. Matching of Graphical Symbols in Line-Drawing Images using Angular Signature Information, *IJDAR,* **6**, pp.115-125.

[Tabbone, 2005] Tabbone, S. 2005. Quelques Contributions à la Reconnaissance de Formes dans des Documents Graphiques, *Memoire-HdR*.

[Terrades, 2006] Terrades, O.-R. 2006. Linear Combination of Multi-resolution Descriptors: Application to graphic recognition, *PhD thesis*.

[TOM, 2000a] Tombre, K., Tabonne. S., Vectorization in Graphics Recognition: To Thin or Not to Thin. 2000. Dans *Proceedings of 15th International Conference on Pattern Recognition*, **2,** pp. 91 - 96. Barcelona, Spain.

[TOM, 2000b] Tombre, K., Ah-Soon, C., Dosch, P., Masini, G., et Tabbone, S. 2000. Stable and Robust Vectorization: How to Make the Right Choices. *Graphics recognition-Recent advances*, *Springer-Verlag LNCS,* **1941**, pp. 3-18. Berlin.

[Tombre, 1998] Tombre, K. 1998. Ten Years of Research in the Analysis of Graphics Documents: Achievements and Open Problems. Dans le *Proceedings of the 10th Portuguese Conference on Pattern Recognition,* pp.11-17. Lisbon, Portugal.

[Tombre, 1997] Tombre, K. 1997. Analysis of Engineering Drawings: State of the Art and Challenges. *In the Proceeding of the 2nd International Conference of Graphics Recognition*, pp. 54-61, Nancy-France.

[Turner, 1996] Turner, M.J. et Wiseman, N.E. 1996. Efficient Lossless Image Contour Coding. Computer Graphics Forum, **15(**2), pp.107–118.

U

[Ullman, 1976] Ullman, J. R. An Algorithm for Subgraph Isomorphism. *Journal of the Association for Computing Machinery*, **23**(1), pp. 31- 42.

[Umeyama, 1988] Umeyama, S. An Eigen Decomposition Approach to Weighted Graph Matching Problems. *IEEE PAMI*, **10**, pp. 695 – 703.

V

[Valveny, 2007] Valveny, E., Dosch, P., Winstanley, A., Zhou, Y., Yang, S., Yan, L., Wenyin, L., Elliman, D., Delalandre, M., Trupin, E., Adam, S., Ogier, J-M. 2007. A General Framework for the Evaluation of Symbol Recognition Methods, *IJDAR,* **9,** pp.59 –74.

[Ventura, 1994] Ventura, A. D., Schettini, R. 1994. Graphic Symbol Recognition using a Signature Technique. *12th International Conference on Pattern Recognition*, **2**, pp. 533 - 535. Jerusalem (Israel).

W

[Wall, 1984] Wall, K., et Danielsson, P. A 1984. Fast Sequential Method for Polygonal Approximation of Digitized Curves. *Computer Vision, Graphics and Image Processing*, **28**, pp. 220–227.

[Wagner, 1974] Wagner, R. et Fischer, M. 1974. The String-to-String Correction Problem. *Journal of the Association for Computing Machinery,* **21**(1), pp.168-173.

[Wang, 2006] Wang, Y., Phillips, I.T., Haralick, R.M. 2006. Document Zone Content Classification and its Performance Evaluation. *Pattern Recognition,* **39**, pp. 57 - 73.

[Wang, 2007] Wang, H.-F. et Hancock, E.-R. 2007. Probabilistic Relaxation Labeling by Fokker-Planck Diffusion on a Graph. Dans *le 6th International Workshop on Graph-Based Representations in Pattern Recognition*, *LNCS*, **4538**, pp. 204 – 214. Alicante, Spain.

[Weindorf, 2002] Weindorf, M. 2002. Structure Based Interpretation of Unstructured Vector Maps. *Workshop on Graphics Recognition (GREC), LNCS,* **2390**, pp.190 –199.

[Wendling, 2002] Wendling, L., Tabbone, A., Matsakis, P. 2002. Fast and robust recognition of orbit and sinus drawings using histograms of forces. *Pattern Recognition Letters*, **23**(14), pp.1687-1693.

[Wenyin, 2007] Wenyin, L., Zhang, W., Yan, L. 2007. An Interactive Example Driven Approach to Graphics Recognition in Engineering Drawings, *IJDAR*, **9**, pp.13 – 29.

[Web-1] "http://xmailserver.org/davide.html". Ras2Vec 1.2, D. Libenzi, a freeware for vectorization

[Web-2] http://www.gupro.de/GXL/

Z

[Zhang, 2005] Zhang, D., Lu, G. 2005. Study and Evaluation of Different Fourier Methods for Image Retrieval. *Image, Vision and Computer*, **23**(1), pp. 33-49.

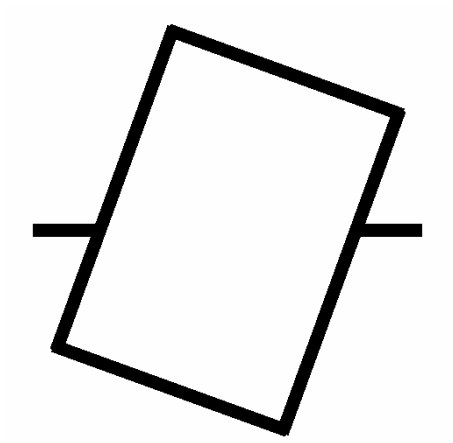
[Zhang, 1998] Zhang, C., Murai, S. et Baltsavias, E. 1998. Road Network Detection by Mathematical Morphology. *In Asian Conference on Remote Sensing (ACRS),* pp. 185–200.

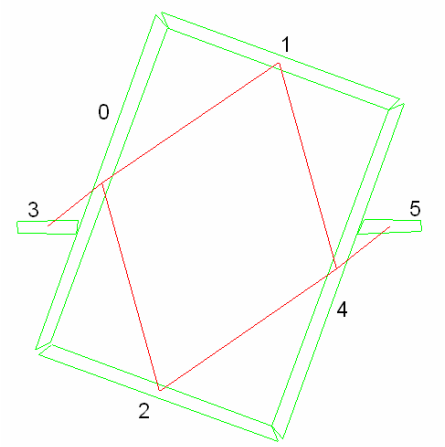
[Zhang, 1984] Zhang, T., Suen, C. 1984. A Fast Parallel Algorithm for Thinning Digital Patterns, *Communications of the ACM*, **27**(3), pp. 236-240.

[Zhang, 1994] Zhang, Y., Wang, P. 1994. A New Parallel Thinning Methodology, *International Journal of Pattern Recognition and Artificial Intelligence*, **8**, pp. 999-1011.

[Zuwala, 2006] Zuwala, D. et Tabbone, S. 2006. Une Méthode de Localisation et de Reconnaissance de Symboles sans Connaissance a Priori. Dans le *Colloque International Francophone Sur l'Ecrit Et Le Document - CIFED'06,* pp. 127-131.

Annexe – I : Un exemple de représentation d'un graphe sous le format GXL.






```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<gxl>
 <graph id="ElectricalD.gxl">
     <node id="node0">
       <attr name="forme">
         <string>1</string>
       </attr>
       <attr name="RelLen">
         <string>1.00</string>
       </attr>
       <attr name="x1i">
         <string>441</string>
       </attr>
       <attr name="y1i">
         <string>129</string>
       </attr>
       <attr name="x1f">
         <string>313</string>
       </attr>
       <attr name="y1f">
         <string>475</string>
       </attr>
       <attr name="x2i">
         <string>325</string>
       </attr>
       <attr name="y2i">
         <string>489</string>
       </attr>
       <attr name="x2f">
         <string>458</string>
       </attr>
       <attr name="y2f">
         <string>122</string>
       </attr>
       <attr name="x1">
         <string>449</string>
       </attr>
       <attr name="y1">
         <string>125</string>
```

```
  </attr>
  <attr name="x2">
    <string>319</string>
  </attr>
  <attr name="y2">
    <string>482</string>
  </attr>
  <attr name="Thickness1">
    <string>14.00</string>
  </attr>
  <attr name="Thickness1">
    <string>15.00</string>
  </attr>
  <attr name="AngleVect1">
    <string>72.00</string>
  </attr>
  <attr name="AngleVect2">
    <string>73.00</string>
  </attr>
</node>


<node id="node1">
  <attr name="forme">
    <string>1</string>
  </attr>
  <attr name="RelLen">
    <string>0.73</string>
  </attr>
  <attr name="x1i">
    <string>313</string>
  </attr>
  <attr name="y1i">
    <string>475</string>
  </attr>
  <attr name="x1f">
    <string>72</string>
  </attr>
  <attr name="y1f">
    <string>386</string>
  </attr>
  <attr name="x2i">
    <string>58</string>
  </attr>
  <attr name="y2i">
    <string>397</string>
  </attr>
  <attr name="x2f">
    <string>320</string>
  </attr>
  <attr name="y2f">
    <string>492</string>
  </attr>
  <attr name="x1">
    <string>316</string>
  </attr>
  <attr name="y1">
    <string>483</string>
  </attr>
  <attr name="x2">
    <string>65</string>
```

```
  </attr>
  <attr name="y2">
    <string>391</string>
  </attr>
  <attr name="Thickness1">
    <string>16.00</string>
  </attr>
  <attr name="Thickness1">
    <string>14.00</string>
  </attr>
  <attr name="AngleVect1">
    <string>158.00</string>
  </attr>
  <attr name="AngleVect2">
    <string>159.00</string>
  </attr>
</node>


<node id="node2">
  <attr name="forme">
    <string>1</string>
  </attr>
  <attr name="RelLen">
    <string>0.73</string>
  </attr>
  <attr name="x1i">
    <string>453</string>
  </attr>
  <attr name="y1i">
    <string>117</string>
  </attr>
  <attr name="x1f">
    <string>192</string>
  </attr>
  <attr name="y1f">
    <string>22</string>
  </attr>
  <attr name="x2i">
    <string>200</string>
  </attr>
  <attr name="y2i">
    <string>40</string>
  </attr>
  <attr name="x2f">
    <string>441</string>
  </attr>
  <attr name="y2f">
    <string>129</string>
  </attr>
  <attr name="x1">
    <string>447</string>
  </attr>
  <attr name="y1">
    <string>123</string>
  </attr>
  <attr name="x2">
    <string>196</string>
  </attr>
  <attr name="y2">
    <string>31</string>
```

```
  </attr>
  <attr name="Thickness1">
    <string>15.00</string>
  </attr>
  <attr name="Thickness1">
    <string>16.00</string>
  </attr>
  <attr name="AngleVect1">
    <string>159.00</string>
  </attr>
  <attr name="AngleVect2">
    <string>158.00</string>
  </attr>
</node>


<node id="node3">
  <attr name="forme">
    <string>1</string>
  </attr>
  <attr name="RelLen">
    <string>0.19</string>
  </attr>
  <attr name="x1i">
    <string>476</string>
  </attr>
  <attr name="y1i">
    <string>250</string>
  </attr>
  <attr name="x1f">
    <string>414</string>
  </attr>
  <attr name="y1f">
    <string>249</string>
  </attr>
  <attr name="x2i">
    <string>407</string>
  </attr>
  <attr name="y2i">
    <string>265</string>
  </attr>
  <attr name="x2f">
    <string>477</string>
  </attr>
  <attr name="y2f">
    <string>262</string>
  </attr>
  <attr name="x1">
    <string>476</string>
  </attr>
  <attr name="y1">
    <string>256</string>
  </attr>
  <attr name="x2">
    <string>410</string>
  </attr>
  <attr name="y2">
    <string>257</string>
  </attr>
  <attr name="Thickness1">
    <string>17.00</string>
```

```
  </attr>
  <attr name="Thickness1">
    <string>13.00</string>
  </attr>
  <attr name="AngleVect1">
    <string>179.00</string>
  </attr>
  <attr name="AngleVect2">
    <string>178.00</string>
  </attr>
</node>


<node id="node4">
  <attr name="forme">
    <string>1</string>
  </attr>
  <attr name="RelLen">
    <string>1.00</string>
  </attr>
  <attr name="x1i">
    <string>71</string>
  </attr>
  <attr name="y1i">
    <string>383</string>
  </attr>
  <attr name="x1f">
    <string>199</string>
  </attr>
  <attr name="y1f">
    <string>39</string>
  </attr>
  <attr name="x2i">
    <string>186</string>
  </attr>
  <attr name="y2i">
    <string>25</string>
  </attr>
  <attr name="x2f">
    <string>54</string>
  </attr>
  <attr name="y2f">
    <string>392</string>
  </attr>
  <attr name="x1">
    <string>62</string>
  </attr>
  <attr name="y1">
    <string>387</string>
  </attr>
  <attr name="x2">
    <string>192</string>
  </attr>
  <attr name="y2">
    <string>32</string>
  </attr>
  <attr name="Thickness1">
    <string>16.00</string>
  </attr>
  <attr name="Thickness1">
    <string>17.00</string>
```

```
  </attr>
  <attr name="AngleVect1">
    <string>72.00</string>
  </attr>
  <attr name="AngleVect2">
    <string>73.00</string>
  </attr>
</node>

<node id="node5">
  <attr name="forme">
    <string>1</string>
  </attr>
  <attr name="RelLen">
    <string>0.18</string>
  </attr>
  <attr name="x1i">
    <string>101</string>
  </attr>
  <attr name="y1i">
    <string>250</string>
  </attr>
  <attr name="x1f">
    <string>34</string>
  </attr>
  <attr name="y1f">
    <string>251</string>
  </attr>
  <attr name="x2i">
    <string>35</string>
  </attr>
  <attr name="y2i">
    <string>264</string>
  </attr>
  <attr name="x2f">
    <string>99</string>
  </attr>
  <attr name="y2f">
    <string>265</string>
  </attr>
  <attr name="x1">
    <string>100</string>
  </attr>
  <attr name="y1">
    <string>257</string>
  </attr>
  <attr name="x2">
    <string>34</string>
  </attr>
  <attr name="y2">
    <string>257</string>
  </attr>
  <attr name="Thickness1">
    <string>14.00</string>
  </attr>
  <attr name="Thickness1">
    <string>16.00</string>
  </attr>
  <attr name="AngleVect1">
    <string>0.00</string>
  </attr>
```

```
      <attr name="AngleVect2">
        <string>1.00</string>
      </attr>
    </node>

    <edge id="edge0" from="node0" to="node1">
      <attr name="relA">
        <string>86</string>
      </attr>
      <attr name="Type">
        <string>L</string>
      </attr>
    </edge>

    <edge id="edge1" from="node0" to="node2">
      <attr name="relA">
        <string>86</string>
      </attr>
      <attr name="Type">
        <string>L</string>
      </attr>
    </edge>

    <edge id="edge2" from="node0" to="node3">
      <attr name="relA">
        <string>74</string>
      </attr>
      <attr name="Type">
        <string>T</string>
      </attr>
    </edge>

    <edge id="edge3" from="node4" to="node1">
      <attr name="relA">
        <string>86</string>
      </attr>
      <attr name="Type">
        <string>L</string>
      </attr>
    </edge>

    <edge id="edge4" from="node4" to="node2">
      <attr name="relA">
        <string>86</string>
      </attr>
      <attr name="Type">
        <string>L</string>
      </attr>
    </edge>

    <edge id="edge5" from="node4" to="node5">
      <attr name="relA">
        <string>72</string>
      </attr>
      <attr name="Type">
        <string>T</string>
      </attr>
    </edge>

  </graph>
</gxl>
```

Annexe – II : Les formes mixtes (BD_4)
(avec des parties pleines et des parties linéaires)