Isomorphisme de sous-graphe pour la recherche d'information audiovisuelle contextuelle

Subgraph isomorphism for contextual Audiovisual information search

Yannick Prié[1,2] Tahar Limane[2] Alain Mille[2]

[1]LISI INSA-Lyon
[2]LISA CPE-Lyon

43 Bd. du 11 novembre 1918, BP 2077, 69616 Villeurbanne Cedex
Yannick.Prie@insa-lyon.fr / limane@cpe.fr / am@cpe.fr

Résumé

Cet article présente un algorithme de recherche d'information dans un système d'information représenté par un graphe. L'approche des Strates Interconnectées par les Annotations pour l'annotation de documents audiovisuels est d'abord expliquée, ainsi que la notion de graphes potentiels caractérisés pour décrire des contextes audiovisuels et des requêtes. Le problème de la résolution des requêtes se ramène alors à un problème de recherche d'isomorphismes de sous-graphes correspondant à un graphe potentiel, dans un graphe global. Après avoir présenté ce problème dans sa généralité, nous détaillons un algorithme de *multi-propagation* reposant sur l'étude d'une heuristique raisonnable en recherche d'informations.

Mots Clef

Recherche d'information, graphes potentiels contextuels, graphes orientés valués, isomorphisme de sous-graphes.

Abstract

This article presents an algorithm for information retrieval in an information system that is represented as a graph. We first explain the AI-Strata approach for audiovisual documents annotation, and the notion of characterized potential graphs, used for audiovisual contexts and queries description. The query resolution problem appears as partial subgraph isomorphism between potential graphs and a global graph search problem. We present this problem in the general case, and the remainder of the article is devoted to the study of a *multi-propagation* algorithm based on a cut of the search space which is reasonable in information retrieval.

Keywords

Information Retrieval, contextual potential graphs, oriented attributed graphs, subgraph isomorphism.

1- Introduction

Les progrès techniques réalisés au niveau des ordinateurs (stockage, puissance de traitement) et des réseaux les reliant (débits) permettent d'envisager des systèmes d'information traitant des documents aussi volumineux que les documents audiovisuels. Si la recherche précise d'un document considéré comme unité (à l'aide de son titre et du nom de l'auteur par exemple) et sa visualisation à l'aide d'un système de « vidéo à la demande » ne pose *a priori* pas de problème, il n'en est pas de même dès que la recherche devient moins précise. Par exemple, une simple requête par mots-clés donnant une dizaine de documents candidats d'une demi-heure chacun nécessite pour vérifier lesquels sont intéressants une visualisation de plusieurs heures. De la même manière, chercher un morceau de document pour un objectif précis nécessite d'une part la possibilité de documenter et d'indexer ce morceau précisément, d'autre part celle d'y accéder directement. La possibilité de *naviguer* dans le document est également requise.

Dans ces objectifs, [1] propose par exemple de découper automatiquement des documents AV ou [2] de les modéliser dans des langages de balises. Notre équipe au sein du projet Sesame[1] s'intéresse à l'*annotation* de documents audiovisuels en vue de leur recherche et de leur utilisation. Les éléments d'annotations temporellement situés, les relations qu'ils entretiennent, ainsi que les abstractions de ces éléments forment un réseau d'annotations partiellement structuré.
Un document audiovisuel est considéré comme un document dont la description est semi-structurée et temporellement située. Bien que diverses extensions aux formalismes de requêtes classiques aient été proposées pour prendre en compte ce genre de description de documents multimédia [12], elles restent insuffisantes [10], et de nouvelles représentations et mécanismes de recherche restent à inventer.

Nous proposons de considérer un système d'information audiovisuelle comme un graphe, et de représenter les requêtes de recherche d'information comme des formes qu'il convient de retrouver dans ce graphe. L'objet du présent article est de présenter l'algorithme de recherche que nous avons mis au point. Dans la première partie, nous présentons l'approche des Strates-IA pour l'annotation de documents audiovisuels, et la manière dont la description des contextes audiovisuels et des requêtes

[1] Sesame : Système d'Exploration de Séquences Audiovisuelles et Multimédia enrichie par l'Expérience. Ce projet est financé par le CNET, Consultation Thématique 96-ME-17. This project is granted by the French Center for Telecommunication Research, contract 96-ME-17.

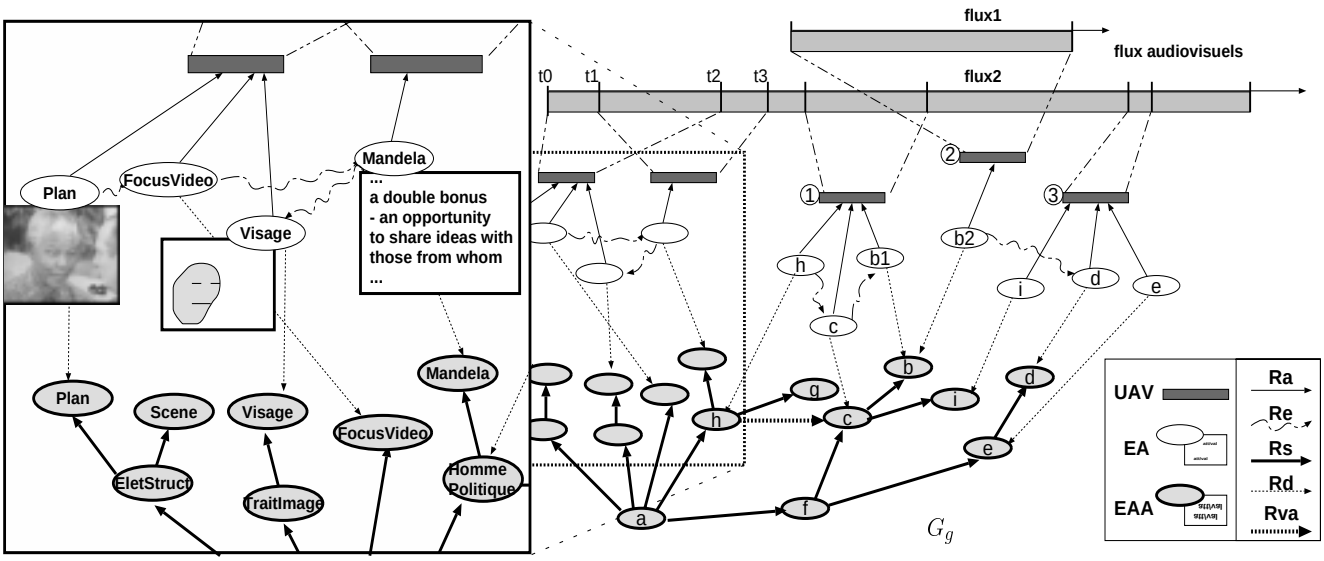

Figure 1 : *Graphe d'annotation et base de connaissances, trois types de sommets pour un graphe*

se fait dans un formalisme équivalent : les graphes potentiels caractérisés. Le problème de la résolution des requêtes se ramène alors à un problème de recherche d'isomorphisme de sous-graphes partiels correspondant à un graphe potentiel dans un graphe global. Après avoir présenté ce problème dans sa généralité, nous nous intéressons à un sous-problème particulier, basé sur une réduction du problème raisonnable en recherche d'informations, et à un algorithme de *multi-propagation* à partir de correspondances connues. Nous nous intéressons ensuite aux propriétés de l'algorithme, avant de conclure en discutant les performances et les extensions possibles.

2- Présentation du modèle des Strates-IA

Nous présentons brièvement dans cette partie le modèle des Strates-IA. Pour une présentation plus détaillée, le lecteur est invité à se référer à [21].

Les documents audiovisuels sont caractérisés par leur multimodalité, leur séquentialité temporelle et l'importance majeure jouée par le *contexte* en leur sein. Un plan est par exemple inséré, du fait du *montage*, entre deux autres plans, ainsi que dans un document et n'est compréhensible que par rapport à eux.

Décrire symboliquement un document audiovisuel signifie l'*annoter*, c'est-à-dire attacher des *annotations* à des *morceaux* du document. Ce principe mis en place, un grand nombre de variations est possible, on peut par exemple faire varier la longueur des morceaux d'un document entier à une image particulière. Les annotations peuvent être mises en place automatiquement, semi-automatiquement ou bien manuellement, et relever de plus ou moins hauts niveaux conceptuels (du simple histogramme de couleur au nom d'un acteur).

Notre approche s'inscrit dans le cadre de la stratification : des morceaux (ou strates) sont définis à chaque fois que l'on pose un élément d'annotation sur le flux, et sont d'extension temporelle quelconque liée à l'objet d'intérêt repéré qui a motivé la pose d'une annotation. Par exemple figure 1, l'annotation <Mandela> définit un morceau limité par *t1* et *t2* sur le flux *flux1*, car l'annotateur a voulu signifier la présence de Mandela à cet endroit précis du document. Les morceaux de documents sont appelés *unités audiovisuelles (UAV)*. Les UAV sont définies par l'identification d'un flux audiovisuel et ses deux limites temporelles dans ce flux. Les *éléments d'annotations (EA)* sont caractérisés principalement par un attribut terme d'annotation (par exemple Mandela), mais possèdent autant d'attribut/valeurs que nécessaire (par exemple du texte libre contenant un discours pour l'EA ⟨*Mandela*⟩ ou une image caractéristique pouvant résumer un ⟨*Plan*⟩). Les éléments d'annotation sont en *relation d'annotation* R_a avec les unités audiovisuelles. Une UAV est définie par un EA primitif et son annotation peut être complétée par autant d'EA que nécessaire. Les éléments d'annotation peuvent être mis en relation les uns avec les autres à l'aide de la *relation élémentaire*, dont la sémantique est libre. Si l'annotateur souhaite préciser la sémantique d'une relation élémentaire entre deux Ea, il peut en utiliser un troisième et deux relations élémentaires. Par exemple ⟨*Mandela*⟩ et ⟨*Plan*⟩ peuvent être mis en relation à l'aide de l'EA ⟨*FocusVideo*⟩ par :

$$\langle Plan \rangle\ R_e\ \langle FocusVideo \rangle\ R_e\ \langle Mandela \rangle.$$

Les termes attachés aux éléments d'annotations forment un vocabulaire contrôlé dans un thésaurus dans laquelle ils sont *a minima* organisés dans une hiérarchie de spécialisation/abstraction d'*éléments d'annotation abstraits (EAA)*. Les éléments d'annotation sont en *relation de décontextualisation* R_d avec les éléments d'annotation abstraits. D'autres relations existent encore dans ce thesaurus (par exemple la relation *voir aussi* R_{va}). Les EAA possèdent en particulier un attribut *valence*, qui exprime leurs possibilités de relation en contexte en tant qu'EA du flux (voir [22,23] pour plus de précisions sur cette notion). L'ensemble des EAA peut être construit manuellement, ou bien en s'aidant par exemple de mécanismes d'extraction automatique de termes à partir de textes. Le degré de figement de cette base de connaissances dépend des visées de l'application documentaire.

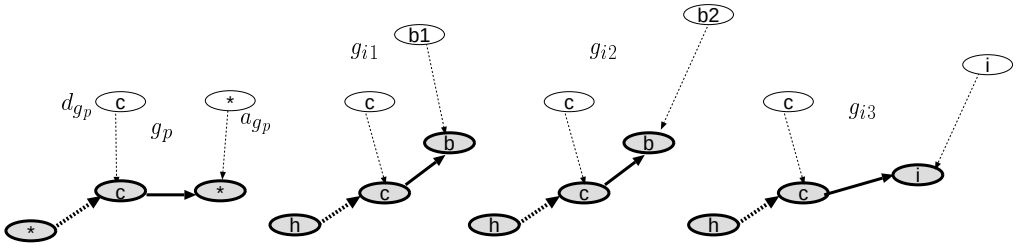

Figure 2 : *Un graphe potentiel et ses trois instances*

Des définitions précédentes il apparaît qu'un document audiovisuel annoté se décompose en unités audiovisuelles annotées par des éléments d'annotation interconnectés par des relations élémentaires et des relations conceptuelles, d'où le nom de l'approche : Strates Interconnectées par les Annotations.

3- Requêtes dans le graphe des Strates-IA : position du problème

Il est possible de considérer l'ensemble du système que nous avons mis en place comme un graphe global *Gg*, composé de sommets de trois types (UAV, EA et EAA) et de relations entre ces sommets.

Utiliser la représentation que nous avons mise en place pour chercher des informations dans le graphe global signifie rechercher des éléments du graphe à partir d'autres éléments connus, c'est-à-dire construire des chemins entre éléments du graphe. Nous considérons les chemins au sens général en faisant abstraction du sens des arc. Par exemple figure 1, on considère qu'il existe un chemin entre l'EAA <Traitimage> et l'EA <Mandela>. Nous définissons le *contexte* d'un sommet *e* du graphe comme l'ensemble des éléments *ei* qui peuvent être mis en relation avec lui, c'est-à-dire tels qu'il existe un chemin *e…ei* dans *Gg* .

Comme il existe des chemins entre tous les sommets le graphe dans son ensemble représente le contexte maximal de tout élément, et il nous faut un moyen de contrôler celui-ci. Pour cela, nous mettons en place la notion de *graphe potentiel*, qui est un graphe construit autour d'une ossature de chemin, chacun des sommets de l'ossature pouvant être précisé par d'autres ossatures de chemins, *etc*. Les contraintes de construction d'un graphe potentiel sont les mêmes que celles du graphe global, à la différence près qu'il est possible d'utiliser des sommets *génériques* pouvant représenter n'importe quel autre sommet de même type. L'attribut *nom de c*es sommets prend alors la valeur de l'identificateur générique *.

Caractériser certains sommets d'un graphe potentiel par des noms en fait des *graphes potentiels caractérisés* utiles (car manipulables à l'aide de leur sommets caractérisés). La figure 2 présente g_p un tel exemple de graphe. Les sommets départ d_{gp} et arrivée a_{gp} du chemin caractérisent le graphe potentiel présenté, qui permet de considérer un certain contexte d'un EA de terme *c*.

Instancier un graphe potentiel dans un graphe global consiste à trouver dans ce dernier un ou plusieurs sous-graphes qui soient isomorphes (aux contraintes de sommets génériques près) avec lui. Par exemple, le graphe potentiel *gp* de la figure 2 s'instancie dans *Gg* de la figure 1 en trois sous-graphes g_{i1}, g_{i2} et g_{i3}. Ce problème se ramène à la reconnaissance d'une *forme* définie par un sous-ensemble de sommets et d'arcs pris dans le graphe général qui s'apparie avec la forme décrite par le graphe potentiel, ce qui conduit à une recherche d'isomorphisme de sous-graphes partiels. L'intérêt d'une telle approche de requêtes dans un graphe global avait été pressentie dans [15].

Pour des raisons de simplification, nous ne nous intéressons pas ici aux autres attributs que l'attribut *nom* des sommets. Il va cependant de soi qu'il est possible de considérer d'autres attributs dans les sommets du graphe potentiel et du graphe global (par exemple des attributs résultats de traitements d'image, ou bien du texte libre). Il convient alors de définir pour chaque attribut une fonction d'appariement booléenne suivant que l'attribut d'un sommet du graphe potentiel correspond ou non à l'attribut du sommet du graphe global étudié pendant la procédure d'instanciation. Posons de façon formelle la définition des graphes généraux et potentiels.

Soit L_S l'ensemble dénombrable des noms-étiquettes des sommets des graphes du système. Soit L_A l'ensemble fini des étiquettes d'arcs du système. Soit * l'étiquette générique.

Définition 1 : Un *graphe* G_Y est défini par le 4-uplet $G_Y=(Y,A_Y,\mu_Y,\nu_Y)$, où Y désigne l'ensemble des sommets de G_Y ; $A_Y \subseteq Y\times Y$ est l'ensemble des arcs de G_Y ; $\mu_Y : Y\rightarrow L_S$ une application qui associe à chaque sommet son étiquette ; $\nu_Y : A_Y \rightarrow L_A$ une fonction qui associe à chaque arc une étiquette.

Définition 2 : Un *graphe potentiel* G_X est une graphe dans lequel l'étiquette d'un sommet ou d'un arc peut être générique (*).

4- Algorithme de multi-propagation

La suite de l'article est consacrée à la présentation de l'algorithme d'isomorphisme de sous-graphes partiels que nous avons mis au point. La première partie présente la définition du problème et donne quelques notations et définitions nécessaires. Dans la deuxième partie, l'algorithme est détaillé ; cet algorithme est basé sur une exploration *multiple* de graphe dite *multi-propagation*.

Afin de simplifier la présentation, nous nous restreindrons à des graphes sans attributs autres que l'étiquette. Par contre, les règles de vérification de cohérence utilisées seront décrites dans leur forme générale; c'est-à-dire en se plaçant dans le cadre des graphes généraux et potentiels définis ci-dessus.

4.1- Graphes et sous-graphes partiels

Un *graphe* G est défini par le couple $G=(X,A)$, où X désigne l'ensemble des sommets de G et $A \subseteq X \times X$ est l'ensemble des arcs de G. x_j est successeur de x_i si $(x_i,x_j) \in A$; l'ensemble des successeurs de x_i est noté $Succ(x_i)$. x_i est prédécesseur de x_j si $(x_i,x_j) \in A$; l'ensemble des prédécesseurs de x_i est noté $Pred(x_i)$.

Etant donné un graphe $G=(X,A_X)$, un *sous-graphe* de G est un graphe $G'=(X',A_{X'})$ tel que $X' \subseteq X$ et $X' \neq \emptyset$ et $A_{X'} = A_X \cap X' \times X'$. Un *graphe partiel* de G engendré par le sous-ensemble d'arcs $B \subseteq A_X$ est le graphe $G'=(X,B)$. Le *sous-graphe partiel* engendré par $X' \subseteq X$ et $B \subseteq A_{X'}$ est le graphe partiel de $G'=(X',A_{X'})$ engendré par $B \subseteq A_{X'}$.

4-2- Isomorphisme de sous-graphe partiel

Le problème de l'isomorphisme est un problème fondamental en informatique. On peut le formuler de la façon suivante : soient deux graphes $G_X=(X,A_X)$ et $G_Y=(Y,A_Y)$. Ils sont *isomorphes* s'il existe une bijection $f: X \rightarrow Y$ telle que :

$$\forall x_i,x_j \in X : (x_i,x_j) \in A_X \Leftrightarrow (f(x_i),f(x_j)) \in A_Y.$$

Une autre variante de ce problème appelée problème de l'isomorphisme de *sous-graphes* consiste à rechercher une relation entre G_X et un sous-graphe partiel de G_Y. L'étude de ce problème répond à la question suivante : G_Y contient-il un sous graphe partiel isomorphe à G_X ; c'est-à-dire un sous-ensemble Y' de sommets et un sous-ensemble $A_{Y'}$ d'arcs tels que $|Y'|=|X|$, $|A_{Y'}|=|A_X|$ et il existe une bijection $f: X \rightarrow Y'$ telle que :

$$\forall x_i,x_j \in X : (x_i,x_j) \in A_X \Leftrightarrow (f(x_i),f(x_j)) \in A_{Y'}.$$

On rencontre le problème de l'isomorphisme de sous-graphes dans de nombreux domaines comme la chimie organique [25,24,27] pour la reconnaissance de molécules qui possèdent une sous-structure particulière - le raisonnement basé sur les cas pour la recherche de cas dans une base de cas [3,20] - les réseaux sémantiques et les grammaires de graphes [7] - l'apprentissage, où les informations symboliques sont représentées par des graphes et l'on désire par exemple savoir si un concept est plus général qu'un autre [9,5] - la reconnaissance de formes et la vision par ordinateur, où l'étude de ce problème a été motivée par le désir de retrouver des objets dans des images [4,28]

Dans cette présentation, nous nous intéresserons au problème de l'isomorphisme de sous-graphes partiels. Etant donnés deux graphes $G_X=(X,A_X)$ et $G_Y=(Y,A_Y)$, l'algorithme présenté ci-dessous détermine tous les sous-graphes partiels de G_Y qui sont isomorphes à G_X. Les méthodes développées pour résoudre le problème de l'isomorphisme de sous-graphes (qui est NP-complet) peuvent être classées dans l'une des deux approches suivantes :

1. L'approche basée sur la recherche de cliques. Dans cette première approche, nous pouvons citer les travaux de [8,19] (cités dans [11]). Le problème de recherche d'isomorphismes de sous graphes dans cette approche, se ramène à la recherche de cliques maximales dans un graphe appelé graphe d'association qui décrit toutes les correspondances possibles entre sommets de G_X et G_Y. Les algorithmes issus de cette approche sont très généraux, puisque le problème est considéré de façon globale mais peu efficaces dans la pratique de par leur complexité et ne s'appliquent qu'à des graphes de petite taille.

2. L'approche dite de relaxation, basée sur l'utilisation combinée de méthodes de filtrage et d'algorithmes d'énumération de l'espace des solutions. [14,13,26] présentent des algorithmes illustrant cette seconde approche. Ces algorithmes présentent l'avantage de pouvoir s'implanter facilement sur des architectures parallèles et d'utiliser ainsi la puissance des machines multiprocesseurs.

Plus récemment, un algorithme de recherche d'isomorphismes basé sur la deuxième approche a été proposé par Cordella et al. [6], appliqué à des graphes générés aléatoirement et comparé aux algorithmes présentés par [25,18]**.** De manière informelle, cet algorithme est basé sur le principe suivant : pour rechercher toutes les solutions, l'algorithme construit un arbre où chaque sommet représente un isomorphisme partiel et un arc correspond à l'ajout d'une paire de sommets conduisant à un autre isomorphisme partiel de cardinalité supérieure. La racine de cet arbre est l'ensemble vide et la profondeur de l'arbre en cours de construction est directement fonction du cardinal de l'isomorphisme partiel associé. Pendant la construction des solutions, si l'ajout d'une paire de sommets à un isomorphisme partiel (à un sommet de l'arbre) conduit à un état incohérent, alors l'algorithme abandonne toute exploration depuis cet état ; c'est-à-dire que le chemin s'interrompt sur cet état incohérent. L'algorithme présenté dans [6], s'avère en moyenne efficace grâce à cette technique d'élagage des chemins incohérents de l'arbre des solutions. L'inconvénient majeur de cet algorithme est son exploration en largeur complète : tous les sommets de l'arbre sont développés en même temps, ce qui multiplie le nombre de sommets à chaque niveau de l'arbre de façon importante. De plus, la présence dans l'arbre de solutions partielles connexes (pouvant être fusionnées) n'est pas gérée, ce qui constitue un autre facteur de multiplication du nombre d'états et nécessite de tester les solutions doublonnées.

Nous proposons une première amélioration de cet algorithme, qui consiste à explorer l'espace des états suivant des directions optimales d'un point de vue du nombre d'états générés. A chaque étape, c'est-à-dire à chaque niveau de l'arbre, une direction est déterminée en étendant un seul état à la fois qui est choisi en fonction d'un critère réduisant le nombre d'états nouveaux. Une deuxième amélioration importante vient de la fusion d'états connexes, réduisant ainsi le nombre d'états au total et évitant la redondance des solutions partielles portées par les sommets de l'arbre. De plus, ces deux améliorations apportées vont nous permettre de fournir des solutions finales dès qu'elles sont établies, sans

attendre d'avoir terminé l'exploration de toutes les solutions possibles.

4.3- Notations et définitions

Nous allons dans cette partie introduire quelques définitions et préciser quelques notations nécessaires pour la description de notre algorithme.

Correspondance **:** Un couple de sommets $[x,y]\in X\times Y$ en relation par f est appelé *correspondance valide*. Pour une correspondance $c=[x,y]$, le sommet x est l'extrémité initiale ou *origine* de c, y est l'extrémité finale ou *extrémité* de c.

Graine **:** Une graine est une correspondance $[x,y]$ valide donnée par l'utilisateur (correspondance de départ).

Dans le modèle des Strates-IA, une graine associe un sommet du graphe potentiel à un sommet connu du graphe global. Ce dernier, peut correspondre à un morceau de document audiovisuel (donc une UAV connue), soit à un élément de la base de connaissances (EAA, unique par définition), soit enfin à un élément d'annotation spécifié par l'utilisateur. Cette limitation de fait sur les graphes potentiels est donc raisonnable en recherche d'information, et revient à considérer que pour tout problème on connaît au moins un point de départ permettant sa résolution. Les graines vont constituer ces points de départ de la recherche d'appariement[2].

Etat **:** Soient deux graphes $G_X=(X,A_X)$ et $G_Y=(Y,A_Y)$. Un état e du processus d'appariement des deux graphes est défini par un ensemble de correspondances valides : $V(e)=\{[x,y]\in X\times Y \mid x \text{ est en correspondance avec } y\}$ appelé *isomorphisme partiel* associé à l'état e. On notera par $V_X(e)$ (resp. $V_Y(e)$) la projection de $V(e)$ sur X (resp. Y) :

$$V_X(e) = \{ x\in X \mid \exists\, y\in Y \mid [x,y] \in V(e) \}$$
$$V_Y(e) = \{ y\in Y \mid \exists\, x\in X \mid [x,y] \in V(e) \}$$

Un état composé uniquement d'une graine est appelé *état initial*. L'ensemble des graines présentes dans un état e est noté $R(e)$.

Extension d'un état **:** $I_X(e)$ (resp. $I_Y(e)$) dénote l'ensemble des sommets incidents intérieurement aux sommets de $V_X(e)$ (resp. $V_Y(e)$). $E_X(e)$ (resp. $E_Y(e)$) dénote l'ensemble des sommets incidents extérieurement aux sommets de $V_X(e)$ (resp. $V_Y(e)$). Plus formellement :

- $I_X(e)=\{x\in X-V_X(e) \mid \exists x'\in V_X(e) \mid (x,x')\in A_X\}$
- $E_X(e)=\{x\in X-V_X(e) \mid \exists x'\in V_X(e) \mid (x',x)\in A_X\}$
- $I_Y(e)$ et $E_Y(e)$ peuvent être définis de façon similaire.

On désigne par $E(e)$ (resp. $I(e)$) l'ensemble des paires de sommets incidents intérieurement (resp. extérieurement) aux sommets de $V(e)$ et défini par :

$$E(e)=E_X(e)\times E_Y(e) \text{ (resp. } I(e)=I_X(e)\times I_Y(e)).$$

On note par $C(e)$ l'ensemble des correspondances candidates à l'*extension* de l'isomorphisme partiel associé à l'état e. La définition de l'ensemble $C(e)$ est réalisée par la séquence ci-dessous :

Si $E(e)\neq\emptyset$ alors $C(e) \leftarrow E(e)$
Sinon $C(e) \leftarrow I(e)$

Soit $C_X(e) = \{x\in X \mid \exists\, y\in Y \mid [x,y]\in C(e)\}$, la projection de l'ensemble $C(e)$ sur X. Pour tout sommet $x\in C_X(e)$ on appelle extension de $C(e)$ suivant x la suite des correspondances de $C(e)$ ayant le sommet x comme origine et notée par $C(e\rightarrow x)$. On note par $|C(e\rightarrow x)|$ la dimension de la suite $C(e\rightarrow x)$ et par $\delta(e)$ la dimension de la plus petite extension associée à l'état e :

$$\delta(e)=\min\{|C(e\rightarrow x)| \text{ avec } x\in C_X(e)\}.$$

Critères de cohérence structurelle **:** le contrôle de cohérence d'une correspondance $[x,y]$ avec un isomorphisme partiel $V(e)$ consiste à vérifier quelques propriétés structurelles telles que les propriétés de connectivité et d'incidence.

Propriété de connectivité : cette propriété exprime le fait que x peut être mis en correspondance avec y si chacun des voisins de x qui est dans $V_X(e)$ est déjà mis en correspondance avec un voisin de y et inversement. D'où les règles qui vérifient le respect de ces contraintes d'adjacence :

$$(\forall x'\in V_X(e)\cap pred_X(x)\Rightarrow \exists y'\in pred_Y(y) \mid (x',y')\in V(e))$$
$$(\forall y'\in V_Y(e)\cap pred_Y(y)\Rightarrow \exists x'\in pred_X(x) \mid (x',y')\in V(e))$$
$$(\forall x'\in V_X(e)\cap succ_X(x)\Rightarrow \exists y'\in succ_Y(y) \mid (x',y')\in V(e))$$
$$(\forall y'\in V_Y(e)\cap succ_Y(y)\Rightarrow \exists x'\in succ_Y(x) \mid (x',y')\in V(e))$$

Propriété d'incidence : Les deux règles suivantes permettent de vérifier cette propriété. La première règle indique si le nombre de successeurs de x incidents intérieurement à $V_X(e)$ est égal au nombre de successeurs de y incidents intérieurement à $V_Y(e)$. La deuxième règle traite les précédesseurs, la troisième et la quatrième sont consacrées au sommets incidents extérieurement.

$$(\ |(succ_X(x)\cap I_X(e)| = |(succ_Y(y)\cap I_Y(e)|)$$
$$(\ |(pred_X(x)\cap I_X(e)| = |(pred_Y(y)\cap I_Y(e)|)$$
$$(\ |(pred_X(x)\cap E_X(e)| = |(pred_Y(y)\cap E_Y(e)|)$$
$$(\ |(succ_X(x)\cap E_X(e)| = |(succ_Y(y)\cap E_Y(e)|)$$

Propriété d'incidence d'ordre 2 : Cette propriété indique si le nombre de successeurs (resp. prédécesseurs) de x hors de l 'état e est bien égal au nombre de successeurs (resp. prédécesseurs) de y :

$$(|pred_X(x)\cap(X-E_X(e)-V_X(e))| \leq |pred_Y(y)\cap(Y-E_Y(e)-V_Y(e))|)$$
$$(|(succ_X(x)\cap(X-E_X(e)-V_X(e))|$$

[2] On peut retrouver cette notion de graine dans [27] sous la forme de sous-structure initiale g_0 d'un sous-graphe cible G. Le graphe g_0 sert comme point de départ d'une recherche d'un sous-graphe maximal de G qui peut être apparié à un graphe présent dans une base de graphes candidats organisée en hiérarchie de subsomption.

$\leq |(succ_Y(y) \cap (Y - E_Y(e) - V_Y(e))|)$

Critères de cohérence sémantique **:** la cohérence sémantique d'une correspondance $[x,y] \in C_X(e) \times C_Y(e)$ peut s'exprimer par des règles simples vérifiant par exemple des relations entre :

- les attributs de ses sommets,
- les attributs liés aux arcs incidents aux sommets de la correspondance.

cohérence des sommets :
((type(x)=type(y) et
((étiquette(x)=étiquette(y))∨(étiquette(x)='*')))

cohérence des arcs :
$\forall(x',y') \mid x' \in V_X(e) \cap pred_X(x) \wedge y' \in V_Y(e) \cap pred_Y(y) \Rightarrow$
(type(x',x) = type(y',y))
$\forall(x',y') \mid x' \in V_X(e) \cap succ_X(x) \wedge y' \in V_Y(e) \cap succ_Y(y) \Rightarrow$
(type(x,x') = type(y,y'))

Etape **:** Une étape P_k est définie par l'ensemble des états $\{e_i(k);\ i=1,...,N_k\}$ atteints à l'itération k du processus d'appariement. L'étape initiale notée P_0 est définie par l'ensemble des états initiaux $\{e_i(0);\ i=1,...,N_0\}$ spécifiés par l'utilisateur.

Structure d'un état : Chaque état e de P_k est décrit par les cinq éléments suivants :

- l'isomorphisme partiel $V(e)$,
- l'extension $C(e)$,
- le nombre $\delta(e)$,
- l'ensemble $R(e)$ des graines présentes dans e,
- l'indicateur d'extensibilité : extensible(e) mis à la valeur Vrai à la création de l'état.

L'indicateur d'extensibilité d'un état permet de spécifier

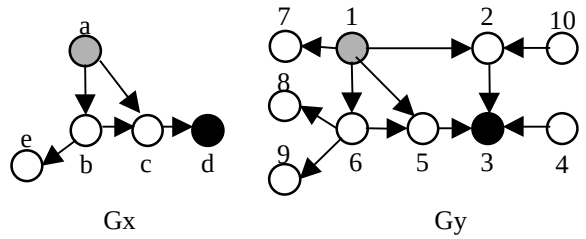

Figure 3 : *{[a,1]} et {[d,3]} sont les états initiaux de l'instanciation de G_X et G_Y*

si celui-ci peut être étendu en un nouvel état au cours du processus d'appariement, ou s'il peut *fusionner* avec d'autres états.

La figure 3 présente deux graphes exemples G_X et G_Y sur lesquels on a distingué deux correspondances initiales (graines), [a,1] et [d,3] compo-sant l'étape initiale P_0={{[a,1]}; {[d,3]}}. La correspondance candidate à l'extension de l'état {[a,1]} suivant b est [b,6] ([b,2] ne vérifie pas la propriété d'incidence d'ordre 2) et $C(\{[a,1]\} \rightarrow b) = \{[b,6]\}$. Les correspondances candidates à l'extension du même état suivant c sont [c,2] et [c,5]. On a alors :

$\delta(\{[a,1]\}) = \min\{|C(\{[a,1]\} \rightarrow b)|, |C(\{[a,1]\} \rightarrow c)|\} = 1$
De la même manière on a :

$C(\{[d,3]\} \rightarrow c) = \{[c,5],[c,2]\}$ et
$\delta(\{[d,3]\}) = |C(\{[d,3]\} \rightarrow c)| = 2$

Fusion d'états **:** à une étape donnée, pour savoir si l'extension d'un état e par une correspondance $[x,y]$ crée un isomorphisme partiel *connexe* à un autre isomorphisme partiel, il faut savoir s'il existe déjà dans l'étape un état e' contenant la correspondance $[x,y]$ et dont l'ensemble des graines est disjoint de l'ensemble des graines présentes dans e. Si c'est le cas, l'extension de e par $[x,y]$ provoque la fusion des deux états (les deux états e et e' sont dits *connexes*). Le nouvel état résultant de la fusion de ces deux états connexes est obtenu en fusionnant leur ensemble de correspondances $V(e'') \leftarrow V(e') \cup V(e)$ et leur ensemble de graines $R(e'') \leftarrow R(e') \cup R(e)$. L'état e devient non extensible (gelé) en positionnant son attribut d'extensibilité à Faux : extensible(e)←Faux. Le gel d'un état consiste à interdire toute extension future de celui-ci *autre que résultant d'un autre état fusionnant avec lui*. Le fait que l'état fusionne signifie qu'il transmet toutes ses propriétés d'extension au nouvel état fusionné, et abandonne donc celles-ci pour lui-même.

4.4- Algorithme de multi-propagation

Les différentes notions que nous venons de définir nous permettent de présenter dans sa totalité l'algorithme de multi-propagation.

L'étape 0 est initialisée avec les états liés aux graines de départ. Le déroulement général de l'algorithme consiste alors à passer d'une étape k à une étape k+1 en étendant un certain nombre d'états de l'étape k suivant leur relation d'extension, et en conservant les autres. Quand un état atteint la taille du graphe qu'on cherche à instancier, cet état représente un isomorphisme solution. On parle d'algorithme de *propagation* au sens où l'extension d'un état se fait par propagation le long des arcs des graphes. L'algorithme de [6] choisit un état quelconque de départ, une seule graine donc, et étend tous les états d'une étape de façon maximale. Au bout d'autant d'étapes qu'il y a de sommets dans le graphe à instancier, l'algorithme aboutit à trouver toutes les solutions, contenues dans la dernière étape.

Dès le moment où l'on dispose de graines à partir desquelles on peut propager les recherches de solutions, il y a lieu de parler de *multi-propagation* de la recherche d'isomorphisme. Dans le cas général, si on part d'une étape k, on retrouvera dans l'étape k+1 :

- les états de l'étape k qui n'ont pas été touchés par l'extension ;
- les états de l'étape k qui ont servi à une fusion (donc ont simplement été marqués non extensibles) ;
- les nouveaux états résultant des extensions simples (l'isomorphisme partiel $V(e)$ d'un état $e \in P_k$ est étendu d'une correspondance en $V(e) \cup \{(x,y)\}$) ou bien des extensions avec fusion.

IsoGP $(G_X=(X,A_X),G_Y=(X,A_Y),P_0)$

$k \leftarrow 0$
Répéter
  Soit $e \in P_k$ extensible / $\delta(e)=\inf\{\delta(a); a \in P_k\}$
  $P_{k+1} \leftarrow P_k - e$
  Choisir un sommet x tel que $|V(e \rightarrow x)| = \delta(e)$
  Pour toute correspondance $[x,y] \in V(e \rightarrow x)$
  Faire
    $P' \leftarrow$ ExtensionFusion(P_{k+1}, [x,y], e)
    Pour tout $e' \in P'$
    Faire
      Si $|e'| = |X|$
      Alors Retourner e' solution
      Sinon $P_{k+1} \leftarrow P_{k+1} + e'$
    Fin Pour
  Fin Pour
  $k \leftarrow k+1$
Jusqu'à $(\forall e \in P_k$, Extensible(e)=Faux)

Figure 4 : *Algorithme de calcul des isomorphismes de sous-graphes partiels*

A la différence de l'approche présentée dans [6] qui étend à chaque étape tous les états suivant toutes les extensions possibles, dans notre approche on n'étend qu'*un seul* état, ce qui permet d'engendrer moins d'états dans l'arbre des solutions (au plus le même nombre).

L'heuristique de choix de l'état à étendre à chaque étape est le critère principal de notre algorithme. A chaque itération, on choisit un état $e \in P_k$ à étendre vérifiant le critère suivant :

$$\delta(e)=\inf\{\delta(a) ; a \in P_k\}.$$

Sur l'exemple de la figure 3, l'état {[a,1]} est ainsi sélectionné pour l'extension car $\delta(\{[a,1]\}) = 1$. Ce choix de l'extension minimale revient à minimiser le nombre de nouveaux états générés dans le passage d'une étape k à une étape k+1, donc à progresser de la façon la plus économique possible vers la solution.

La figure 4 présente l'algorithme principal IsoGP de recherche d'isomorphisme de sous-graphe partiels, qui prend en entrée les graphes G_X et G_Y et l'étape initiale P_0 comprenant les *n* états initiaux (*n* graines). A chaque étape k, le choix de l'état e à étendre fait, on reporte tous les états présents dans P_k dans P_{k+1} à l'exception de l'état e. Pour chaque correspondance [x,y] de $V(e \rightarrow x)$, la fonction ExtensionFusion(P_{k+1}, [x,y], e) (cf. figure 5) qui calcule l'extension de l'état e suivant [x,y] est appelée. La taille de chacun des états retournés par ExtensionFusion est ensuite testée. Si celle-ci correspond à la taille du graphe potentiel, alors l'état décrit un isomorphisme solution, et cette solution est rendue immédiatement, sinon on ajoute l'état à l'ensemble des états de P_{k+1}.

ExtensionFusion(P, [x,y], e)

NouvEtats $\leftarrow \varnothing$
Pour tout $t \in P$ / extensible(t)=Vrai
Faire
  Si $R(t) \cap R(e) = \varnothing$ et $[x,y] \in t$
  Alors
    $e' \leftarrow t \cup e$; $R(e') \leftarrow R(t) \cup R(e)$;
    extensible(t) $\leftarrow$ Faux
    NouvEtats $\leftarrow$ e' + NouvEtats
Fin Pour
Si NouvEtats = $\varnothing$
Alors $e' \leftarrow e+[x,y]$;
    NouvEtats $\leftarrow$ e'
Pour tout $t \in$ NouvEtats
Faire
  Si $|t| \neq |X|$ *(t n'est pas solution)*
  Alors C(e') $\leftarrow$ CalculExtension (e')
    Si C(e') = $\varnothing$
    Alors Enlever t de NouvEtats
Fin Pour
Retourner NouvEtats

Figure 5 : *Fonction réalisant une extension ou des fusions d'états connexes*

L'algorithme s'achève lorsqu'il ne reste plus que des états gelés dans l'étape courante, c'est-à-dire lorsque tous les isomorphismes de sous-graphes partiels solutions ont été trouvés.

La fonction ExtensionFusion retourne :
- soit l'ensemble des états obtenus par fusion de e avec des états de P_{k+1} connexes (i.e. ayant [x,y] dans leurs correspondances et toutes leurs graines différentes de celles de e) ;
- soit un état résultant d'une simple extension de e par la correspondance [x,y] ;
- soit $\varnothing$ si l'état n'a pas d'extension possible et ne peut donc mener à une solution.

CalculExtension calcule l'ensemble des extensions possibles pour un état, ainsi que son extension minimale.

4.5- Un exemple complet

On considère les deux graphes orientés présentés figure 6 $G_X=(X,A_X)$ avec $X=\{a,b,c,d,e\}$ et $G_Y=(Y,A_Y)$ avec $Y=\{1,2,3,4,5,6,7,8,9,10\}$. La lettre à l'intérieur du cercle représentant un sommet indique la valeur d'un attribut associé au sommet. On cherche les sous-graphes partiels de G_Y isomorphes au graphe d'entrée G_X à partir des deux états initiaux suivants : $V(e_{01})=\{[a,3]\}$ et $V(e_{02})=\{[b,4]\}$.

L'exécution de l'algorithme sur notre exemple se fait suivant les étapes représentées dans le graphe états-transitions figure 6. Sur ce graphe, nous avons fait figurer à coté des sommets le paramètre δ associé à l'état correspondant. Sur les arcs sont indiqués les sommets

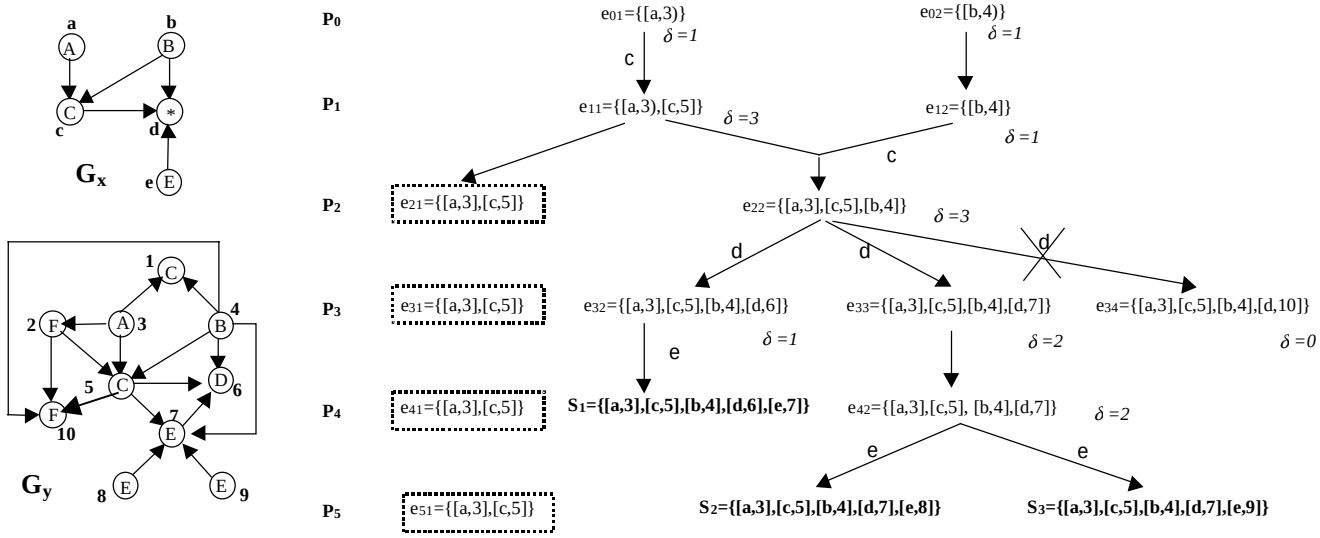

Figure 6 :. *Le graphe états-transition présente les six étapes de la recherche des trois instances de G_X dans G_Y*

suivant lesquels des extensions sont réalisées. Un arc en pointillés indique une simple recopie d'un état d'une étape k dans l'étape k+1. Un arc avec une croix représente une extension non viable. Un sommet entouré d'un rectangle en pointillé indique la non extensibilité de l'état associé. Les sommets en gras S_i représentent les solutions finales trouvées

Le passage de l'étape P_0 à l'étape P_1 se fait par une extension simple de l'état e_{01} suivant c. Le passage de l'étape 1 à l'étape 2 voit l'extension de l'état e_{12} suivant c et sa fusion subséquente avec e_{11}, de laquelle découle l'état e_{22} tandis que e_{11} (et ses diverses recopies) sont gelés. Le déroulement suivant ne pose pas de problème particulier, et l'algorithme s'arrête car l'unique état présent dans l'étape 5 n'est pas extensible. On remarquera également que l'ensemble de toutes les graines des états de cette étape ne couvre plus l'ensemble des graines de départ, c'est-à-dire que toutes les potentialités offertes par les différents états de départ ont été exploitées au maximum. A l'issue de cette exécution, on obtient trois isomorphismes de sous graphes partiels de G_X dans G_Y :

- S_1={(a,3),(c,5),(b,4),(d,6),(e,7)} à l'étape P_4
- S_2={(a,3),(c,5),(b,4),(d,7),(e,8)} à l'étape P_5
- S_3={(a,3),(c,5),(b,4),(d,7),(e,9)} à l'étape P_5

5- Résultats et Discussion

La phase d'expérimentation répond aux tests validant notre approche et permettant d'analyser les performances générales de l'algorithme mis en œuvre. Pour cela, nous avons utilisé la bibliothèque LEDA [16,17] développée à l'Institut Max Planck à Saarbrücken, qui offre une gamme très complète de fonctionnalités relatives à la représentation et aux traitements de graphes. Un composant générique *graphe* est fourni avec les opérateurs permettant de réaliser les traitements de base les plus courants. Les algorithmes sur les graphes sont présents en LEDA sous forme de fonctions paramétrées par le type de graphe. Nous avons évalué notre algorithme et l'algorithme présenté dans [6] sur une série de graphes généraux aléatoires répondant aux contraintes des StratesIA (UAV, EA, EAA, relations)[3]. Les graphes potentiels composés de 10 sommets en moyenne, sont construits par extraction de sous-graphes partiels des graphes généraux ainsi générés, et possèdent au moins un sommet UAV ou EAA ayant une correspondance unique (graine).

Un test élémentaire sur un graphe général $G_Y=(Y,A_Y)$ et un graphe potentiel $G_X=(X,A_X)$ générés aléatoirement, consiste à prendre des mesures (temps d'exécution, nombre d'états générés, etc.) en exécutant :

- une fois notre algorithme avec N graines de départ et
- *n* fois l'algorithme [6] en fixant à chaque exécution une nouvelle graine de départ. A partir de cette série d'exécutions, une moyenne des mesures visées est calculée.

Il est ainsi possible de comparer l'algorithme de multi-propagation prenant *n* graines en compte en même temps à une moyenne des *n* lancements de l'algorithme de propagation.

*

Le tableau 1 résume les résultats des séries de tests réalisées en donnant la taille en nombre de sommets du graphe général généré, le nombre de solutions trouvées, le numéro d'état correspondant à la première solution, le nombre total d'états générés par notre algorithme, et enfin le nombre total d'états générés par l'algorithme présenté dans [6].

Ces résultats sont importants car ils confortent la stratégie choisie pour réduire l'espace des solutions explorés par notre algorithme. On montre ainsi qu'en pilotant le choix des états à l'extension par quelques paramètres simples, il est possible d'obtenir des solutions plus rapidement que l'algorithme présenté dans [6] et des espaces de solutions beaucoup plus réduits. Par exemple, en moyenne la première solution est fournie par notre algorithme avant

[3] Un graphe Strates-IA annotant par exemple deux journaux télévisés quotidiens sur une année représenterait à peu près 100.000 nœuds.

d'atteindre le tiers de la taille totale de l'arbre d'états ; alors que dans l'algorithme [6], on est contraint d'attendre la construction totale de l'arbre d'états pour en extraire la première solution. De plus, on voit clairement que la réduction sensible du nombre total d'états générés peut atteindre 98%.

Tableau 1 : *Comparaison des résultat des algorithmes de propagation et de multipropagation*

Taille du Graphe général	Nombre de solutions	Etat Première Solution	Nombre total d'états	Nombre total d'états [6]
130	48	23	118	5803
250	17	17	68	2696
500	26	15	36	706
1500	14	19	40	2001
2000	20	14	28	615
8000	43	27	93	4875
15000	37	27	70	4527
30000	51	27	105	4597
80000	84	19	132	3837

En perspective de ce travail, quelques orientations peuvent être proposées. Nous avons en effet fait le choix d'étendre un seul état par étape, suivant une unique direction, et l'heuristique de choix considère qu'une bonne direction est une direction qui générera le moins d'états possibles dans la nouvelle étape. Cette heuristique, qui donne de bons résultats peut être discutée. Par exemple, en gardant l'extension d'un unique état, il serait possible de considérer une heuristique variable qui, à partir d'un certaine étape du déroulement de l'algorithme stipulerait de favoriser les états considérés comme étant les plus près d'une solution (i.e. possédant un nombre de graines proche du nombre de graines total).

D'autre part, si notre algorithme s'intègre dans le domaine de la modélisation de documents audiovisuels, il reste général. Ceci provient d'une volonté de considérer l'ensemble du système comme un unique graphe, et de s'attacher à décrire les opérations sur ce système comme étant liées à des manipulations de contextes exprimés par des graphes potentiels. Deux conséquences en découlent : en premier lieu, la généralité de l'algorithme permet d'envisager de l'appliquer à tout ensemble de données s'apparentant à un graphe. Ainsi on peut envisager d'en étudier l'application à des bases de données semi-structurées à partir non plus d'une unique graine (racine d'un arbre de données), mais à partir de plusieurs graines différentes. En deuxième lieu, les particularités de l'application devraient permettre d'envisager de nouvelles heuristiques et améliorations de performance par rapport aux principes généraux présentés dans cet article. Celles-ci pourraient s'appuyer sur les types de sommets et de relations inter-sommets de l'application. Par exemple, la complexité liée aux fonctions de calcul des similarités entre sommets peut être prise en compte. Dans le cadre de notre application, il peut ainsi être utile d'étendre un état dans une direction impliquant un calcul de similarité basé sur des attributs issus de traitements d'image, s'il y a peu de candidats à la comparaison. Au contraire, si l'extension d'un état implique des calculs coûteux sur les attributs d'un grand nombre de sommets, il peut y avoir lieu de retarder ceux-ci, et d'explorer des branches nécessitant des calculs de similarité plus simples. L'heuristique de choix de l'état à étendre est donc paramétrable et adapter celle-ci à l'application permet d'améliorer encore les performances.

6- Conclusion

Nos objectifs consistaient à proposer un modèle à la fois souple et suffisamment général pour permettre la représentation de documents audiovisuel et une méthode efficace de recherche et d'extraction de formes spécifiques du modèle. Le premier objectif s'est traduit par la proposition d'un modèle de graphe attribué permettant d'une part d'organiser les documents audiovisuels (graphe général) et d'autre part, d'exprimer des requêtes de l'utilisateur décrivant des contextes, sous forme de graphes potentiels. Au niveau de la méthode, un algorithme efficace de recherche des instances d'un graphe potentiel dans un graphe général a été réalisé, dont le traitement s'apparente à une recherche d'isomorphismes de sous-graphe partiels. Après une revue générale du problème d'isomorphisme, nous avons présenté un algorithme, basé sur l'exploration étape par étape d'un arbre d'états modifiant et améliorant l'algorithme présenté dans [6]. L'efficacité de cet algorithme repose d'une part sur la stratégie (heuristique) de choix d'un seul état à étendre par étape, et d'autre part sur la notion de fusion d'états connexes. Deux propriétés majeures sont d'une part la possibilité de paramétrer de façon aisée l'heuristique de choix de l'état à étendre, d'autre part le fait de rendre des solutions dès qu'elles sont trouvées, ce qui est important en recherche d'information. Après avoir montré les performances de notre proposition par rapport à l'algorithme général [6], nous discutons les perspectives que celui-ci offre.

Notre travail va désormais s'orienter dans deux directions. La première est liée à notre application audiovisuelle et consiste à étudier l'adaptation de notre algorithme à des graphes de Strates-IA décrits et stockés sous forme d'une suite de fichiers XML et non plus stockés en mémoire. Chacun des fichier XML peut décrire un sous-graphe du graphe général (par exemple correspondant à un flux et à ses annotations). La deuxième direction consiste à étudier et analyser le comportement de notre algorithme de façon plus fine en mettant en place une plate-forme d'expérimentation.

Références

[1] P. Aigrain, P. Joly et V. Longueville, V. Medium Knowledge-Based Macro-Segmentation of Video into Sequences. *In Intelligent Multimedia Information Retrieval*. Ed. Mark Maybury. AAAI Press, MIT Press. pages 159-173, 1997.

[2] G. Auffret, J. Carrive, O. Chevet, T. Dechilly, R. Ronfard, R. et B. Bachimont. Audiovisual-based hypermedia Authoring: using structured representations for efficient access to AV documents. In ACM Hypertext'99, Darmstadt, pages 169-178, 1999.

[3] S. Bradtke et W.G. Lehnert. Some experiments with case based search. In AAAI-88 : The Seventh National Conf. on Artificial Intelligence, pages 133-138, 1988.

[4] C.J. Cho et J.J. Kim. Recognizing 3-D objects by forward checking constrained tree search. Pattern Recognition Letters, 13(8) :587-597, 1992.

[5] D.J. Cook et L.B. Holder. Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research, pages 231-255, 1994.

[6] L.P. Cordella, P. Foggia, C. Sansone et M. Vento. Subgraph Transformations for inexact Matching of Attribued Relational Graphs. Computing Supplement, volume 10, pages 43-52, 1998.

[7] H. Ehrig. Introduction to graph grammers with applications to semantic networks. Computer and Mathematics with applications, 23 :557-572, Septembre 1992.

[8] B. Falkenhainer, K.D. Forbus et D. Gentner. The structure mapping engine : Algorithms and examples. Artificial Intelligence, 41 :1-63, 1989/90.

[9] D.H. Fisher. *Readings in Machine Learning*, San Francisco, CA : Morgan Kaufmann. Chap. Knowledge acquisition via incremental conceptual clustering, pages 267-283, 1990.

[10] Grosky, W.I., Managing Multimedia Information In Database Systems, Communications of the ACM, 40:12, Dec. 1997, pages 73-80.

[11] R. Horaud et T. Skordas. Stero correspondence through feature grouping and maximal cliques. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI, 11 (11) : 1168-1180, 1989.

[12] J. McHugh, S. Abiteboul, R. Goldman, D. Quass et J. Widom, Lore: a database management system for semistructured data, SIGMOD Record, vol. 3, pages 54-66, septembre 1997.

[13] W.Y. Kim et A.C. Kak. 3-D object recognition using bipartite matchnig embedded in discrete relaxation. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI, 13 :224-251, 1991.

[14] L.Kitchen et A. Rosenfeld. Discrete relaxation for matching relational structures. IEEE Transactions on Systems, Man, and Cybernetics, 9(12) :869-874, 1979.

[15] P. Martin, *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l'acquisition de connaissances et la recherche d'informations*, Thèse de doctorat, Univ. de Nice - Sophia Antipolis, 1996, 300 p.

[16] K. Mehlhorn, S. Näher, M. Seel et C. Uhrig, *The LEDA User Manual, Version 3.7.1*, Max-Planck-Institut für Informatik, Saarbrücken-Germany, 1998, 415 p.

[17] K. Mehlhorn et S. Näher., Algorithm Design and Software libraries : Recents Developments in LEDA Project-Algorithms, Software Architecture, In IFIP'92, Madid, 1992, p. 493-505.

[18] B.T. Messmer. *Efficient Graph Matching Algorithms for preprocessed Model Graph*. Ph.D. Thesis, Institute of Computer Science and Applied Mathematics, University of Bern, 1996. 163 p.

[19] S.H. Myaeng et A. Lopez-Lopez. Conceptual graph matching : a flexible algorithm and experiments. Journal of Experimental and Theorical Artificial Intelligence, 4 :107-126, avril 1992.

[20] J. Poole. Similarity in legal case based reasoning as degree of matching in conceptual graphs. In MM. Richter, S. Wess, K.D. Althoff, and F. Maurer, editors. Proceedings : First European Workshop on Case-Based Reasoning, Kaiserslautern, pages 54-58, 1993.

[21] Y. Prié, A. Mille et J.-M. Pinon, AI-Strata : a user-centered model for content-based description and retrieval of audiovisual sequences., *Int. Conf. on Advanced Multimedia Content Processing*, Osaka, Japon. LNCS 1554, pages 328-343, Nov. 1998.

[22] Y. Prié, A. Mille et J.-M. Pinon, Modèle d'utilisation et modèles de tâches pour l'aide à l'utilisateur fondée sur l'expérience : le cas d'un système d'information audiovisuelle. Ingénierie des Connaissances 1999, Palaiseau, pages 21-30, juin 1999.

[23] Y. Prié, A. Mille et J.-M. Pinon A context-Based Audiovisual Representation Model for Audiovisual Information Systems, 2nd International and Interdisciplinary Conf. on Modeling and using context, Trente, Italie. LNAI 1688, pages 296-309. Sept. 1999.

[24] J. C. Régin. *Développement d'outils algorithmiques pour l'intelligence artificielle. Application à la chimie organique.* Thèse de Doctorat, Univ. Montpellier 2, 1995.

[25] D.H. Rouvray et A. T. Balaban. Chemical applications of graph theory. In R.J. Wilson and L.W. Beineke, editors, *Applications of Graph Theory*, Academic Press. Pages 177-221., 1979.

[26] J.R. Ullman. An Algorithm for subgraph isomorphism. Journal of the Association for Computing Machinery, 23(1) :31-42, 1976.

[27] P. Vismara, P. Jambaud, C. Laurenço et J. Quinqueton. RESYN : objets, classification et raisonnement distribué en chimie organique. In *Langages et modèles à objet, État des recherches et perspectives*, Ed. par Ducournau, Euzenat, Masini et Napoli, INRIA, pages 397-419, 1997.

[28] E.K. Wong. Model matching in robot vision by subgraph isomorphism. Pattern Recognition, 25(3) :287-304, 1992.