Comparaison de l'optimisation par colonies de fourmis et d'une recherche taboue réactive sur des problèmes d'appariement de graphes

Sébastien Sorlin [2], Olfa Sammoud [1,2], Christine Solnon [2] et Khaled Ghédira [1]

[1] SOIE, Institut Supérieur de Gestion de Tunis,
41 rue de la Liberté, Cité Bouchoucha, 2000 Le Bardo, Tunis
[2] LIRIS, CNRS UMR 5205, bât. Nautibus, Université Lyon I
43 Bd du 11 novembre, 69622 Villeurbanne cedex, France
{ssorlin,csolnon}@liris.cnrs.fr, {olfa.sammoud,khaled.ghedira}@isg.rnu.tn

Résumé

De nombreuses applications nécessitent de mesurer la similarité d'objets. Lorsque ces objets sont représentés par des graphes, la mesure de similarité se ramène à un problème d'appariement de graphes, *i.e.*, à la recherche d'une meilleure mise en correspondance des sommets de deux graphes. Dans cet article, nous nous intéressons au calcul d'une similarité de graphes basée sur des appariements multivoques des sommets des graphes. Cette mesure a été montrée générique dans le sens où les autres mesures classiques de similarité de graphes peuvent être vues comme un cas particulier de celle-ci. Nous proposons deux algorithmes de calcul de cette similarité : un algorithme basé sur l'optimisation par colonies de fourmis et un algorithme de recherche locale taboue réactive. Nous comparons l'efficacité de ces deux algorithmes sur deux classes de problèmes d'appariements de graphes difficiles et nous montrons que ces algorithmes obtiennent des résultats complémentaires.

Abstract

Many applications involve measuring object similarity. When the objects are modeled by graphs, measuring the similarity turn into a graph matching problem, *i.e.*, into the search for a best matching between the graphs vertices. In this paper, we address the problem of computing a graph similarity measure based on a multivalent graph matching. This measure has been shown generic in the sense that other well-known graph similarity measures can be viewed as special cases of it. We propose and compare two different kinds of algorithms : an Ant Colony Optimization based algorithm and a Reactive Tabu Local Search. We compare the efficiency of these two algorithms on two different kinds of difficult graph matching problems and we show that they obtain complementary results.

1 Introduction

Les graphes sont souvent utilisés pour représenter des objets structurés : les sommets représentent les composants de l'objets et les arcs représentent les relations binaires entre ces composants. Les graphes sont par exemple utilisés pour représenter des images [4, 6], des objets de conception [9], des molécules ou des protéines [1], des emplois du temps [8]... Dans ce contexte, la reconnaissance ou la classification d'objets nécessite de comparer des graphes, *i.e.*, mettre en relation leurs sommets (les apparier) afin d'identifier leurs points communs. Cette comparaison peut se faire à travers la recherche d'une relation d'isomorphisme de graphes ou de sous-graphes afin de montrer l'existence d'une relation d'équivalence ou d'inclusion entre les deux graphes. Cependant, deux objets *similaires* ne sont pas nécessairement *identiques* et présumer de l'existence d'une relation permettant de retrouver tous les sommets et tous les arcs d'un seul graphe (ou des deux graphes) est généralement utopique. Par conséquent, des techniques de comparaison de graphes à tolérance d'erreurs telles que la recherche du plus grand sous-graphe commun ou la distance d'édition de graphes ont été proposées ([7, 10]). Le problème n'est alors plus de trouver un appariement parfait mais un *meilleur* appariement, *i.e.*, celui qui préserve un maximum de sommets et d'arcs.

Plus récemment, quatre articles [9, 6, 4, 11] ont proposé d'aller un cran plus loin dans la comparaison de graphes en introduisant la notion d'appariements multivoques, *i.e.*, des appariements où chaque sommet d'un graphe peut être apparié à un ensemble de sommets de l'autre graphe. Les appariements multivoques permettent notamment de comparer des objets décrits à différents niveaux de granularité tels que des images

sur- et sous-segmentées [4] ou le modèle schématique d'une image à une image réelle et sur-segmentée [6, 11].

Nous nous intéresserons particulièrement à la mesure de similarité de graphes multi-étiquetés de Champin et Solnon [9] car il est montré dans [19] que cette mesure est générique. Cette mesure est paramétrée par des fonctions de similarité permettant d'exprimer des connaissances de similarité et des contraintes propres au problème considéré et par conséquent, elle peut être rendue équivalente à de nombreuses autres mesures de similarité ou de distance de graphes, en particulier celles proposées par Boeres et al. [6], Ambauen et al. [4] et Deruyver et al. [11].

En section 2, nous présentons brièvement la mesure générique de similarité de graphes de Champin et Solnon [9]. En section 3 et 4, nous proposons deux algorithmes pour calculer cette mesure de similarité. Le premier est basé sur l'optimisation à base de colonies de fourmis et le second sur une recherche locale taboue réactive. En section 5, nous comparons expérimentalement ces deux algorithmes sur deux types de problèmes d'appariements de graphes. Nous concluons sur la complémentarité de ces deux algorithmes et nous proposons quelques perspectives à ce travail.

2 Une mesure générique de la similarité de graphes multi-étiquetés

Graphes multi-étiquetés. Un graphe orienté est défini par un couple $G = (V, E)$, où V est un ensemble fini de sommets et $E \subseteq V \times V$ est un ensemble d'arcs orientés. Les sommets et les arcs peuvent être associés à des étiquettes qui décrivent leurs propriétés. Sans perte de généralité, nous supposons qu'au moins une étiquette est associée à chaque sommet et chaque arc. Etant donnés un ensemble L_V d'étiquettes de sommets et un ensemble L_E d'étiquettes d'arcs, un graphe multi-étiqueté est défini par un triplet $G = \langle V, r_V, r_E \rangle$ tel que :

- $r_V \subseteq V \times L_V$ est une relation associant des étiquettes aux sommets, *i.e.*, r_V est l'ensemble des couples (v, l) tel que le sommet v est étiqueté par l'étiquette l,
- $r_E \subseteq V \times V \times L_E$ est une relation associant des étiquettes aux arcs, *i.e.*, r_E est l'ensemble des triplets (v, v', l) tel que l'arc (v, v') est étiqueté par l'étiquette l. Notons que l'ensemble E des arcs du graphe G peut être défini par $E = \{(v_i, v_j) | \exists l, (v_i, v_j, l) \in r_E\}$.

Les éléments des ensembles r_V et r_E sont appelés les *caractéristiques* des sommets et des arcs du graphe G.

Mesure de similarité. Nous décrivons maintenant la mesure de similarité de graphe introduite par Champin et Solnon dans [9]. Nous invitons le lecteur désirant plus de détails à se référer à cet article.

La similarité de deux graphes est calculée par rapport à un appariement des sommets de ces deux graphes. L'appariement considéré est multivoque, *i.e.*, chaque sommet d'un graphe peut être apparié à un ensemble –éventuellement vide– de sommets de l'autre graphe. Plus formellement, un appariement multivoque de deux graphes étiquetés $G = \langle V, r_V, r_E \rangle$ et $G' = \langle V', r_{V'}, r_{E'} \rangle$ est une relation $m \subseteq V \times V'$ contenant tous les couples (v, v') de sommets tel que v est apparié avec v'. Etant donné un appariement m, nous notons $m(v)$ l'ensemble des sommets appariés à v :

$$\begin{aligned} \forall v \in V, m(v) &= \{v'/(v, v') \in m\} \\ \forall v' \in V', m(v') &= \{v/(v, v') \in m\} \end{aligned}$$

Etant donné un appariement multivoque des sommets de deux graphes G et G', l'étape suivante consiste à identifier l'ensemble $G \sqcap_m G'$ des caractéristiques communes aux deux graphes par rapport à l'appariement considéré. Cet ensemble contient toutes les caractéristiques (v, l) des sommets des deux graphes telles que m apparie le sommet v à au moins un autre sommet possédant la même étiquette l. Cet ensemble contient aussi toutes les caractéristiques (u, v, l) d'arcs des deux graphes telles que m apparie le sommet u à au moins un sommet u' et le sommet v à au moins un sommet v' tels que l'arc (u, v) posséde la même étiquette l. Plus formellement, l'ensemble des caractéristiques communes $G \sqcap_m G'$ de deux graphes multi-étiquetés $G = \langle V, r_V, r_E \rangle$ et $G' = \langle V', r_{V'}, r_{E'} \rangle$ (définis sur les mêmes ensembles d'étiquettes) par rapport à un appariement de sommets $m \subseteq V \times V'$ est défini par :

$$\begin{aligned} G \sqcap_m G' \doteq & \{(v, l) \in r_V \mid \exists v' \in m(v), (v', l) \in r_{V'}\} \\ & \cup \{(v', l) \in r_{V'} \mid \exists v \in m(v'), (v, l) \in r_V\} \\ & \cup \{(v_i, v_j, l) \in r_E \mid \exists v'_i \in m(v_i), \exists v'_j \in m(v_j), (v'_i, v'_j, l) \in r_{E'}\} \\ & \cup \{(v'_i, v'_j, l) \in r_{E'} \mid \exists v_i \in m(v'_i), \exists v_j \in m(v'_j), (v_i, v_j, l) \in r_E\} \end{aligned}$$

Etant donné un appariement m, il faut aussi identifier l'ensemble des sommets éclatés (les "splits"), *i.e.*, l'ensemble des sommets qui sont appariés à plus d'un sommet, chaque sommet éclaté v étant associé à l'ensemble s_v des sommets avec lesquels il est apparié :

$$splits(m) \doteq \{(v, m(v)) | v \in V \cup V', |m(v)| \geq 2\}$$

La similarité de deux graphes $G = \langle V, r_V, r_E \rangle$ et $G' = \langle V', r_{V'}, r_{E'} \rangle$ par rapport à un appariement m est alors définie par :

$$sim_m(G, G') = \frac{f(G \sqcap_m G') - g(splits(m))}{f(r_V \cup r_{V'} \cup r_E \cup r_{E'})} \quad (1)$$

où $f : \wp(r_V \cup r_{V'} \cup r_E \cup r_{E'}) \rightarrow \Re$ est une fonction qui pondère les caractéristiques de sommets et d'arcs

et où $g : \wp(\{V \times \wp(V')\} \cup \{V' \times \wp(V)\}) \rightarrow \Re$ est une fonction qui pondère les éclatements de sommets. f et g sont définies en fonction de l'application considérée : elles sont utilisées pour introduire les connaissances de similarité et les contraintes liées à l'application dans la mesure de similarité.

Finalement, la similarité $sim(G, G')$ de deux graphes multi-étiquetés $G = \langle V, r_V, r_E \rangle$ et $G' = \langle V', r_{V'}, r_{E'} \rangle$ est la plus grande similarité qu'il est possible d'obtenir par rapport à l'ensemble de tous les appariements possibles entre les deux graphes, *i.e.* :

$$sim(G, G') \quad = \quad \max_{m \subseteq V \times V'} sim_m(G, G')$$

Notons que le dénominateur de la formule (1) ne dépend pas de l'appariement réalisé –ce dénominateur est introduit pour normaliser la similarité entre zéro et un. Par conséquent, l'appariement m qui maximise la similarité est aussi l'appariement qui maximise la fonction *score* suivante :

$$score(m) \quad = \quad f(G \sqcap_m G') - g(splits(m))$$

Généricité de cette mesure. Il a été montré dans [19] que la mesure de similarité de Champin et Solnon est générique : il est possible de définir les fonctions f et g de la formule (1) de telle sorte qu'un calcul de la mesure de similarité puisse résoudre de nombreux problèmes d'appariements. En particulier, le problème de l'isomorphisme de (sous-) graphe, celui de la distance d'édition de graphes, la recherche du plus grand sous-graphe (partiel) commun à deux graphes et les problèmes d'appariements multivoques de graphes de Boeres et al. [6] et Ambauen et al. [4] peuvent être modélisés en un calcul de similarité de graphes.

Complexité et approches complètes. Le problème du calcul de la similarité de Champin et Solnon est NP-difficile (il est plus général que la recherche du plus grand sous-graphe commun à deux graphes [19]). Un algorithme de recherche complète de l'appariement qui maximise la formule (1) est proposé dans [9]. Cet algorithme, basé sur une exploration exhaustive de l'espace de recherche combiné à des méthodes de filtrage de cet espace, garantit l'optimalité de l'appariement retourné. Cependant, il a été expérimentalement montré que l'explosion combinatoire limite cet algorithme à la comparaison de très petits graphes (moins de 10 sommets dans le pire des cas). Dès lors, les algorithmes incomplets, *i.e.*, ne garantissant pas l'optimalité de la solution retournée mais ayant une complexité polynomiale, semblent être une bonne alternative aux approches complètes.

3 Mesurer la similarité de graphes avec des colonies de fourmis

La méta-heuristique d'optimisation par colonies de fourmis (ACO, Ant Colony Optimization) est une approche bio-inspirée [13, 12] fréquement utilisée pour la résolution de nombreux problèmes combinatoires difficiles. Ce paradigme consiste à modéliser le problème à résoudre en une recherche d'un meilleur chemin dans un graphe –appelé le graphe de construction– et à utiliser des fourmis artificielles pour rechercher les "bons" chemins dans ce graphe. Le comportement de ces fourmis artificielles est inspiré des fourmis réelles : *(i)* les fourmis déposent de la phéromone pour marquer les chemins prometteurs, *(ii)* elles se déplacent dans le graphe de construction en choisissant leur chemin selon une probabilité dépendant des traces de phéromones précédemment déposées, et *(iii)* la quantité de phéromone déposée sur les chemins décroît à chaque cycle de l'algorithme afin de simuler le phénomène d'évaporation de la phéromone observé dans la nature.

Afin de résoudre des problèmes d'appariements de graphes, nous avons proposé dans [17] un premier algorithme ACO appelé "ANT-GM" (ANT-Graph Matching). Cependant, bien que cet algorithme soit compétitif avec la recherche locale taboue sur des problèmes d'isomorphisme de sous-graphes, il ne l'était plus du tout sur les problèmes d'appariements multivoques de graphes. L'algorithme que nous proposons ici est une amélioration de l'algorithme "ANT-GM" appelée "ANT-GM'06". L'amélioration porte sur trois points :

1. une nouvelle fonction heuristique est utilisée dans la probabilité de transition ;
2. une nouvelle stratégie phéromonale est proposée ;
3. une recherche locale est utilisée afin d'améliorer les solutions construites par les fourmis.

Algorithme. A chaque cycle de l'algorithme, chaque fourmi de la colonie construit un appariement en partant d'un appariement initialement vide et en ajoutant itérativement de nouveaux couples de sommets. Le couple de sommets inséré à chaque étape est choisi selon une probabilité qui dépend des traces de phéromone précédemment déposées et d'une heuristique. Une fois que chaque fourmi a construit son appariement, une procédure de recherche locale est lancée afin d'essayer d'améliorer la qualité du meilleur appariement trouvé lors de ce cycle. Les traces de phéromone sont par la suite mises à jour en fonction de cet appariement amélioré. Ce processus est réitéré jusqu'à ce qu'une fourmi ait trouvé l'appariement optimal ou que le nombre maximum de cycle autorisé soit atteint.

Contrairement à l'algorithme "ANT-GM" proposé dans [17], "ANT-GM'06" suit la stratégie *Max-Min Ant System*[20] : une borne minimale τ_{min} et une borne maximale τ_{max} sont imposées aux traces de phéromone (avec $0 < \tau_{min} < \tau_{max}$). Cela permet de minimiser les différences relatives entre les traces de phéromone et d'éviter une convergence prématurée de la recherche vers une région probablement sous-optimale. Afin de favoriser l'exploration de l'espace de recherche lors des premiers cycles, les traces de phéromone sont initialisées à leur borne maximale τ_{max}.

Graphe de construction. Le graphe de construction est le graphe sur lequel les fourmis déposent de la phéromone. Les sommets de ce graphe sont les composants de solutions que les fourmis peuvent sélectionner pour progressivement construire leurs solutions. Dans notre application de recherche du meilleur appariement entre deux graphes, un sommet du graphe de construction est un couple de sommets des graphes à apparier. Plus formellement, étant donnés deux graphes multi-étiquetés $G = \langle V, r_V, r_E \rangle$ et $G' = \langle V', r_{V'}, r_{E'} \rangle$ à apparier, le graphe de construction est le graphe complet non-orienté qui associe un sommet pour chaque couple de sommets $(u, u') \in V \times V'$.

Traces de phéromone. Un point clé lors du développement d'un algorithme à base de colonies fourmis est de choisir une stratégie phéromonale, *i.e.*, de décider où les traces de phéromone doivent être déposées et comment elles doivent être exploitées et mises à jour. Dans notre cas, deux possibilités peuvent être envisagées :

- la stratégie "Vertex" où la phéromone est déposée sur les sommets du graphe de construction. Ainsi, une quantité de phéromone déposée sur le sommet (u, u') du graphe de construction reflète l'expérience de la colonie concernant l'intérêt d'apparier le sommet u au sommet u'.
- la stratégie "Edge" où la phéromone est déposée sur les arcs du graphe de construction. Une quantité de phéromone sur l'arc $< (u, u'), (v, v') >$ représente alors la désirabilité d'apparier en même temps u à u' et v à v' lors de la construction de nouveaux appariements.

Les résultats expérimentaux présentés dans [17] ont été obtenus avec la stratégie "Edge" : c'est la stratégie la plus performante sur les problèmes de recherche de cliques maximales [18] et sur les problèmes de sac-à-dos multiples [2, 3]. Cependant, contrairement aux autres problèmes combinatoires, sur les problèmes d'appariements de graphes, nos expérimentations ont montré que la stratégie "Vertex" (qui mémorise l'expérience concernant chaque couple de sommets individuellement) donne de meilleurs résultats que la stratégie "Edge" (qui mémorise l'expérience concernant chaque paire de couples de sommets et donc l'expérience d'apparier deux arcs entre eux). En outre, la stratégie "Vertex" est beaucoup plus rapide : le dépôt et l'évaporation de la phéromone ont une complexité linéaire par rapport au nombre de sommets du graphe de construction alors qu'elle est quadratique quand la stratégie "Edge" est utilisée.

Par conséquent, dans "ANT-GM'06", la phéromone est déposée sur les sommets du graphe de construction et non plus sur ses arcs comme dans la version initiale de "ANT-GM". La quantité de phéromone sur un sommet (u, u') est notée $\tau(u, u')$.

Construction des appariements par les fourmis. A chaque cycle, chacune des fourmis construit un appariement. En partant d'un appariement m vide (*i.e.* $m = \emptyset$), les fourmis ajoutent à chaque itération un couple de sommet à m choisi parmi l'ensemble des couples non encore sélectionnés $cand = \{(u, u') \in (V \times V') - m\}$. Comme de coutume dans les algorithmes ACO, le couple de sommets à ajouter à m est choisi selon une probabilité qui depend des traces de phéromone et d'un facteur heuristique propre au problème considéré. Plus formellement, étant donné un appariement courant m et un ensemble de candidat $cand$, la probabilité $p_m(u, u')$ d'ajouter le couple de sommets (u, u') à m est défini comme suit :

$$p_m(u, u') = \frac{[\tau(u, u')]^\alpha \cdot [\eta_m(u, u')]^\beta}{\sum_{(v,v') \in cand} [\tau(v, v')]^\alpha \cdot [\eta_m(v, v')]^\beta} \qquad (2)$$

où

- $\tau(u, u')$ est le facteur phéromonal. Lors du choix du premier couple, $\tau(u, u') = 1$, la probabilité ne dépend alors que du facteur heuristique.
- $\eta_m(u, u')$ est le facteur heuristique, introduit pour favoriser les couples de sommets qui maximisent la fonction *score*, *i.e.*, $\eta_m(u, u') = score(m \cup \{(u, u')\}) - score(m)$ si l'ajout de (u, u') fait croître le *score*, $\eta_m(u, u') = 0$ sinon.
- α et β sont deux paramètres qui déterminent l'importance relative des deux facteurs.

Les fourmis arrêtent leur construction quand tous les couples de sommets candidats font décroitre le *score* de l'appariement ou quand les trois derniers ajouts n'ont pas permis d'accroître ce *score*.

Procédure de recherche locale. Pour de nombreux problèmes combinatoires, les algorithmes ACO les plus performants sont ceux qui combinent la construction de solutions par fourmis avec de la recherche locale.

Nous avons donc essayé d'améliorer "ANT-GM'06" en ajoutant une procédure de recherche locale simple offrant un bon compromis entre la qualité des solutions trouvées et son temps d'exécution. Cette recherche locale est appliquée sur le meilleur appariement trouvé lors de chaque cycle. Le voisinage d'un appariement m est l'ensemble des appariements obtenus en ajoutant ou en supprimant un seul couple de sommets à m. L'heuristique utilisée est élitiste : à chaque itération, le meilleur voisin (au sens de la fonction *score*) est sélectionné. A l'obtention d'un maximum local m, la recherche est diversifiée en supprimant les trois plus mauvais couples de sommets à m (au sens de la fonction *score*) et en interdisant leur rajout lors des itérations suivantes. Ce processus d'intensification et de diversification est réitéré tant qu'il permet d'améliorer l'appariement.

Mise à jour de la phéromone. Une fois que chaque fourmis a construit un appariement et que le meilleur appariement trouvé a été amélioré par recherche locale, les traces de phéromone sont mises à jour selon la stratégie *Max-Min Ant System*. Toutes les traces de phéromone sont diminuées par un coefficient d'évaporation ρ ($0 \leq \rho \leq 1$). La meilleure fourmi du cycle est alors autorisée à déposer de la phéromone. Plus formellement, soit m_k le meilleur appariement (par rapport à la fonction *score*) trouvé durant le cycle et amélioré par la recherche locale et m_{best} le meilleur appariement trouvé depuis le début de la recherche (y compris le cycle courant), la quantité de phéromone déposée est inversement proportionelle à la différence entre le *score* de m_k et celui de m_{best}, *i.e.*, $1/(1 + score(m_{best}) - score(m_k))$. Comme nous avons choisi de déposer la phéromone sur les sommets du graphe de construction, cette quantité de phéromone est ajoutée sur chaque sommet $< u, u' >$ tel que $(u, u') \in m_k$.

4 Recherche taboue réactive pour les problèmes d'appariements de graphes

Algorithme glouton. Champin et Solnon [9] proposent un algorithme glouton pour résoudre leur problème de mesure de similarité de graphes. Nous décrivons rapidement cet algorithme car nous l'utilisons pour calculer l'appariement initial de notre recherche locale taboue réactive (RTS, Reactive Tabu Search). Pour plus de détails sur cet algorithme, nous invitons le lecteur à se référer à l'article original. L'algorithme glouton construit un appariement en démarrant d'un appariement vide $m = \emptyset$ et en ajoutant à chaque itération un couple de sommets à m. Le couple de sommets à ajouter à un appariement courant m est choisi parmi les couples candidats de l'ensemble $cand = (V \times V') - m$. Des couples de sommets sont ajoutés à m jusqu'à ce que m soit localement optimal, *i.e.*, jusqu'à ce qu'il n'existe plus de couple de sommets dont l'ajout puisse augmenter la similarité induite par l'appariement. A chaque étape, le couple à ajouter est choisi aléatoirement parmi les couples de *cand* qui maximisent la fonction *score*. Cet algorithme glouton a une complexité polynomiale de $\mathcal{O}((|V| \times |V'|)^2 \times cfg)$ (où cfg est la complexité du calcul des fonctions f et g par rapport à la taille des appariements). En contrepartie de cette faible complexité, cet algorithme ne fait jamais de retour-arrière et n'est pas complet. Notons que le choix des couples de sommets est aléatoire : il est donc possible d'exécuter plusieurs fois cet algorithme et de ne conserver que le meilleur appariement trouvé.

Recherche locale. L'algorithme glouton proposé dans [9] retourne un appariement "localement optimal" : ajouter ou enlever un seul couple de sommets à cet appariement ne permet pas d'augmenter la similarité induite par cet appariement. Cependant, l'ajout et le retrait de plusieurs couples de sommets peuvent rendre possible cette amélioration. Une recherche locale [14, 15] essaye d'améliorer une solution en explorant son voisinage de façon opportuniste. Les voisins d'un appariement m sont les appariements qui peuvent être obtenus en ajoutant ou en retranchant un couple de sommets à m.

$$\forall m \subseteq V \times V', voisins(m) = \{m \cup \{(v,v')\} | (v,v') \in (V \times V') - m\} \cup \{m - \{(v,v')\} | (v,v') \in m\}$$

En partant d'un appariement initial obtenu avec l'algorithme glouton, la recherche locale explore l'espace de recherche de voisin en voisin jusqu'à trouver l'appariement optimal (quand le *score* de celui-ci est connu) ou jusqu'à ce que le nombre maximum de mouvements autorisé soit atteint. Le voisin à sélectionner à chaque itération de l'algorithme est choisi selon la meta-heuristique taboue.

Meta-heuristique Taboue. La recherche *Taboue* [14, 16] est une des meilleures meta-heuristiques connues pour le choix du voisin à explorer. A chaque étape, le meilleur voisin (*i.e.*, celui qui maximise le *score*) est choisi. Notons que ce meilleur voisin peut-être moins bon que l'appariement courant et c'est pourquoi, pour éviter de rester autour d'un optimum local en ajoutant et en enlevant sans cesse les mêmes couples de sommets, une liste de mouvements tabous est utilisée. Cette liste, de longueur k, mémorise les k derniers mouvements réalisés (*i.e.*, les k derniers couples de sommets ajoutés ou enlevés) et il est interdit de réaliser un mouvement inverse à un mouvement de

la liste taboue (*i.e.*, ajouter/enlever un couple récemment enlevé/ajouté). Une exception nommée "aspiration" est ajoutée : l'interdiction ne s'applique pas aux mouvements permettant d'atteindre un appariement de meilleur qualité que le meilleur appariement connu.

Recherche taboue réactive. La longueur k de la liste taboue est un paramètre critique difficile à paramétrer : si la liste est trop longue, la diversification de la recherche est trop importante et l'algorithme ne découvre pas de bons appariements ; si la liste est trop courte, l'intensification est trop forte, l'algorithme reste autour d'un optimum local et ne peut pas améliorer la solution courante. Pour pallier ce problème de paramétrage, Battiti et Protasi [5] proposent une *recherche réactive* où la longueur de la liste taboue est dynamiquement adaptée pendant la recherche. Afin de rendre la recherche réactive, il est nécessaire d'évaluer les besoins de diversification pendant la recherche : quand un même appariement est exploré deux fois, il est nécessaire de diversifier la recherche. Afin de détecter ces redondances, une clé de hachage est mémorisée pour chaque appariement visité. Quand une collision survient dans la table de hachage, signe que l'appariement courant a déjà été exploré, la liste taboue est allongée. Quand au contraire il n'y a pas eu de collisions pendant un certain nombre de mouvements (signe que la recherche ne stagne pas dans un optimum local), la liste taboue est raccourcie. Comme seules les clés de hachage des appariements sont mémorisées et que ces clés peuvent être calculées de façon incrémentale par rapport à la définition du voisinage, rendre l'algorithme tabou réactif est très peu coûteux.

Tabou réactif itéré. La recherche locale taboue réactive peut être lancée à partir de différents appariements initiaux : le nombre maximum de mouvements $maxMoves$ autorisé est divisé par $nbExec$ et $nbExec$ exécutions de $maxMoves/nbExec$ mouvements sont lancées. Le meilleur appariement trouvé durant ces $nbExec$ exécutions est alors conservé.

5 Comparaison expérimentale de RTS et d'ACO

Problèmes considérés. Nous comparons nos algorithmes sur deux types de problèmes d'appariements multivoques de graphes : des instances générées aléatoirement et les 7 instances du problème d'appariement non-bijectif de graphes de Boeres et al. [6].

Jeu d'essais 1. Nous avons utilisé un générateur aléatoire pour générer des paires de graphes "similaires" : un graphe est tout d'abord généré aléatoirement puis quelques fusions et éclatements de sommets et quelques suppressions et insertions d'arcs et de sommets lui sont appliquées afin de générer un deuxième graphe similaire au premier. Lorsque les composants des graphes ont peu d'étiquettes similaires, le meilleur appariement est généralement trouvé de façon trivial. Par conséquent, afin d'obtenir des instances plus difficiles, les arcs et les sommets des graphes que nous avons générés ont tous la même étiquette. Nous avons généré 100 graphes similaires. Ces graphes ont entre 80 et 100 sommets et entre 200 et 360 sommets. Le second graphe a été obtenu en faisant 5 fusions ou éclatements de sommets et 10 insertions ou suppressions de sommets ou d'arcs. La fonction f de la formule (1) est définie comme la fonction cardinalité et la fonction g est définie par : $g(S) = w * \sum_{(v,s_v)\in S} (|s_v| - 1)$ où w est le poids d'un éclatement de sommet.

Le choix du poids w des éclatements de sommets peut changer radicalement la difficulté d'une instance. Quand le poids est nul, le problème est résolu de façon trivial : il est possible de faire autant d'éclatements de sommets que nécessaire pour retrouver une étiquette de sommet ou d'arc. Quand le poids est très élevé, les solutions optimales ne peuvent pas éclater de sommets et le problème devient alors un problème d'appariement univoque de sommets généralement plus simple qu'un problème d'appariement multivoque. Lorsqu'un poids "intermédiaire" est choisi, le problème est généralement beaucoup plus difficile : il est nécessaire de trouver un équilibre entre retrouver des étiquettes de sommets et d'arcs et éclater des sommets. Afin de mesurer la capacité de nos algorithmes à trouver cet équilibre, nous les avons testés avec deux différents poids "intermédiaires". Le premier poids w est égal à 1 (benchmark 1.a), si bien qu'il y a généralement un grand nombre d'éclatements de sommets dans la solution optimale. Le second poids w est égal à 3 (benchmark 1.b), si bien qu'il y a peu d'éclatements de sommets dans la solution optimale. Sur les 100 instances générées, nous n'avons gardé que les 13 instances les plus difficiles (*i.e.*, celles qui n'ont jamais été résolues par l'algorithme glouton de Champin et Solnon [9]).

Jeu d'essais 2. Un problème d'appariement non bijectif a été introduit par Boeres et al. dans [6] afin de trouver le meilleur appariement entre une image de cerveau sur-segmentée et le modèle d'un cerveau. Etant donnés un graphe modèle $G = (V, E)$ et un graphe image $G' = (V', E')$, un appariement est défini comme une fonction $\phi : V \rightarrow \wp(V') - \emptyset$ associant à chaque sommet du graphe modèle G un ensemble non-vide de sommets du graphe image G'. L'apparie-

ment doit respecter trois contraintes : *(i)* chaque sommet du graphe image doit être associé à exactement un sommet du graphe modèle, *(ii)* il existe un ensemble de couples de sommets $(v, v') \in V \times V'$ tels que v' ne doit pas appartenir à $\phi(v)$, *(iii)* le sous-graphe induit par chaque ensemble $\phi(v)$ de sommets de l'image doit être connexe (afin de ne fusionner que des régions adjacentes). Un poids $s^v(v_i, v'_i)$ (resp. $s^e(e_i, e'_i)$) est associé à chaque couple de sommets $(v_i, v'_i) \in V \times V'$ (resp. arcs $(e_i, e'_i) \in E \times E'$). Le but est alors de trouver un appariement qui respecte les contraintes et qui maximise une fonction de similarité dépendant du poids des sommets et des arcs appariés.

Il est possible de définir les fonctions f et g de la mesure de Champin et Solnon de façon à ce que l'appariement qui maximise la formule (1) soit l'appariement qui maximise la similarité de Boeres et al. [6]. Nous invitons le lecteur à consulter [19] pour plus de détails sur la définition de ces deux fonctions. Nous avons donc transformé les 7 instances du problème de Boeres et al. en un problème de similarité de graphe multi-étiquetés. Pour ces 7 instances, le graphe modèle contient entre 10 et 50 sommets et le graphe image entre 30 et 250 sommets. Sur ces instances, nous comparons nos algorithmes avec l'algorithme $LS+$, l'algorithme de construction aléatoire d'appariement proposé par Boeres et al. [6]. Cet algorithme est basé sur une génération gloutonne d'un ensemble d'appariements respectant les contraintes puis sur une amélioration par recherche locale du meilleur appariement trouvé par l'algorithme glouton. Nous invitons le lecteur à consulter [6] pour plus de détails sur les instances du problème et sur l'algorithme $LS+$.

Paramétrage d'ACO. Pour *ANT-GM'06*, le poids α du facteur phéromonal a été fixé à 1 pour les instances du benchmark 1 et à 2 pour celles du benchmark 2. Le taux de persistance de la phéromone ρ est fixé à $0,98$, le poids du facteur heuristique est fixé à 10, le nombre de cycles maximum est fixé à 1000 (resp. 2000) pour le benchmark 1 (resp. 2). Le nombre de fourmis $nbAnts$ de la colonie est 20. Les bornes τ_{min} et τ_{max} de la quantité de phéromone sont fixées à 0.01 et 6.

Afin d'évaluer l'intérêt d'hybrider l'optimisation par colonies de fourmis avec de la recherche locale, `ANT-GM'06` est testé avec et sans la phase d'amélioration des appariements par recherche locale. Ces algorithmes sont alors noté `ANT-GM'06+LS` et `ANT-GM'06`.

Paramétrage de RTS. La recherche locale taboue réactive nécessite 5 paramètres : la longueur minimale min et maximale max de la liste taboue, la longueur d'allongement et de raccourcissement *diff* de la liste quand le processus de réaction se déclenche, la fréquence $freq$ de raccourcissement de la liste (en nombre de mouvements) et le nombre $nbMoves$ de mouvements autorisés. La longueur initiale de la liste taboue est toujours fixée à sa longueur minimale. Pour les deux benchmarks, max est fixé à 50, *diff* à 15 et $maxMoves$ à 50000. Afin d'obtenir de meilleurs résultats, les deux autres paramètres doivent être choisis en fonction du benchmark : sur le premier benchmark, min est fixé à 15 et $freq$ à 5000 alors que sur le second benchmark, min est fixé à 10 et $freq$ à 1000. Comme 50000 mouvements de recherche locale sont réalisés beaucoup plus rapidement que 1000 cycles de `ANT-GM'06`, notre recherche locale est réitérée à partir de différents appariements initiaux. Le nombre d'itérations de notre recherche locale taboue est fixé de telle sorte que le temps d'exécution soit le même pour ACO et pour Tabou. Notons cependant que l'exécution de notre algorithme sur les instances du benchmark 2 est déterministe : les poids utilisés sont des nombres réels et par conséquent, il n'y a jamais de couples de sommets ex-æquo à départager aléatoirement. Il est donc inutile de réitérer Tabou et une seule exécution de Tabou est réalisée.

Pour chacun des algorithmes, les résultats sont donnés pour au moins 20 exécutions de chaque instance.

Résultats. Le tableau 1 présente les résultats obtenus sur les 13 instances du benchmark 1.a (poids des splits fixé à 1). Nous pouvons tout d'abord constater que nos algorithmes sont robustes dans le sens où le *score* moyen sur 20 exécutions est généralement très proche du meilleur *score* trouvé (pour 9 instances sur 13, le *score* moyen est égal au meilleur *score* trouvé). La recherche locale taboue réactive donne de meilleurs résultats que `ANT-GM'06` : elle obtient un *score* meilleur que `ANT-GM'06` sur une instance et est toujours au moins aussi rapide. L'ajout d'une procédure de recherche locale améliore les résultats de `ANT-GM'06` : `ANT-GM'06+LS` obtient un meilleur (resp. moins bon) *score* que RTS sur 3 (resp. 1) instances et les même résultats que RTS sur 9 instances. Sur ce benchmark, `ANT-GM'06+LS` et RTS obtiennent donc des résultats complémentaires : `ANT-GM'06+LS` obtient plus fréquemment de meilleurs résultats que RTS mais RTS est plus rapide que `ANT-GM'06+LS`, même lorsque ces deux algorithmes obtiennent des *scores* identiques. Notons enfin que si un cycle de `ANT-GM'06+LS` est plus long qu'un cycle de `ANT-GM'06` (du fait de l'ajout de la recherche locale), `ANT-GM'06+LS` ne nécessite généralement pas plus de temps CPU que `ANT-GM'06` car la procédure de recherche locale accélère la convergence de l'algorithme et que par conséquent, un nombre plus faible de cycles est nécessaire pour trouver la solution optimale.

Problème				RTS			ANT-GM'06			ANT-GM'06+LS		
Nbr	($\mid V_1 \mid$,$\mid E_1 \mid$)	($\mid V_2 \mid$,$\mid E_2 \mid$)	L	Best	Moy	T.	Best	Moy	T.	Best	Moy	T.
1	(80, 200)	(74, 186)	1512	511	511.00	57	511	511.00	131	**512**	**511.10**	140
2	(80, 240)	(82, 261)	1415	644	644.00	60	644	644.00	266	644	644.00	239
3	(80, 320)	(83, 362)	1445	821	820.97	279	821	820.50	498	**822**	**821.20**	660
4	(80, 340)	(72, 302)	1174	753	753.00	55	753	753.00	111	753	753.00	130
5	(80, 360)	(77, 367)	1139	**856**	**855.97**	187	855	855.00	321	855	855.00	249
6	(80, 360)	(78, 367)	1196	863	863.00	21	863	863.00	187	**864**	**863.94**	565
7	(90, 300)	(91, 307)	1670	762	762.00	98	762	762.00	326	762	762.00	213
8	(90, 320)	(87, 310)	1611	780	780.00	51	780	780.00	572	780	780.00	409
9	(90, 320)	(90, 339)	1716	816	**816.00**	69	816	815.45	546	816	815.45	602
10	(100, 260)	(96, 263)	2093	697	696.63	628	697	696.90	976	697	**697.00**	812
11	(100, 300)	(100, 304)	2078	780	780.00	148	780	780.00	278	780	780.00	279
12	(100, 320)	(98, 331)	2080	828	828.00	46	828	828.00	286	828	828.00	218
13	(100, 360)	(99, 371)	2455	915	915.00	90	915	915.00	267	915	915.00	152

TAB. 1 – Jeu d'essais 1.a (appariements multivoques de graphes avec le poids des splits fixé à 1). Pour chaque instance, le tableau précise le nombre de sommets et d'arcs des deux graphes, la limite de temps CPU L pour chaque exécution (sur un Pentium IV 1.7 GHz). Pour chaque algorithme est précisé le meilleur *score* trouvé et le *score* moyen obtenu pour 20 exécutions, le temps moyen en secondes nécessaire à l'obtention du meilleur *score*.

Problème				RTS			ANT-GM'06			ANT-GM'06+LS		
Nbr	($\mid V_1 \mid$,$\mid E_1 \mid$)	($\mid V_2 \mid$,$\mid E_2 \mid$)	L	Best	Moy	T.	Best	Moy	T.	Best	Moy	T.
1	(80, 200)	(74, 186)	659	496	496.00	28	496	496.00	132	496	496.00	121
2	(80, 240)	(82, 261)	798	624	624.00	26	624	624.00	108	624	624.00	88
3	(80, 320)	(83, 362)	896	801	801.00	17	801	801.00	213	801	801.00	218
4	(80, 340)	(72, 302)	737	732	732.00	27	732	732.00	185	732	732.00	194
5	(80, 360)	(77, 367)	852	846	846.00	198	846	846.00	116	846	846.00	77
6	(80, 360)	(78, 367)	855	840	840.00	36	840	840.00	94	840	840.00	67
7	(90, 300)	(91, 307)	1140	748	748.00	82	748	748.00	186	748	748.00	150
8	(90, 320)	(87, 310)	1079	766	766.00	44	766	766.00	187	766	766.00	187
9	(90, 320)	(90, 339)	1127	802	802.00	70	802	802.00	167	802	802.00	163
10	(100, 260)	(96, 263)	1346	683	683.00	114	683	682.75	556	683	683.00	354
11	(100, 300)	(100, 304)	1466	769	769.00	358	769	769.00	274	769	769.00	285
12	(100, 320)	(98, 331)	1463	814	814.00	51	814	814.00	241	814	814.00	201
13	(100, 360)	(99, 371)	1528	900	900.00	54	900	900.00	245	900	900.00	243

TAB. 2 – Jeu d'essais 1.b (appariements multivoques de graphes avec le poids des splits fixé à 3). Pour chaque instance, le tableau précise le nombre de sommets et d'arcs des deux graphes, la limite de temps CPU L pour chaque exécution (sur un Pentium IV 1.7 GHz). Pour chaque algorithme est précisé le meilleur *score* trouvé et le *score* moyen obtenu pour 20 exécutions, le temps moyen en secondes nécessaire à l'obtention du meilleur *score*.

Le tableau 1 ne reporte pas les résultats du premier algorithme ANT-GM que nous avions proposé dans [17] : notre nouvel algorithme ANT-GM'06 est largement plus efficace que le précédent. Sur toutes les instances considérées, ANT-GM'06 trouve de meilleurs solutions en un temps plus court que ANT-GM. Par exemple, sur la première instance, le meilleur *score* trouvé par ANT-GM est 505 et 8648 secondes sont nécessaires à l'obtention de ce résultat alors que ANT-GM'06 trouve un score de 511 en 131 secondes.

Tableau 2 présente les résultats sur les 13 instances du benchmark 1.b (poids des splits fixé à 3). Sur chacune des instances, nos trois algorithmes trouvent toujours le même meilleur *score* et le même *score* moyen (sauf ANT-GM'06 sur une instance). Cependant, RTS trouve ces solutions en un temps plus court que ANT-GM'06 et ANT-GM'06+LS à l'exception de deux instances. Ces résultats montrent donc que sur ces instances, il est nettement préférable d'utiliser la recherche locale taboue réactive.

Le tableau 3 présente les résultats sur les 7 instances du problème de l'appariement non-bijectif de graphes [6] du benchmark 2. Nos trois algorithmes obtiennent de meilleurs résultats que l'algorithme de référence proposé par Boeres et al. [6] (6 instances sur 7 sont mieux résolues par RTS et 7 instances sur 7 par les algorithmes ACO). Notons que, comme les poids considérés sont des nombres réels, une exécution de RTS est déterministe : il n'y a jamais d'ex-æquo à départager. Par conséquent, glouton retourne toujours le même résultat et il n'est pas nécessaire de réitérer RTS. Les résultats obtenus sur ces instances montrent qu'une nouvelle fois, les algorithmes ACO donnent de meilleurs résultats que RTS mais nécessitent aussi un temps d'exécution nettement supérieur.

En conclusion, ANT-GM'06+LS obtient généralement de meilleurs résultats que RTS mais nécessite un temps d'exécution plus long. Notons que la limite de temps accordée à nos algorithmes a été fixée par rapport aux algorithmes ACO qui, contrairement à la recherche locale, convergent assez lentement : l'algorithme RTS a donc été pénalisé par cette limite de temps. La convergence de RTS est beaucoup plus rapide que celle des fourmis et ACO aurait obtenu de moins bons résultats que RTS avec une limite de temps plus courte. ANT-GM'06+LS et RTS obtiennent donc des résultats complémentaires : pour calculer rapidement de bonnes solutions des instances difficiles ou pour résoudre des instances faciles, il est préférable d'utiliser RTS. A contrario, en consacrant plus de temps de calcul, pour calculer de meilleures solutions ou pour résoudre des instances très difficiles il est préférable d'utiliser l'algorithme ANT-GM'06+LS.

6 Conclusion et perspectives

Nous nous intéressons dans cet article au problème du calcul de la mesure générique de la similarité de deux graphes multi-étiquetés de Champin et Solnon [9]. Nous proposons et nous comparons deux types d'algorithmes : un algorithme nommé ANT-GM'06+LS basé sur l'optimisation à base de colonies de fourmis dopé par un algorithme de recherche locale, et l'algorithme RTS, une recherche locale taboue réactive. Nous comparons ces algorithmes sur deux types de problèmes difficiles d'appariements de graphes : des problèmes d'appariements multivoques de graphes de Champin et Solnon [9] et des problèmes d'appariements non-bijectifs de graphes de Boeres et al. [6]. Nous montrons que ces deux algorithmes obtiennent des résulats complémentaires : l'algorithme ANT-GM'06+LS obtient généralement de meilleurs résultats mais est plus lent que RTS. Par conséquent, pour calculer rapidement de "bonnes" solutions des instances difficiles ou pour résoudre des instances faciles, nous préconisons l'utilisation de RTS. Si au contraire le temps de calcul disponible est important ou pour résoudre des instances particulièrement difficiles, nous préconisons l'utilisation de ANT-GM'06+LS.

Nous aimerions améliorer la rapidité de convergence de ANT-GM'06+LS. Cette accélération pourrait par exemple se faire en utilisant une recherche locale plus performante pour la réparation des appariements trouvés par les fourmis. Nous aimerions aussi que la recherche locale réactive soit plus diversifiée afin de sortir plus facilement des maxima locaux. Cette amélioration de la diversification pourrait se faire en choisissant une méthode de génération des appariements initiaux moins élitiste que l'algorithme glouton de Champin et Solnon et en démarrant un plus grand nombre de recherches locales plus courtes à partir de différents points de l'espace de recherche.

Références

[1] T. Akutsu. Protein structure alignment using a graph matching technique. Technical report, Bioinformatics Center, Institute for Chemical Research, Kyoto University. Uji, Kyoto 611-0011, Japan, 1995.

[2] I. Alaya, C. Solnon, and K. Ghédira. Ant algorithm for the multi-dimensional knapsack problem. *International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004)*, pages 63–72, 2004.

[3] I. Alaya, C. Solnon, and K. Ghédira. Optimisation par colonies de fourmis pour le problème du

Problem			LS+	RS		ANT-GM'06			ANT-GM'06+LS		
GM-i	($\mid V_1 \mid, \mid V_2 \mid$)	L	Sim	Sim	T.	Sim	Moy	T.	Sim	Moy	T.
5	(10, 30)	18	.5474	.5481	0.9	.5601	.5598	16	**.5608**	.5604	15
5a	(10, 30)	19	.5435	.5529	4.6	.5638	.5638	10	**.5645**	.5641	7
6	(12, 95)	269	.4248	.4213	0.0	**.4252**	.4251	211	**.4252**	.4251	215
7	(14, 28)	13	.6319	.6333	2.1	.6369	.6369	7	**.6376**	.6369	5
8	(30, 100)	595	.5186	.5210	1.3	.5229	.5226	462	**.5232**	.5228	229
8a	(30, 100)	595	.5222	.5245	1.3	.5263	.5261	456	**.5269**	.5264	241
9	(50, 250)	6018	.5187	.5199	81.7	.5201	.5201	4133	**.5203**	.5202	2034

TAB. 3 – Résultats sur les problèmes d'appariements non-bijectifs de graphes de Boeres et al. [6] (colonne LS+). Pour chacune des instances est précisé le nom de l'instance, le nombre de sommets des deux graphes, la limite en temps CPU L de chacune des exécutions (sur un Pentium IV à 1,7GHz), la similarité obtenue avec les meilleurs solutions trouvées par LS+[6], `ANT-GM'06`, `ANT-GM'06+LS` et RTS (meilleur solution trouvée sur 20 exécutions, moyenne des meilleurs solutions et temps moyen en secondes.

sac-à-dos multidimensionnel. *To appear in Techniques et Sciences Informatiques (TSI)*, 2006.

[4] R. Ambauen, S. Fischer, and H. Bunke. Graph edit distance with node splitting and merging and its application to diatom identification. *4th IAPR-TC15 Wk on Graph-based Representations in Pattern Recognition*, LNCS 2726-Springer :95–106, 2003.

[5] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. In Springer-Verlag, editor, *Algorithmica*, volume 29, pages 610–637, 2001.

[6] M. Boeres, C. Ribeiro, and I. Bloch. A randomized heuristic for scene recognition by graph matching. In *Wkshp on Experimental and Efficient Algorithms (WEA 2004)*, pages 100–113, 2004.

[7] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *PRL : Pattern Recognition Letters*, 18 :689–694, 1997.

[8] E.K Burke, B. MacCarthy, S. Petrovic, and R. Qu. Case based reasoning in course timetabling : An attribute graph approach. *Proc. 4th Int. Conf. on Case-Based Reasoning (ICCBR-2001)*, LNAI 2080-Springer :90–104, 2001.

[9] P.-A. Champin and C. Solnon. Measuring the similarity of labeled graphs. In *5th Int. Conf. on Case-Based Reasoning (ICCBR 2003)*, volume LNAI 2689-Springer, pages 80–95, 2003.

[10] D. Conte, P. Foggia, C. Sansone, and M. Vento. 30 years of graph matching in pattern recognition. *Int. Jour. of Pattern Recogn. and AI*, 18(3) :265–298, 2004.

[11] A. Deruyver, Y. Hodé, E. Leammer, and J.-M. Jolion. Adaptive pyramid and semantic graph : Knowledge driven segmentation. In Luc Brun and Mario Vento, editors, *Graph-Based Representations in Pattern Recognition : 5th IAPR International Workshop, GbRPR 2005, Poitiers, France, April 11-13, 2005. Proceedings*, volume 3434 of *LNCS*, page 213. Springer, 2005.

[12] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. *In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization. McGraw Hill, London, UK*, pages 11–32, 1999.

[13] M. Dorigo and T. Stützle. Ant colony optimization. *MIT Press*, 2004.

[14] F. Glover. Tabu search - part I. *Journal on Computing*, pages 190–260, 1989.

[15] S. Kirkpatrick, S. Gelatt, and M. Vecchi. Optimisation by simulated annealing. In *Science*, volume 220, pages 671–680, 1983.

[16] S. Petrovic, G. Kendall, and Y. Yang. A tabu search approach for graph-structured case retrieval. In IOS Press, editor, *Proc. of the STarting Artificial Intelligence Researchers Symposium (STAIRS 2002)*, pages 55–64, 2002.

[17] O. Sammoud, C. Solnon, and K. Ghédira. An ant algorithm for the graph matching problem. *5th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2005)*, LNCS 3448 - Springer :213–223, 2005.

[18] C. Solnon and S. Fenet. A study of aco capabilities for solving the maximum clique problem. *To appear in Journal of Heuristics- Springer*, 2006.

[19] S. Sorlin and C. Solnon. Reactive tabu search for measuring graph similarity. *5th IAPR Workshop on Graph-based Representations in Pattern Recognition (GbR 2005)*, LNCS 3434 - Springer :172–182, 2005.

[20] T. Stützle and H.H. Hoos. Max-min Ant System. *Journal of Future Generation Computer Systems*, 16 :889–914, 2000.