Plateforme GraMAP

1. Description de la plateforme

GraMAP (*Graph Matching Algorithms Platform*) est une plateforme web dédiée aux algorithmes d'appariement de graphes qui implémente l'ensemble des algorithmes conçus et développés au sein de l'équipe G2Ap du laboratoire GAMA dans le cadre du projet AOC (Appariement d'Objets Complexes). En effet, puisque un grand nombre de modèles de données décrivant des objets complexes peuvent êtres représentés sous forme de graphes, GraMAP se veut une plateforme générique d'expérimentation et de comparaison pour les algorithmes d'appariement des différentes descriptions de ces objets.
Dans GraMAP, les algorithmes sont classés suivant le type d'appariement qu'ils réalisent (exact ou non-exact) et le modèle qu'ils traitent. Dans ce qui suit, nous situons, par rapport à ces critères, et nous décrivons chacun des algorithmes que comporte GraMAP.

1.1 TwigStack++

La jointure structurelle est l'une des premières méthodes proposées pour résoudre l'appariement exact des documents XML. L'algorithme holistique de jointure appelé *TwigStack* [1] est l'algorithme le plus répandu dans la littérature. Notre algorithme *TwigStack*++ [2] vise principalement à diminuer le coût de la jointure et le calcul inutile de l'algorithme TwigStack.
L'algorithme TwigStack++ est composé de trois étapes. La première permet de décomposer l'arbre de la requête en un ensemble de petites composantes connexes (chemins et étoiles). Le but de cette étape est de réduire le nombre de composantes générées et par conséquent le nombre d'opérations de jointures exécutées dans la troisième étape. Ensuite, des solutions intermédiaires pour chaque composante de la requête sont trouvées. Dans cette étape nous avons développé une nouvelle technique de *Buffring* qui consiste a lire prématurément des éléments XML qui pourraient être requises plus tard en le stockant dans une mémoire cache ou tampon. Enfin, les résultats intermédiaires sont joints pour obtenir la solution finale.

1.2 SCM4BP

SCM4BP (*String Comparators matchmaking for Business Processes*) [3] est un algorithme générique qui utilise la technique de comparateurs de chaînes de caractères pour l'appariement non-exact de services basé sur les graphes de ces services. Un comparateur de deux chaines de caractères peut être une distance ou une similarité. Une distance de valeur élevée signifie que les deux chaines de caractères sont très différentes alors qu'une valeur élevée de la similarité montre une grande similitude entre les deux chaines de caractères. Nous avons utilisé dans l'algorithme SCM4BP trois distances à savoir, *Hamming*, *Sous-mots* et *Levenshtein*, et deux similarités, à savoir *Jaro* et une métrique que nous avons proposé. Cette dernière améliore la similarité de Jaro en considérant les deux sens de comparaisons de chaînes de caractères.
Chaque service (processus métier), représenté par un graphe orienté, est d'abord décomposé en ses séquences d'exécutions possibles. Par la suite, la distance entre deux graphes de processus métiers est approchée par un algorithme de calcul de distances entre les séquences d'exécutions possibles des deux processus métiers.
Le processus métier utilisé pour représenter les services est basé sur le même modèle que [4] qui couvre les principales fonctionnalités des langages tels que BPMN, EPC et OWL-S.

L'algorithme prend en entrée deux services décrits par des fichiers XML et retourne la distance ou la similarité entre eux (suivant la métrique).

1.3 SubMatch

SubMatch (*Subgraph Matching*) [6] [7] [8] est une approche dédiée à l'appariement non-exact de modèles de processus.
Le but est de calculer la similarité entre deux services web décrits avec OWLS. La comparaison s'effectue sur la partie comportement (*process model*) des services web représentés par des graphes. L'algorithme prend en entrée deux graphes représentants des modèles de processus OWLS et retourne le degré de similarité entre eux.

1.4 SSD

SSD (*String Strong coloring based Distance algorithm*) [9] est un algorithme pour l'appariement non-exact d'arbres étiquetés non-ordonnés et/ou non-enracinés, qui se base sur une coloration de graphes spécifique, à savoir, la *coloration forte stricte*.
L'algorithme SSD est composé de trois étapes. La première permet de décomposer chaque arbre en un ensemble de petites composantes connexes. Cette étape est essentiellement basée sur la coloration forte stricte qui met en évidence des nœuds particuliers d'un arbre, appelés *nœuds de base*. Ces derniers permettent de déterminer deux types de composantes de l'arbre, les *étoiles* et les *bi-étoiles*. Dans le but d'avoir une décomposition unique pour chaque arbre, nous avons défini un ensemble de critères qui permettent de sélectionner de manière déterministe les composantes à extraire de l'arbre à chaque itération. La seconde étape calcule la distance d'édition exacte entre les composantes résultantes de la décomposition des arbres à comparer. Enfin, la troisième étape permet de détermine la distance entre deux arbres en se basant sur les distances d'édition entre leurs composantes. Ceci, en calculant le coût du meilleur couplage entre les composantes des deux arbres.
L'algorithme SSD permet de prend en entrée deux fichiers XML décrivant deux objets quelconques, qu'il représente sous forme d'arbres étiquetés, et retourne la distance entre eux.

En plus des algorithmes que nous avons conçus, d'autres algorithmes d'appariement que nous avons utilisés pour des comparaisons sont implémentés dans la plateforme GraMAP, comme par exemple, celui de Riesen and Bunke [10] pour le calcul de la distance d'édition entre deux graphes étiquetés, appelé EXACT (basé sur A*) et celui de Zeng et al. [11] pour l'approximation de la distance d'édition entre deux graphes étiquetés, appelé ZMD (Zeng's Mapping Distance).

2. Evaluations et collections

Tous les algorithmes d'appariement de graphes que nous avons implémentés dans GraMAP ont été étudiés analytiquement et expérimentalement, et comparés à d'autres algorithmes. De plus, la plateforme a été testée et validée avec différents benchmarks et collections de données synthétiques et réelles. Parmi ces données :

- Xmark (The XML benchmark project: http://www.xml-benchmark.org),
- Une base de représentation XML de la syntaxe du langage naturel, TreeBank (56385 arbres),
- Une base de services contenant 623 processus métiers [5],
- Deux bases de séquences de protéines, PSD (Protein Sequence Database : 262525 arbres) et SwissProt (50000 arbres). Une base de représentation XML de la syntaxe du langage naturel, TreeBank (56385 arbres).

Des échantillons de ces collections sont fournis sur la plateforme. Néanmoins l'utilisateur peut exécuter les différents algorithmes avec ses propres collections (les formats de données requis pour chaque algorithme sont donnés).

3. Résultats

Les résultats obtenus par les algorithmes mis en place dans la plateforme GraMAP sont décrits dans [1] [3] [6] [7] [8] [9]. Nous présentons ici quelques interfaces de la plateforme.
La page d'accueil est donnée par la Figure 1. Elle liste les différents algorithmes avec une brève explication pour chacun.

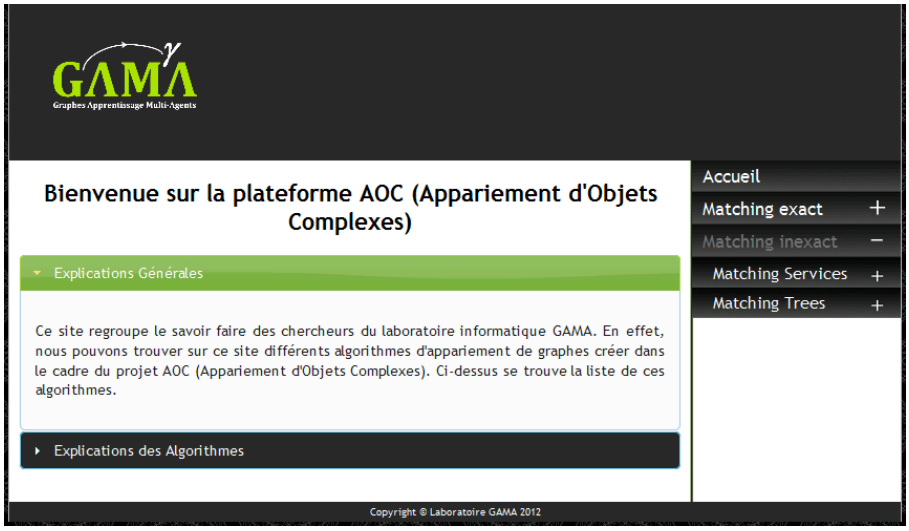

Figure 1 : Page d'accueil de la plateforme GraMAP

Une fois que l'utilisateur a choisi un des algorithmes à exécuter, il sera invité à choisir un fichier requête et un ensemble de fichiers cibles auxquels sera comparée la requête. Ces fichiers sont soit fournis par la plateforme soit par l'utilisateur. Puis, suivant l'algorithme, il est possible de choisir soit la métrique soit les algorithmes avec lesquels comparer.
Les Figures 2 et 3 présentent les formulaires à renseigner pour exécuter les algorithmes SCM4BP et SSD, respectivement.

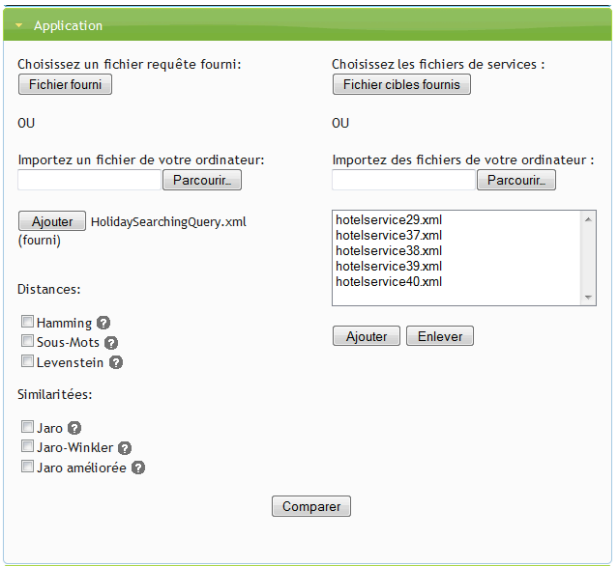

Figure 2 : Formulaire pour l'algorithme SCM4BP

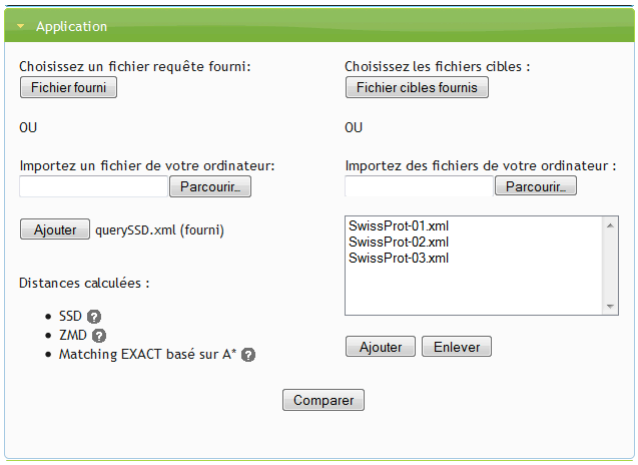

Figure 3 : Formulaire pour l'algorithme SSD

Les résultats des exécutions sont donnés par les Figures 4 et 5. La liste des fichiers cibles retournée est triée par ordre de pertinence par rapport au fichier requête.

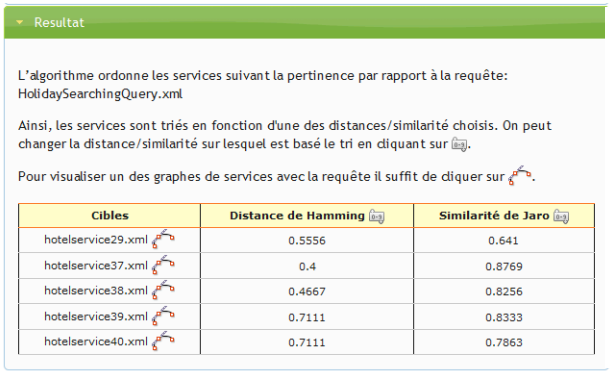

Figure 4 : Résultats de l'exécution de SCM4BP

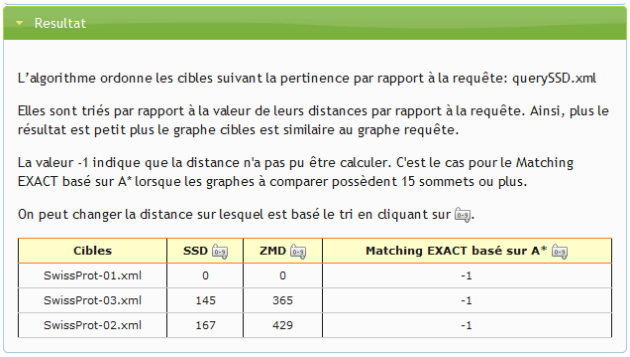

Figure 5 : Résultats de l'exécution de SSD et ZMD

4. Références

[1] N. Bruno, D. Srivastava, and N. Koudas. Holistic twig joins: optimal XML pattern matching, in Proceedings of ACM Special Interest Group on Management of Data, pp. 310-321, 2002.

[2] M.A. Tahraoui, H. Kheddouci. TwigStack++. A New Efficient Holistic Twig Join Algorithm, In International Journal of Information-Intelligence-Interaction (I3), 2011 N°2.

[3] Yacine Belhoul, Mohammed Haddad, Eric Duchêne and Hamamache Kheddouci. String Comparators Based Algorithms for Process Model Matchmaking: In the proceedings of the 9th International Conference on Service Computing (IEEE SCC 2012), Honolulu, Hawaii, USA, June 24-29, 2012.

[4] R. M. Dijkman, M. Dumas, and L. Garcia-Banuelos. Graph matching algorithms for business process model similarity search: In proceedings of BPM'09, 2009, pp. 48–63.

[5] A. Gater. Semantic process model benchmark. Rapport technique, PRiSM Laboratory, 2011.

[6] H. Seba, S. Lagraa, H. Kheddouci. Web Service Matchmaking by Subgraph Matching. Filipe and J. Cordeiro (Eds.): Web Information Systems and Technologies, Lecture Notes in Business Information Processing: LNBIP 101, pp. 43–56, 2012. Springer-Verlag Berlin Heidelberg (2012).

[7] S. Lagraa, H. Seba, H. Kheddouci. Matchmaking OWL-S processes: an approach based on path signatures. In the proceedings of The ACM International Conference on Management of Emergent Digital EcoSystems (MEDES 2011), San Francisco USA, Novembre , 2011 (doi>10.1145/2077489.2077521).

[8] S. Lagraa, H. Seba, R. Khennoufa and H. Kheddouci. A Graph decomposition approach to web service matchmaking, In the proceedings of 7th International Conference on Web Information Systems and Technologies (WEBIST), pp. 31-40, May, Hollande, 2011.

[9] Saïd Yahiaoui, Mohammed Haddad, Brice Effantin, Hamamache Kheddouci. Coloring Based Approach for Matching Unrooted and/or Unordered Trees. Submitted

[10] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing,27(7):950–959, 2009.

[11] Z. Zeng, A. Tung, J. Wang, J. Feng, and L. Zhou.Comparing stars: On approximating graph edit distance. Proceedings of the VLDB Endowment,2(1):25–36, 2009.