Le système D2CP. Un prototype pour la découverte dynamique des e-services.

Christophe Rey

Le système D^2CP

Un prototype pour la découverte dynamique des e-services.

Christophe Rey

Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)
CNRS UMR 2239 – Université Blaise-Pascal Clermont-Ferrand II
24 avenue des Landais 63177 Aubière cedex

rey@isima.fr

RÉSUMÉ. *Les e-services (ou services web) représentent un nouveau paradigme pour le e-commerce et promettent beaucoup : découverte dynamique, communication et négociation automatique, composition en des services complexes,... Jusqu'à ce jour peu de systèmes existants fournissent des implémentations alliant à la fois un stockage des e-services et des traitements évolués basés sur le Web sémantique pour les gérer. Cet article détaille le prototype D^2CP, une des premières implémentations d'un tel système, permettant à la fois la définition et le stockage des e-services dans des ontologies XML, et la découverte dynamique de combinaisons de e-services pour répondre à une requête utilisateur. Après un exemple de découverte dynamique, les fonctionnalités de D^2CP sont présentées (génération aléatoire d'ontologies, choix de l'algorithme de découverte), ainsi que les résultats des tests qui ont permis de le valider qualitativement et quantitativement.*

ABSTRACT. *E-services (or web services) are un new promising paradigm for e-commerce: dynamic discovery, automatic communication and negociation, composition into complex services,.. Until now, very few systems implements a way to store e-services and to process them intelligently, based on the Semantic Web principles. This paper is about such a prototype called D^2CP. D^2CP allows both the definition and the storage of e-services into XML-based ontologies, and the dynamic discovery of combinations of e-services to answer a user query. After an example of dynamic discovery are presented D^2CP capabilities (random ontologies generation, selection of the discovery algorithm), and also tests results that valited D^2CP.*

MOTS-CLÉS : *e-services, services web, découverte dynamique, ontologie, prototype*

KEYWORDS: *e-services, web services, dynamique discovery, ontology, prototype*

1. Introduction

Internet est en train de révolutionner la façon dont les entreprises interagissent avec leurs fournisseurs, partenaires et autres clients. Durant la dernière décennie, le nombre et la diversité des services se sont considérablement accrus sur le web, ce qui a abouti à l'apparition du e-commerce. Une initiative industrielle a récemment défini un nouveau paradigme pour le e-commerce dans lequel les applications sont encapsulées et présentées sous la forme de services électroniques intégrés : les e-services (ou services web) [Wei01, Vld01]. Un e-service peut être défini comme une application rendue disponible sur Internet par un fournisseur, et accessible à des clients [CAS 01b]. Des services de réservation en ligne, de gestion de comptes bancaires, ou même des applications métiers entières en sont des exemples actuellement courants. Ce qui rend la notion de e-service si attirante est qu'il semble que les e-services soient en mesure de permettre une meilleure automatisation dans la gestion des applications, par exemple en permettant leur découverte dynamique, en facilitant la communication et la négociation, voire en permettant leur composition en des services plus complexes [CAS 01a, Wei01].

Dans ce contexte, le problème de la découverte dynamique des e-services est le suivant : étant donnés une requête utilisateur et un ensemble de e-services, comment trouver la combinaison de e-services qui répond au mieux à la requête ?

Actuellement, les e-services sont utilisés au travers de registres qui permettent leur définition, stockage et découverte [CAS 01b]. Ce sont par exemple les registres UDDI utilisant le langage de description de services WSDL, ou des registres ebXML[1] utilisant UML. Les algorithmes de découverte y sont néanmoins limités par le fait qu'ils sont basés sur une recherche par mots-clés, ou par tables de correspondance de couples (clé, valeur) [BER 02]. De plus, pour une requête, ces algorithmes découvrent des e-services séparés, et non des combinaisons de e-services couvrant l'ensemble de la requête.

Dans le cadre du Web sémantique, dont le but est de baser les traitements d'informations du Web sur une définition sémantique de celles-ci, de nouveaux langages pour une description formelle plus riche et précise des e-services ont été proposés : les logiques de description (LD), famille de langages issus de la logique du premier ordre limitée à deux variables, y occupent ainsi une place centrale, grâce aux différents niveaux d'expressivité qu'elles permettent et au nombre important de mécanismes d'inférence (appelés raisonnements) dont elles disposent [HOR 02b, HOR 02a, FEN 02, LUT 02, ANK 02]. Ces langages, comme par exemple DAML-S[2], ont permis la création de dictionnaires structurés, les ontologies, regroupant des définitions de e-services basés sur des concepts définis eux-mêmes dans l'ontologie. Grâce à ces langages et ontologies associées, de nouvelles approches pour une découverte dynamique basée sur la définition des e-services, et plus uniquement sur des mots-clés, ont été définies [BER 02, SYC 02, PAO 02, GON 01]. Dans ce contexte, nous avons proposé, sous

1. Voir *http ://www.uddi.com*, *http ://www.w3.org/TR/wsdl*, *http ://openebxml.sourceforge.net*
2. Voir *http ://www.daml.org/services/*

la forme d'un nouveau raisonnement pour les LD, une découverte qui a l'avantage d'être la seule à notre connaissance à permettre la découverte de *combinaisons* de e-services pour répondre à une requête, et à fournir la différence entre la requête et les combinaisons découvertes sous la forme d'un concept réutilisable pour d'autres traitements (initiation d'un dialogue système-utilisateur par exemple). Nous présentons ces résultats dans [HAC 02b] et [HAC 02c], et l'algorithme correspondant appelé *computeBCov* dans [REY 03].

Dans cet article, nous nous présentons le système D^2CP qui est le prototype qui implémente *computeBCov* et qui permet ce nouveau raisonnement pour la découverte dynamique de e-services, étant données une ontologie de e-services et une requête utilisateur au format XML. Nous présentons la conception, les expérimentations et les évaluations de ce système qui, à notre connaissance, est un des premiers prototypes qui va aussi loin dans l'implémentation d'un algorithme de découverte fondé sur l'approche Web sémantique des e-services et utilisant les LD.

Cet article est organisé de la manière suivante. La section 2 résume, à partir d'un exemple, le problème de la découverte dynamique de e-services (détaillé dans [HAC 02b] et [HAC 02c]). Dans la section 3 le système D^2CP et ses principales caractéristiques sont présentés. Dans la section 4, nous discutons des tests effectués avec D^2CP. Enfin, nous concluons en envisageant les travaux à venir.

2. Découverte dynamique de e-services

Nous détaillons ici un exemple de découverte dynamique de e-services, avec une ontologie décrivant des concepts du domaine du tourisme. (voir figure 1).

L'ontologie présentée en figure 1 est organisée en trois couches : une pour les définitions de concepts généraux comme heure et date, une pour les concepts sur le tourisme comme voyage et logement, et une pour les e-services. Chaque couche est définie en fonction des concepts définis dans les précédentes, en fonction de concepts de base appelés concepts atomiques ("chaîne", "entier" par exemple) et en fonction des rôles atomiques qui représentent les liens entre concepts (comme par exemple "lieu de départ" qui relie le concept défini "voyage" au concept atomique "chaîne"). Comme cela est expliqué dans [COR 03], cette architecture à trois couches est une particularité de MKBEEM : cela permet de définir des e-services indépendamment de tout fournisseur, ce qui rend la plate-forme de e-commerce bien plus réactive face à la grande mouvance du monde des e-services.

En figure 2 est donnée une requête Q qui pourrait être posée par un utilisateur. Les combinaisons de e-services qui sont découvertes dynamiquement à partir de Q et de l'ontologie de la figure 1 sont détaillées dans la figure 3. Après la réécriture de la requête comme une combinaison de e-services, nous pouvons voir en figure 3 que l'utilisateur peut connaître la partie de sa requête qui n'a pas été comprise (i.e. qui ne correspond à aucun e-service), c'est le "rest", et la partie de la combinaison des e-services qui n'a pu être mise en correspondance avec aucun élément de la requête,

Ontologie

Ontologie	Concept défini	Définition de concept
Ontologie globale	*heure*	Conjonction d'un entier pour les heures et d'un entier pour les minutes.
	date	Conjonction d'un entier pour le jour, un pour le mois, un pour l'année et d'une chaîne pour le jour de la semaine.
	nombre	Un entier.
Ontologie du tourisme	*voyage*	Conjonction d'une chaîne pour le lieu de départ, une pour le lieu d'arrivée et d'un moyen de transport.
	logement	Conjonction d'une chaîne pour le lieu et d'une *date* pour le premier jour.
Ontologie des e-services	*hôtel*	Sorte de *logement* avec un nombre de lits et une chaîne indiquant la catégorie de l'hôtel.
	appartement	Sorte de *logement* avec un nombre de chambres et une chaîne pour la catégorie d'appartement.
	horaire1	Sorte de *voyage* avec une *heure* de départ et une *date* de départ.
	horaire2	Sorte de *voyage* avec une *heure* d'arrivée et une *date* d'arrivée.

Description des e-services

hôtel	Permet de consulter une liste d'hôtels en donnant quelques informations parmi le lieu, la date du premier jour, le nombre de lits ou la catégorie de l'hôtel.
appartement	Permet de consulter une liste d'appartements en donnant quelques informations parmi le lieu, la date du premier jour le nombre de chambres et la catégorie d'appartement.
horaire1	Permet de consulter une liste de voyage étant donnés les lieux de départ et arrivée et l'heure et la date de départ.
horaire2	Permet de consulter une liste de voyage étant donnés les lieux de départ et arrivée et l'heure et la date d'arrivée.

Figure 1. *Une petite ontologie à trois couches sur le tourisme.*

Q	J'arriverai à Paris lundi prochain et je voudrais un logement avec piscine. Demande d'une combinaison de e-services contenant un lieu d'arrivée, associé avec une date d'arrivée, et un logement, associé avec un équipement de loisir.

Figure 2. *Exemple de requête utilisateur Q.*

c'est le "miss". Les combinaisons découvertes sont meilleures par rapport aux autres combinaisons possibles par le fait que ces rest et miss ont été tour à tour minimisés. Afin que les e-services découverts puissent être exécutés, il faut que l'utilisateur renseigne toutes les informations des e-services pour ne plus avoir de miss. Le système peut alors se baser sur ce miss pour établir un dialogue avec l'utilisateur pour qu'il complète sa requête.

Dans [HAC 02b] et [HAC 02c], nous avons formalisé la découverte dynamique de e-services dans le contexte du Web sémantique : nous avons défini un nouveau raisonnement complexe pour les LD appelé la recherche des meilleures couvertures d'un concept en utilisant une terminologie (i.e. une ontologie dans le contexte LD). Nous avons montré que c'était en fait une nouvelle instance du cadre formel du pro-

Combinaisons de e-services	rest	miss
horaire2, appartement	équipement de loisir	lieu de départ, moyen de transport, heure d'arrivée, nombre de chambres et catégorie d'appartement
horaire2, hôtel	équipement de loisir	lieu de départ, moyen de transport, heure d'arrivée, nombre de lits et catégorie d'hôtel

Figure 3. *Résultats de la découverte dynamique de e-services pour la requête Q.*

blème de réécriture de concepts en utilisant une terminologie, cadre formalisé dans [BAA 00a, BAA 00b].

Il est à noter que l'ontologie de la figure 1, la requête de la figure 2 et les résultats de la figure 3 sont ici présentés de manière intuitive, alors qu'ils sont exprimés, dans D^2CP, à l'aide d'une logique de description, moins immédiatement compréhensible. Par ailleurs, cette logique doit avoir la propriété de subsomption structurelle définie dans [TEE 94]. En effet, c'est pour ces langages qu'a été démontrée l'équivalence de la recherche des meilleures couvertures d'un concept avec le problème **NP**-difficile de la recherche des minimaux transversaux de coût minimal dans un hypergraphe [HAC 02a]. Dans [REY 03], nous avons proposé un algorithme appelé *computeBCov* pour résoudre ce problème. Ainsi, *computeBCov* est le cœur du système D^2CP.

3. Le prototype D^2CP

Dans cette partie, nous présentons le prototype Java dans lequel est implémenté *computeBCov*. Nous l'avons appelé D^2CP pour "Dynamic Discovery of Concepts Prototype", ou prototype de découverte dynamique de concepts. Après la présentation générale des caractéristiques de D^2CP (section 3.1), nous présenterons les fondements de l'algorithme *computeBCov* (section 3.2), nous montrerons comment ce système peut aider l'utilisateur à générer des ontologies de tests (section 3.3), à choisir les différentes variantes de *computeBCov* pour effectuer une découverte dynamique de e-services (section 3.4) et sous quelles formes les résultats d'exécution peuvent être obtenus (section 3.5).

3.1. *Vue d'ensemble de* D^2CP

Le but de ce système est d'être une plate-forme de tests de l'algorithme *computeBCov* de découverte dynamique de e-services. Pour ceci, D^2CP permet :

(i) de générer aléatoirement des ontologies de e-services et des requêtes associées sous la forme de fichiers XML. Voir la section 3.3.

(ii) de découvrir dynamiquement les meilleures combinaisons de e-services étant données une requête Q et ontologie :

- fournies par l'utilisateur,
- ou générées par l'utilisateur à l'aide de D^2CP.

(iii) de faire tourner les 6 variantes de l'algorithme *computeBCov* (voir la section 3.2).

(iv) d'obtenir les résultats de la découverte dynamique.

- sous la forme d'une trace d'exécution en arbre
- ou sous la forme de mesures de temps de chaque étape de l'algorithme.

La figure 4 résume toutes ces fonctionnalités.

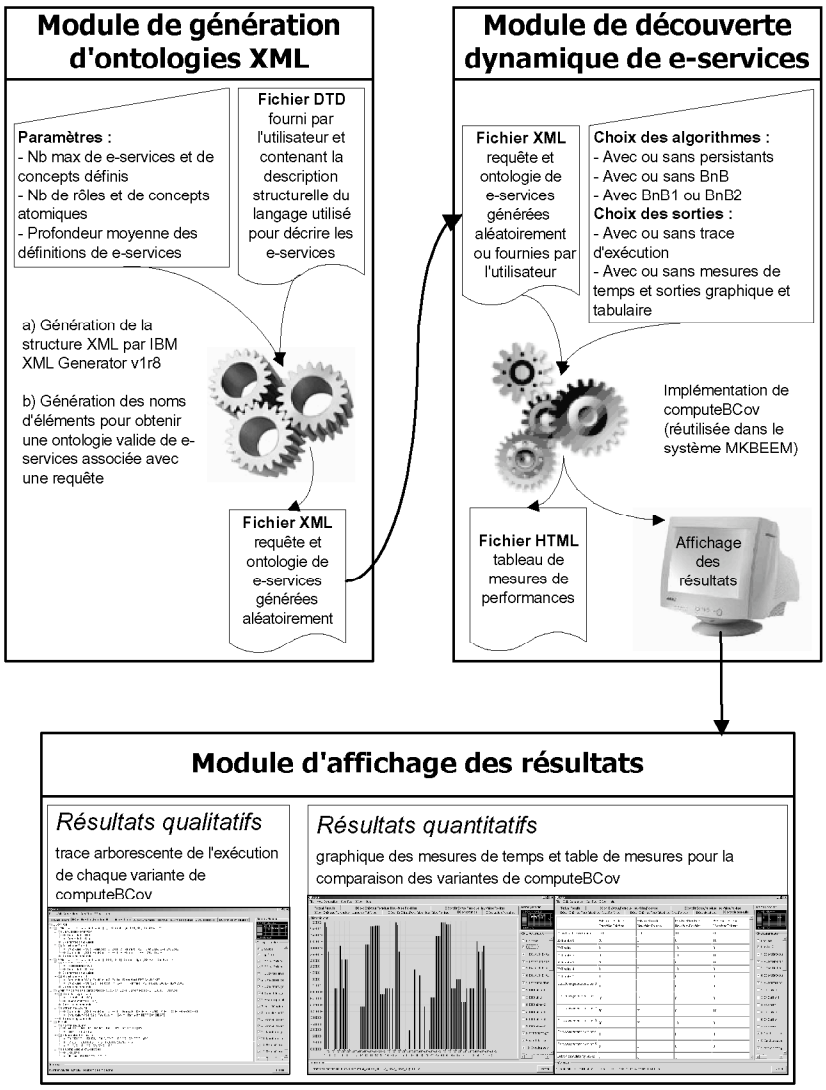

Figure 4. *Vue d'ensemble des fonctionnalités de D^2CP.*

3.2. *L'algorithme computeBCov*

computeBCov repose sur une propriété classique des hypergraphes donnée dans [BER 89] et reprise dans [MAN 94, EIT 95], qui en fait un algorithme itératif qui, à chaque itération, génère un certain nombre de candidats pour ne conserver que les plus petits (au sens de l'inclusion ensembliste) à la fin de l'itération. Dans [REY 03], on voit que cette phase de génération de candidats peut être optimisée par deux ajouts :

– La propriété des "persistants" établit une condition nécéssaire et suffisante de minimalité d'un candidat. Grâce à elle, on peut donc ne générer que les candidats minimaux à chaque itération, et ce au prix d'un coût bien moindre que celui impliqué par l'élagage des non minimaux après la génération de tous les candidats possibles.

– Une technique d'optimisation combinatoire appelée "Branch and Bound" (BnB) peut être ajoutée pour améliorer encore l'algorithme de base. Elle permet d'élaguer de nombreuses branches dans l'arbre des candidats possibles grâce à l'évaluation, à chaque itération, du coût d'une solution possible, cette évaluation pouvant être calculée de deux manières possibles (BnB1 ou BnB2).

On a donc au total 6 variantes pour *computeBCov* : avec ou sans les persistants, avec ou sans le BnB, et si avec le BnB, alors avec BnB1 ou avec BnB2. Ces 6 variantes sont toutes implémentées dans D^2CP (voir [REY 03] pour les algorithmes détaillés et tous les détails techniques).

3.3. *Génération d'ontologies*

Le module de génération d'ontologies de D^2CP permet de générer des ontologies sous forme de fichiers XML selon une DTD donnée par l'utilisateur qui décrit la structure de base du langage utilisé pour décrire les e-services dans l'ontologie. Ce langage doit appartenir à l'ensemble des LD à subsomption structurelle pour suivre le cadre formel de la découverte dynamique de e-services défini dans [HAC 02b]. Cette génération est divisée en deux parties :

– d'abord la structure de l'ontologie XML est créée grâce à l'utilisation du IBM XML Generator v1r8[3],

– ensuite, les noms des éléments sont générés afin d'obtenir une ontologie acyclique, valide pour le langage utilisé dans le processus de découverte dynamique de e-services.

Le résultat de la génération est un fichier XML qui contient un ensemble de définitions de concepts construits avec d'autres concepts précédemment définis et avec les rôles et concepts atomiques, ainsi que des définitions de e-services construits de la même façon. De plus, une requête est aussi générée aléatoirement dans le fichier XML. La génération assure l'absence de cycle dans les définitions générées. Le fichier XML peut alors être pris comme entrée du module de découverte dynamique.

3. Voir http ://www.alphaworks.ibm.com/tech/xmlgenerator.

Pour ajuster la génération aléatoire de toutes ces informations, l'utilisateur de D^2CP peut ajuster 7 paramètres, soit en modifiant directement le fichier DTD, soit par l'intermédiaire de l'interface du système :

– Le nombre maximal de définitions de concepts et de définitions de e-services à générer dans l'ontologie.

– Le nombre maximal de clauses dans la définition des e-services (chaque e-service est une conjonction de clauses, une clause étant une information précise décrivant une petite partie d'un e-service).

– La proportion de concepts définis par rapport aux e-services.

– La proportion de chaque constructeur[4] de la logique de description choisie dans les définitions de concepts ou de e-services.

– Le nombre de concepts atomiques maximal dans l'ontologie.

– Le nombre de rôles atomiques maximal dans l'ontologie.

– La profondeur moyenne des e-services, c'est à dire le nombre moyen de concepts imbriqués dans la définition des e-services.

Pour plus de détails sur le réglage de ces paramètres, voir [REY 03].

3.4. *Les variantes de computeBCov*

D^2CP permet à l'utilisateur d'exécuter les 6 variantes de *computeBCov* pour effectuer la découverte dynamique de e-services. Ceci se fait par l'intermédiaire de l'interface présentée en figure 5. D'ailleurs, sur cette figure[5], on peut voir que l'utilisateur a décidé d'exécuter 3 variantes de *computeBCov* :

– une avec BnB (et BnB1) et avec les persistants,

– une avec BnB (et BnB2) et avec les persistants

– et une sans BnB et avec les persistants.

3.5. *Affichage des résultats*

La figure 5 montre aussi que l'utilisateur peut choisir le type d'affichage (voir la figure 4) qu'il souhaite parmi :

(i) un affichage textuel simple des meilleures combinaisons de e-services (pas de trace ni de statistique),

---

4. Chaque logique de description est définie par un ensemble de constructeurs logiques qui assure le niveau d'expressivité de la logique correspondante. Les quantificateurs universel "$\forall$" et existentiel "$\exists$" sont des exemples de constructeurs.

5. La ligne en grisé évoquant une "scheduling heuristic" représente une heuristique d'ordonnancement qui n'a pas encore été implémentée dans D^2CP.

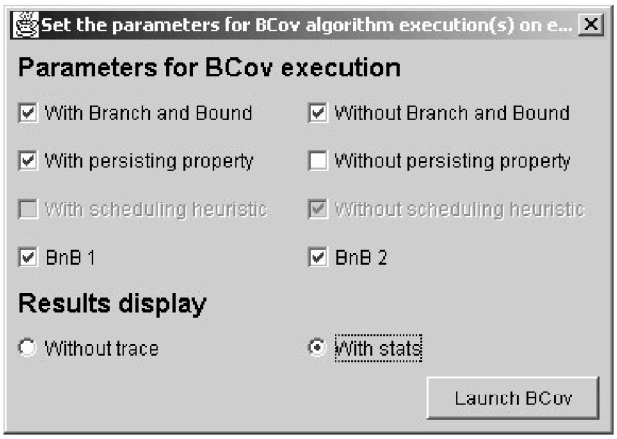

Figure 5. *Interface de D^2CP qui permet de choisir les variantes de computeBCov à exécuter.*

(ii) une trace arborescente de chaque exécution, didactique et utile pour vérifier le bon déroulement de chaque variante exécutée,

(iii) et un affichage des statistiques d'exécution qui sont des mesures de chaque étape de *computeBCov* destinées à identifier les performances respectives de chaque variante. Ces statistiques peuvent être présentées :

- sous la forme d'un histogramme

- ou sous la forme d'un tableau de valeurs, par ailleurs stocké dans un fichier HTML afin de servir d'entrée à un tableur.

4. Expérimentations

Dans cette partie, nous présentons les expérimentations qui ont été menées avec D^2CP et qui ont permis de valider la découverte dynamique de e-services ainsi que de déterminer les variantes les plus efficaces de *computeBCov*. Les tests présentés dans cette partie, sauf en ce qui concerne la validation MKBEEM, ont été effectués sur une machine de type PC, avec un processeur Intel Pentium 3 à 500MHz et 320 Mo de RAM.

4.1. *Validation du noyau de D^2CP avec le système MKBEEM*

Le noyau de D^2CP est l'ensemble des classes Java qui implémente les structures de données (l'hypergraphe principalement) et les variantes de *computeBCov* néces-

saires pour la réalisation de la découverte dynamique de e-services à partir d'un fichier XML contenant une ontologie de e-services et une requête.

Ces classes ont été utilisées dans le cadre du projet MKBEEM[6], un médiateur pour le e-commerce dont le but est d'être le support de marchés électroniques offrant des services multilingues et intelligents, basés sur des connaissances regroupées en ontologies, dans le domaine du tourisme et de la vente par correspondance. L'exemple de la partie 2 est extrait des scenarii rendus possibles par le prototype MKBEEM. Ces scenarii, de type vente en ligne et e-services de voyage/tourisme, ont permis de valider le système, et entre autres la découverte dynamique mise en jeu, à une échelle européenne (France et Finlande) et en trois langues (Finlandais, Anglais et Français).

Dans les premières expériences menées avec le prototype MKBEEM, nous avons modélisé de petites ontologies (environ 500 concepts définis et 50 e-services) sur le domaine du tourisme notamment. Le traitement des requêtes utilisateurs s'est avéré qualitativement satisfaisant, les combinaisons de e-services découverts étaient pertinentes, et quantitativement intéressant, le temps total dédié à la découverte n'a jamais dépassé quelques secondes.

Les tests n'ont pas encore été menés sur des ontologies de tailles plus réalistes pour des raisons de temps : c'est en effet un travail de longue haleine que de modéliser un domaine complet ainsi qu'une offre importante et réaliste de e-services sous la forme d'une ontologie. Pour pallier à ce problème, et poursuivre les tests de la découverte dynamique, nous avons choisi de générer des ontologies. D'abord nous avons généré des ontologies qui représentaient les pires cas théoriques pour *computeBCov*, puis nous avons généré des ontologies aléatoires de plus grandes tailles pour le passage à l'échelle. Les tests sur ces ontologies sont respectivement présentés dans les deux sections qui suivent.

4.2. *Pires cas pour computeBCov*

Dans [REY 03], une étude de complexité de l'algorithme *computeBCov* aboutit à la construction de deux hypergraphes H_1 et H_2 qui représentent des pires cas pour cet algorithme. A partir de H_1 et H_2, nous avons construit les ontologies XML correspondantes et nous avons testé les 6 variantes de *computeBCov* sur celles-ci. Les résultats sont montrés en figure 6. Nous insistons ici sur le fait que H_1 et H_2 représentent des cas vraiment mauvais pour *computeBCov* et par là même irréalistes car aboutissant à des milliers de combinaisons de e-services possibles : 26 e-services et 8192 meilleures combinaisons pour H_1 et 28 e-services et 13056 meilleures combinaisons pour H_2.

La figure 6 nous montre que même pour ces cas très difficiles à traiter en raison du nombre élevé de meilleures combinaisons, il existe des variantes de *computeBCov* (celles avec les persistants) qui s'exécutent en un temps raisonnable.

---

6. MKBEEM est l'acronyme de Multilingual Knowledge Based European Electronic Marketplace (IST-1999-10589, 1er Fév. 2000 - 1er Déc. 2002). Voir *http ://www.mkbeem.com*

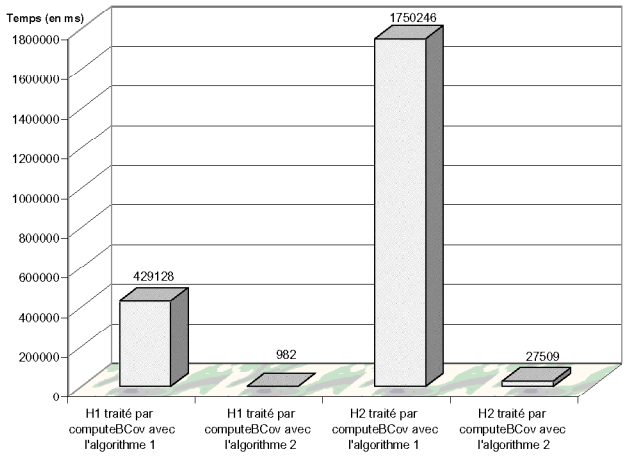

Figure 6. *Temps total d'exécution, en millisecondes, de computeBCov sur H_1 et H_2 par D^2CP. "Algorithme 1" repère les variantes de computeBCov sans les persistants et "Algorithme 2" celles avec les persistants.*

4.3. *Ontologies synthétiques*

Nous étudions ici trois ontologies générées par D^2CP, qui sont de plus grandes tailles que celles issues du contexte MKBEEM et sur lesquelles nous avons testé *computeBCov*. Après avoir présenté ces trois cas, nous discutons des temps d'exécution de chaque variante sur chacun des cas.

Les caractéristiques de ces trois cas sont résumées à la figure 7. Pour étudier ces exemples et leurs caractéristiques, nous considérons dans la figure 7 une "ontologie repère" qui représente une ontologie possédant des caractéristiques qui nous semblent se rapprocher d'une ontologie réaliste[7].

La figure 8 positionne chaque cas par rapport à l'ontologie repère, d'après les valeurs de la figure 7. Pour établir si le cas est favorable ou non par rapport à *computeBCov*, on raisonne sur les nombres de concepts et rôles atomiques : plus ces valeurs sont petites, plus ces concepts et rôles atomiques auront été réutilisés durant la génération aléatoire. Comme la requête est aussi générée aléatoirement, de petites valeurs pour ces nombres impliquent que la requête partage des informations communes avec beaucoup de e-services. Ceci implique beaucoup de combinaisons possibles, et donc une découverte dynamique plus lente a priori.

7. Ces valeurs prennent en compte l'expérience acquise avec le système MKBEEM. Elles restent arbitraires et sujettes à ajustements.

Caractéristique	Ontologie repère	Cas 1	Cas 2	Cas 3
Nombre de concepts définis	2500	365	1334	3405
Nombre de e-services	500	366	660	570
Nombre de clauses dans la requête	15	6	33	12
Nombre moyen de e-services pour une clause (i.e. de e-services dans lesquels se retrouve une clause de la requête)	20	10.83	20.84	30.75
Nombre de concepts atomiques	150	13	30	12
Nombre de rôles atomiques	100	13	30	12
Proportion de concepts définis réutilisés dans une définition de concept par rapport aux concepts atomiques.	30%	30%	20%	33%

Figure 7. *Caractéristiques de l'ontologie repère et des trois cas générés avec D^2CP.*

Ontologie	Description
Cas 1	Assez différente de l'ontologie repère de par sa petite taille, petite requête, cas très défavorable pour *computeBCov*.
Cas 2	Proche de l'ontologie repère mais de petite taille, requête de taille normale, cas défavorable pour *computeBCov*.
Cas 3	Très proche de l'ontologie repère, petite requête, cas très défavorable pour *computeBCov*.

Figure 8. *Résumé des trois ontologies générées avec D^2CP.*

Les temps globaux d'exécution des 6 variantes de *computeBCov* dans D^2CP sont donnés en figure 9.

En ce qui concerne l'efficacité globale de *computeBCov* dans D^2CP, la figure 9 montre que, pour les cas 1 et 3, il existe au moins une variante qui effectue la découverte dynamique en moins de deux secondes, et en 30 secondes pour le cas 2. En ce qui concerne l'efficacité relative de chaque variante, ces tests montrent qu'il y a une grande différence de performance suivant la variante utilisée. Pour résumer, nous pouvons dire d'après la figure 9 que :

(i) L'optimisation par le Branch and Bound est très efficace :

- pour le cas 1, même sans les persistants, le BnB permet d'obtenir une solution en temps réel (moins d'une seconde)

- et pour les cas 2 et 3, le BnB permet d'envisager l'obtention d'une solution dans un temps limité sans les persistants, voire presque en temps réel s'il est associé aux persistants. Sans le BnB, ces cas sont intraitables (plus de 12 heures).

L'effet espéré du Branch and Bound, qui est de limiter l'explosion combinatoire du problème, est donc bien présent. Cet effet est encore accentué par l'usage de BnB2 au lieu de BnB1 : comme BnB2 permet une meilleure évaluation d'une solution possible

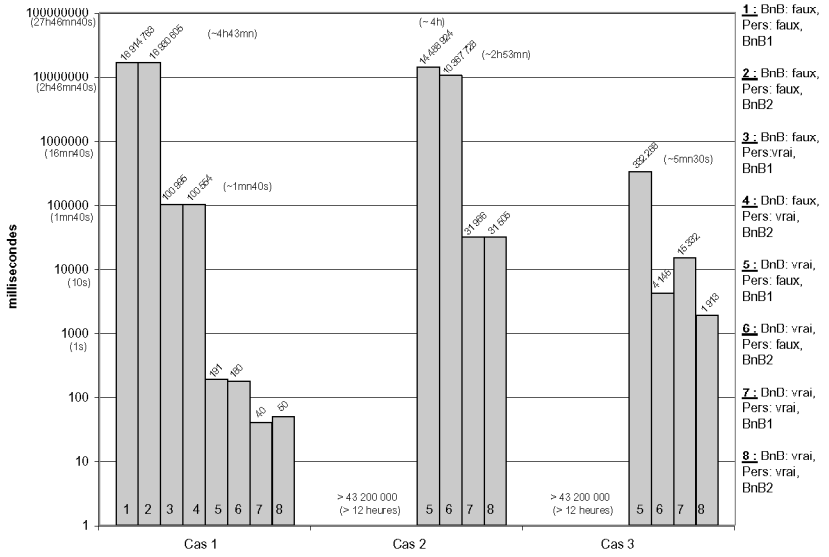

Figure 9. *Temps global d'exécution, en millisecondes, de chaque variante de computeBCov pour les 3 cas d'étude dans* D^2CP.

à chaque itération, le BnB implémenté avec BnB2 élague plus de branches dans l'arbre des combinaisons.

(ii) Les persistants sont aussi très intéressants, mais uniquement associés avec le BnB. Ils n'ont pas d'effet sur le nombre de combinaisons explorées, et donc ne limitent pas l'explosion combinatoire, mais ils accélèrent très sensiblement la génération des combinaisons à explorer. Dans le cas 2, associés avec le BnB, ils permettent l'obtention des solutions en un temps raisonnable (environ 30 secondes). Dans le cas 3, ils permettent l'obtention des solutions presque en temps réel (quelques secondes).

Ainsi la variante la plus performante est celle qui associe le BnB implémenté avec la méthode BnB2 et les persistants. Cette variante de *computeBCov* permet d'obtenir dans D^2CP une découverte dynamique de e-services performante pour les cas d'études précédents. C'est évidemment cette variante qui a été utilisée pour valider la découverte dynamique de e-services au sein du projet MKBEEM. Les cas étudiés étant de tailles respectables, proches de l'ontologie repère et de nature assez défavorable pour *computeBCov*, on peut être optimiste quant aux performances de D^2CP sur des ontologies réelles qui restent à construire.

5. Conclusion

Dans cet article, nous avons présenté le système D^2CP qui permet la découverte dynamique de combinaisons de e-services, étant données une ontologie de e-services et une requête stockées dans un fichier XML. Cet découverte dynamique est permise par l'algorithme *computeBCov*[REY 03]. Grâce aux ontologies synthétiques de quelques milliers de concepts générées par D^2CP, on a pu tester avec succès le passage à l'échelle de *computeBCov*. Qualitativement, le noyau de D^2CP a été validé au sein du projet européen MKBEEM sur de petites ontologies. Le travail à venir est donc la construction d'ontologies réalistes en taille et en contenu dans un ou plusieurs domaines pour tester les performances de D^2CP dans des conditions réelles. Les résultats des tests déjà effectués sont cependant très encourageants et de bonne augure. Le prototype D^2CP est à notre connaissance un des tous premiers système à proposer une implémentation aussi poussée d'un processus de découverte dynamique de e-services, et le seul qui permet la découverte de combinaisons de e-services. C'est aussi un des premiers systèmes qui prouve concrètement le fort potentiel de l'approche Web sémantique dans le traitement des e-services.

6. Bibliographie

[ANK 02] ANKOLEKAR A., BURSTEIN M., HOBBS J. R., LASSILA O., MCDERMOTT D., MARTIN D., MCILRAITH S. A., NARAYANAN S., PAOLUCCI M., ET AL. T. P., « DAML-S : Web Service Description for the Semantic Web », *Proceedings of the 1st Int'l Semantic Web Conf. (ISWC 02)*, 2002.

[BAA 00a] BAADER F., KÜSTERS R., MOLITOR R., « Rewriting Concepts Using Terminologies », COHN A. G., GIUNCHIGLIA F., SELMAN B., Eds., *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, San Francisco, CA, 2000, Morgan Kaufmann Publishers, p. 297–308.

[BAA 00b] BAADER F., KÜSTERS R., MOLITOR R., « Rewriting Concepts Using Terminologies – Revisited », Report n° 00-04, 2000, LTCS, RWTH Aachen, Germany, See http ://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[BER 89] BERGE C., *Hypergraphs*, vol. 45 de *North Holland Mathematical Library*, Elsevier Science Publishers B.V. (North-Holland), 1989.

[BER 02] BERNSTEIN A., KLEIN M., « Discovering Services : Towards High-Precision Service Retrieval. », *Proceedings of the Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27-28, 2002, Revised Papers*, vol. 2512 de *Lecture Notes in Computer Science*, Springer, 2002, p. 260–276.

[CAS 01a] CASATI F., SHAN M.-C., « Dynamic and adaptive composition of e-services », *Information Systems*, vol. 26, n° 3, 2001, p. 143-163.

[CAS 01b] CASATI F., SHAN M.-C., « Models and Languages for Describing and Discovering E-Services », *Proceedings of SIGMOD 2001, Santa Barbara, USA*, May 2001.

[COR 03] CORCHO O., GOMEZ-PEREZ A., LÉGER A., REY C., TOUMANI F., « An ontology-based mediation architecture for E-commerce applications », *Proceedings of Intelligent Information Systems 2003, Zakopane, Poland, June 2-5*, To appear, 2003.

[EIT 95] EITER T., GOTTLOB G., « Identifying the Minimal Transversals of a hypergraph and Related Problems », *SIAM Journal on Computing*, vol. 24, n° 6, 1995, p. 1278–1304.

[FEN 02] FENSEL D., BUSSLER C., MAEDCHE A., « Semantic Web Enabled Web Services », *International Semantic Web Conference, Sardinia, Italy*, juin 2002, p. 1–2.

[GON 01] GONZÁLEZ-CASTILLO J., TRASTOUR D., BARTOLINI C., « Description Logics for Matchmaking of Services », *Proc. of the KI-2001 Workshop on Applications of Description Logics Vienna, Austria*, vol. 44, September 2001.

[HAC 02a] HACID M.-S., REY C., TOUMANI F., « Dynamic Discovery of E-services (A Description Logics Based Approach). », Report, 2002, LIMOS, Clemont-Ferrand, France, see http ://lisi.insa-lyon.fr/∼mshacid/eServices.pdf.

[HAC 02b] HACID M., LÉGER A., REY C., TOUMANI. F., « Computing concept covers : a preliminary report. », *Proceedings of the International Workshop on Description Logics (DL02), April 19 to April 21, 2002. Toulouse. France*, April 2002.

[HAC 02c] HACID M., LÉGER A., REY C., TOUMANI F., « Dynamic discovery of e-services in a Knowledge Representation and Reasoning context. », *Proceedings of the 18th French conference on advanced databases (BDA), Paris, France*, October 2002.

[HOR 02a] HORROCKS I., P.F.PATEL-SCHNEIDER, VAN HARMELEN F. ., « Reviewing the Design of DAML+OIL : An Ontology Language for the Semantic Web », *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002)*, 2002, To appear.

[HOR 02b] HORROCKS I., « DAML+OIL : a Reason-able web ontology language », *Proceedings of the Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27*, vol. 2287 de *Lecture Notes in Computer Science*, Springer, 2002.

[LUT 02] LUTZ C., SATTLER U., « A Proposal for Describing Services with DLs », *Proceedings of the 2002 International Workshop on Description Logics*, 2002.

[MAN 94] MANNILA H., RäIHä K.-J., *The Design of Relational Databases*, Addison-Wesley, Wokingham, England, 1994.

[PAO 02] PAOLUCCI M., KAWAMURA T., PAYNE T., SYCARA K., « Semantic Matching of Web Services Capabilities. », *Proc. of the Int. Semantic Web Conference, Sardinia, Italy*, June 2002, p. 333–347.

[REY 03] REY C., TOUMANI F., HACID M.-S., LÉGER A., « An algorithm and a prototype for the dynamic discovery of e-services », Report, 2003, LIMOS, Clemont-Ferrand, France, see http ://www.isima.fr/∼rey/.

[SYC 02] SYCARA K., WIDOFF S., KLUSCH M., , LU J., « LARKS : Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace », *Autonomous Agents and Multi-Agent Systems*, vol. 5, 2002, p. 173–203.

[TEE 94] TEEGE G., « Making the Difference : A Subtraction Operation for Description Logics. », DOYLE J., SANDEWALL E., TORASSO P., Eds., *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, May 1994, p. 540–550.

[Vld01] « The VLDB Journal : Special Issue on E-Services », 10(1), Springer-Verlag Berlin Heidelberg, 2001.

[Wei01] « Data Engineering Bulletin : Special Issue on Infrastructure for Advanced E-Services », 24(1), IEEE Computer Society, 2001.