Composition Sémantique et Dynamique à base d'Agents des Services Cloud pour ERP

Hamza Reffad[1], Adel Alti [1], Philippe Roose [2]

1. *LRSD/Département d'informatique*
Université Farhat Abbas de Sétif, DZ-19000 Sétif
reffadh@yahoo.fr ;alti.adel @.univ-setif.dz

2. *LIUPPA – T2i*
IUT de Bayonne – Pays Basque, 64600, Anglet – France
Philippe.Roose@iutbayonne.univ-pau.fr

RESUME. Actuellement, la technologie Cloud est largement adoptée par les entreprises pour développer des solutions informatiques de qualité. En effet, les Petites et les Moyennes Entreprises (PME) sont à la recherche d'ERP « sur mesure » afin d'automatiser leurs activités commerciales. La complexité de la tâche de sélection et de composition de services augmente selon les évolutions des différents besoins fonctionnels et non fonctionnels des PME (contraintes et préférences). La plupart des systèmes ERP cloud existants (SAP, Oracle ERP cloud, etc.) ne sont pas suffisamment flexibles pour prendre en charge l'autoadaptation des processus d'affaire ERP. Cet article présente une nouvelle approche sémantique et dynamique à base d'agents pour la composition de services cloud afin d'obtenir un processus d'affaire ERP personnalisé. Une ontologie est proposée pour la description sémantique et la gestion du processus de construction ERP. Elle génère un processus d'affaire ERP virtuel selon les besoins fonctionnels du PME. Un algorithme en deux phases pour la composition des services cloud est proposé pour obtenir un processus d'affaire ERP optimal, concret et personnalisé. Les résultats expérimentaux montrent l'efficacité de l'approche proposé.

ABSTRACT. Nowadays, cloud technology is widely adopted by companies to develop quality computing solutions. Indeed, Small and Medium Enterprises (SMEs) are looking for the best customized ERP to automate their business activities. The complexity of the task of selection and composition of services increases with changes in the different functional and non-functional needs of SMEs (constraints and preferences). Most existing cloud ERP systems (SAP, Oracle, etc.) are not flexible enough to support ERP business process auto-adaptation. This article presents a new semantic and dynamic agent-based approach to cloud service composition to obtain a personalized ERP business process. A new ontology is proposed for the semantic description and management of the ERP construction process. It generates a virtual ERP business process according to the SME functional needs. A two-stage algorithm for the cloud services composition is proposed to obtain an optimal personalized concrete ERP business process. Experimental results show the effectiveness of the proposed approach.

Mots-clés : Cloud, optimisation, ontologie, service, NSGA-II, violation des contraintes.

KEYWORDS: Cloud; optimization, ontology, service, NSGA-II algorithm, constraints violation.

1. Introduction

Un logiciel Enterprise Resource Planning (ERP) est un outil qui permet le pilotage de l'entreprise. Il embarque, en un même logiciel et une seule base de données ainsi que les fonctionnalités nécessaires à la gestion de l'ensemble de l'activité d'une entreprise : gestion comptable, gestion commerciale, etc.

Vue la complexité et le cout élevé de ces ERP, les Petites et les Moyennes Entreprises (PME) recherchent un ERP sur mesure en tenant compte des évolutions de leurs activités. Avec la prolifération du Cloud Computing, les grands fournisseurs de systèmes ERP tels que SAP, Sage et Microsoft positionnent leurs offres ERP sous forme de modèle SaaS (Johansson et Ruivo, 2013). Néanmoins, ces systèmes ne sont pas suffisamment flexibles pour supporter les évolutions des besoins des entreprises (Li, 2009). La tendance vers une approche de composition de services cloud pour avoir un Processus d'Affaire (PA) offre deux avantages : la facilité d'intégration et la réduction des coûts (Tarantilis *et al.*, 2008). Ainsi, la disponibilité d'un grand nombre des services cloud hétérogènes avec différentes QoS sont offerts par plusieurs fournisseurs de services Cloud. Les développeurs ont tiré parti de ces services pour fournir un PA répondant aux besoins fonctionnels et non fonctionnels spécifiques des clients. Plusieurs méthodes d'optimisation de composition des services ont été proposées afin d'optimiser les paramètres de QoS (Sasikaladevi et Arockiam, 2012 ; Yu *et al.*, 2015 ; Asghari et Navimipour, 2016). Cependant, ces mécanismes ne tiennent pas compte des évolutions des contraintes et préférences du client. En plus, ils ne gèrent pas de manière efficace et flexible un nombre large de services hétérogènes. Cette hétérogénéité implique une dégradation de qualité de contrôle dans la sélection et la composition des services (Chang *et al.*, 2014).

Les clients souhaitent que les applications ERP puissent être personnalisées automatiquement en fonction de leurs besoins fonctionnels actuels, de leurs contraintes et préférences. Un filtrage sémantique permet d'améliorer la sélection des services en pénalisant les services qui violent les contraintes des clients. Cependant, le nombre des services Cloud et la qualité de service d'un fournisseur de Cloud n'est pas statique et peut évoluer dans le temps. Face aux besoins évolutifs des entreprises ainsi qu'à l'augmentation des services Cloud offrant différentes QoS, le développement des ERP personnalisés nécessite de proposer des solutions pertinentes qui répondent aux attentes des clients. Notre approche consiste à offrir au client un processus d'affaire ERP qui satisfait ses besoins fonctionnels en optimisant les QoS selon ses contraintes et préférences contextuelles.

Les contributions de ce travail sont : (1) l'extension de l'ontologie définit dans (Reffad et al., 2016) par l'inclusion de la description sémantique des contraintes et préférences contextuelles de l'entreprise cliente, (2) - développement d'un algorithme sémantique dynamique collaboratif en deux phases pour la composition des services Cloud. Dans la première phase, nous avons utilisé l'algorithme NSGA-II auquel nous avons ajouté une nouvelle relation de dominance basée sur une fonction de pénalité. Cette phase vise à sélectionner les services cloud composites pertinents qui respectent les contraintes des clients. La deuxième phase sélectionne un service cloud composite parmi les services engendrés par la première phase,

selon les préférences du client en utilisant la somme pondérée des QoS. À ce stade, nous avons proposé une nouvelle technique de calcul des poids de QoS.

Le reste de l'article est structuré comme suit : la section 2 détaille des travaux connexes. Dans la section 3, nous présentons notre approche en détails. Dans la section 4, des résultats expérimentaux sont présentés à l'aide des jeux de données aléatoires. Enfin, la section 5 conclut l'article en présentant quelques perspectives.

2. Travaux Connexes

Plusieurs approches ont été proposées pour l'optimisation de la composition des services Cloud. Les trois principales catégories d'approches sont les suivantes : approches basées sur les contraintes (Rosenberg *et al.,* 2009 ; Deng *et al.,* 2016), approches basées sur la classification sémantique (Alti *et al.,* 2014 ; Reffad *et al.,* 2016) et les approches basées sur les métaheuristiques (Sasikaladevi et Arockiam, 2012 ; Yu *et al.,* 2015 ;Chang *et al.,* 2015).

Pour les approches basées sur les contraintes, (Rosenberg *et al.,* 2009) ont proposé l'approche semi-automatique CaaS (Composition as a Service) combinée avec un langage spécifique pour le domaine nommé VCL (Vienna Composition Language). Ce langage est riche mais manque de descriptions sémantiques des services cloud hétérogènes. Ceci complique la tâche de décision au cours de la sélection des services. (Alti *et al.,* 2014) ont proposé une approche de composition sémantique automatique basée sur l'utilisation d'ontologies. Ils ont décrit une ontologie qui permet la génération automatique de l'assemblage de services hétérogènes de qualité. Ce travail ne tient pas compte de la croissance d'une variété de fournisseurs de services hétérogènes. Les approches basées sur les métaheuristiques distinguent deux principales catégories d'optimisation, pour la première, le problème multi-objectif peut être résolu en le réduisant à un problème mono-objectif via une technique de scalarisation. Nous pouvons distinguer les méthodes suivantes : recherche taboue (Pop et al., 2011), algorithme génétique (Sasikaladevi et Arockiam, 2012), optimisation des essaims de particules (Jun et Weihua, 2009), algorithme de concurrence impérialiste (Jula et al., 2011) et optimisation des colonies de fourmis (Yu *et al.,* 2015). Le principal inconvénient commun des méthodes de scalarisation est que la découverte du service composite avec le meilleur fitness peut pénaliser quelques QoS (i.e. si la fonction de fitness est la somme pondérée des critères QoS de sécurité et de temps de réponse, un service composite peut avoir le meilleur fitness avec une mauvaise sécurité). Pour le second, les approches d'optimisation multi-objectifs sont souvent utilisées lorsque deux objectifs conflictuels ou plus doivent être considérés simultanément afin de négocier un ensemble de solutions Pareto-optimales (NSGA-II, SPEA2 et E3-MOGA) (Huang *et al.,* 2012). Les approches Pareto-optimales génèrent un ensemble de solutions, mais dans de nombreuses situations, les clients n'ont besoin que d'une solution qui doit être sélectionnée automatiquement parmi les solutions résultantes. L'algorithme NSGAII est le plus populaire des algorithmes d'optimisation multi-objectif. Il permet d'atteindre le front Pareto-optimal par un minimum d'itérations.

Nous avons basé sur NSGA-II pour avoir un ensemble de solutions Pareto-optimal qui respecte les contraintes du client. Ensuite, une autre phase est exécutée pour sélectionner la solution finale selon les préférences du client.

Chen et al. (Chen *et al.*, 2015) ont défini une plate-forme CloudERP sur laquelle l'entreprise cliente pouvait personnaliser un système ERP entier correspondant à ses besoins sous forme d'un service composite. La méthode de composition de service proposée est basée sur un algorithme génétique avec la théorie 'rough set theory'.

Les travaux existants n'utilisent pas de mécanismes sémantiques au niveau du profil client pour satisfaire explicitement ses contraintes et ses préférences. De plus, ces travaux ne prennent pas la nature dynamique lié aux changements du contexte.

3. Approche dynamique collaboratif sémantique pour la composition des services cloud

3.1 Architecture générale

L'architecture générale de notre approche, illustrée à la figure 1, se base sur les agents durant la découverte, la sélection et la composition dynamique sémantique des services cloud afin d'obtenir un processus d'affaire ERP optimal selon les exigences du client. Elle se compose de trois entités :

– C**lient :** c'est l'entreprise qui a besoin d'un ERP personnalisé

– **Service Sémantique Proxy-Cloud :** il est constitué d'un agent planificateur ERP Virtual Composite Service (ERP-VCS), d'un agent maitre et des agents esclaves d'optimisation de service composite appelé ERP Cloud Composite Service (ERP–CCS-2S) et d'un gestionnaire du contexte.

- **Agent planificateur ERP-VCS :** est un moteur d'inférence sémantique chargé de générer le PA virtuel sous forme d'un Service Composite Virtuel (SCV) selon les besoins fonctionnels du client. Ce SVC se compose par des services virtuels (tâches). Chaque service virtuel (SV) est lié à un ensemble des services cloud concrets (SC) ayant les mêmes fonctionnalités avec différentes QoS.

- **Agent ERP–CCS-2S** : il prend en entrée les contraintes et les préférences du client décrites dans notre ontologie et la description sémantique des services Cloud (catégorie, rôle, paramètres d'entrées/sorties, paramètres de QoS et les spécifications de contexte), ensuite il produit un PA optimal sous forme d'un service cloud composite (SCC) en **fonction** des contraintes et préférences du client.

- **Agent de gestion sémantique du contexte** : le composant de gestion sémantique évalue dynamiquement le profil utilisateur (*besoins, contraintes, préférences*) et le contexte du service cloud (*connexion, taux de charge, services disponibles, etc.*) pour **sélectionner** les services cloud convenables à l'utilisateur. Chaque changement de contexte incite une réévaluation sémantique du contexte.

- **Le registre des services Cloud et profils clients** : contient la description sémantique des services cloud de tous les fournisseurs de services cloud et la description des profils des clients au sein de notre ontologie.

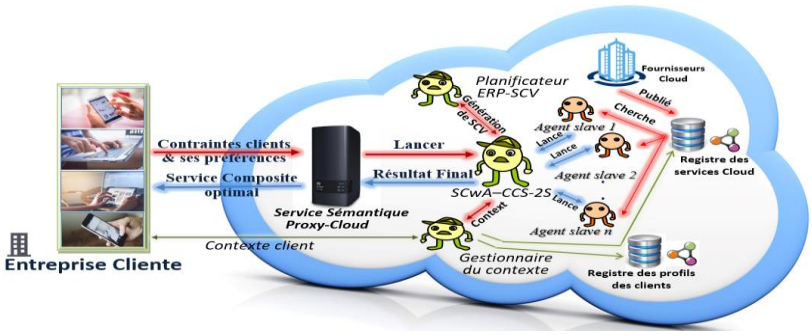

Figure 1. L'architecture générale de notre approche

3.2. Modèle ontologique

L'idée clé de l'ontologie proposée est l'assignement d'un niveau sémantique qui permet la description des services Cloud et des PAs et le filtrage sémantique des services. La figure 2 présente l'ontologie définie par les classes suivantes :

– **Client :** à un *nom*, des besoins fonctionnels, des contraintes et des préférences contextuelles. Un client spécifie ses besoins fonctionnels, ses contraintes et ses préférences visuellement à travers une interface utilisateur (GUI).

- Besoins fonctionnels : Les besoins fonctionnels peut être exprimé par des clauses 'AND' combinant des contraintes fonctionnelles simples. *Exemple*: (TaskCategory = `Buy' AND TransactionType=`Internet' AND TaskInput = `ProductDetails' AND TaskOutput =`Receipt`). Le moteur d'inférence génère un PA virtuel optimal sous forme d'une composition de services virtuels.

- Contraintes : Le client spécifie ses contraintes à partir d'une interface utilisateur. Il sélectionne une contrainte pour chaque QoS (temps de réponse : rapide, prix : moins cher, etc.). Pour unifier les valeurs de ces contraintes, elles sont transformées en valeurs sémantiques (*faible, moyenne, haute*). Ensuite, ces valeurs sémantiques sont mappées sur des intervalles voisins $\{I_1, I_2, ..., I_{nb}\}$ tels que $I_i \in [0,1]$. Finalement, la $i^{ème}$ contrainte sémantique de l'attribut q_i (notée C_i) est convertie en valeur quantitative :

$$C_i = \text{qmin}_i^{\text{semantic\_value}} \tag{1}$$

- Préférences : les préférences de QoS sont spécifiées par le client sous forme d'une liste ordonnée. Cet ordre attribue un nombre à chaque critère de QoS appelé explicitPriority. Ensuite, un poids w_i est assigné à chaque critère de QoS en fonction de son importance. Ce poids est calculé par l'équation 2.

$$w_i = \frac{e^{P_i}}{\sum_{j=1}^{n} e^{P_j}} \tag{2}$$

$$P_i = explicitPriority_i + constraintRank_i \times n \tag{3}$$

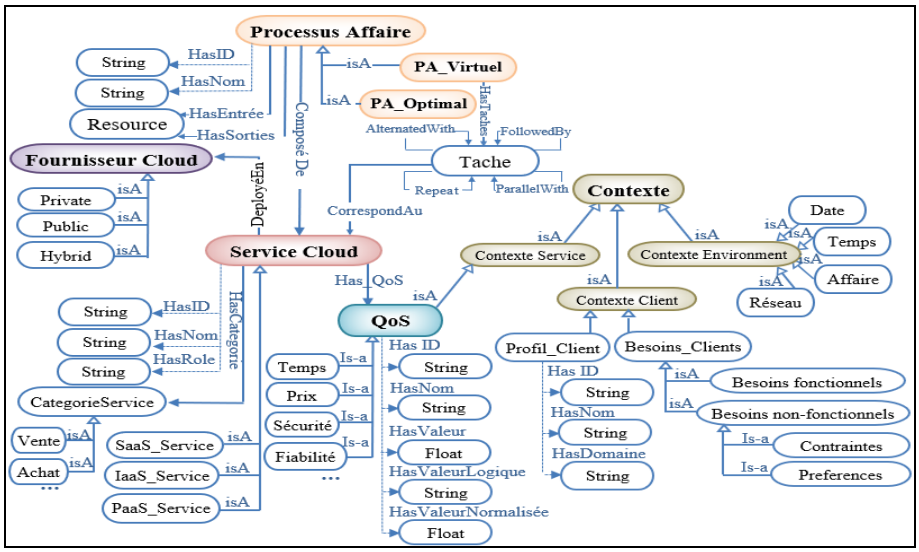

Figure 2. Ontologie générique d'un processus d'affaire pour ERP.

Où : n est le nombre de QoSs ; *explicitPriority* est un nombre entre 1 et *n* indiquant la priorité explicite d'une QoS. Si une préférence n'est pas spécifiée, son *explicitPriority est nulle* ; *constraintRank* est un nombre entier qui indique l'importance de la contrainte de QoS (0 : *aucun,* 1 : *faible,* 2 : *moyen* and 3 : *haut*).

– **Service :** L'ontologie couvre les services Cloud d'affaires, tels que l'achat, vente, finance, etc. Chaque service est étoffé par un ensemble d'attributs de QoS.

– **Processus d'affaire :** cette classe décrit tous les processus d'affaires en termes de séquences de tâches. Chaque tâche regroupe des services ayant une même fonctionnalité avec des QoS différentes.

– **Contexte :** le contexte est tout type d'informations qui peut être collecté à partir du service (*nom, version, besoins en ressources*), de l'environnement (*heure, lieu*) et de l'utilisateur (*contraintes et préférences*).

3.3 *Algorithme détaillé*

Le but de notre travail est de proposer une approche afin de satisfaire aux besoins fonctionnels du client et d'optimiser ses contraintes et ses préférences évolutives. L'approche proposée est implémentée par une ontologie générique contextuelle cloud qui implémente le concept des Agents et NSGA-II. Pour notre travail, nous avons retenu l'algorithme NSGA-II qui est connu comme très performant et efficace dans le domaine de l'optimisation multi-objectif (Huang *et al.*, 2012). Nous avons inclus l'approche multi-agents qui permet d'améliorer le temps d'exécution à cause de son aspect parallèle. L'objectif principal est la

sélection du service cloud composite optimal en termes de QoS avec un minimum dégrée de violation de contraintes du client. Il se base sur des services sémantiques concrets disponibles enregistrés dans un registre de l'ontologie. La capacité des agents de supervision de contexte est exploitée pour réagir aux changements de contexte des services et/ou aux changements des préférences des clients. L'utilisation des solutions trouvées permet d'exploiter l'expérience de recherche acquise par les agents de construction de solutions dans les itérations futures de l'algorithme. Les étapes de l'algorithme sont :

– **Etape1 : Génération automatique et sémantique du processus virtuel :** le modèle de PA global est généré automatiquement en utilisant un algorithme de chainage arrière et les liens sémantiques entre les SVs (Alti *et al.,* 2014) selon les besoins fonctionnels des clients. Il est optimisé en se basant sur la réputation de chaque service virtuel pour éviter la redondance de services.

– **Etape 2 (phase 1) : Optimisation parallèle multi-objectifs au niveau de chaque agent esclave :** dans cette étape chaque agent esclave a comme objectif de vérifier la satisfaction des contraintes clients afin de sélectionner le meilleur service cloud composite (SCC). L'optimisation est effectuée en adoptant le (NSGA-II) guidée par une fonction de pénalité afin de sélectionner le meilleur service composite en termes de QoS en respectant les contraintes du client.

1. **Codage des solutions :** Un chromosome ou individu représente un service composite (PA) sous forme d'une chaine de services Cloud (gènes).
2. **Evaluation** : On évalue chaque SCC par l'agrégation des valeurs de QoS normalisées des SC.
3. **Sélection** : Dans notre approche on a on a assigné un degré de violation pour chaque attribut de QoS pour pénaliser les services qui ne respectent pas les contraintes du client. Ce degré est calculé comme suit :

$$\deg_{v\_CCS} = \sum_i w_i \times \deg^{i}_{v\_ccs} \tag{4}$$

$$\text{Avec } \deg^{i}_{v\_ccs} = \begin{cases} C_i - q_i & \text{If } C_i > q_i \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

Où : w_j : est le poids du ième attribut de QoS ; q_i : est la valeur agrégée du ième attribut de QoS ; C_i : est la valeur de contrainte du client du ième attribut de QoS ;

La nouvelle relation de dominance considère que le SCC dominant est celui qui a le plus faible degré de violation de contrainte. Si les degrés de violation de deux SCC sont égaux, la relation de dominance classique de NSGA-II est appliquée.

– **Etape 3 (phase 1) : Fusion et resélections des Pareto-optimaux au niveau d'agent maître SCwA–CCS-2S.** A la fin de toutes les itérations les résultats obtenus par les agents esclaves sont fusionnés et ordonnées. Ensuite, les meilleurs services composites (Pareto-optimal global) sont sélectionnés.

– **Etape 4 (phase 2) : Sélection d'une solution optimale.** Cette étape sélectionne une solution unique (SCC) à partir de l'ensemble du Pareto-optimal (PCCS) obtenu à la première phase. Le score de chaque solution est calculé par l'équation 6. La solution avec le meilleur score (maximum) est sélectionnée.

Où : nb est le nombre de paramètre de QoS ; w_j est le poids du j^{ème} attribut de QoS ; q'_j est la valeur agrégée du j^{ème} attribut de QoS ;

Le gestionnaire de contexte évalue en permanence le contexte du client (les préférences et contraintes du client) pour fournir une solution SCC en conséquence. En cas de modification des préférences du client, la solution est prise directement à partir des solutions du Pareto-optimal actuel sans devoir relancer la phase 1.

4. Résultats d'expérimentations

Nous avons évalué la satisfaction des contraintes et préférences du client, plusieurs expérimentations ont été réalisées sur un PC Intel (R) 4.0 GHz, 4 Go de RAM, Windows 7 (32 bits) et NetBeans. Le processus d'affaire étudié contient 10 services virtuels et chaque service virtuel est lié à un ensemble de candidats comprenant 100 services cloud. Le client choisi ses contraintes comme suit : Prix : *moins cher*, Temps de réponse : *rapide ;* qui se transforment en valeur sémantique « Haut ». La figure 3 montre clairement que tous les SCCs de l'ensemble du Pareto-optimal respectent les contraintes du client. En d'autres termes, toutes les valeurs des attributs de QoS de tous les SCCs du Pareto-optimal appartiennent à l'intervalle [0.7, 1]. Elle montre aussi la solution finale à fournir au client selon les trois différentes préférences du client. Afin de justifier l'utilisation de l'aspect multi-agents, des tests de comparaison sont effectuées entre l'approche SCwA-CCS (multi-agents) avec l'approche SCw-CCS (sans agents).

La figure 4 montre la performance de l'aspect multi-agents en termes de temps d'exécution. En particulier lorsque le nombre d'itérations augmente.

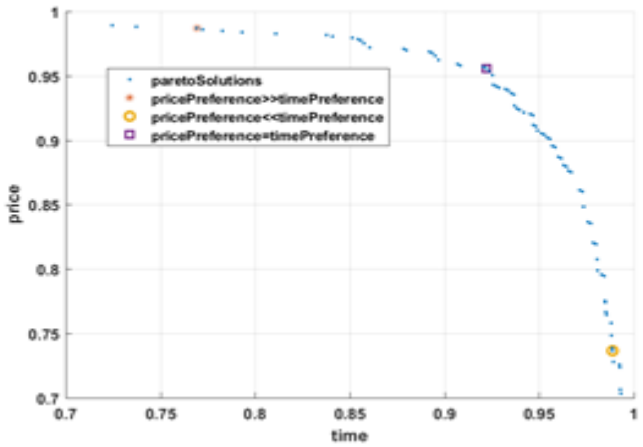

Figure 3. Pareto-optimal et solutions finales selon les différentes préférences du client (contraintes : temps ϵ [0.7, 1]et prix ϵ [0.7, 1]).

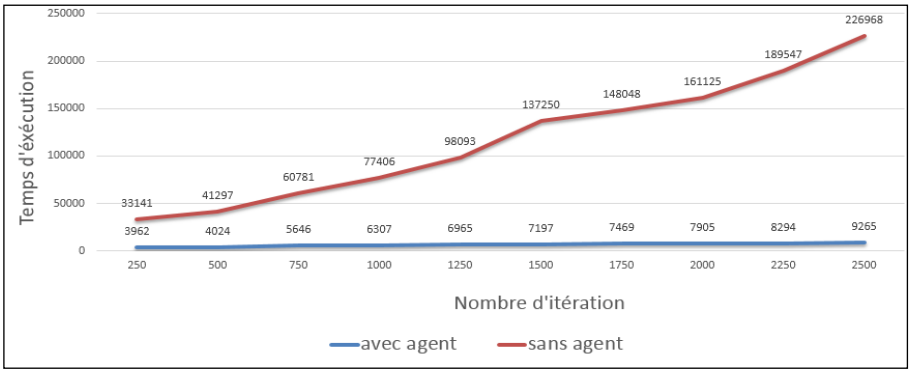

Figure 3. Temps d'exécution de SCw–CCS vs SCwA–CCS.

5. Conclusion

Cet article présente une nouvelle approche sémantique et dynamique à base d'agents afin de composer des services Cloud pour obtenir un ERP personnalisé. En effet, nous avons proposé une ontologie afin de guider et d'adapter le processus de composition dynamique. Cette ontologie est basée sur le potentiel des règles d'inférences pour créer des processus d'affaires virtuels satisfaisant les besoins fonctionnels du client. Ainsi, un algorithme à deux phases est proposé pour l'optimisation du processus d'affaire en termes de QoS. La première phase consiste à engendrer un ensemble de SCCs (Pareto-optimal) qui respectent les contraintes des clients. Cette phase utilise NSGA-II à base d'agents en introduisant un nouveau critère de dominance basé sur la violation des contraintes. La deuxième phase sert à sélectionner le SCC parmi les SCCs obtenus à partir de la première phase afin de le fournir au client selon ses préférences. Dans cette phase, on a utilisé la somme pondérée des attributs de QoS pour évaluer par score les différents SCCs du Pareto-optimal afin de sélectionner le SCC ayant le meilleur (maximum) score. Une nouvelle technique est proposée pour calculer les poids des attributs de QoS. Les résultats de l'expérimentation montrent l'efficacité de l'approche proposé en termes des contraintes et préférences du client, ainsi que le facteur multi-agents améliore clairement le temps d'exécution du système. Dans ce travail nous avons utilisé des valeurs agrégées de QoS déterministes (i.e. la moyenne). Dans les futurs travaux, nous envisageons d'étendre ce travail par l'intégration de l'aspect incertain dans la mesure des QoS, ce qui rend les résultats plus précis.

Bibliographie

Alti A., Laborie S., Roose, P. (2014). Dynamic semantic- based adaptation of multimedia documents. *Transactions on Emerging Telecommunications Technologies*, vol. 25, n° 2, p. 239-258.

Asghari S., Navimipour N. J. (2016). Review and Comparison of Meta-Heuristic Algorithms for Service Composition in Cloud Computing, Majlesi, *Journal of Multimedia Processing*, vol. 4, 2016.

Chang H., Liu H., Leung Y.W., Chu X. (2014). Minimum latency server selection for heterogeneous cloud services. In IEEE Global Communications Conference (GLOBECOM), p. 2276-2282, December 2014.

Chen C.S., Liang W.Y , Hsu H.Y. (2015). A cloud computing platform for ERP applications. *Applied Soft Computing* , vol. 274, p. 127-136.

Deng S., Huang L., Wu H., Wu, Z. (2016). Constraints-Driven Service Composition in Mobile Cloud Computing., *IEEE International Conference on Web Services (IEE ICWS 2016)*, San Francisco, CA, USA, June 27 - July 2, 2016. p. 228-235.

Huang, X., Lei, X., & Jiang, Y. (2012). Comparison of Three Multi-Objective Optimization Algorithms for Hydrological Model. *In Computational Intelligence and Intelligent Systems*, p. 209-216, Springer, Berlin, Heidelberg.2012.

Johansson B., Ruivo, P. (2013). Exploring factors for adopting ERP as SaaS. *Procedia Technology*, vol. 9, p. 94-99.

Jula A., Zalinda, O., Sundararajan, E. (2013). A hybrid imperialist competitive gravitational attraction search algorithm to optimize cloud service composition. *In IEEE Workshop Memetic Computing (MC)*, p. 37–43.

Jun L., Weihua G. (2009). An environment-aware particle swarm optimization algorithm for services composition. International Conference on Computational Intelligence and Software Engineering, p. 1-4.

Li B., Li M. (2009). Research and design on the refinery ERP and eERP based on SOA and the component-oriented technology. *IEEE International Conference on Networking and Digital Society, ICNDS'09*., vol. 1, p. 85-88., 2009.

Reffad H., Alti A., Roose P. (2016). Cloud-based Semantic Platform for Dynamic Management of Context-aware mobile ERP applications. ACM MEDES International Conference – DOI: 10.1145/3012071.3012076, 2016, Hendaye, France.

Rosenberg F., Leitner P., Michlmayr A., Celikovic P., Dustdar, S. (2009). Towards Composition as a Service -A Quality of Service Driven Approach, *IEEE 25th International Conference on Data Engineering(ICDE)*, p. 1733 -1740

Sasikaladevi N., Arockiam L. (2012). Genetic approach for service selection in composite web service. *International Journal of Computer Applications*, vol. 44, n° 4, p. 22-29.

Tarantilis C.D., Kiranoudis C.T., Theodorakopoulos N.D. (2008). A web-based ERP system for business services and supply chain management: Application to real-world process scheduling. European Journal of Operational Research, vol.187, n° 4, p.1310-1326.

Pop, C.B., Vlad, M., Chifu, V.R., Salomie, I., Dinsoreanu, M. (2011). A tabu search optimization approach for semantic web service composition. In 10th IEEE International Symposium Parallel and Distributed Computing (ISPDC'2011), p. 274-277.

Yu Q., Chen L., Li B. (2015). Ant colony optimization applied to web service compositions in cloud computing. Computers & Electrical Engineering, vol.41, p.18-27.