Composition de services Web et équité vis-à-vis des utilisateurs finaux

J. El Haddad[1] and O. Spanjaard[2]

[1] LAMSADE, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16 France.
elhaddad@lamsade.dauphine.fr

[2] LIP6, Université Pierre et Marie Curie, 104 avenue du Président Kennedy, 75016 Paris France.
olivier.spanjaard@lip6.fr

1 Composition de services Web

La migration des systèmes d'information des entreprises vers un schéma orienté *Business to Business* (i.e., des entreprises mettant leurs services à disposition d'autres entreprises sous forme de composants logiciels) a favorisé l'émergence des services Web depuis le début des années 2000 [1]. Par exemple, une compagnie aérienne peut mettre en ligne à disposition des agences de voyage un service logiciel de réservation de vols. Pour fournir une valeur ajoutée à l'*utilisateur final*, une agence de voyage peut décider de composer plusieurs services Web afin de faciliter l'organisation d'un séjour. Par exemple, une agence peut proposer un pack "déplacement" comportant d'une part la réservation d'un billet pour un événement sportif, d'autre part une réservation de billet de train ou d'avion. On parle alors de *composition de services Web* [3]. Ce type de composition peut être représentée à l'aide d'un *graphe de tâches*, comme celui indiqué sur la Figure 1. Les tâches t_1, t_2 et t_3 représentent respectivement une réservation de place, une réservation de train ou de vol. Dans ce graphe, certains utilisateurs emprunteront le *chemin d'exécution* $\langle t_1, t_2 \rangle$ (ceux réservant un billet de train) et d'autres le chemin d'exécution $\langle t_1, t_3 \rangle$ (ceux réservant un billet d'avion). Il arrive très fréquemment que plusieurs services répondent à un même ensemble de besoins fonctionnels. Par exemple, pour réaliser la réservation d'une place au stade, un utilisateur peut mettre en concurrence plusieurs services de billeterie. Une *sélection de services Web* consiste alors à assigner un service Web à chaque tâche. Cette affectation est appelée *plan d'exécution*. L'ensemble des plans d'exécution possibles est combinatoire : pour n tâches avec un choix parmi m services pour chacune, il y a m^n compositions possibles. Afin de discriminer parmi ces plans d'exécution, on utilise la notion de *qualité de service*. En effet, les services Web *simples* asurant les mêmes propriétés fonctionnelles se distinguent les uns des autres par des critères de qualité de service, tels que réputation, coût, durée d'exécution etc. Etant donné un chemin d'exécution, la qualité de service assurée par un service Web *composite* (résultant d'un plan d'exécution) sur un critère peut être calculée selon une règle d'agrégation propre à chaque critère. Le vecteur de qualités de service ainsi obtenu est ensuite lui-même scalarisé en un indicateur unique de qualité via une fonction d'agrégation. Le problème combinatoire qui se pose alors est de déterminer un plan d'exécution optimisant cet indicateur. Néanmoins, pour un même plan d'exécution, la qualité de service diffère selon le chemin d'exécution emprunté par l'utilisateur final.

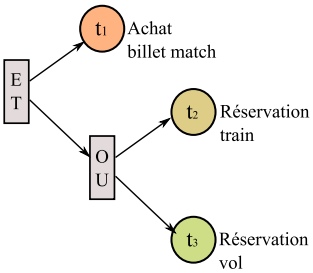

Fig. 1. Organisation d'un déplacement par composition de services Web

2 Equité vis-à-vis des utilisateurs finaux

Dans le contexte de la composition de services Web, l'*équité* [5] va désigner le soucis de ne pas léser un utilisateur final vis-à-vis d'un autre dans le plan d'exécution utilisé pour assurer les différentes tâches. La méthode proposée par Zeng *et al.* [2] pour prendre en compte la multiplicité des scénarios possibles d'utilisation (i.e., des chemins d'exécution) consiste à : 1) déterminer indépendamment la sélection optimale pour chacun des chemins d'exécution, puis 2) "fusionner" ces sélections en un plan d'exécution unique. Pour ce faire, les auteurs assignent à chaque tâche le service choisi dans le chemin d'exécution correspondant au *hot path* de cette tâche. Le *hot path* d'une tâche désigne le chemin d'exécution le plus souvent emprunté lors des exécutions de cette tâche. Cette manière de procéder peut néanmoins conduire à des sélections globalement peu satisfaisantes, comme nous l'illustrerons dans notre exposé. Pour palier cet inconvénient, nous chercherons ici à prendre en compte *explicitement* les différents scénarios possibles d'utilisation. Nous nous inspirerons pour cela des travaux menés en optimisation *robuste* [4], cette dernière notion étant proche formellement de la notion d'équité. Nous viserons en particulier à minimiser le critère de *regret maximum*, classiquement utilisé pour mesurer la robustesse d'une solution. Dans notre contexte, nous qualifierons d'*équitable* une solution optimale au sens de ce critère. Nous proposerons une formulation du problème de recherche d'un tel plan d'exécution équitable avec un programme linéaire en nombres entiers. Enfin, nous présenterons les résultats d'expérimentations numériques menées sur le sujet.

Références

1. Austin, D., and Barbir, A., Ferris, C. and Garg, S. : Web Services Architecture Requirements. W3C Working Draft, http ://www.w3c.org/TR/2002/WD-wsa-reqs-20020819 (2002).
2. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J. and Chang, H. : QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Sofware Engineering, Vol. 30, No. 5, p. 311–327 (2004).
3. Dustdar, S. and Schreiner, W. : A survey on web services composition. International Journal of Web and Grid Services, Vol. 1, No. 1, p. 1–30 (2005).
4. Kouvelis, P. and Yu, G. : Robust discrete optimization and its applications. Kluwer Academic Publisher (1997).
5. Moulin, H. : Axioms of Cooperative Decision Making. Monograph of the Econometric Society, Cambridge University Press (1988).