Un modèle de classement de services par contribution et utilité*

Camelia Constantin
LIP6, Université Paris 6
8, rue du Capitaine Scott
75015 Paris, France
camelia.constantin@lip6.fr

Bernd Amann
LIP6, Université Paris 6
8, rue du Capitaine Scott
75015 Paris, France
bernd.amann@lip6.fr

David Gross-Amblard
Le2i, Université de Bourgogne
BP 47870
21078 Dijon Cedex
david.gross-amblard at u-bourgogne.fr

Résumé

Les architectures orientées services (AOS) se sont imposées pour la construction d'applications web grâce à l'élaboration de standards ouverts et compatibles avec les technologies web existantes. Le nombre de services sur le web grandit chaque jour et les développeurs d'applications sont souvent confrontés au problème de choisir parmi les services disponibles. Cet article présente un modèle de classement de services fondé sur des scores d'importance qui reflètent pour chaque service son activité et sa contribution à la qualité d'autres services. Nous décrirons également un algorithme distribué pour le calcul de ces scores exploitant les connexions déjà existantes entre services. Le modèle et les algorithmes ont été validés par simulation.

Mots clés

Systèmes coopératifs, algorithmes distribués, classement de services.

1 Introduction

La tâche essentielle d'un modèle de classement est de définir des scores pertinents pour classer un ensemble d'entités en fonction de certains critères. Un grand nombre de modèles et d'algorithmes de classement ont été proposés pour des documents, des données et des services dans divers domaines d'applications. Les modèles de classement basés sur le contenu classent les entités d'après la *pertinence* de leur contenu ou d'autres métadonnées associées à une requête donnée en entrée. Ce type de classement a prouvé son efficacité pour les systèmes de recherche de documents et la sélection de données et de services s'appuyant sur la qualité. Le classement basé sur les liens exploite des relations qui existent entre les entités afin d'estimer leur *importance* dans un graphe de liens. L'exemple le plus célèbre de cette approche de classement est l'algorithme PageRank implanté par Google pour estimer l'importance d'une page dans le graphe du Web. Cet article présente un nouveau modèle de classement fondé sur les liens d'appels qui existent entre des services distribués. L'idée principale est de calculer pour chaque service un score d'importance globale en exploitant des connaissances spécifiques sur sa contribution aux autres services.

La prochaine section illustre notre approche dans le contexte d'un système fondé sur une architecture orientée-services pour la syndication de nouvelles (news). La section 3 présente l'état de l'art général des travaux liés à notre approche. Le modèle est défini formellement dans la section 4, suivi par la présentation de deux algorithmes distribués pour calculer l'importance d'un service, permettant ainsi le passage à l'échelle (section 5). La section 7 démontre experimentalement la validité de notre approche, et la section 8 conclue.

*Ce travail a été partiellement financé par le programme de recherche français ACI-MD dans le cadre du projet SemWeb.

2 Exemple d'application

Nous considérons des applications orientées services reposant sur un ensemble de services qui collaborent afin d'exécuter certaines tâches. Nous n'imposons aucune restriction particulière ni sur la fonctionnalité, les données locales ou l'implantation de chaque service, ni sur la structure ou le contenu des messages échangés entre services. En conséquence, notre modèle s'applique à des services simples comme le partage ou l'impression de documents, ou plus complexes comme les services d'interrogation de bases de données et les services web. De nombreux exemples d'applications correspondent à notre définition d'un service et peuvent bénéficier de notre modèle de classement. Par exemple, dans un moteur de recherche distribué en pair-à-pair, chaque pair peut être considéré comme un service cherchant certains documents localement et interrogeant d'autres pairs (services). Les résultats des recherches peuvent être classés suivant des critères basés sur le contenu, comme la pertinence du document pour la requête. Mais pour trier des documents au contenu similaire, il peut également être utile de supposer qu'un pair important (par exemple ayant du renom) produit des résultats plus importants que les autres. Un autre exemple est la découverte et la sélection de services web. Bien qu'il soit possible de comparer et classer des services web en fonction de leur descriptions WSDL et UDDI, ces techniques sont généralement basées sur des descriptions de services homogènes et sémantiquement riches. Nous soutenons l'hypothèse que des mesures s'appuyant sur des "liens de collaboration" entre services sont une alternative intéressante pour classer des services présentant des descriptions pauvres ou hétérogènes et des fonctionnalités équivalentes.

Nous illustrons notre approche à travers l'exemple d'un système de partage de nouvelles distribué (news), où chaque nœud peut jouer le rôle d'un fournisseur de nouvelles (serveur), d'un consommateur de nouvelles (client) ou des deux à la fois (portail). La figure 1 montre un exemple avec cinq nœuds qui collaborent. Les services s_4 et s_5 sont fournis par les deux agences de presse *Agence France Presse (AFP)* et *Reuters*. Les services s_2 et s_3 sont publiés par les deux journaux *Le Monde* et *20 Minutes* et le service s_1 est proposé par le moteur de recherche *Google*.

2.1 Scores de contribution des services

Tous les services collaborent en échangeant des nouvelles via des appels de services. Nous supposons qu'une partie de la qualité de chaque service dépend des services qu'il appelle. Notre notion de "qualité de service" (QoS) est abstraite et chaque service pris individuellement peut utiliser des critères de qualité différents afin d'estimer la contribution des autres services.

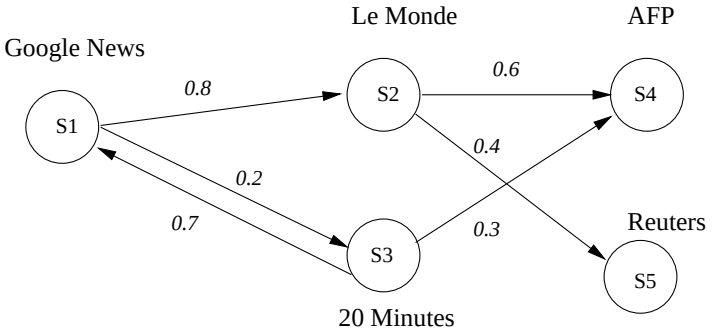

FIG. 1 – Graphe de contributions à la qualité

Par exemple, un journal peut considérer que la qualité d'un article dépend de son intérêt pour ses lecteurs.

En suivant ces critères, le journal français *Le Monde* estime alors qu'en général, l'agence de presse française *AFP* fournit des informations plus intéressantes pour ses lecteurs que l'agence américaine *Reuters*. D'un autre côté, *20 Minutes* essaie de réduire les coûts et préfère donc *Google News*, qui est gratuit, à *AFP*, qui propose un service payant. Ces mesures sont intégrées dans le graphe de la figure 1 où chaque arête $s_i \rightarrow^c s_j$ est étiquetée par la valeur $c \in [0, 1]$ représentant la *contribution* du service s_j à la qualité de son client s_i *relative aux autres services* utilisés par s_i.

Nous ne sommes pas intéressés par la façon dont chaque service définit et acquiert les mesures sur la contribution relative des autres services à sa qualité. Bien que, pour simplifier, ces informations sont considérées comme publiques dans notre modèle, la section 5 montre que l'accès public n'est pas nécessaire pour calculer l'importance d'un service.

Chaque service participe directement à l'amélioration de la qualité de ses clients, à la qualité des clients de ses clients, et ainsi de suite. Par exemple, *AFP* contribue indirectement à la qualité de *Google News* via *Le Monde*. Une situation plus complexe apparaît lorsqu'il existe des cycles dans la construction du graphe. Les cycles de contribution peuvent s'expliquer par le fait que chaque service peut contenir différentes données et implanter différentes fonctionnalités. Par exemple, *20 Minutes* utilise *Google News* parce qu'il est gratuit et *Google News* pourrait appeler *20 Minutes* afin de collecter des nouvelles de *AFP* (rappelons qu'il n'y a pas de lien direct entre *Google News* et *AFP*).

2.2 Scores d'utilisation de services et d'appels

Un service peut effectivement contribuer à la qualité d'autres services seulement s'il est appelé. Ceci est illustré dans le graphe de la figure 2 ci-dessous, où chaque arête $s_i \hookrightarrow^t s_j$ correspond à un appel de service à un moment t. Il montre que même si *Reuters* est supposé contribuer à la qualité de *Le Monde*, cette contribution est encore "virtuelle" puisqu'il n'a pas été encore appelé. Nous appelons cette propriété qui dépend des appels effectivement reçus l'*utilisation du service*.

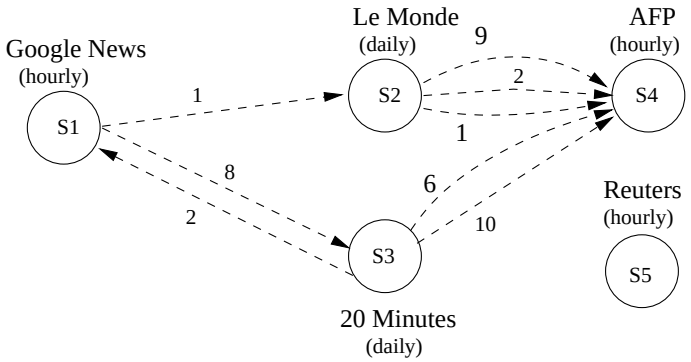

FIG. 2 – Traces d'appels de service à l'instant 10

L'utilisation d'un service s_i est définie pour un certain instant τ et dépend des appels de services reçus par s_i avant τ. Nous supposons que pour chaque couple de services (s_i, s_j) il existe une fonction qui calcule le *score d'utilisation de service* $u_{ji}(\tau)$ de s_j par s_i au moment τ. Supposons que le service s_1 (*Google News*) soit intéressé par *toutes* les nouvelles publiées par le service s_2 (*Le Monde*) qui met à jour ses nouvelles quotidiennement. Alors, l'utilisation du service *Le Monde* par *Google News* à l'instant τ peut être estimé par l'*âge* $|\tau - t|$ du dernier appel $s_1 \hookrightarrow^t s_2$ émis par le service s_1 avant τ. Si cet âge est inférieur à une

journée, *Le Monde* peut être considéré comme complètement utile pour *Google News* à l'instant τ, i.e. $u_{21}(\tau) = 1$. D'un autre côté, si l'âge excède un certain nombre de jours, e.g. un mois, *Le Monde* peut être considéré comme parfaitement inutile ($u_{21}(\tau) = 0$) pour *Google News* à τ. En effet, si *Google News* appelle *Le Monde* trop rarement, il manque une partie de ses nouvelles et *Le Monde* est considéré comme étant moins utile pour *Google News*.

La fonction de calcul du score d'utilisation peut différer d'une application à l'autre. Par exemple, le calcul du score d'utilisation $u_{ji}(\tau)$ du service s_j par s_i peut également prendre en compte l'état du service client s_i. Si nous supposons dans notre exemple que *Google News stocke* les résultats obtenus de certains services, les scores d'utilisation de service pour ces services pourraient être plus grands que ceux des services dont les nouvelles ne sont pas mémorisées.

La notion d'utilisation peut être étendue aux appels de services en estimant l'*utilisation* $u_{jki}(\tau)$ *des résultats d'appels de service sortants* $s_k \hookrightarrow^t s_j$ *pour répondre aux appels de service entrants* $s_i \hookrightarrow^{t'} s_k$.

De façon similaire à l'exemple précédent, ce score d'utilisation $u_{jki}(\tau)$ pourrait être exprimé, par exemple, par la proximité temporelle entre les appels de service entrants $s_i \hookrightarrow^{t'} s_k$ et les appels de service sortants $s_k \hookrightarrow^t s_j$ avant τ. En appliquant cette sémantique à la figure 2, l'appel $s_2 \hookrightarrow^1 s_4$ est plus utile pour l'appel $s_1 \hookrightarrow^1 s_2$ que les appels $s_2 \hookrightarrow^{2,10} s_4$ à l'instant 2 ou à l'instant 10. En général, $u_{jki}(\tau)$ exprime un taux d'utilisation des appels issus de s_i des résultats retournés par les appels de s_j et dépend de l'implantation de s_k. Par exemple, si le service *20 Minutes* stocke les résultats reçus par le service *AFP*, l'appel issu de *20 Minutes* vers *AFP* à l'instant 6 pourrait être utile pour l'appel ultérieur reçu par *20 Minutes* à l'instant 8.

2.3 Importance de service

L'idée principale de ce travail est d'estimer l'*importance d'un service* en combinant les scores de contribution et d'utilisation des services avec lesquels il collabore directement ou indirectement. Nous illustrons ceci avec l'exemple ci-dessus. Tout d'abord, si nous considérons seulement le graphe de contribution à la qualité de la figure 1, il paraît sensé de dire que *AFP* devrait être plus important que *Reuters* puisqu'il contribue globalement plus aux autres services. Cet argument est confirmé par la figure 2 qui montre que *Reuters* n'a jamais été appelé par aucun service. Cependant, si nous essayons de comparer *Le Monde* et *20 Minutes* de la même façon, nous observons que les deux critères (contribution et utilisation) sont indépendants l'un de l'autre. Par exemple, *Le Monde* devrait sans doute être plus important que *20 Minutes* si l'on considère la valeur de sa contribution à *Google News*. Cependant, comme on l'a vu dans la figure 2, à $\tau = 10$ les deux services ont été appelés une fois seulement par *Google News* à différents moments. Si l'on estime que l'utilisation d'un service dépend de l'âge du dernier appel reçu, à l'instant 1 *Le Monde* est dans ce cas plus important que *20 Minutes* (qui n'a pas été encore utilisé), mais il perd la première place à l'instant 8 lorsque *20 Minutes* est appelé.

3 État de l'art

Le classement de services est un problème fondamental dans la construction d'applications web qui nécessitent la découverte et la sélection dynamique de services. De nombreux critères peuvent être utilisés pour le classement de services. Par exemple, [7] propose d'utiliser des techniques d'échantillonage avancées afin de comparer et de classer des services d'interrogation en fonction de leurs données locales. Les techniques s'appuyant sur les recommandations exploitent les appréciations d'utilisateurs [17, 11] ou d'autres services [10] pour la sélection dynamique de service. D'autres critères de classement s'appuient sur

la conformité des critères de qualité de service durant des périodes données [22, 11]. La variation dans la conformité des niveaux de qualité projetés pendant une fenêtre de temps est aussi un paramètre [11]. Tous ces paramètres peuvent être combinés en utilisant des poids différents suivant les exigences spécifiques de l'utilisateur [22, 17]. Dans ce contexte, notre méthode peut être vue comme une approche basée sur des recommandations pour classer les services où chaque appel de service correspond à un vote implicite qui prend en compte la qualité de service et l'utilisation observée lors d'une période donnée.

Les méthodes de classement de pages web comme PageRank [18] et HITS [13] sont basées sur l'exploration des liens entre ces pages et ont été développées et appliquées avec succès dans les moteurs de recherche (voir [16] pour une description détaillée). Cette famille de modèles et d'algorithmes de classement considère que chaque page p propage une fraction de son importance à toutes les autres pages p' qu'elle référence. La plupart des algorithmes existants calculent l'importance d'une page en exploitant la structure d'hyperliens du web. OPIC [1] évite la construction de la matrice du web grâce à un algorithme adaptatif permettant d'estimer l'importance d'une page dynamiquement lors du processus d'acquisition des pages web. Des approches récentes considèrent également des mesures d'importance dépendantes du temps. Par exemple, [4] observe que l'importance basée sur les liens pénalise les pages nouvellement apparues sur le web et propose de diminuer le PageRank pour des pages plus anciennes. Le même type d'argument est utilisé par [8] qui calcule l'importance d'une page en se basant sur PageRank et sa dérivée en fonction du temps. [9] exploite les informations temporelles afin de classer des nouvelles (news) et leurs sources : un article récent est plus important qu'un ancien, et une source qui produit des nouvelles importantes récentes est plus importante que les autres sources.

Les modèles d'importance ont également été appliqués dans le contexte d'infrastructures pair-à-pair. Par exemple, [23] définit et calcule des scores d'importance pour des pages distribuées dans un réseau pair-à-pair. L'importance d'une page est définie comme une combinaison de son importance locale au sein de son pair et de l'importance globale de son pair vis-à-vis d'autres pairs. Une approche similaire est présentée par [19], où chaque pair affine le score de ses pages locales en se concertant périodiquement avec d'autres pairs.

Un algorithme synchrone pour la gestion de la réputation dans des systèmes pair-à-pair est décrit dans [12]. Chaque pair calcule sa valeur de confiance globale basée sur les valeurs de confiance d'autres pairs et sur la confiance locale de ceux-ci en lui-même. [20] propose une version asynchrone et distribuée de *PageRank* en s'appuyant sur des itérations chaotiques. Un calcul entièrement asynchrone est présenté dans [14]. Nous proposons un algorithme itératif et asynchrone pour le calcul d'importance de services qui est inspiré de [20, 14] et qui prend en considération la modification proposée par [5] pour la terminaison du calcul asynchrone.

4 Classement de services basé sur les liens

Une application orientée services est constituée d'un ensemble de services qui échangent des messages. Nous supposons que tous les messages sont estampillés par une horloge continue et asynchrone produisant un ensemble infini de valeurs d'horloge $\mathcal{T}$[1]. Pour simplifier, nous ne faisons pas la distinction entre les différents utilisateurs finaux et les applications externes et nous supposons qu'ils sont tous représentés par un seul service dans le système.

[1] Notre modèle ne requiert pas une synchronisation exacte.

4.1 Fonction de trace

Nous définissons une *fonction de trace* qui associe à chaque pair de services (s_i, s_j) les estampilles temporelles de tous les messages envoyés par s_i à s_j. Plus formellement :

Définition 1 (fonction de trace)**.** *Soit $\mathcal{S}$ un ensemble fini de services identifiés qui échangent des messages. Une* fonction de trace $\Lambda : \mathcal{T} \times S \times S \rightarrow 2^{\mathcal{T}}$ *retourne pour chaque instant $t \in \mathcal{T}$ et chaque paire de services distincts (s_i, s_j), l'ensemble d'estampilles $\Lambda(t, s_i, s_j) = \{t_i | t_i \leq t\} \subseteq \mathcal{T}$ de tous les messages envoyés par s_i à s_j avant t.*

La fonction de trace Λ enregistre les collaborations entre les services comme un ensemble de messages *échangés*. Notons que Λ ignore les messages locaux. À un niveau d'abstraction plus élevé, il est possible de composer les messages en *appels de service*. Par exemple, dans un appel de service "requête-réponse", le service envoyant le premier message (requête) est considéré comme étant le client du service recevant cette requête, alors que dans un appel "sollicitation-réponse", le service émetteur du premier message (sollicitation) sera considéré comme le serveur du deuxième. Bien que notre modèle s'applique à n'importe quel type de protocole, nous supposons par la suite que la fonction de trace Λ enregistre seulement les estampilles t d'appels de service de type requête-réponse $s_i \hookrightarrow^t s_j$ où t correspond à l'estampille du message représentat la requête envoyée par s_i à s_j.

En pratique il pourrait être utile de distinguer différents types d'appels de service $s_i \rightarrow^t s_j$ entre deux services s_i et s_j. Les messages ne sont pas typés dans notre modèle, mais il est possible d'introduire une certaine forme d'appel de service typé de la manière suivante. Afin de faire la distinction entre n types d'appels différents reçus par un service s_j, nous pouvons remplacer s_j par n services s_j^k différents, $1 \leq k \leq n$. La fonction de trace Λ enregistre alors chaque message $s_i \rightarrow^t s_j$ de type k comme un message échangé entre le service s_i et s_j^k.

Exemple 1. L*e Monde est un service fournissant des articles sur différentes thématiques comme le* sport*, la* culture*, l'*économie*, etc. Afin de faire la distinction entre les différentes demandes de nouvelles liées à une thématique précise, on peut remplacer* Le Monde *par plusieurs services* Le Monde^{sport}*,* Le Monde^{culture}*,* Le Monde^{conomie}*, etc.*

Nous notons par $Out(t, s_i) = \{s_j \mid s_j \in \mathcal{S} \wedge \Lambda(t, s_i, s_j) \neq \emptyset\}$ l'ensemble des services appelés par s_i jusqu'à l'instant t. De même, $In(t, s_j) = \{s_i \mid s_i \in \mathcal{S} \wedge \Lambda(t, s_i, s_j) \neq \emptyset\}$ est l'ensemble des services qui ont appelé s_j avant t.

Définition 2 (graphe d'appels de services)**.** *La fonction Λ observe l'activité entre des services distincts et génère un* graphe d'appels de service *orienté $SC(t) = (\mathcal{S}, \mathcal{C}, \Lambda)$ où (i) $\mathcal{S}$ est l'ensemble de sommets (ii) $\mathcal{C}$ est l'ensemble d'arêtes, telles que $(s_i \rightarrow s_j) \in \mathcal{C}$ ssi $\Lambda(t, s_i, s_j) \neq \emptyset$.*

Exemple 2. *Suivant le graphe d'appel montré dans la figure 2, nous avons $\Lambda(10, s_2, s_4) = \{1, 2, 9\}$, $\Lambda(10, s_4, s_2) = \emptyset$ (s_4 n'a pas appelé s_2 avant $t = 10$), $Out(10, s_2) = \{s_4\}$ et $Out(10, s_3) = \{s_1, s_4\}$.*

4.2 Contribution d'un service

Un service contribue directement ou indirectement à d'autres services dans le système. Plus précisément, nous considérons qu'un service s_j peut contribuer directement à un certain moment t à tous les services s_i, avec $s_i \in In(t, s_j)$, et par transitivité à tous les services s_k auxquels s_i contribue.

Nous faisons la distinction entre la contribution d'un service s_j à la *qualité* d'un service s_i et la manière dont s_i utilise s_j pendant une période de temps. Par conséquence, notre définition de contribution d'un

service prend en compte une contribution à la qualité de service *statique* (indépendante des appels) et une contribution dynamique due à l'utilisation (dépendante des appels).

4.2.1 Contribution à la qualité de service

Chaque service donné estime sa qualité en fonction de certains critères spécifiques à l'application. L'idée essentielle légitimant cette estimation est que la qualité d'un service dépend de la qualité des services qu'il appelle. Par exemple, dans un moteur de recherche pair-à-pair, la qualité d'un service peut être définie comme le nombre moyen de documents pertinents retournés à ses clients. D'autres critères de qualité peuvent englober la fraîcheur moyenne de ses résultats (dans le cas de la réplication de données avec des mises à jour), le temps de réponse moyen, ou des critères commerciaux s'appuyant sur le prix de chaque appel de service.

Nous définissons le score de contribution d'un service s_j à la qualité d'un service s_i par une fonction Υ :

Définition 3 (contribution locale à la qualité)**.** *Soit $\mathcal{S}$ un ensemble de services. La* fonction de contribution $\Upsilon : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ *définit pour chaque paire de services distincts (s_i, s_j) un* score de contribution locale $\Upsilon(s_i, s_j) = \pi_{ji}$ *du service s_j à la qualité du service s_i tel que $\sum_{s_j \in Out(s_i)} \pi_{ji} = 1$.*

La fonction $\Upsilon(s_i, s_j)$ ne retourne pas une valeur quantitative mais un *score* π_{ji} qui compare la contribution à la qualité du service s_i de tous les services s_j directement utilisés par s_i. Nous supposons également que ces scores sont statiques et statistiquement indépendants d'un appel particulier entre s_i et s_j [2].

Chaque service s_i définit la contribution locale π_{ki} de tous les services s_k qu'il utilise, indépendamment des services s_j qui sont utilisés par s_k. Néanmoins la contribution de s_j à la qualité de s_i à travers s_k peut être estimée comme étant la partie de π_{ki} qui est due à s_j, *i.e* $\pi_{ki} * \pi_{jk}$. Le graphe de la *contribution à la qualité* $SC(t, \Upsilon)$ est obtenu en ajoutant à chaque arête $s_i \rightarrow s_j$ du graphe d'appels de service $SC(t)$ une étiquette $\Upsilon(s_i, s_j) = \pi_{ji}$ (voir par exemple le graphe de la figure 1). Chaque chemin $p = s_i \rightarrow^{\pi_{ki}} s_k \rightarrow \ldots s_l \rightarrow^{\pi_{jl}} s_j$ dans ce graphe représente alors une contribution d'un service s_j à un service s_i avec un score de contribution $\pi_p = \pi_{ki} * ... * \pi_{jl}$.

Par conséquent la *contribution globale* d'un service donné s_j à la qualité de *n'importe quel* service $s_i \in \mathcal{S}$ est calculée à partir de tous les chemins de contribution :

Définition 4 (contribution globale à la qualité)**.** *Nous notons par $\mathcal{P}_{ij}$ l'ensemble de tous les chemins reliant s_i à s_j dans le graphe de contribution $SC(t, \Upsilon)$. La* contribution globale π^*_{ji} *du service s_j à la qualité de s_i est la somme des contributions de s_j pour s_i sur tous les chemins p allant de s_i à s_j dans $SC(t, \Upsilon)$:*

$$\pi^*_{ji} = \sum_{p \in \mathcal{P}_{ij}} \pi_p. \tag{1}$$

Exemple 3. *Dans la figure 1, le service* Google News *estime que les nouvelles offertes par* Le Monde *contribuent localement avec un score de* 0.8 *à sa qualité. Puisque* Le Monde *appelle* AFP*,* AFP *contribue indirectement à la qualité de* Google News *avec un score de* $0.8 * 0.6 = 0.48$*.*

A cause des cycles dans la graphe de contribution l'ensemble de chemins de contribution entre deux services s_i et s_j peut être infini. Néanmoins, la contribution globale de cet ensemble de chemins converge vers un point fixe (on peut montrer que pour tout $\varepsilon > 0$, l'ensemble de chemins p pour lesquels $\pi_p < \varepsilon$ est fini).

[2] Cette restriction peut être relâchée mais elle simplifie la présentation et la compréhension de notre modèle d'importance.

Exemple 4. *Il existe un ensemble infini de chemins de contribution allant de* Google News *à* AFP *qui passent par* 20 Minutes*. La contribution de* AFP *à la qualité de* Google News *peut alors être calculée comme étant* $(0.8 * 0.6 + 0.2 * 0.3) * \sum_{i \geq 0} (0.7 * 0.2)^i = 0.63$.

4.2.2 Utilisation de service

Les appels enregistrés dans $\Lambda(t, s_i, s_j)$ représentent l'utilisation effective de s_j par s_i à un instant t donné. Un service peut effectivement contribuer à la qualité d'autres services seulement s'il est appelé. Pour chaque couple de services (s_i, s_j) nous introduisons un *score d'utilisation de services* qui exprime la façon dont s_j est utilisé par s_i, en fonction des appels reçus par s_i :

Définition 5 (utilisation locale de service)**.** *La fonction d'utilisation de service* $U_d : \mathcal{T} \times \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ *retourne pour chaque instant* t *et chaque paire de services* (s_i, s_j) *un* score d'utilisation locale de service $U_d(t, s_i, s_j) = u_{ji}(t)$ *du service* s_j *par le service* s_i *obtenu d'après les appels enregistrés dans* $\Lambda(t, s_i, s_j)$.

La définition concrète de la fonction d'utilisation dépend de la sémantique de l'application et n'importe quelle fonction agrégeant les estampilles d'appels de service dans $\Lambda(t, s_i, s_j)$ avec une sémantique appropriée peut être choisie. Des fonctions d'*utilisation de service* différentes peuvent être définie pour chaque couple (s_i, s_j). Chaque service s_j peut appliquer une fonction spécifique pour chaque service $s_i \in In(t, s_j)$, ou la même fonction pour tous les services $s_i \in In(t, s_j)$.

Dans la suite, nous supposons que le score d'utilisation $u_{ji}(t)$ est spécifique au service s_j et qu'il décroît avec l'ancienneté des derniers appels reçus de la part de ses clients s_i, avec un facteur de décroissance qui dépend de la fonctionnalité et du comportement de s_j. Par exemple, un fournisseur de données qui met à jour ses informations très fréquemment pourrait considérer que son score d'utilisation par les services s_i décroît très rapidement, car ses données sont utiles pour ses clients seulement pendant une courte période.

Exemple 5. *Le service* s_1 *(*Google News*) est intéressé par toutes les nouvelles produites par le service* s_2 *(*Le Monde*). Puisque* Le Monde *met à jour ses nouvelles tous les jours,* Google News *doit l'appeler quotidiennement pour maintenir un score d'utilisation maximale de* $u_{21}(t) = 1$*. Si le seul appel* $s_1 \hookrightarrow^1 s_2$ *de* Google News *vers* Le Monde *a eu lieu plusieurs jours avant l'instant* 10*, l'utilisation de*Le Monde *pour* Google News *est inférieure à* 1*, reflétant le fait qu'il a probablement manqué plusieurs nouvelles.*

L'utilisation de service ne prend pas en compte les relations logiques ou temporelles possibles entre les appels entrants et sortants d'un service s_i. Nous introduisons la notion d'*utilisation d'appels* qui décrit ce type de relation entre les appels de service entrants et sortants.

Exemple 6. *Dans l'exemple de la figure 2, le service* 20 Minutes *a reçu un appel de la part de* Google News *à l'instant* 8 *et a émis deux appels à destination du service* AFP *avant (à l'instant* 6*) et après (à l'instant* 10*) cet appel. Le score d'utilisation de ces appels à destination de* AFP *pour les appels émis par* Google News *calculé par le service* 20 Minutes *peut alors prendre en considération le fait que les appels de services entrants utilisent uniquement des résultats obtenus avant ces appels. En particulier, l'appel de service à l'instant* 8 *utilise uniquement des résultats récupérés par* 20 Minutes *de la part de* AFP *à l'instant* 6 *et l'appel de service à l'instant* 10 *est inutile pour cet appel.*

Les scores d'utilisation d'appel sont estimés par une fonction combinant les informations sur les appels de service entrants dans $\Lambda(t, s_i, s_k)$ avec les informations sur les appels de services sortants dans $\Lambda(t, s_k, s_j)$ d'un même service s_k :

Définition 6 (utilisation d'appel)**.** *Soit $\mathcal{S}$ un ensemble de services observés par une fonction de trace Λ. La* fonction d'utilisation d'appel $U_i : \mathcal{T} \times \mathcal{S} \times \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ *retourne pour chaque instant t et chaque triplet de services (s_i, s_k, s_j) le* score d'utilisation d'appel $U_i(t, s_i, s_k, s_j) = u_{jki}(t)$ *des appels de service $s_k \rightarrow^{t'} s_j \in \Lambda(t, s_k, s_j)$ au service s_j pour les appels de service $s_i \rightarrow^{t''} s_k \in \Lambda(t, s_i, s_k)$ reçus du service s_i.*

Le score $u_{jki}(t)$ est dynamique et montre le degré avec lequel les appels de s_k vers s_j sont liés aux appels reçus par s_k de la part de s_i. De manière analogue à l'utilisation de service, la définition des scores d'utilisation d'appel dépend de l'application et peut comparer de différentes façons les appels de services entrants et sortants d'un service donné. Par exemple un service s_i dont les appels vers un service s_k déclenchent régulièrement des appels de s_k vers un autre service s_j, conduit à un score d'utilisation d'appel $u_{jki}(t)$ élevé. De même, si un service s_i utilise des réponses stockées par un service s_k de la part du service s_j, le score d'utilisation d'appel $u_{jki}(t)$ correspondant peut dépendre de l'obsolescence de ces données.

Exemple 7. Google News *a appelé* Le Monde *à l'instant* 1*. Si nous considérons que cet appel exploite seulement les nouvelles générées avant cet instant, l'utilisation des appels vers* AFP *issus ultérieurement est égal à* 0*. Dans le cas où l'utilisation d'appel exprime la part des appels vers* AFP *qui ont été déclenchés par les appels issus de* Google News*, la valeur de l'utilisation d'appel à l'instant* 1 *est* 1*, et* $1/3$ *à l'instant* 10 *(nous supposons que chaque appel de service entrant a déclenché tous les appels de service sortants avec la même estampille).*

L'utilisation d'appel permet de généraliser la notion d'utilisation de service en définissant le *score d'utilisation de service global* d'un service s_j pour n'importe quel service $s_i \in \mathcal{S}$ à travers les chemins d'appels $p = s_i \rightarrow s_l \rightarrow s_m \rightarrow \ldots \rightarrow s_n \rightarrow s_k \rightarrow s_j$ dans $SC(t)$.

Définition 7 (utilisation globale de service)**.** *Soit $p = s_i \rightarrow s_l \rightarrow s_m \rightarrow \ldots \rightarrow s_n \rightarrow s_k \rightarrow s_j$ un chemin d'appel dans $SC(t)$. Le* score d'utilisation global de s_j pour s_i à travers p *est défini par le produit des scores d'utilisation d'appels et le score d'utilisation locale de s_j sur ce chemin :*

$$u_p(t) = u_{mli}(t) * \ldots * u_{jkn}(t) * u_{jk}(t). \tag{2}$$

Soit $\mathcal{P}_{ij}$ l'ensemble de tous les chemins possibles de s_i à s_j dans le graphe des appels de service $SC(t)$. Le score d'utilisation global de service $u^*_{ji}(t)$ *d'un service s_j par un service s_i à l'instant t est la somme des scores d'utilisation de service de s_j pour s_i à travers les chemins dans $\mathcal{P}_{ij}$:*

$$u^*_{ji}(t) = \sum_{p \in \mathcal{P}_{ij}} u_p(t). \tag{3}$$

4.3 Contribution effective et importance

Notre objectif est de comparer et de classer des services en fonction de leur activité observée à l'aide d'une fonction de trace Λ combinée avec les informations sur la façon dont chaque service contribue à la qualité de tous les autres services. L'importance d'un s_j dépend de sa *contribution effective* aux autres services $s_i \in \mathcal{S}$, qui est calculée en combinant les scores de contribution et les scores d'utilisation sur tous les chemins d'appels de service dans $SC(t)$. L'idée principale de la définition suivante est de pondérer pour chaque chemin de s_i à s_j le score de contribution avec le score d'utilisation correspondant.

Définition 8 (contribution effective globale)**.** *Pour un chemin d'appels de service donné* $p = s_i \rightarrow s_l \rightarrow s_m \rightarrow \ldots \rightarrow s_n \rightarrow s_k \rightarrow s_j$ *dans* $SC(t)$*, la* contribution effective globale *de* s_j *pour* s_i *à travers* p*, noté par* $\tilde{\pi}_p(t)$*, est le produit du score de contribution et du score d'utilisation de service de* s_j *pour* s_i *à travers* p *:*

$$\tilde{\pi}_p(t) \quad = \quad \pi_p * u_p(t) \tag{4}$$

Soit $\mathcal{P}_{ij}$ *l'ensemble de tous les chemins possibles de* s_i *à* s_j *dans le graphe d'appels de service* $SC(t)$*. Le* score de contribution effective globale $\tilde{\pi}^*_{ji}(t)$ *de* s_j *pour* s_i *est obtenu comme étant la somme des contributions effectives à travers tous les chemins* $p \in \mathcal{P}_{ij}$ *:*

$$\tilde{\pi}^*_{ji}(t) = \sum_{p \in \mathcal{P}_{ij}} \tilde{\pi}_p(t). \tag{5}$$

Un service est alors important si sa valeur de contribution effective à tous les autres services est élevée.

Définition 9 (importance d'un service)**.** *L'*importance *d'un service* s_j *est définie comme étant la somme de toutes ses contributions effectives globales à tous les autres services* $s_i \in \mathcal{S}$ *:*

$$I_j(t) = \sum_{s_i \in \mathcal{S}} \tilde{\pi}^*_{ji}(t). \tag{6}$$

Exemple 8. *Les figures 1 and 2 montrent que le service* s_4=AFP *contribue au service* s_2=Le Monde *avec un score* $\pi_{42} = 0.6$ *et qu'il a été appelée par* Le Monde *à l'instant* 9*. Si l'utilisation de service* $u_{42}(t)$ *à l'instant* $t = 10$ *est estimée par l'âge du dernier appel et si le temps écoulé entre les instants* 10 *et* 9 *est seulement quelques heures, la contribution effective de* AFP *à* Le Monde *est élevée (proche du score de la contribution à la qualité). D'autre part, même si* Le Monde *contribue beaucoup à la qualité de* Google News*, il a été appelé par* Google News *plusieurs jours avant l'instant* 10*, ce qui conduit à un score de contribution effective faible.*

Par un raisonnement similaire, nous pouvons voir que même si la contribution à la qualité $\pi_{42} * \pi_{21}$ *de* AFP *pour* Google News *à travers* Le Monde *est élevée, la contribution de service de* AFP *pour* Google News *à travers* Le Monde *est faible. Ceci est dû au faible score de contribution d'appel des appels vers* AFP *pour les appels émis par* Google News *(lorsque l'appel à un instant donné exploite les nouvelles générées avant cet instant, comme décrit dans l'exemple 7).*

D'autre part, la contribution à la qualité $\pi_{43} * \pi_{31}$ *de* AFP *pour* Google News *à travers* 20 Minutes *est renforcée par un score d'utilisation de service élevé* $u_{431}(10) * u_{43}(10)$ *si l'utilisation de service* $u_{43}(10)$ *dépend de l'âge du dernier appel* $S_3 \hookrightarrow^{10} S_4$ *à l'instant* 10 *(âge =* 0*) et l'utilisation d'appel* $u_{431}(10)$ *est élevée à cause de la proximité temporelle des deux derniers appels correspondants (*8 *et* 10*).*

5 Calcul d'importance

Dans cette section nous présentons deux algorithmes pour calculer l'importance des services dans un système orienté services. L'idée fondamentale est d'exploiter l'infrastructure distribuée et les liens de communication existants afin de distribuer le calcul sur les différents nœuds de service. Nos simulations montrent que les algorithmes présentés peuvent être déployés efficacement dans des architectures de services distribués à grande échelle.

5.1 Principe de la distribution

Dans les deux algorithmes, chaque service s_j calcule sa propre importance $I_j(\tau)$ à l'instant τ et échange des messages avec les services voisins dans $In(\tau, s_j)$ et $Out(\tau, s_j)$. Plus précisément, le service s_j débute son calcul avec une valeur d'importance initiale $I_j^0(\tau)$ qu'il met régulièrement à jour s'appuyant sur les *valeurs d'importance* $\nu_{jk}(\tau)$ reçues de ses voisins $s_k \in In(\tau, s_j)$. La valeur d'importance $\nu_{jk}(\tau)$ reçue par s_j de la part de chaque $s_k \in In(\tau, s_j)$ exprime la contribution à la qualité π_{jk} du service s_j au service s_k et, *récursivement*, la contribution à la qualité et l'utilisation d'appel du service s_j pour d'autres services s_i du système à travers s_k. Plus formellement, la *valeur d'importance* $\nu_{jk}(\tau)$ *reçue par* s_j *de la part d'un voisin* s_k est définie récursivement en utilisant les valeurs d'importance $\nu_{ik}(\tau)$ reçues par s_k de la part de ses clients s_i :

$$\nu_{jk}(\tau) = \pi_{jk} + \pi_{jk} * \sum_{s_i \in In(\tau, s_k)} \nu_{ik}(\tau) * u_{jki}(\tau) \tag{7}$$

L'équation 7 montre que si aucun appel du service s_k vers le service s_j n'a été utile aux clients du service s_k, la valeur d'importance $\nu_{jk}(\tau)$ reçue par s_j de s_k est égale à la contribution locale π_{jk} de s_j à la qualité de s_k.

L'importance $I_j(\tau)$ d'un service s_j à un moment donné τ est la somme des valeurs d'importance reçues $\nu_{jk}(\tau)$ pondérée par les scores d'utilisation directe $u_{jk}(\tau)$:

$$I_j(\tau) = \sum_{s_k \in In(\tau, s_j)} \nu_{jk}(\tau) * u_{jk}(\tau) \tag{8}$$

Grâce à cette définition, chaque service s_j est capable de calculer son importance avec les valeurs d'importance reçues de la part de ses clients s_k. On remarquera que chaque service s_j connaît les valeurs de ses scores d'utilisation $u_{jk}(\tau)$ au début du calcul. La preuve montrant que l'équation 8 calcule les mêmes valeurs d'importance que celles définies par l'équation 6 est donnée dans la section 6.

Les deux algorithmes que nous allons décrire maintenant diffèrent essentiellement par le protocole utilisé pour échanger les valeurs d'importance. Dans le premier algorithme les services calculent leur importance de manière synchrone par l'échange de nouveaux messages de calcul. Ceci conduit à un coût de communication et une charge du réseau supplémentaire au moment du calcul. Le second algorithme utilise un protocole asynchrone pour le calcul et réduit la charge du réseau en intégrant les valeurs d'importance dans les appels de service "normaux" (applicatifs) sans générer des messages supplémentaires.

5.2 Calcul distribué synchrone

L'algorithme suivant obtient le vecteur de valeurs d'importance à un instant donné τ par un calcul de point fixe implanté sous forme d'itérations distribuées et synchronisées. À chaque itération, un service s_k collecte les valeurs d'importance $\nu_{ki}(\tau)$ de tous ses clients $s_i \in In(\tau, s_k)$, recalcule son importance et l'envoie aux services $s_j \in Out(\tau, s_k)$. L'échange des valeurs d'importance est synchronisé et le service s_k doit attendre les valeurs d'importance $\nu_{ki}(\tau)$ calculées à l'itération précédente avant de recalculer son importance à l'itération courante. Les itérations s'arrêtent quand les valeurs d'importance des services sont proches du point fixe (pour un seuil ε donné).

L'algorithme synchrone COMPUTSYNC(τ) est exécuté par chaque service $s_k \in \mathcal{S}$ pour calculer son importance à un instant τ. Les variables O_k et N_k sont des vecteurs tels que o_{ki} et n_{ki} sont des valeurs d'importance reçues par s_k de la part de $s_i \in In(\tau, s_k)$ avant (o_{ki}) et pendant (n_{ki}) l'itération en cours. La variable globale ρ désigne l'ensemble des services qui ont convergé après chaque pas de l'algorithme. Au

début, la variable ρ est initialisée à l'ensemble vide et tous les services s_k choisissent un temps commun τ et une valeur d'importance initiale I_k^0 égale à 0.

COMPUTSYNC(τ)
Entrée : un instant τ
Sortie : importance du service s_k à l'instant τ
début
$O_k = N_k = 0$
faire
// envoi des valeurs d'importance
pour chaque $s_j \in Out(\tau, s_k)$ **faire**
$\nu_{jk}(\tau) = \lambda * \pi_{jk} + \lambda * \pi_{jk} * \sum_{s_i \in In(s_k)} n_{ki} * u_{jki}(\tau)$
envoie $\nu_{jk}(\tau)$ à s_j
finpour
//les services de la part desquels s_k n'a pas encore reçu l'importance
$Tmp = In(\tau, s_k)$
tant que $Tmp \neq \emptyset$ **faire**
//attend qu'un service s_i envoie $\nu_{ki}(\tau)$
$n_{ki} = \nu_{ki}(\tau)$
$Tmp = Tmp \setminus \{s_i\}$
fintantque
si $|N_k - O_k|/O_k \geq \varepsilon$ **alors**
$\rho = \rho \setminus \{s_i\}$
alors
$\rho = \rho \cup \{s_i\}$ // convergence locale
finsi
//calcule l'importance du service
$I_k(\tau) = \sum_{s_i \in In(s_k)} n_{ki} * u_{ki}(\tau)$
$O_k = N_k$
// stoppe quand tous les services ont convergé
tant que $\rho \neq \mathcal{S}$
fin

L'algorithme COMPUTSYNC est une version distribuée des itérations de Jacobi qui calculent l'importance reçue comme étant la solution du système linéaire de l'équation 9 (voir section 6). Les valeurs d'importance sont multipliées par une valeur constante $\lambda \in (0, 1)$ (de manière similaire à [18]) ce qui garantit que la solution du système existe, qu'elle est unique, et que les itérations de Jacobi convergent vers la solution du système indépendamment des valeurs d'importance initiales I_k^0 (voir section 6 pour une preuve formelle).

L'importance reçue par un service s_j converge *localement* pendant une itération i lorsque la condition de convergence $|N_k - O_k|/O_k < \varepsilon$ est satisfaite pour un $\varepsilon > 0$ donné. Néanmoins, la convergence locale ne garantit pas que la condition de convergence restera satisfaite pour tous $j >= i$ (convergence globale) et chaque service arrête le calcul seulement lorsque tous les services $s_k \in \mathcal{S}$ ont convergé localement ($\rho = \mathcal{S}$).

5.3 Calcul distribué asynchrone

L'algorithme suivant évite l'envoi de messages de calcul supplémentaires ainsi que les retards dûs à la synchronisation, en intégrant les valeurs d'importance dans des appels de services propres à l'application. Chaque service s_k recalcule son importance locale à chaque appel de service entrant, sans attendre que tous les clients $s_i \in In(\tau, s_k)$ aient envoyé leur importance. D'une manière similaire, le service s_k communique ses nouvelles valeurs d'importance à un service $s_j \in Out(\tau, s_k)$ seulement lorsqu'il fait un nouvel appel à s_j. Nous obtenons ainsi un protocole asynchrone où chaque service calcule son importance à sa propre vitesse, en utilisant des valeurs d'importance qui peuvent être dépassées. Le seul point de synchronisation entre les services réside dans le choix de l'instant τ auquel le calcul est débuté. De manière analogue à l'algorithme précédent, nous utilisons un vecteur N_k dont les éléments sont initialisés à 0 au début de l'algorithme.

À chaque nouvel appel de service $s_k \hookrightarrow^t s_j$:

– l'émetteur s_k

1. calcule la valeur d'importance $\nu_{jk}(\tau) = \lambda * \pi_{jk} + \lambda * \pi_{jk} * \sum_{s_i \in In(\tau, s_k)} n_{ki} * u_{jki}(\tau)$;
2. ajoute cette valeur dans l'appel de service (comme un élément supplémentaire d'un message SOAP par exemple) ;
3. appelle le service s_j ;

– le receveur s_j

1. mémorise la valeur d'importance reçue dans n_{jk} et
2. met à jour son importance $I_j(\tau) = \sum_{s_k \in In(\tau, s_j)} n_{jk} * u_{jk}(\tau)$.

L'algorithme ci-dessus implante les itérations *totalement asynchrones* pour calculer la solution du système linéaire 9 présenté dans la section 6. Nous supposons que notre système satisfait l'hypothèse d'*asynchronisme total* [5] qui suppose que chaque valeur d'importance est mise à jour indéfiniment souvent et que les anciennes valeurs sont purgées du système. Chaque service s_k doit aussi être informé à un certain moment des mises à jour d'importance de tous les services $s_i \in In(\tau, s_k)$. Lorsque ces conditions sont satisfaites les itérations totalement asynchrones convergent vers la solution du système linéaire (voir section 6).

Pour garantir la terminaison de l'algorithme (voir [5]), une valeur d'importance $\nu_{jk}(\tau)$ est envoyée par s_k à s_j seulement si elle diffère de plus de ε de la dernière valeur d'importance envoyée par s_k à s_j. Lorsque le calcul a convergé, s_k n'envoie plus de mises à jour d'importance à s_j. La terminaison de l'algorithme est détectée lorsque i) plus aucune valeur d'importance ne transite et ii) pour chaque service s_k, tout recalcul de $I_k(\tau)$ ne change pas sa valeur (voir [5]).

6 Preuve d'exactitude et de convergence

Cette section montre que les deux algorithmes présentés dans la section 5 convergent vers l'unique solution d'un système linéaire (équation 9) qui représente pour chaque service s_j sa contribution globale à tous les autres services du système système (équation 6).

Nous construisons tout d'abord un graphe qui représente les chemins de propagation de valeurs d'importance entre services. À partir d'un graphe d'appels de service $SC(\tau) = (\mathcal{S}, \mathcal{C}, \Lambda)$ (définition 2), une fonction de contribution locale Υ et une fonction d'utilisation d'appel U_i nous pouvons construire un graphe orienté et étiqueté, appelé *graphe d'importance* qui contient :

– un nœud n_{ki} pour chaque arête $s_i \rightarrow s_k \in \mathcal{C}$ et

– un arc de n_{ki} à n_{jk}) pour chaque pair d'arcs adjacents $s_i \rightarrow s_k$ et $s_k \rightarrow s_j$ dans $SC(\tau)$. L'arc de n_{ki} à n_{jk} est étiqueté par la fraction ω_{jki} de la valeur d'importance $\nu_{ki}(\tau)$ reçue de s_i que s_k envoie à s_j(équation 7) :

$$\omega_{jki}(\tau) = \Upsilon(s_k, s_j) * U_i(\tau, s_i, s_k, s_j) = \pi_{jk} * u_{jki}(\tau).$$

Exemple 9. *Le graphe d'importance correspondant à notre exemple est représenté dans la figure 3. Puisque le service* s_1 *(*Google News*) utilise le service* s_2 *(*Le Monde*) il y a un nœud* n_{21} *dans le graphe d'importance représentant les appels de* s_1 *à* s_2*. De même, nous avons un nœud* n_{42} *pour le couple* (Le Monde, AFP)*. Les deux nœuds sont reliés par une arête étiqueté par* $\omega_{421}(10) = 0.6 * u_{421}(10)$ *à l'instant* 10*, représentant la fraction de l'importance reçue* $\nu_{21}(10)$ *que* Le Monde *fait suivre à* AFP.

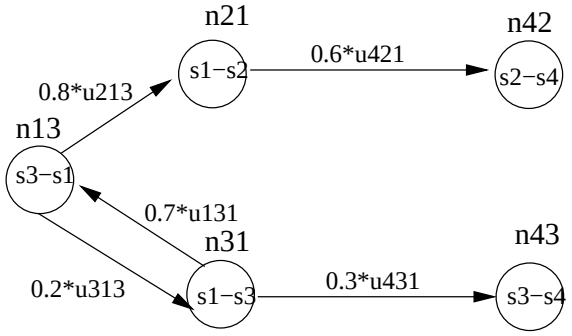

FIG. 3 – Graphe d'importance

Soit $\mathcal{S}$ un ensemble de n services. Nous notons par :

– $\tilde{\Pi}(\tau)$ la matrice d'adjacence du graphe d'importance de taille de $n^2 \times n^2$ (le nombre de couples (s_i, s_j) possibles est n^2). L'élément $\omega_{jki}(\tau) = \pi_{jk} * u_{jki}(\tau)$ est à la ligne $(j-1) * n + k$ et à la colonne $(k-1) * n + i$,
– $R_j(\tau) = [\nu_{j1}(\tau), \ldots, \nu_{jn}(\tau)]$ le vecteur des valeurs d'importance reçues par s_j , où $\nu_{ji}(\tau) = 0$ pour tous les $s_i \notin In(\tau, s_j)$, et
– $Q_j = [\pi_{j1}, ..., \pi_{jn}]$ le vecteur des contributions à la qualité de s_j, où $\pi_{ji} = 0$ pour tous les $s_i \notin In(\tau, s_j)$.

Les vecteurs $R(\tau) = [R_1(\tau), ..., R_n(\tau)]$ et $Q = [Q_1, ..., Q_n]$ de longueur n^2 contiennent alors les valeurs d'importance et les scores de contribution pour tous les $s_j \in \mathcal{S}$.

Théorème 1. *La solution de l'équation 7 correspond à la solution du système linéaire suivant :*

$$R(\tau) = \lambda * \tilde{\Pi}(\tau) * R(\tau) + \lambda * Q, \tag{9}$$

où $\lambda \in (0, 1)$.

Démonstration. La preuve est triviale, par la définition de la multiplication et l'addition sur les matrices. □

Lemma 1. *L'ensemble de valeurs d'importance* $\nu_{jk}(\tau)$ *reçues par chaque service* s_j *de la part de chaque service* $s_k \in In(t, s_k)$ *à la fin du calcul correspond à la solution du système linéaire 9.*

Dans la suite nous considérons τ fixe et nous allons utiliser la notation $\tilde{\Pi}$ au lieu de $\tilde{\Pi(\tau)}$. Il a été montré [6, 21] que si la valeur propre maximale de $\lambda * \tilde{\Pi}$, notée $\rho(\lambda * \tilde{\Pi})$, est inférieure à 1 :

1. $(\mathbb{I}_{N\times N} - \lambda * \tilde{\Pi})$ (où $\mathbb{I}_{N\times N}$ est la matrice identité) est inversible et la solution du système linéaire 9 existe et elle est unique :

$$R \quad = \quad (\mathbb{I}_{N\times N} - \lambda * \tilde{\Pi})^{-1} * \lambda * Q. \tag{10}$$

2. les itérations de Jacobi (implantées par le calcul synchronisé à la section 5) qui calculent la solution du système linéaire convergent, la solution pouvant être exprimée comme :

$$R \quad = \quad (\sum_{k=0}^{\infty} \lambda^k * \tilde{\Pi}^k) * \lambda * Q. \tag{11}$$

3. l'algorithme totalement asynchrone décrit à la section 5 converge vers la solution du système linéaire lorsque l'hypothèse d'*asynchronisme total* est satisfaite [5].

Pour montrer que $\rho(\lambda * \tilde{\Pi}) < 1$ pour $\lambda \in (0, 1)$, il suffit de montrer que $\rho(\tilde{\Pi}) \leq 1$. En s'appuyant sur le théorème de Gerschgorin [21], il suffit de montrer que la somme des éléments sur chaque colonne c de la matrice $\tilde{\Pi}$ est inférieure ou égale à 1. Considérons la colonne $c = (k-1) * n + i$ qui représente les poids des arêtes du nœud n_{ki} vers tous les nœuds n_{jk}. La somme des poids de cette colonne est :

$$\sum_{j\in[1,n]} \omega_{jki}(\tau) = \sum_{s_j \in Out(t,s_k)} \pi_{jk} * u_{jki}(\tau) \leq 1,$$

puisque $\sum_{s_j \in Out(t,s_k)} \pi_{jk} = 1$ et $u_{jki}(\tau) \leq 1$.

Le théorème suivant déclare que la valeur d'importance reçue par s_j de s_k multipliée par l'utilité de service de s_j pour s_k correspond à la fraction d'importance de s_j reçue par tous les chemins d'appels qui se terminent par un appel de s_k vers s_j.

Théorème 2. *Pour chaque nœud n_{jk} dans le graphe d'importance représentant l'arête $(s_k \rightarrow s_j)$ dans $SC(\tau)$, la valeur d'importance $\nu_{jk}(\tau) * u_{jk}(\tau)$ obtenue en multipliant la solution du système linéaire dans l'équation 9 avec l'utilisation de service de s_j par s_k est la somme des contributions de s_j à tous les services $s_i \in \mathcal{S}$ sur les chemins d'appels de service qui se terminent par l'arête $(s_k \rightarrow s_j)$.*

Démonstration. Afin de simplifier les équations, nous ne prenons pas en considération la constante λ. Chaque chemin $p = (s_i \rightarrow s_l \rightarrow s_m \rightarrow ... \rightarrow s_n \rightarrow s_k \rightarrow s_j)$ dans $SC(\tau)$ correspond dans le graphe d'importance à un chemin $p' = (n_{li} \rightarrow n_{ml} \rightarrow ... \rightarrow n_{kn} \rightarrow n_{jk})$, de poids $\omega_{p'} = \omega_{mli}(\tau) * \ldots * \omega jkn(\tau) = \pi_{ml} * u_{mli}(\tau) * \ldots * \pi_{jk} * u_{jkn}(\tau)$.

L'ensemble $\mathcal{P}_{ij}$ de chemins de s_i à s_j dans le graphe d'appels de service $SC(\tau)$ peut être séparé en $|Out(t, s_i)| \times |In(t, s_j)|$ sous-ensembles où chaque sous-ensemble contient les chemins débutant par $(s_i \rightarrow s_l)$ et finissant par $(s_k \rightarrow s_j)$ pour tous les $s_l \in Out(t, s_i)$ et les $s_k \in In(t, s_j)$. Par conséquent, cela signifie que tous les chemins possibles dans le graphe d'importance de n_{li} à n_{jk} correspondent à l'ensemble de tous les chemins possibles dans le graphe d'appels de service de s_i à s_j qui débutent par $(s_i \rightarrow s_l)$ et se terminent par $(s_k \rightarrow s_j)$.

La somme $\sum_{k=0}^{\infty} \tilde{\Pi}^k$ calcule une matrice où un élément à la position $((j-1)*n+k, (l-1)*n+i)$ est la somme des poids de tous les chemins possibles de n_{li} à n_{jk} dans le graphe d'importance. La multiplication matrice-vecteur $(\sum_{k=0}^{\infty} \tilde{\Pi}) * Q$ multiplie le poids $\omega_{p'}$ de chaque chemin p' défini comme ci-dessus par π_{li}, *i.e* $\omega_{p'} * \pi_{li} = \pi_{li} * \pi_{ml} * u_{mli}(\tau) * ... * \pi_{jk} * u_{jkn}(\tau)$. Observons que $u_{jk}(\tau) * \omega_{p'} * \pi_{li} = \pi_{li} * \pi_{ml} * u_{mli}(\tau) * ... * \pi_{jk} * u_{jkl}(\tau) * u_{jk}(\tau)$ est la contribution de s_j à s_i sur le chemin p défini comme ci-dessus.

Nous en déduisons que la solution $\nu_{jk}(\tau)$ du système linéaire multipliée par $u_{jk}(\tau)$ peut être exprimée en prenant en considération tous les chemins possibles des services n_{li} vers le service n_{jk} suivant : $u_{jk}(\tau) * \nu_{jk}(\tau) = \sum_{n_{li}} \sum_{p'} \omega_{p'} * \pi_{li} * u_{jk}(\tau)$ où p' sont tous les chemins possibles de s_i à s_j qui débutent par $(s_i \rightarrow s_l)$ et terminent par $(s_k \rightarrow s_j)$. Les services n_{li} à partir desquels il existe un chemin vers le service n_{jk} représentent un sous-ensemble des arêtes sortantes de s_i, plus précisément celles sur les chemins dans $SC(\tau)$ allant de s_i à s_j qui finissent par $(s_k \rightarrow s_j)$.

Ceci représente la contribution du service s_j au services s_i sur le sous-ensemble de chemins d'appels de services de s_i à s_j qui se terminent par $(s_k \rightarrow s_j)$.

□

Et, pour conclure, la contribution de s_j au service s_i sur les chemins d'appels de service se terminant par $(s'_k \rightarrow s_j)$, avec $s_{k'} \neq s_k$ est exprimée par $\nu_{jk'}(\tau) * u_{jk'}(\tau)$, pour tous les $s_{k'} \in In(t, s_j)$. Cela implique que l'importance $I_j(\tau) = \sum_{s_k \in In(t,s_j)} \nu_{jk} * u_{jk}(\tau)$ représente la somme des contributions du service s_j à tous les services s_i sur tous les chemins possibles dans $SC(\tau)$.

7 Évaluation expérimentale

Nous avons implanté les deux algorithmes en Java (JDK 1.5) sur un portable AMD Turion 64 (1.6GHz, 2Gb RAM) sous l'environnement Linux SUSE 10.0. Dans les expériences suivantes, nous avons considéré un réseau de 1000 services collaborant qui ont été simulés sous la forme de fils d'execution Java. Nous avons généré différentes configurations de réseau (décrites plus tard) qui se distinguent par la façon dont chaque service s_i choisit les services voisins $s_j \in Out(\tau, s_i)$ qui contribuent à sa qualité à un instant τ donné. La contribution à la qualité est distribuée uniformément entre chaque service dans $Out(\tau, s_i)$: $\pi_{ji} = 1/|Out(\tau, s_i)|$. Afin de modéliser l'utilisation de service et d'appel de service, nous affectons à chaque arc de s_i à s_j une valeur aléatoire $\delta_{ji} \in [0, 10]$ qui représente l'ancienneté du dernier appel de s_i à s_j relativement à l'instant τ. Nous considérons que tous les services utilisent les mêmes fonctions d'utilisation $u_{ji}(\tau) = 1 - \alpha * \delta_{ji}$ (utilisation de service) et $u_{jki} = 1 - \alpha * |\delta_{ki} - \delta_{jk}|$ (utilisation d'appel). Le facteur d'utilisation α contrôle l'influence de la fonction d'utilisation sur le calcul d'importance ($\alpha = 0$ signifie que tous les liens de contribution de service sont pris en compte lors du calcul en ignorant l'ancienneté des derniers appels de service).

Chaque service s_i débute son calcul avec une valeur d'importance de 0 et stoppe son calcul après avoir atteint une *convergence locale* (lorsque $|N_i - O_i|/O_i < \varepsilon$). Dans l'algorithme synchrone décrit à la section 5, certains services peuvent converger plus vite que d'autres et nous considérons qu'un service s_i réalise une *itération* lorsqu'il reçoit des mises à jour d'importance de tous les services dans $In(\tau, s_i)$ qui n'ont pas encore convergé (ceci est différent de l'algorithme présenté, où chaque service attend les mises à jour de tous les services dans $In(\tau, s_i)$). Pour le calcul asynchrone, nous supposons qu'un service s_i réalise une itération quand il fait un nombre de mises à jour d'importance qui est égal au nombre de ses voisins dans $In(\tau, s_i)$. Dans la suite, sauf si cela est explicitement spécifié, toutes les expériences ont été effectuées avec un seuil $\varepsilon = 10^{-4}$, un facteur d'utilisation $\alpha = 0$ et $\lambda = 0.85$.

7.1 Génération des graphes de services

Dans la suite de la section, nous appellerons les services dans $Out(\tau, s_i)$ les voisins de s_i. Les voisins d'un service sont choisis à l'aide des quatre stratégies suivantes, conduisant à des graphes d'importance de services avec des topologies différentes :

Graphe-max [MAX]. Ce graphe est semblable au modèle de graphe du web proposé par [2]. Chaque service choisit pour voisins avec une probabilité de 0.75 cinq services "populaires", c'est à dire des services qui ont déjà été choisis par beaucoup d'autres services.

Graphe Linear-copying [LC]. Chaque service sélectionne aléatoirement un service "prototype" p parmi tous les services existants. Il choisit alors cinq voisins parmi tous les services où chaque tel voisin est aussi un voisin de p avec une probabilité de 0.75. Le graphe obtenu est similaire au modèle de graphe du web proposé par [15].

Réseau Small-world [SW]. Cette configuration simule un *réseau small-world* en créant 20 communautés composées chacune de 50 services. Chaque service dans une telle communauté se connecte aléatoirement à 5 voisins dans la même communauté et chaque communauté interagit en moyenne avec 5 services d'autres communautés choisies aléatoirement.

Configuration client-serveur [CS]. Cette configuration combine les trois stratégies ci-dessus afin de modéliser un environnement client-serveur avec 80 communautés de clients qui appelent les services d'une unique communauté de serveurs (SW). Chaque communauté de clients contient 10 services avec un service "prototype" connecté à des services serveurs choisis aléatoirement. Les 9 autres services au sein de la même communauté sont connectés à un service de leur communauté choisi au hasard, et à au plus 5 services serveurs, chaque serveur étant avec une probabilité de 0.75 un voisin du prototype de la communauté (stratégie LC). La communauté de serveurs est un graphe-Max composé de 200 services serveurs connectés en moyenne à 5 autres services serveurs.

7.2 Résultats expérimentaux

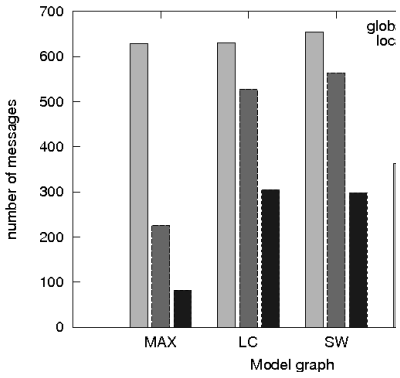

FIG. 4 – Nb. de messages pour différents modèles de graphe

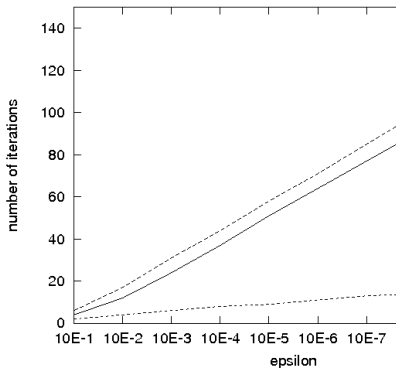

FIG. 5 – Nb. d'itérations pour la convergence en fonction de ε

La figure 4 montre, pour chacune des quatre configurations, le nombre moyen de messages de calcul générés par service jusqu'à la convergence, en utilisant la synchronisation globale, la synchronisation locale (ne pas attendre les services qui ont déjà convergé) et aucune synchronisation. Nous voyons que la synchronisation globale génère le plus grand nombre de messages puisque les services qui ont déjà convergé localement à un certain pas d'itération i continuent encore d'envoyer des messages jusqu'à ce que tous les services aient convergé. Avec le synchronisation locale, les services qui ont convergé localement arrêtent d'envoyer des messages. En utilisant l'algorithme asynchrone les services convergent plus vite qu'avec l'algorithme localement synchrone (comme nous le montre la figure 4). La raison pour ce gain est qu'un service n'envoie pas systématiquement les valeurs d'importance qu'il vient de calculer à tous ses voisins, comme cela se fait pour l'algorithme synchrone, mais il "optimise" la communication en envoyant des nouvelles valeurs d'importance seulement de temps en temps. Nous voyons que SW et LC génèrent plus de messages que les autres configurations. Les services dans SW sont connectés aléatoirement, ce qui peut conduire à la présence de nombreux cycles et des chemins de contribution longs pour beaucoup de services. Le même argument s'applique pour les services prototypes dans LC qui sont choisis au hasard. Au contraire, les chemins de contribution dans MAX sont plutôt courts puisque tous les services sont liés à des services populaires. Il y a beaucoup de services qui ne sont voisins d'aucun autre service et qui convergent très vite. Dans CS il y a beaucoup de petites communautés indépendantes avec peu de connexions.

La figure 5 montre l'influence de la valeur seuil ε utilisée pour la convergence sur le nombre d'itérations. Comme attendu, des valeurs faibles de ε conduisent à un nombre élevé d'itérations, puisque nous devons prendre en considération des chemins plus longs (avec des contributions plus faibles) avant d'atteindre la convergence. Néanmoins la figure 5 montre que la croissance du nombre d'itération est logarithmique, indépendamment du modèle de graphe. Des résultats similaires ont été relevés par [6] sur la convergence de PageRank.

Les figures 6 et 7 montrent le taux de valeurs d'importance qui ont atteint leur classement final en fonction du nombre d'itérations.

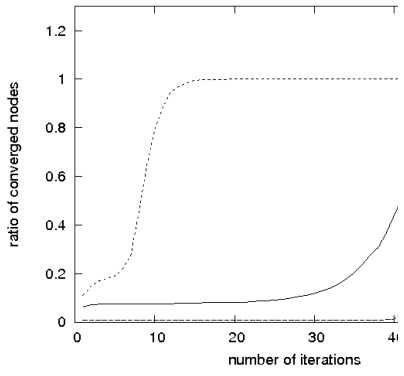

FIG. 6 – Taux de nœuds ayant convergé en fonction du nombre d'itérations (algorithme synchrone)

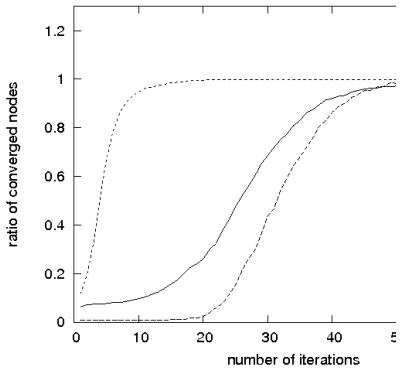

FIG. 7 – Taux de nœuds ayant convergé en fonction du nombre d'itérations (algorithme asynchrone)

Avec l'algorithme synchrone (figure 6) pour LC (resp. SW) seulement 10% (resp. 1%) des services ont convergé après 30 itérations. La convergence globale est atteinte après environ 50 itérations. Cela peut s'expliquer par la connectivité élevée des communautés small-world qui conduit à un grand nombres de chemins possibles à explorer. Au contraire, les services dans CS convergent plus rapidement (après 10 itérations, presque tous les services ont convergé) puisque les services clients sont regroupés en beaucoup de petites communautés.

En comparant la figure 6 avec la figure 7 nous remarquons tout d'abord que la suppression de la synchronisation permet aux services de converger plus rapidement. La raison est qu'un service envoie dans chaque message d'importance plus d'informations sur les chemins de contribution que dans les algorithmes synchrones, ce qui accélère la convergence. En d'autres mots, avec l'algorithme asynchrone, les services "apprennent" plus rapidement quels sont tous leurs chemins de contribution.

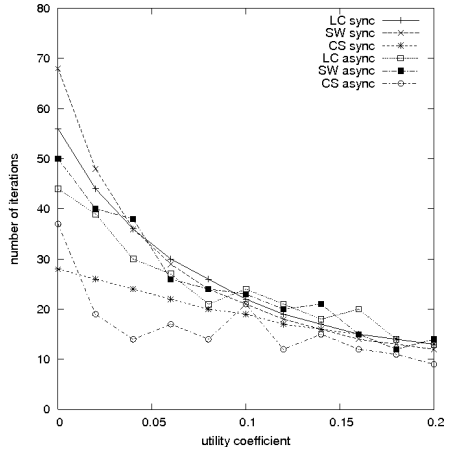

FIG. 8 – Nombre d'itération pour différentes fonctions d'utilité

α	LC		SW		CS	
	10	50	10	50	10	50
0.0	10	50	10	50	10	50
0.04	9	46	7	39	10	47
0.08	8	45	6	30	10	44
0.12	7	42	4	26	10	42
0.16	6	39	2	17	8	38
0.2	4	25	2	10	4	27

FIG. 9 – Influence de l'utilisation sur l'ensemble des résultats

Nous avons également réalisé plusieurs expériences pour illustrer l'impact de la fonction d'utilisation sur le calcul d'importance. Sur la figure 8 nous voyons que le nombre d'itérations décroît lorsque la valeur de α augmente, indépendamment du modèle du graphe. Ce résultat était attendu puisque des petites valeurs d'utilisation conduisent à une faible connectivité ainsi que des chemins plus courts puisque la contribution à la qualité d'un service sur un chemin est réduite par le facteur d'utilisation α.

Sur la figure 9 nous étudions l'influence du facteur d'utilisation α sur le classement des services pour trois configurations de graphe. Nous considérons comme référence le classement obtenu pour $\alpha = 0.0$ (classement obtenu en considérant seulement les liens de contribution de service sans l'utilisation de service). Pour différentes valeurs de α nous calculons la fraction des services qui sont encore parmi les 10 premiers et les 50 premiers. Nous voyons que l'utilisation de service a un impact fort sur le classement obtenu. Par exemple, dans la configuration SW et pour $\alpha = 0.2$ seulement 20% des services contribuants le plus sont encore parmi les 10 (resp. 50) premiers.

Enfin la table 1 illustre l'influence d'un service sur l'importance des autres services. Après un calcul

	sans utilisation						utilisation : $\alpha = 0.1$					
	sync			async			sync			async		
	nœuds	chemin	mess	nœuds	chemin	mess	nœuds	chemin	mess	nœuds	chemin	mess
LC	938	18	16682	887	99	7670	938	14	12694	764	61	2979
SW	990	32	12577	450	210	16325	995	16	2223	256	85	2854
CS	896	200	230	101	31	648	898	126	200	94	20	245

TAB. 1 – Coût du recalcul au départ d'un pair

d'importance initial, nous avons enlevé un nœud choisi au hasard et avons recalculé les valeurs d'importance de tous les services. La table montre le nombre de services impliqués dans le recalcul de l'importance, le nombre de messages qui sont échangés pour ce recalcul et la longueur maximale des chemins qui sont pris en considération. Nous voyons qu'avec l'algorithme localement synchrone, presque tous les services sont impliqués dans le calcul d'importance, alors qu'avec l'algorithme asynchrone, seulement une partie des services dans le voisinage des nœuds retirés recalculent leur importance. Par exemple avec $\alpha = 0.1$ et le graphe CS, seulement 94 nœuds participent au calcul avec l'algorithme asynchrone. Notons également que les longueurs des chemins et que le nombre de messages échangés sont plus faibles pour $\alpha = 0,1$. Les différences dans les longueurs des chemins et dans le nombre de messages échangés entre les algorithmes synchrone et asynchrone semblent dépendre de la topologie du graphe. Par exemple, avec le graphe LC, le nombre de messages de l'algorithme synchrone est plus important que celui de l'algorithme asynchrone pour les deux valeurs de α, alors que pour SW c'est le contraire.

8 Conclusion et travail futur

Cet article est un premier pas vers de nouveaux modèles de classement de services s'appuyant sur les liens. Bien que notre approche ait été illustrée dans le contexte d'une infrastructure de services pour l'abonnement à des nouvelles, nous croyons que le modèle et les algorithmes proposés sont utiles pour de nombreuses autres applications comme la recherche et la sélection de services web [7, 22] ou le stockage en entrepôts de données pair-à-pair basés sur des services web [3].

Nous prévoyons de continuer ce travail dans différentes directions. Tout d'abord, nous envisageons de valider notre approche dans le contexte des moteur de recherche pair-à-pair. L'idée essentielle est d'utiliser notre modèle pour classer des résultats de recherche en fonction de l'importance sémantique et basée sur l'utilisation de leurs sources. Deuxièmement, nous sommes en train d'étendre notre modèle en introduisant plus de connaissances sur le comportement et sur l'état des services. Par exemple, dans le modèle actuel, l'utilisation de service ne prend pas en compte si le résultat des appels de service sont mémorisés ou non par son client. Cette question apparaît naturellement dans le contexte des applications d'entrepôts de données pair-à-pair avec réplication de données. Une autre direction de recherche intéressante concerne la combinaison des importances des données et des services et consiste à définir un modèle d'importance flexible pour des données intentionnelles [3].

Les problèmes de sécurité liés au calcul d'importance ne sont pas traités dans cet article et nous sommes en train d'étudier les possibilités d'adapter des solutions existantes dans d'autres contextes d'applications (comme dans [12]) à notre environnement.

Références

[1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive On-Line Page Importance Computation. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 280–290, 2003.

[2] R. Albert, H. Jeong, and A.-L. Barabási. The Diameter of the World Wide Web. *Science*, 286 :509–512, 1999.

[3] The Active XML Project. http ://activexml.net.

[4] R. A. Baeza-Yates, C. Castillo, and F. Saint-Jean. Web Dynamics, Structure, and Page Quality. In *Web Dynamics*, pages 93–112. 2004.

[5] D. P. Bertsekas and J. N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms-a survey. *Automatica*, 27(1) :3–21, 1991.

[6] M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Transactions on Internet Technology (TOIT)*, 5(1) :92–128, 2005.

[7] J. Caverlee, L. Liu, and D. Rocco. Discovering and ranking web services with BASIL : a personalized approach with biased focus. In *Proc. Intl. Conf. on Service-Oriented Computing (ICSOC)*, pages 153–162, 2004.

[8] J. Cho, S. Roy, and R. Adams. Page Quality : In Search of an Unbiased Web Ranking. In *Proc. ACM Symp. on the Management of Data (SIGMOD)*, 2005.

[9] G. M. D. Corso, A. Gulli, and F. Romani. Ranking a Stream of News. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 97–106, 2005.

[10] F. Emekçi, O. D. Sahin, D. Agrawal, and A. E. Abbadi. A Peer-to-Peer Framework for Web Service Discovery with Ranking. In *Proc. Intl. Conf. on Web Services (ICWS)*, pages 192–199, 2004.

[11] S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(User Ranking, Compliance, Verity). In *Proc. Intl. Conf. on Web Services (ICWS)*, pages 200–207, 2004.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 640–651, 2003.

[13] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *J. ACM*, 46(5) :604–632, 1999.

[14] G. Kollias, E. Gallopoulos, and D. B. Szyld. Asynchronous Iterative Computations with Web Information Retrieval Structures : the PageRank Case. In *Proc. Intl. Conf. on Parallel Computing (PARCO)*, 2005.

[15] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Random Graph Models for the Web Graph. In *Proc. Intl. Symp. on Foundations of Computer Science (FOCS)*, pages 57–65, 2000.

[16] A. N. Langville and C. D. Meyer. A Survey of Eigenvector Methods of Web Information Retrieval. *The SIAM Review*, 47(1) :135–161, 2005.

[17] Y. Liu, A. H. H. Ngu, and L. Zeng. QoS computation and policing in dynamic web service selection. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 66–73, 2004.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking : Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[19] J. X. Parreira and G. Weikum. JXP : Global Authority Scores in a P2P Network. In *Proc. Intl. Workshop on the Web and Databases (WebDB)*, pages 31–36, 2005.

[20] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed Pagerank for P2P Systems. In *Proc. Intl. Symp. on High Performance Distributed Computing (HPDC)*, pages 58–69, 2003.

[21] R. S. Varga. *Matrix Iterative Analysis*. Englewood Cliffs, NJ : Prentice-Hall, 1962.

[22] L.-H. Vu, M. Hauswirth, and K. Aberer. QoS-Based Service Selection and Ranking with Trust and Reputation Management. In *Proc. Intl. Conf. on Cooperative Information Systems (CoopIS)*, pages 466–483, 2005.

[23] Y. Wang and D. J. DeWitt. Computing PageRank in a Distributed Internet Search Engine System. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 420–431, 2004.