A graph b-coloring based scheme for Composition-Oriented Web Services Abstraction: COWSA

Lyes DEKAR
supervised by Hamamache KHEDDOUCI

Université de Lyon, Lyon, F-69003, France; Laboratoire LIESP, Université Lyon1
Batiment Nautibus (ex.710), 43 bd du 11 Novembre 1918 F-69622 Villeurbanne Cedex, France
{ldekar,hkheddou}@bat710.univ-lyon1.fr

Abstract. We propose in this paper a self-learning scheme named COWSA, which aims to enhance the performances of existing Web services composition algorithms. Our scheme is based on a new dynamic clustering of Web services, that is oriented to Web services composition. This clustering is performed through using the b-coloring of graphs. We conduct a series of experiments to evaluate the contribution and the performances of our scheme.

1 Introduction

The Web services composition has drawn a great deal of attention recently. With the growth of the Web services number, it is essential to organize them in order to facilitate the discovery of services participating to the composition process. In this paper, we propose to regroup the Web services according to the compositions made by the users (through their requests for composed services). This approach seems to be a new approach for the Web services classification since the majority of Web services classification methods [3] is based on the similarity between Web services. To achieve our clustering, we use a b-coloring of graphs [2]. The b-coloring can be defined as follows : Let $G = (V, E)$ be an undirected connected and simple graph with a vertex set V and an edge set E. The b-coloring of G is a vertex coloring function c from V to the set of colors $\{1, 2, ..., k\}$ such that: 1- for each pair of adjacent vertices $(v_i, v_j) \in E$, $c(v_i) \neq c(v_j)$ (proper coloring). 2- In each color class, there exists at least one vertex having neighbors in all other color classes. Such a vertex is called a *dominating vertex.* A color that has a dominating vertex is called a *dominating color.*

2 The Composition-Oriented Web Services Abstraction: COWSA

In this paper, we identify two kinds of services: *atomic services* and *composite services* (CS). An atomic service is a Web service that fulfils requests without depending on other Web services. A composite service is a Web service that includes a set of atomic services, called *component Web services*, and can be itself a part of another composite service. We define a *complex request* as a user's request that consists of a set of service functions $F_1, ..., F_k$, which cannot be satisfied by one atomic service. A complex request is satisfied by a composite service. In order to satisfy a complex request, a composite service should be constructed through the Web services discovery and composition operations. The aim of this paper is to propose a method to organize the Web services in such a way to enhance the performances of the web services composition methods. Our proposed method consists of regrouping Web services that are often composed together in clusters. A set of services is assumed often composed together if they appear often in a same composite service when satisfying users complex requests. This approach aims to identify stable services sets, which contain services that are regularly and frequently composed together. Then, we obtain an *Abstraction* of Web services, where every identified set of services represent a template of composite services, or what we name a *meta composite service* (MCS). A MCS exhibits a WSDL interface and can be invoked as an atomic service. The MCS are used by the Web services composition methods to accelerate the discovery of component Web services that are required in the composition process. Then, the MCS are used to fulfill the users complex requests. Since the users requests can change over the time, then the MCS can change too. Therefore, we propose to give our method following a self-learning model.

We model our system by an undirected edge-weighted graph $G = (V, E)$, where V is the vertex set and E is the edge set. Vertices in G represent services, edges correspond to the composition relation between services, and edge weights represent the number of times two linked services are composed. This information is presented in the Composition Weight Matrix (CWM). The clustering here consists to regroup services such that we obtain a large intracluster composition weight and a small intercluster composition weight.

The filtered graph: In order to regroup the services joined by a large weighted

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
S_1	0								
S_2	2	0							
S_3	1	13	0						
S_4	12	3	2	0					
S_5	5	14	23	3	0				
S_6	9	14	4	5	15	0			
S_7	1	17	6	0	9	12	0		
S_8	12	19	1	18	12	4	14	0	
S_9	13	6	17	20	3	2	16	2	0

Fig. 1. A composition weight matrix (CWM).

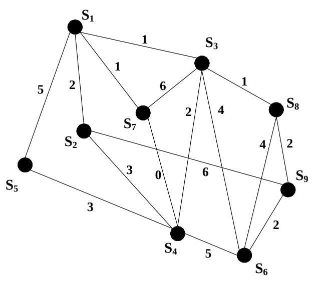

Fig. 2. The filtered graph.

link in the same cluster, we remove all edges with a weight larger than a threshold α. Consequently, after removing these edges, we obtain a *filtered graph* $G_{<\alpha} = (V, E_{<\alpha})$, such that $E_{<\alpha} = \{(v_i, v_j) \mid CWM(v_i, v_j) < \alpha\}$. Figure 2 gives the filtered graph $G_{<9}$ corresponding to the composition weight matrix given in Figure 1.

3 A partial dynamic algorithm for a b-coloring of graphs

The proposed algorithm is composed of two parts. The first part of our algorithm constructs a b-coloring to obtain a partition of the filtered graph $G_{<\alpha}$ into disjoined color classes $\{C_1, C_2, ..., C_k\}$ that represent clusters. The second part of our algorithm maintains this clustering when edges are added or removed from the filtered graph. Then, the proposed algorithm is a partial dynamic algorithm. Before presenting the algorithm, we first give some notations and definitions. We let Δ be the maximum degree of G and $c(v)$ be the color of the vertex v in the graph G. For every vertex v, we define $N(v)$ its open neighborhood, as the set of vertices adjacent to v. The set of colors of $N(v)$ is denoted $N_c(v)$. We note L the color set used in the graph. For each color c used in the graph, we associate a variable $Dom[c]$ that indicates if the color c is a dominating or a non-dominating color (*true*: if c is dominating. *false*: otherwise). Finally, we define a function *weight(v,c)* that indicates the *composition weight* between the vertex v and the color c. This function is defined by: $weight(v, c) = \max\{CWM(v, v') \mid c(v') = c\}$.

3.1 The clustering construction algorithm

In this subsection, we give the first part of our algorithm, which performs the b-coloring and constructs the clusters. The b-coloring is made in two steps:
***Procedure 1: the coloring initialization*:** In the first procedure, the graph is initialized by coloring it with a maximum number of colors $(\Delta + 1)$. The procedure starts by coloring the vertex having maximum degree Δ by the color 1 and adds it to a list S. Then, the procedure color the remaining vertices as follows: the vertex v_i with the largest degree among all colored vertices belonging to S is selected. If there is non-colored vertices v_j adjacent to v_i then a new color is assigned to every one of them and are added to S. The assigned color must be different from those appearing in the neighborhood of v_j or v_i, and must not Exceed $\Delta + 1$. Finally, the procedure checks if the color of v_i is a dominating color. In this case, this color is marked as dominating. After that, the vertex v_i is removed from S. The operation is repeated for every colored vertex until all the graph is colored. Let us consider the filtered graph $G_{<9}$ obtained in Figure 2. By performing the procedure 1, the filtered graph $G_{<9}$ has an initial coloring with a maximum number of colors $\Delta + 1 = 6$, as shown in Figure 3. Among this colors, only the vertex S_4 is a dominating vertex, and then only the color 1 is dominating ($Dom[1] = true$).

Procedure 2: find a b-coloring of G**:** In the coloring obtained after the execution of the previous procedure, some colors could be not dominating. Then, the Procedure 2 finds a b-coloring of a graph G where all the colors are dominating. Hence, the strategy consists to remove a non-dominating color p from the graph by recoloring every vertex colored with p by an already used color not appearing in its neighborhood. If there is choice between many colors, then the color that has the largest composition weight with the vertex is selected. After that, the procedure checks if there is non-dominating colors that have dominating vertex. The operation is repeated until all the colors are marked as dominating. Let us consider the colored graph obtained after the execution of the procedure 1 and given in Figure 3. By performing the procedure 2, we obtain a b-coloring of a graph $G_{<9}$, as shown in Figure 4. We can observe that four colors appear in the

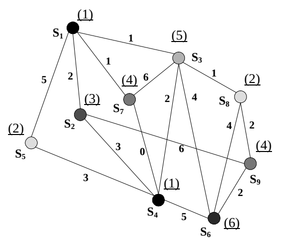

Fig. 3. A graph $G_{<9}$ Coloring initialization.

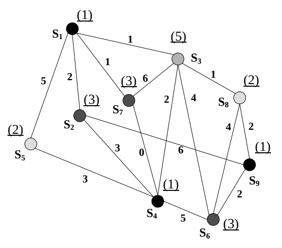

Fig. 4. the b-coloring of a graph $G_{<9}$.

graph $G_{<9}$. This means that the Weighted composition graph is partitioned into four color classes. The composition weight between the services that are in the same cluster (color class) is larger than α.

3.2 The clustering maintenance algorithm

The composition weights between services can evolve in the time because the self-learning of our method. Then, an edge can appear (resp. disappear) in (resp. from) the filtered graph if its weight becomes under the threshold value (resp. its weight exceeds the threshold value). Then, we propose an edge-dynamic algorithm to maintain the b-coloring when edges are added or removed from the graph. We assume in this algorithm that if a color class contains several dominating vertices, then only one of them represents this class. This one is called *representative dominating* vertex. A representative dominating vertex x is said *satisfied* if for any color q in the graph, there exists at least one vertex $y \in N(x)$ such that $c(y) = q$. If there exists only one such a vertex then this one is called a *Satisfaction vertex.* Any change of the satisfaction vertex color can affect the b-coloring. The vertices that are neither representative dominating vertices nor satisfactions vertices are called *Normal vertices.*

The edge adding When an edge (v, y) is added to the graph, we can distinguish three different cases, according to the endpoints of the added edge:
1- The edge is added between two vertices having different colors: In this case, the b-coloring of the graph is not affected. **2- The edge is added between a normal vertex v and another vertex having the same color**: In this case, the coloring is not anymore proper, and the b-coloring conditions are not verified. Then, the color of the normal vertex must be changed. Hence, we can distinguish four different cases: (a) *The vertex v is adjacent to a dominating vertex of every color in the graph*: In this case, we give the vertex v a new color for which v will be dominating. (b) *There exists at least a color c to which the vertex v is not adjacent*: In this case, we give the vertex v this color. (c) *The vertex v is adjacent to all the colors in the graph, and there exists at least one color c that appears only on normal vertices w*: in this case, the vertex v takes the color c, which causes a not proper coloring. Then, the vertex w takes another color c' not appearing in its neighborhood. (d) *The vertex v is adjacent to satisfaction and/or dominating vertices with every color in the graph and there exists at least one color c that does not appear on a dominating vertex adjacent to v, but on satisfaction vertices w adjacent to v*: in this case, the vertex v takes the color c. Hence, the coloring is not anymore proper. Then, we give to the satisfaction vertex w another color not appearing in its neighborhood. The color change of w implies that the dominating vertex x that was satisfied by w is not anymore satisfied. If this dominating vertex is the only one for its colors then the b-coloring is not satisfied. In order to reestablish the b-coloring without systematically removing the non-dominating color, we try to put the previous color of w on another normal vertex z adjacent to v (by respecting a proper coloring) to reestablish the dominating condition. If such a vertex does not exist then we remove the color $c(x)$ from the graph and we color every uncolored vertex with the smallest color not appearing in its neighborhood. **3- The edge is added between a satisfaction vertex v and another satisfaction or representative dominating vertex y having the same color**: The coloring is not proper and then the color of one of the endpoints vertices must be changed. Then, we change the color of a satisfaction vertex v. Hence, the dominating vertex x satisfied by v does not respect anymore the dominating condition. Therefore, we make the same actions as in the point (d) of the previous case.
The edge removing If an edge between a unique dominating vertex v of a color c in the graph and one of its satisfaction vertices y is removed, then v is not anymore dominating. Hence, we perform the same actions as in the point *(d)* of the second case of edge adding.

4 Experiments

In this section, we evaluate the performance and the contribution of our scheme. We implement the Web services composition method given in [1]. Then, we compare the performances of this method with the same method using our scheme. Two metrics are defined to evaluate the performance of our scheme: the *Clus-*

ters stability rate and the *average search time* (AST). The clusters stability rate is the ratio of Web services that change cluster during an interval of time Δt. The average search time AST is the average time required by the service search engine of the composition process to find all the Web services implied in the composition. Figure 5 shows the clusters stability rate according to the simulation

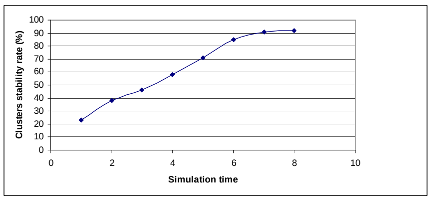

Fig. 5. Clusters (Meta composite services) stability rate vs. Simulation time.

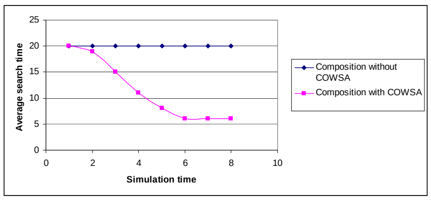

Fig. 6. The average search time vs. Simulation time.

time. We can observe that the cluster stability rate increase with time. This is explained by the self-learning process of COWSA, which enables it to learn more about the behavior of users in the construction of their complex requests. Then, the clusters become more stable and less sensitive to new composite services. Figure 6 shows the average search time (AST) of the Web services composition method given in [1] with using COWSA and without using it. We can observe that from the third time, the AST of the composition using COWSA becomes less than the AST of the composition not using COWSA. We can explain this behavior by the self-learning process of COWSA appearing in Figure 5.

5 Conclusion and future works

In this paper, we present a composition-oriented Web services abstraction scheme, called COWSA. COWSA is a self-learning method that aims to enhance the performances of Web services composition methods.

References

1. Pat. P. W. Chan and M. R. Lyu. Dynamic web service composition: A new approach in building reliable web service. *22nd International Conference on Advanced Information Networking and Applications*, pages 20–25, 2008.
2. B. Effantin and H. Kheddouci. A distributed algorithm for a b-coloring of a graph. *International Symposium on Parallel and Distributed Processing and Applications (ISPA-2006), Lecture Notes in Computer Science*, 4330:430–438, 2006.
3. S. RAM, Y. Hwang, and H. Zhao. A clustering based approach for facilitating semantic web service discovery. *15th Annual Workshop on Information Technolgies and Systems (WITS)*, 2006.