International Journal of Informatics and Communication Technology (IJ-ICT)
Vol.6, No.3, December 2017, pp. 179~188
ISSN: 2252-8776, DOI: 10.11591/ijict.v6i3.pp179-188

An Agent Architecture for QoS-based Web Service Composition Using the Skyline Algorithm

El-Alami Ayoub*, Hair Abdellatif
Laboratory of Applied mathematics and Scientific Calculation, Faculty of Sciences and Technology, Sultan Moulay Slimane University, B.P. 523, Beni Mellal, Morocco.

Article Info

Article history:

Received Aug 12th, 2017
Revised Oct 20th, 2017
Accepted Nov 26th, 2017

Keyword:

Multi-Agent System
Qos
Skyline
Web Service
Web Service Composition

ABSTRACT

Web service composition is a concept based on the built of an abstract process, by combining multiple existing class instances, where during the execution, each service class is replaced by a concrete service, selected from several web service candidates. This approach has as an advantage generating flexible and low coupling applications, based on its conception on many elementary modules available on the web. The process of service selection during the composition is based on several axes, one of these axes is the QoS-based web service selection. The Qos or Quality of Service represent a set of parameters that characterize the non-functional web service aspect (execution time, cost, etc...). The composition of web services based on Qos, is the process which allows the selection of the web services that fulfill the user need, based on its qualities. Selected services should optimize the global QoS of the composed process, while satisfying all the constraints specified by the client in all QoS parameters. In this paper, we propose an approach based on the concept of agent system and Skyline approach to effectively select services for composition, and reducing the number of candidate services to be generated and considered in treatment. To evaluate our approach experimentally, we use a several random datasets of services with random values of qualities.

Corresponding Author:

El-Alami Ayoub,
Laboratory of Applied mathematics and Scientific Calculation,
Faculty of Sciences and Technology,
Sultan Moulay Slimane University,
B.P. 523, Beni Mellal, Morocco.
Email: ayoubalami6@gmail.com

1. INTRODUCTION

Due to the perpetual increase of web services with similar functionality, the composition process becomes costly in terms of response time, take into consideration the number of candidate services in each class of the composition. For this purpose several works and researches have proposed methods to solve this problem of composition based on the Qos. These methods allow the selection of a particular service from among several candidate services, for each domain or class of the composition, in the aim to generate a composition with a better quality of composition.

Using the exhaustive search, can be useful to find the best combinations of services, which they have the optimal levels of QoC. But this method of search still non-practical, with a very large number of possible combinations that can be generated and tested in every execution. This complexity of exhaustive search relating to the number of classes in a composition, also to the number of service candidates in each class. So, this problem can be modeled as a combinatorial problem with NP-hard complexity. Among the useful methods to solve this composition problem, is to use the Skyline selection technique. This technique of Skyline aims to reduce the number of candidate services in each composition class, and therefore reduces the total number of combinations. This reduction is accomplished by the elimination of all dominated services, and keeping dominant ones, where we say that one service is dominated by another, if at least one of its qualities is small than another service.

In this paper, we have focused our work on using an agent-based method, in order to search and perform compositions between two Skylines services, where each Skyline contains the dominant candidate services of a class. The role of agents is to generate a new Skyline by the generation of a partial composition from two previous classes, and keeping just the dominant compositions, and removing the rest of the service candidates. The method consist to repeat the same operation between the partial created composition, and the following class until the end of the composition, to obtain finally a Skyline includes the instances of the optimal combinations of the main composition.

2. QOS-BASED COMPOSITION PROCESS

A service class is defined by a set of web services that provide the same functionality, but they have probably different non-functional characteristics (different values of QoS). In our work we will consider a cloud computing in which, different web services are deployed, and that contain a classification of services according to their functional characteristics (class of services), also a description of non-functional characteristics for each service (Qos). This description of functional and non-functional services are stored and published in a service registry (UDDI), which is accessible via the web, and meet customer needs for simple web services and composition services. Our objective is to create a process in the cloud using system agent to be able to satisfy customer's composition requests.

2.1. QoS parameters

The quality of service parameters is a set of non-functional quantitative characteristics, which determines and describes the performance of a web service. These parameters can include various attributes, like price, reputation, availability, reliability, response time, bandwidth, throughput, etc. All these previous parameters can be evaluated with real positive or negative values. For positive parameters, service customers should search to maximize them, such as reputation and availability, and for negative parameters, its need to be minimized such as cost and response time. For better presentation we will work just with negative parameters, for the positive parameters, they will be transformed into negative by multiplying their values by -1. We suppose the vector $Q_s = \{q_1 (s), ..., q_n (s)\}$ to represent Qos values of web service S, and $q_i(s)$ determines the values of the i-th attribute of the service S.

2.2. QoS calculation for composite services

The QoS parameters for composite services or composition are calculated based on the QoS parameters of its component services. Considering C = {S1, S2, ..., Sn} as a composition of the following web services: {S1, S2, ..., Sn}, the quality of composition C is defined as $Q(C) = \{q_1 (C), q_2 (C),\ldots, q_n (C)\}$, where $q_i(C)$ is estimated value of i-th attribute, and which can be calculated using an aggregation function. This function aims to aggregate the values of corresponding Qos attributes of all component services. Usual aggregation functions for QoS calculation are summation, multiplication, and minimization function [1]. The Table 1 give an example of some QoS attributes and its aggregation function.

Table 1. Examples of QoS aggregation functions

Type	Attributes	Functions
Summation	Response time, Price	$q(C) = \sum_{i=0}^{n} q(\text{Si})$
	Reputation	$q(C) = 1/n \sum_{i=0}^{n} q(\text{Si})$
Multiplication	Availability	$q(C) = \prod_{i=0}^{n} q(Si)$
Minimum	Throughput	$q(C) = \min q(S_i)$

2.2. Problem statement

QoS-based service composition is a problem, which aims at finding the service combination that maximizes the QoS values of a composition.

The simple method for finding the optimal combination is to use an exhaustive search, to generate and compare all possible combinations of candidate web services. For an example composition request with N classes and M services per class, there exist N^M possible combinations to compare. Therefore, the cost of performing an exhaustive search can be very expensive in terms of computation time and memory occupation, needed to store all generated combination. Consequently, Exhaustive search method is inappropriate for run-time service composition in systems with dynamic needs [1]. In the following section, we will propose a solution for this problem, based on the selection and composition of web services, using dominance relationship between candidate services in each service class, to select skyline services. The proposed solution use the multi agent paradigm.

3. SKYLINE AND MULTI-AGENT SYSTEM FOR QOS-BASED COMPOSITION

The purpose of QoS-based composition is to select a set of services, one from each class of services, in such a way that the quality of the selected combination be maximized. Namely that, the selection of the optimal candidate service from each class, does not provide necessarily an optimal composition. Therefore, to find the correct solution, other combinations of services need to be considered. On the other hand, we should mention that not all services are potential candidates to generate the solution. Consequently, the idea of our method is to use the skyline method to distinguish between the candidate services for a given composition, and not potential ones for each class. The skyline is a method that aim to reduce the search space of a given set of elements, by using the dominance relations between services based on QoS values, to identify and eliminate services that are dominated by other services in the same class [1]. A service A is said dominated by another service B, if B is better than or equal to A in all attributes, and B is strictly better in at least one Qos attribute.

Definition [1]. (Dominance) Consider a service class S, and two services X, Y ∈ S, characterized by a set of Q of QoS attributes. X dominates y, denoted as $X \prec Y$, if X is as good or better than Y in all parameters in Q and better in at least one parameter in Q, $\forall k \in [1, |Q|] : q_k(X) \leq q_k(Y)$ and $\exists k \in [1, |Q|] : q_k(X) < q_k(Y)$.

Definition [1]. (Skyline Services) The skyline services of a service class S, comprise those services in S that are not dominated by any other service, $\{x \in S | \neg\exists y \in S: y \prec x\}$.

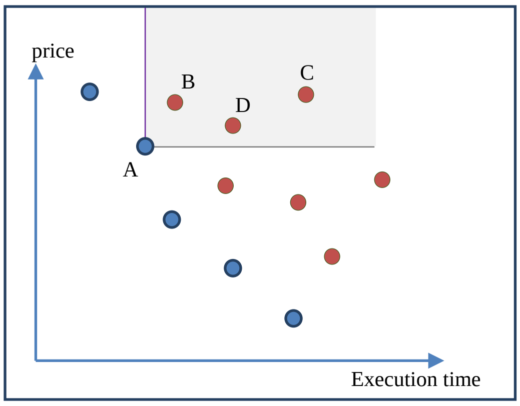

Figure 1. Measuring the Distance to the Skyline

Figure 1 shows an example of application of the Skyline method for a given class, taking into consideration two Qos parameters to evaluate each service, cost and execution time. Each point of the graph represents a candidate service, with the coordinates of a point corresponding to the values of those parameters. Using the method of skyline allowing to reduce the optimal services by distinguish dominant and dominated services, the dominant candidate services in blue color, and the services dominated in orange color. Taking the example of A, B, C and D services, we see that B, C and D all dominated by the service A, because all Qos Attributes of service A, are betters (less) than Qos attributes of other services. Taking the case of services E and A, we can observe that service A is better than E on price, but it is inferior than E on execution time attribute, in this case we said that the services E and A are incomparable services, and both belongs to the optimal skyline services.

3.1. Simple Skyline composition

The idea of the Simple Skyline Composition SSC method is to take advantage of reducing the number of candidate services to be considered in composition, by using the Skyline method on each composite class. After eliminating of dominated services, the SSC performs an exhaustive search to generate all possible combination just from the optimal skyline services, before performing again a Skyline query on generated combinations to get the optimal skyline services as illustrated in figure 1. This method aims at accelerate the composition, also ensuring the ability of application to find the optimal composition, just among the selected service candidates.

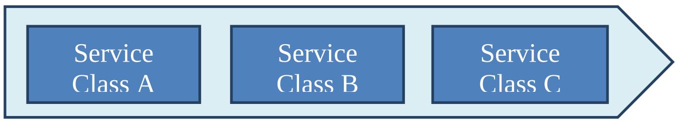

Figure 2. Example of composition of 3 service classes

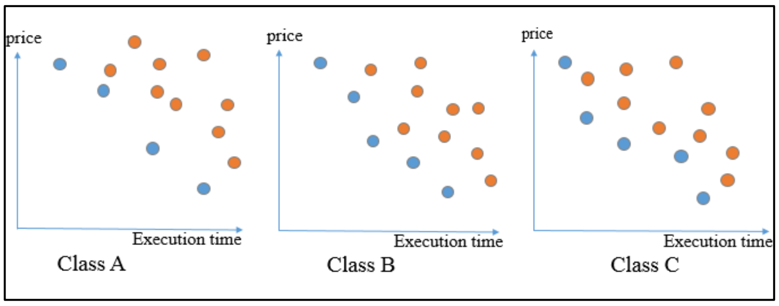

Figure 3. Example of services represented by two quality attributes

Consider the example of Figure 2, where a composition request is processed, with three composite service classes A, B and C, and described by two QoS parameters Price and Execution time. Figure 3 represent the result of first step of execution of SSC algorithm. The SSC performs a skyline request on each service class of composition, to reduce the number of candidate services to combined, represented in blue color. Comparing the number of candidate combinations to generate in each case, we find that SSC algorithm generate 4*5*5=100 candidate combinations. On the other hand, to perform an exhaustive search without SSC we must generate 13*14*14=2584 combinations.

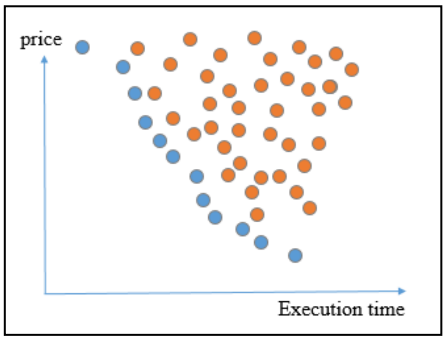

Figure 4. Measuring the Distance to the Skyline

The second step of SSC algorithm is performing a new skyline query on the generated combinations, to select finally the best and optimal skyline compositions. Each blue points in figure 4, represents an optimal composition, composed from three services belongs A, B and C service classes. The Simple Skyline Composition is a simple method aims to performing exhaustive search on a reduce space of services, and maintain the same set of optimal combinations.

```
**Algorithm 1**: Simple Skyline Composition
Input: a set of service classes Comp.
Output: a set of optimal services So.
1: for all Cls ∈ Comp do
2:         $S_{sky}$.add( Skyline(Cls))
3: end for
4: So = $S_{sky}$ .pop()
5: for all c ∈ $S_{sky}$ do
6:         So = combine(So, c)
7: end for
8: So = Skyline(So)
```

3.2. Improved Skyline composition

In order to improve the performance of the previous algorithm of composition. We propose a method of composition based on skylines selection and agent system paradigm, in such a way that, the application of skyline will be performed after each combination of two service classes, and the result of this skyline set, will be combined again with the following class of composition. These two operations of binary composition and skyline reduction must be repeated until the last service class of the global composition figure 5. This approach reduce more and more the total number of candidate combinations, and at the same time, it keep the same results of the preceding algorithms.

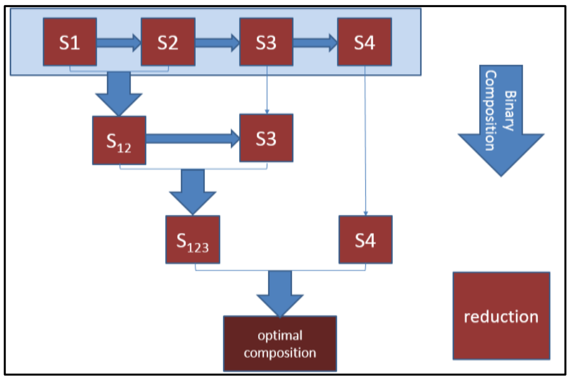

Figure 5. Global presentation of improved skyline composition

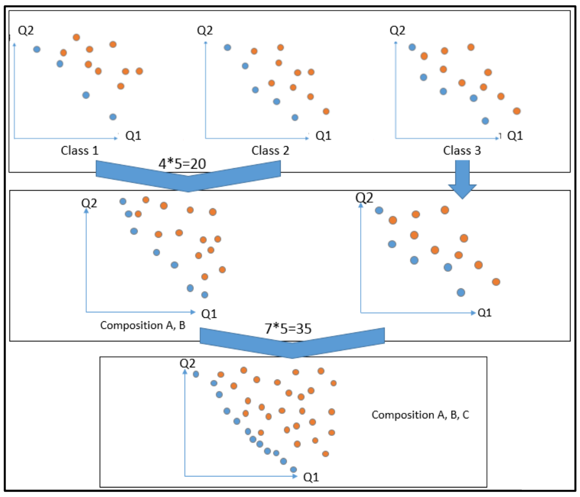

Figure 6. Example of application of improved skyline composition

To implement our approach, we propose an algorithm based on the notion of system agents, to benefit from their capabilities in terms of synchronization of tasks and parallelism. This implementation requires the definition of two functionalities, the binary combination and skyline reduction, using an agent-oriented architecture. In this context, two agents has been proposed to accomplish these tasks: composition agent and control agent. Figure 7 and 8 show an overall working diagram and an activity diagram, which indicates the tasks and activities performed by each agent.

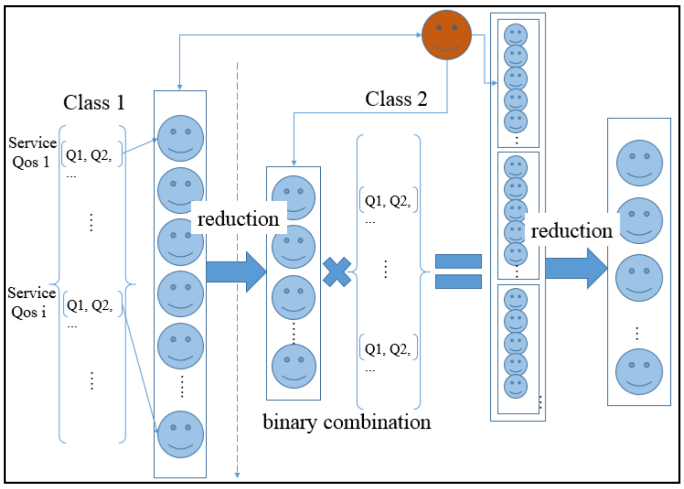

Figure 7. demonstration of reduction and binary composition in a composition process

The idea of the approach is to take advantage of the collective intelligent of system agents to resolve this composition problem. At the first step, a set of initial composition agents generated, where each agent is according to one concrete service of the first service class of composition. A composition agent can be initialized with a vector of QoS attributes of a corresponding web service or with the vector QoS of a partial composition. After the creation of all agents of first class, they send an ACL messages with their vector of qualities to the control agent. After receiving all messages, the control agent triggers the process of reduction, by applying the Skyline algorithm on the received data (QoS attributes) to distinguish the dominant and dominated agents. In order to reduce the candidates agents, the control agent send an **Agree** message to dominant agents and a **Cancel** message to others. When a composition agent receives an **Agree** message from the control agent, the agent starts operation of reproduction. This operation aims to create a new set of composition agents, corresponding to the services of the following service class, where each new agent initialized with the Qos vector of a web service of following class, and the Qos vector of its creator agent. To calculate his own Qos Vector, a composition agent use aggregation functions to calculate new Qos attributes, from the Qos Vector of its own service, and the Qos vector of its own creator agent. After the creation of all new composition agents of the following class, the creator composition agent stops its execution. In the case where the composition agent receiving a **Cancel** message, the agent stops its execution. Once all composition agents of old generation have stopped their execution, the new composition agents starts the communication with the control agent by sending again their Qos vector. Those operations of reduction and reproduction repeated until arriving at the last service class of composition. The dominant composition agents remained at last represents the solution of service composition. The description of composition agent and control agents can be summarized as follows:

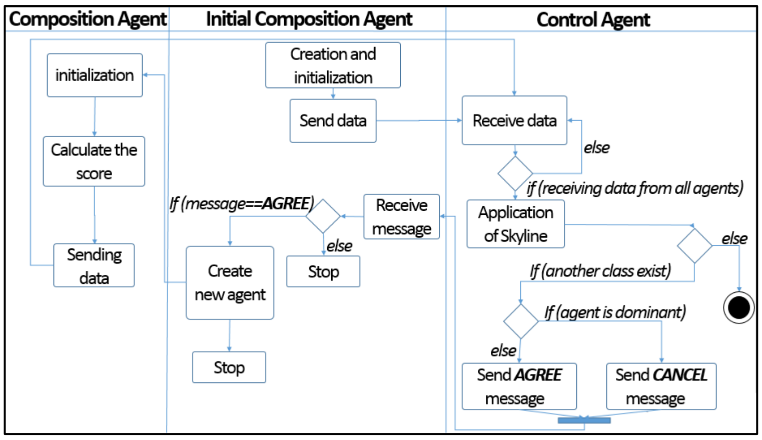

Figure 8. Activity diagram of composition and control agents

Control agent: is the component responsible for communication with all compositions agents. Its role consists on calculating of dominance relationship, and generating of skyline services, based on the messages received from current composition agents. This operation aims to determine the composition agents to be eliminated or to be preserved for the next level composition, or to determine the global optimal composition.

Composition Agent: is the component that contains the QoS vector of a service or of a partial composition. Its role is to send a message contains the QoS vector to the control agent, and consequently based on received decision from the controller, the composition terminate its execution, or reproduce other composition agents with the service qualities of the following classes, before similarly completing its execution.

Algorithm 2 describe the different behaviors of the control agent from initialization until the last step and returning the optimal result. It take as input a set of service classes of composition C= {c1, c2, c3…}, where each entry c_i denotes a set of services that belongs to the i-th service class. It use also an integer variable I to preserve the current level of composition.

***Algorithm 2*: Control Agent algorithm**
Input: a set of service classes of a composition: C.
Output: optimal combinations of the composition C: Cop

```
1: nb_agent = size(C[I])
2: while (i < nb_agent)
3:     Q.add(receive())
3:     i++
4: end while
5: $Q_{op}$ = Skyline (Q)
6: while (i < nb_agent)
3:     for q in $Q_{op}$
7:         if (q == Q[i])
8:             Send (Q.get(i).getSender(), AGREE)
/*  sending an AGREE message to dominant agents */
9:             Q.remove(i)
10:        End if
11:        end for
12: end while
13: for each q in Q
           Send (q.getSender(), CANCEL)
/*  sending a CANCEL message to dominated agents */
14: end for
15: return $Q_{op}$
```

Algorithm 3 define the behaviors of the composition agents, knowing that during execution of a composition, several composition agents can be generated in function of the number of classes, and number of services in each class. On the contrary, one instance of control agent is generated during an execution. A composition Agent take as input a Qos vector of one web service Q, or a Qos of a partial composition. The Qos of partial composition is calculated in the beginning of its life cycle, by using aggregation function between Qos vector of its creator agent Q_1, and the Qos of the associated service Q.

***Algorithm 3*:** Composition agent algorithm
Input: Qos vector of a service: Q, Qos vector of control agent: Q_1, identity of control agent: AgC, set of service classes of the composition: C, the current level of composition to which the correspond service belongs: I.

```
1: if (I>0)
2:     Q=U(Q, Q_1)          // calculate new Qos attributes of partial composition with aggregation functions U
3: end if
4: send(AgC ,Q)  // send a message with Vector Qos to control agent
5: Msg=receive(AgC)   // receive controller response
6: if (Msg == AGREE)
7:     for each S in C[I+1]
8:          createCompositionAgent(S.getQualite(), Q, I+1)          // create a new composition agent
9:     end for
10: stop() // terminate the execution
11: else if (Msg == CANCEL)
12:    stop()
13: end If
```

4. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of our improved skyline composition approach, in terms of the quantity of candidate combinations to be calculated and generated, also in terms of execution time required to reach the optimal solution.

Our experience is performed using a set of web services generated with random quality attributes. These services are identified by two abstract parameters of Qos to minimized, and which take random values between 1 and 50. In the evaluation, we have initialized our program by a given composition with 15 service classes, where each class contains 20 generated services, which means that we have a total of 15^{20} candidate combinations for this experimental composition. The following figures 10 and 11 represent the variation of the execution time, and of the number of generated combinations as a function of the numbers of the classes and services of the composition. The displayed values represents the average of 10 executions of the algorithm, with the same parameters (number of classes and services), and with different values of services.

We implemented the improved skyline composition algorithm described previously in Java, and using JADE platform as a multi-agent system, to implement our system composition and control agents. The experiments were conducted on a Sony laptop machine with an Intel core i5 2.40GHz CPU and 8 GB RAM.

We measured the average execution time and candidate composition generated, required by algorithm for solving the given random composition, defining the number of candidate services by 20 services per class, and 15 classes par composition. The results of this experiment are presented in figure 10 and 11.

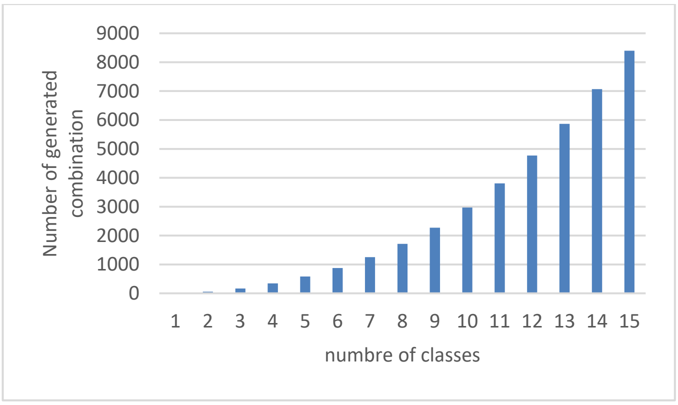

Figure 10: Measuring the variation of number of compositions

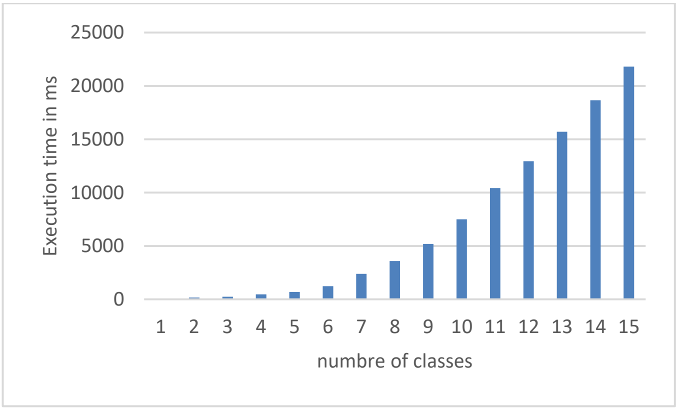

Figure 11: Measuring the variation of execution time for a composition

5. CONCLUSION

In this paper, we propose a method to solve the problem of composition based on the qualities of the services, using the Skyline method and the agent-oriented architecture. The proposed method perform by using Skyline method, which aim to reduce the number of services to be combined and generated, based on domination relationship between candidate services. On the other hand, to increase performance and speed of algorithm, we decide to use the paradigm of system agents to take advantage of its collective and intelligence behaviors to decompose and solve easily the composition problem. To implement the composition method we use a multi-agent system as platform for our program, which contains two types of agents, one for to perform the composition process, and another one to control and the orchestration composition operations. Our experiments have shown that the performance of our method can be useful for composition with a very large quantity of candidate combinations, and which cannot be solved with simple exhaustive search. Finally, using agent paradigm and skyline method can reduce the execution time, also optimize the memory taken for storing data of all these combinations.

REFERENCES

[1] M. Alrifai; D. Skoutas; T. Risse, "Selecting Skyline Services for QoS-based Web Service Composition", *Raleigh NC USA*, pp.11-20, 2010.

[2] M. Alrifai; T. Risse. "Combining global optimization with local selection for efficient qos-aware service composition", In WWW, pp.881-890, 2009.

[3] D. Skoutas; D. Sacharidis, "A. Simitsis, and T. Sellis. Serving the sky: Discovering and selecting semantic web services through dynamic skyline queries*", International Conference on Semantic Computing*, pp.222-229, 2008.

[4] Benouaret; D. Benslimane; A. Hadjali, *"On the Use of Fuzzy Dominance for Computing Service Skyline Based on QoS"*. In the 9th International Conference on Web Services (IEEE ICWS 2011).

[5] Yang Yu; Jian Chen; Shangquan Lin; Ying Wang, ”A Dynamic QoS-Aware Logistics Service Composition Algorithm Based on Social Network”, *IEEE transactions on emerging topics in computing*, volume 2, no. 4, december 2014.

[6] Mike P; Papazoglou Willem-Jan van den Heuvel, “Service oriented architectures: approaches, technologies and research issues” 3 March 2007 Springer-Verlag.

[7] G.Canfora, *“An approach for QoS-aware service composition based on genetic algorithms*”, Proceedings of the 2005 conference on Genetic and evolutionary computation, pp.1069-1075, 2005.

[8] Abourezk; A. Idrissi, *“Introduction of an outranking method in the Cloud computing research and Selection System based on the Skyline”,* Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on. 1-12, 2014.

[9] Paolo Busetta; Massimo Zancanaro, *“Open social agent architecture for distributed multimedia. In Workshop on Agents at Work: Deployed applications”,* 2nd International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2003), Melbourne, Victoria, Australia, July 2003.

[10] Al-Masri E; Mahmoud QH, *“QoS-based discovery and ranking of web services”,* In: 16th International Conference on Computer Communications and Networks (ICCCN 2007), pp 529–534.

[11] Börzsönyi S; Kossmann D; Stocker K, *The skyline operator*. In: Proceedings of the 17th international conference on data engineering (ICDE’01), pp 421–430, 2011.

[12] S. Wang; Q Sun; H. Zou; F.Yang, “Particle Swarm Optimization with Skyline Operator for Fast Cloud-based Web Service Composition”, *Mobile Netw Appl*, pp. 116–121, 2013.

[13] T. Yu; Y. Zhang; K.-J. Lin, “Efficient algorithms for web services selection with end-to-end qos constraints”, *ACM Trans. on the Web*, 1(1), 2007.

[14] M. Abourezq; A. Idrissi, “Introduction of an outranking method in the Cloud Computing Research and Selection System based on the Skyline,”, 28-30 May 2014.

[15] Kang; K. M. Sim, *"Cloudle An Agent-based Cloud Search Engine that Consults a Cloud Ontology*", Cloud Computing and Virtualization Conference, 2010.

[16] A. Idrissi; M. Abourezq, "Skyline in Cloud Computing", *Journal of Theoretical and Applied Information Technology,* Vol. 60, No. 3, February 2014.

[17] C. Zeng, X. Guo, W. Ou and D. Han, Cloud Computing Service Composition and Search Based on Semantic, *Cloud Computing,* Vol. 5931, pp. 290-300, 2009.