DOI 10.12694/scpe.v18i2.1283
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 2, pp. 105–116. http://www.scpe.org

ARANK: A MULTI-AGENT BASED APPROACH FOR RANKING OF CLOUD COMPUTING SERVICES

AREZOO JAHANI, FARNAZ DERAKHSHAN, AND LEYLI MOHAMMAD KHANLI *

Abstract. Cloud computing enables access to computing, processing and storage resources as a service. These services offered to users through internet and based on payment obligations. There are various service providers and services for users, so users have the challenge of choosing appropriate service, matching their needs. Therefore, having a system which helps users to choose the best service based on their need is very important. In this paper, we propose a new multi-agent based method named ARank, which is applied for ranking algorithm to reduce the waiting time of users. ARank method uses intelligent agents which they choose some candidate services and rank these services based on the quality of service values. Furthermore, the agents of ARank include the satisfaction rate of the earlier users in ranking process. The results of our evaluation show reduction in the waiting time of the users using ARank method, compared to the existing related work, Analytic Hierarchical Process (AHP) and Singular Value Decomposition (SVD).

Key words: Cloud computing, Cloud services, Multi agent system (MAS), Ranking, Quality of service.

AMS subject classifications. 68M14, 68T42

1. Introduction. Cloud computing enables using different resources as a service [1]. While there are various service providers with different services, the act of choosing one service from a set of services is a challenge [2]-[4]. Because users have different requirements, therefore, reaching the highest performance with the lowest cost depends on choosing an appropriate service. It can be said that if the best service is selected, then we can use the total capacity of the provider [5], [6].

Cloud computing offers three types of services: Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). The difference of these services is the way they grant access of resources and applications available to user. IaaS provides user with the highest level of flexibility and management control over her/his IT resources [7], [8]. PaaS allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. In this type of service, user do not need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running users application. With a SaaS offered user do not have to think about how the service is maintained; user only need to think about how she/he will use that particular software [9]. Each specific service from these services is selected based on different attributes. For example, each service can be proposed with attributes like accountability, cost, security and usability [2].

In cloud environment, users have different requirements and the services have many kinds of attributes with variable quality. So selecting a service match with the requirement of users is difficult, and does not have a clear solution. But we can find an answer near to optimum one [10]- [12]. In other words, it would be a repetitive task to select some candidate services then compare them with users requirements to rank, while this task can be done in parallel [13]. Therefore, it can be found that ranking services is a problem that can be solved with the distributed problem solving. This kind of problems can be solved using evolutionary algorithms, parallel algorithms and multi agent systems (MAS [14]).

The main challenge of this paper is to select some services based on the request of users, then ranking them. Most existing ranking methods do not include all standard attributes, and they rank services based on some limited attributes [14]- [16]. Some methods include all of the existing services in the ranking [17]- [19]; while it is obvious that there is no need to rank all available services and in fact services with rank 100 and more are not usable for users [20]. Therefore, it can be concluded that it is necessary to have a ranking method including all attributes (or proposed methods must be flexible to add new attributes to it in the future). Also, such a system is not required to rank all services, and one can choose some candidates in advance [21]- [23]. This paper try to address all of these challenges.

*Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran. (a.jahani@tabrizu.ac.ir, derakhshan@tabrizu.ac.ir, l-khanli@tabrizu.ac.ir)

In this paper, we use multi-agent systems, and called our proposed method ARank (Agent Rank). In fact, agents can do works in parallel and in distributed way. We use the concepts of cooperation and coordination of agents in our system. In ARank, agents found some candidate services that can satisfy the requirements of users. Then, after comparing the candidate services with requirements, rank the cloud computing services. The ranking method in ARank has steps as follows: at the first step, ARank chooses multiple candidates services that taking advantage of agents. Then, ranks the selected candidates based on quality of services and scores that given by earlear users. As a result, ARank can achieve higher performance during candidate selection phase, because of the cooperation and parallel operations of agents.

After this introduction, we continue this paper as follows: Section 2 reviews and analyses the related work. Section 3 explores the attributes of services for cloud computing. In Section 4, we present our proposed method and architecture for ranking cloud computing services using MASs which we called ARank. Section 5 introduces the ranking phases of ARank and shows the cooperation and negotiation between agents, followed by Section 6 that offers the analyses results. Finally, Section 7 gives a conclusion and future work ideas.

2. Related work. During recent years, several methods have been proposed for ranking cloud computing services. Most proposed methods included all the services in the ranking process [24]. Some of these methods rank all services and show k top rankings to user at the end of ranking, however, ranking all services is not needed. It is effective to choose the best choices of candidate services, in order to avoid ranking all of them. Our proposed ARank method select some candidate services in parallel with ranking them using agents intelligence and their ability in cooperation and coordination.

Chan and Chieu [25] proposed a data collection and analyses algorithm, based on Singular Value Decomposition (SVD) [25]. This mechanism determines a service among all the services for a specific application with a limited set of requirements with each runtime. The architecture of the proposed method is that a user enters his/her requirements and the system finds a service provider using the proposed method and allocates the users request to the service. So user does not interact with the service selection process, and all is done by the system. This method is useful for those kinds of users which are unaware of service types and their attributes.

Choudhury et al. [26] proposed a method named Service Ranking Systems (SRS) for Cloud Vendors. This method performs ranking in two ways: dynamic, and static. In static mode, SRS ranked all available services in cloud market by not noticing to users requirements. But in dynamic mode, it ranked services according to users requirements.

Zheng et al. [27] proposed a framework named QoS Ranking which ranks services by predicting quality of services values and finding similar users to recall exact value of services and avoid time wastage.

Qu et al. [14] proposed a method which used two types of information for service ranking. The information is gained by earlier users feedback from services and evaluating quality of services values or monitoring services. This info is processed using fuzzy methods and various evaluation tools and then ranks services.

Rajkumar et al. [6] proposed a framework named SMICloud (Service Measurement Index) [30] to rank services based on Analytic Hierarchical Process (AHP) [28] which ranks services in 4 phases: receiving users requirements hierarchically, receiving weight of attributes, finding relations between attributes weight and calculating decision number, and ranking services. SMICloud ranked services based on SMI (Service Measurement Index) attributes [30] that presented by Cloud Service Measurement Index Consortium (CSMIC) and was especially designed for evaluate and compare cloud computing services.

Jahani and Mohammad Khanli [29] (our earlier work) proposed a ranking system (NSGA_SR) that can select some candidate services and rank them based on multi-objective optimization problem. This method is so useful and need less time than other methods in literature. However ARank improve NSGA_SR; Similar to NSGA_SR, ARank selects some candidate services and ranks them based on users requirements, however ARank perform these works in parallel by agents abilities that be more efficient than NSGA_SR. NSGA_SR rank services based on all essential and non-essential requirements. But ARank uses only the requirements that presented by users.

It can be seen that, except the approach offered by Rajkumar [6] and Jahani [29], other proposed methods do ranking based on limited attributes and they do not consider SMI attributes, in their ranking approach. In this way, user often searches for a service with special attributes which the ranking process do not perform ranking based on those attributes. To have more information on SMI, SMI attributes will be explained in the

next section.

3. Quality of service attributes . Each cloud service is identified by its quality attributes. Until 2012, there were not any standard way to compare quality of attributes. In 2012, universal consortium CSMIC announced a standard, named SMI (Service Measurement Index) attributes, for representing the quality of service attributes [30]- [32]. After that, these attributes become the basis for evaluation and comparison of cloud services.

SMI attributes are designed based on ISO standards and they are a standard method to compare cloud services. Generally, there are seven main attributes defined in SMI. Accountability, agility, financial, performance, assurance, security and privacy and usability, also these attributes include some sub-attributes. Each of these attributes is given in the following [30]- [32]:

1. **Accountability:** If a system or a service does not have accountability, no one or organization will attempt to use it, and they will not use these systems for assigning critical data. Because, it is possible that they cannot access their critical data when they need. The sub-attributes of this attribute are: audit, fill, data ownership and tolerability.
2. **Agility:** This attribute allows users to change or expand their service charge free. Compatibility, capacity, elasticity and extensibility are some of its sub-attributes.
3. **Financial:** It is important for users to know that whether the charge that they pay for a service is cost efficient or not. Pay as use is a sub-attribute of this one too.
4. **Performance:** Performance means the least time or energy use for the largest task done. This attribute measures the usability of services to the request of users. Having performance attribute helps to find the largest load that a system can resist against it. Accuracy and competency are the sub-attributes of this one.
5. **Assurance:** Assurance shows the success or failure probability of a system or cluster completing its tasks within a limited time without any system failure.
6. **Security:** The security of a system is depended by the protection of the data and creating security for the transferred data to the system.
7. **Usability:** This attribute represents the fast compromise of cloud service with various requests and different environment. This attribute allows users to use the service and easily delegate their tasks to the system. Our proposed method, ARank, which will be explained in detail in the following, uses all seven attributes and their sub-attributes for cloud service ranking.

4. Our Proposed Architecture. In this section, we present the architecture of our new proposal for ranking cloud computing services. There are two main features for our method: First, it is based on multi-agent system (MAS), so it takes the advantage of distribution of MASs. Second, our proposed method prevents to rank all cloud computing services and the ranking is based on ranking of the candidate services. Thus, the waiting time of users for ranking process will be decreased.

Our proposed method called ARank (Agent Rank), which is based on multi-agent systems and it uses multiple agents which have cooperation and coordination with others. ARank avoids ranking all the services, by choosing some candidates services based on the requirements of user. Then, it ranks candidate services. This architecture contains three types of agents: user agent, manager agent and zone agent. These agents are autonomous in the environment. The agents can negotiate with others when it is necessary, so they can rapidly reach general goals of the system. The general goal of the system has been assigned to agents as a distributed solvable problem for cloud services ranking. The architecture of ARank, which represents agents of the system and the communications between them, is shown in Fig. 4.1.

User agent is an agent which is assigned to use ranking service by the user, as shown in Fig. 4.1. So it is possible that some user agents exist simultaneously and they concurrently do their tasks for their users. Manager agent acts as a central coordinator in the ranking system and coordinates agents to reach maximum performance of the system. Zone agent has the specifications of service providers. These agents are able to communicate with others, to perform parallel tasks and also to get the best result in the least possible time. Each agent is described in detail as follows:

User agent: This agent is created when user requests for service ranking. So the number of this agent is dynamic in the system, and it depends on the number of active users in the ranking system. This agent sends

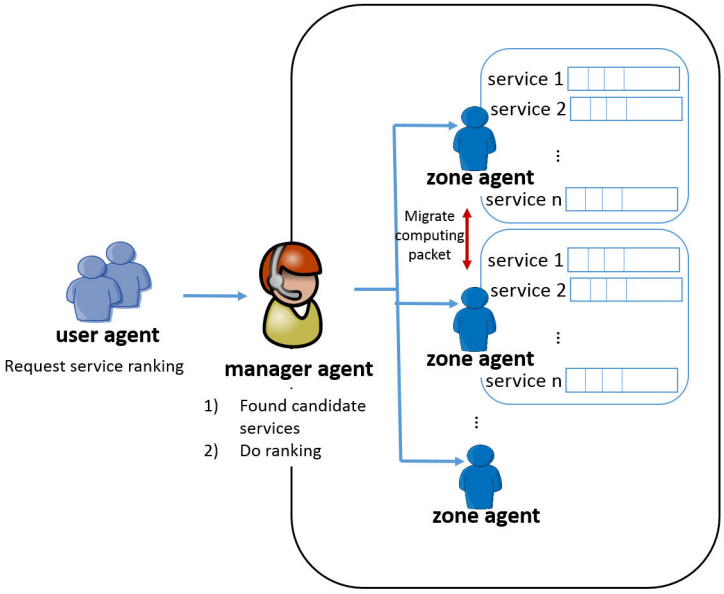

FIG. 4.1. *Multi-agent architecture of proposed ARank method*

user request to manager agent in the form of a message. User agent also has another important task: it asks user to enter his/her degree of satisfaction from used service(s) as a score, after it gives the user the ranking results. Then, user agent returns the scores to the zone agent related to each service. Because, the next levels of ranking is the consideration of the user satisfaction feedbacks and involves them in the ranking. This method makes a competition between service providers. Because, the scores are saved by zone agents (away from the accessibility of service provider), which denotes the history of service provider and record of those services. It is important that each request of the users includes some quality requirements of SMI attributes.

Manager Agent: This agent is responsible for the general coordination of the system. Manager agent receives the requests from user agents, and then for each request, it sends a message (containing information about the request) to every zone agent. In fact, it requests the zone agent to find some candidate for the received message. In response to this message, zone agent sends the name and sum of the scores (collected from earlier users) to manager agent.

Candidate services are chosen based on the quality of services that user requests. Therefore, all candidate services chosen are good for user. To determine how good candidate services are, it is important to rank them. Ranking the candidate services is a time-consuming task. Therefore, the manager agent creates a package, named *'Computing Package'*, which includes the data received from the zone agent, including the name of candidate services and their scores. Ranking must be done based on the content of these packages.

In our proposed method, the manger agent selects one of the zone agents to compute the package and takes the advantage of its computing capability. With this method, multiple *computing packages* are given to the zone agents simultaneously. If multiple users use the ranking system concurrently, the system can compute the *computing package* of each user simultaneously. One of the benefits of this method is the ability of coordination between agents, which causes fast result in ranking the systems that provides better and more accurate services to the users. In order to compute the *computing packages* for service ranking, we used the concept of domination, which will be explained completely in Section 5.3.

Zone agent: Each zone agent contains some services from multiple providers. It is necessary for service providers to register their services in zone agents to benefit the ranking system and introduce themselves to the users. Each zone agent can determine the kind of services that users need. For example, some zone agents only register PaaS type of services, some of them register only for SaaS or only for IaaS type of clouds. So, it is obvious that if the ranking system can do its job perfect, more providers will register in the system. Therefore,

the ranking system will be powerful enough to offer best choices for the requests of users.

The other task of zone agents is the duty of collecting the request of users, and generating candidate services for each request. These agents send the value of attributes and scores of the chosen candidate services to the manager agents. Their next task is receiving *computing packages* and sending them to manager agents. Every zone agent can decide how to compute a *computing package* after receiving a compute message. If it is possible for the zone agent to do the computing and sending the result of a *computing package* by itself, so does it. However, if the zone agent cannot compute the package in the required time, the zone agent negotiates with other zone agents and allows other zone agent calculate the package. The computing result is sent to the manager agent. This agent also stores and uses scores given by user from the user agent. In the architecture, each agent has some exploratory functions for completing task, negotiation and coordination with other agents. Each of these functions are detailed in the next section.

5. Ranking with ARank. The architecture of our proposed method (ARank) is shown in Fig. 4.1. This architecture includes the agents and all communications between them. The main purpose of this architecture is to find some candidate service and ranking them based on users requests in the shortest time. Also the system must have the ability to respond to more users in every time slot simultaneously.

5.1. Modeling the Input Variables of ARank. The inputs for ARank method are the requirements of users on the basis of SMI, and also the priority of those attributes. According to Section 3, in SMI, there are 7 main attributes and some sub-attributes for recognizing each service. These attributes are classified into two types. Within the first type, values are presented with one number, so the value of the attributes must be the same with the users request.

However, the second type of values is presented with two numbers which are the upper bound and lower bound values of a range. Our ranking system has offered a method which is adaptable with both kinds. The system converts each single input into a two number range (with the starting and the ending of the range). For example, if the assurance attribute value is 14, given by user, the system converts it into range [14,14]. The set of requirements by the user is shown in Eq.(5.1).

$$R_u =< r_1, r_2, ..., r_i, ..., r_Q > \qquad u \in [1, m] \tag{5.1}$$

In Eq. (5.1), R_u is an ordered pair of requirements of user which is shown with r_i, and each r_i is the requirement for attribute i. Q is the number of all attributes. Variable u is the id number of system users, while the maximum of users would be m.

Other input from the user is the priority of each attribute. Because from the viewpoint of the user, some attributes are too important and they must be guaranteed at any cost. in cotrast, some attributes are not so important. The user input values for each attribute ranging from 0 to 1. Attributes with priority 1 must be guaranteed at highest quality by proposed services. The set of user priority is shown in Eq. (5.2).

$$C_u =< c_1, c_2, ..., c_i, ..., c_Q > \qquad u \in [1, m] \tag{5.2}$$

As shown in Eq. (5.2), C_u is an ordered pair of priorities of user u which is shown by c_i, each c_i is the value related to attribute i. Q shows the total quality attributes.

According to the presented architecture, service providers must register in zone agents and send them their services information. The services information is shown by SMI attributes quality which is represented in Eq. (5.3).

$$S_j =< q_1, q_2, ..., q_i, ..., q_Q > \qquad j \in [1, n] \tag{5.3}$$

As shown in Eq. (5.3), S_j is the mark for service j which is represented by an ordered pair of quality values q_i. If the total number of quality attributes is Q, variable j describes the id of the services of the corresponding zone agent. The number of services in each zone agent is equal to n. By the way, it is noticeable that the qualities of services values are measured using network observers or by the ranking system itself, so that service providers cannot violate registration phase.

5.2. Supplement of Candidate Services in ARank. A message is sent to a manager agent when a user agent gathered users required values. Following that, the manager agent sends the users required values to all zone agents. Each zone agent starts searching for the services which satisfy users requirements in its set of services after it receives the message from the manager agent. The search process is shown in Algorithm 1.

Algorithm 1 finding candidate services (by zone agents)

Require: R_u, S_j
Ensure: $candidateservices, feedback$;
1: **for** all S_i **do**
2: **if** S_i is in range of R_u **then**
3: add S_i to candidate services
4: **end if**
5: **end for**
6: return candidate services to manager agent
7: return feedback for candidate services to manager agent

According to Algorithm 1, each service is analyzed by zone agent to find the matching services to users requirements (lines 1 and 2). If a service matched the requirements, then it is added to the candidate services set (line 3). At the end, zone agent returns the set of candidate services and scores of each service given by earlier users as an output (line 6 and 7).

5.3. Cooperation Protocol in ARank. When candidate services are found and sent to manager agent, in the next step, manager agent packs the list of candidate services and their scores (given from different zone agents) into a message called *computing package*. The ranking result will be revealed when the contents of *computing package* is computed.

Each manager agent is free to choose a zone agent as a destination of *computing package* and it is possible to use different methods to send the message. It might send the message in circular way, or it decides based on the load on each zone agent. But to be aware of other agents, it should always send a message to be informed of agents status. Sending this message will increase the load on connections, and results in execs of messages in inter-agent communications. So it is better to deal the problem like the load balancing problem in the cloud computing environment and use methods presented for them in [21]- [22].

Load balancing is used in service providers unit. Service providers use servers which might be in different geographical places. Each service provider tries to allocate each service to a low loaded server to increase quality of service. This process improves the quality of service and called load balancing [21].

When load balancing is being processed in cloud computing environments, a dynamic method is used to choose the best server for current requests of system [21]. The first allocation is done with random selection method or in circular turns. After that to prevent the incrementing the load of servers, it is allowed to servers that if they cannot handle the task, they search for a server with a lower load to autonomously assign the task to them. In this case, the load on the central service provider is decreased, and this agent does not get busy with complicated initial task allocation process [22].

Similarly, in order to decline the working load, manager agent get the task of computing the *computing package* to each zone agent randomly, or in circular turn way. Following that, each zone agent autonomously select to compute the *computing package* or assign it to another zone agent through negotiation. The protocol of sending packages and also sending the user requirements to find candidate services is shown in Fig. 5.1.

As shown in Fig. 5.1, manager agent sends a message to zone agents with each request from user. Zone agents choose some candidate services and send their name and scores to the manager agent. Followed by that, the manager agent creates a package including quality attributes of services and their scores and sends it to a zone agent using random algorithm or circular turn method. Next, the ranking result is given to the manager agent, and this agent guides the user to the selected service. Ranking steps (computing the *computing package*) and inter-agent negotiation protocol is described in the following.

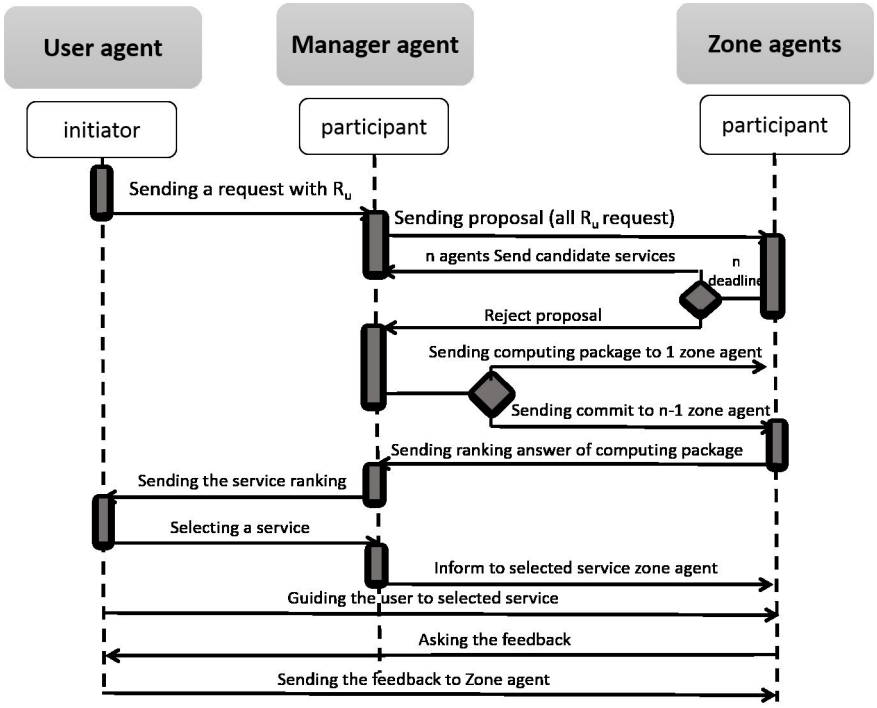

FIG. 5.1. *Cooperation protocol between agents in ARank*

5.4. Ranking Candidate Services (*Computing Package*). The *computing package* is sent by manager agent. The content of this *computing package* is quality attributes and scores given by earlier users to the service. Computing of rankings is assigned to one of the zone agents. If only quality attribute of the services is received, ranking of services is done easily with sorting the services in anticlimactic order. But two types of information (the quality of services attributes, and scores given by earlier users) exist. Use of these two types of information is one of the main advantage of the proposed method. Ranking of the services must be done based on these two goals. Eq. (5.4) and Eq. (5.5) present computing functions.

$$F_1 = \sum_{i=1}^{7} c_i \times q_i \tag{5.4}$$

$$F_2 = \sum_{i=1}^{7} c_i \times q_i{}^{feedback} \tag{5.5}$$

As shown in Eq. (5.4), function $F1$ shows the sum of quality attributes of each candidate service, variable q_i shows the ith quality attribute of each candidate service. According to Eq. (5.5), function $F2$ presents the sum of users feedback or users scores for each quality attribute, variable presents the score given to each quality attribute for attribute i of each candidate service. The input of *computing package* is a number of candidate services with two goal functions. Input is shown in Fig. 5.2.

According to Fig. 5.2, the input of ranking algorithm is some candidate services found by zone agents. Each candidate services are presented with two values, sum of quality of attributes values ($F1$), and sum of scores given by previous users ($F2$).

Now, in order to compare services based on two separate goal functions, domination method is used. In the domination method, services are compared pairwise. Service 1 dominates and wins over service 2, if values of goal services are no worst (bigger, smaller, or any other comparing standard) then service 2, and is better from

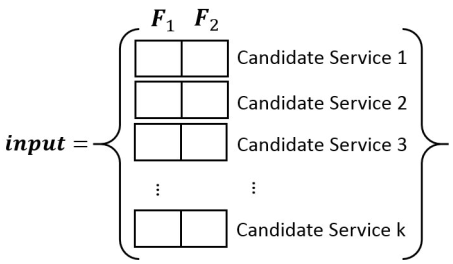

FIG. 5.2. *Input of computing package*

service 2 in one aspect at least. Number of goal functions is not important in domination method; this method is accountable to number of goal functions. This paper has two goal functions $F1$ and $F2$. Therefore, these two methods are compared in both services, and services dominate others will rank as first. The domination method of two is shown in Eq. (5.6) and Eq. (5.7).

$$S_1 \gg S_2 \Rightarrow \forall j \in [1,2] \qquad F_j^{S_1} \geq F_j^{S_2} \tag{5.6}$$

$$\exists j \in [1,2] \qquad F_j^{S_1} > F_j^{S_2} \tag{5.7}$$

In Eq. (5.6) and Eq. (5.7), $\gg$ means S_1 dominates S_2. and respectively show the value of jth function of service 1 and value of jth function of service 2. So, for ranking candidate services using domination of services (computing contents of *computing package*), algorithm 2 is presented.

Algorithm 2 Algorithm 2: Ranking candidate services (by zone agents)

Require: F_1, F_2
Ensure: Ranked services;
R=1;
2: **while** there is not-Ranked candidate services **do**
find all services that are dominated to other services and set there Rank to R
4: **if** there is not any service which dominated to others **then**
set the Rank of all services to R
6: **end if**
R=R+1;
8: **end while**
return the Rank of candidate services

As shown in Algorithm 2, the goal functions $F1$ and $F2$ are used as inputs to compute *computing package*. The algorithm runs until there is no not-ranked services (line 2). Within each run, services which dominate other services, but are similar to each other, are given a rank (line 3). If the algorithm could not find any services that dominated to others, put all remained services in same rank (line 4). With each run, a unit is added to rank given to services for the next session (line 7). As a result, rank of services is returned (line 9).

5.5. Inter-agent negotiations protocol. In the previous section, it is said that ranking is done based on running some operations on *computing package*. And also each zone agent can choose between two options when it receives a package. In the first option, it can take care of operations, start ranking services and return the result to manager agent. Second option is that this agent for any reason (over load or busy ranking candidate services) cannot perform ranking, therefore, convinces one of other zone agents by negotiating to rank services and finally sends the *computing package* to that agent. Negotiation between each initial zone agent and others is shown in Fig. 5.3.

According to Fig. 5.3, the zone agent is not capable of computing a *computing package*, and starts negotiating with other zone agents by sending a message asking for computing the *computing package*. The

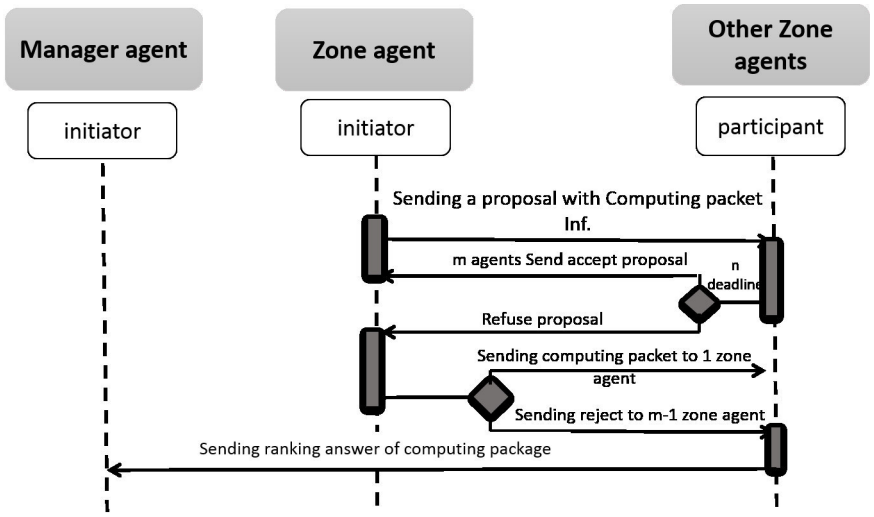

FIG. 5.3. *Inter-agent negotiations protocol*

S_1= < 10 4 1 20 1 2 7 >	Feedback S_1= < 0,5 0.5 1 1 0.2 0,3 1 >
S_2= < 3 6 1 8 6 7 9 >	Feedback S_2= < 1 0.2 0.8 0.7 1 1 0,5 >
S_3= < 1 4 2 0 2 3 17 >	Feedback S_3= < 0.5 0.6 1 1 1 0.4 0,8 >
S_4= < 4 10 1 4 5 16 7 >	Feedback S_4= < 0.1 1 0.2 0.3 0.5 1 0.7 >
S_5= < 2 5 1 17 32 50 30 >	Feedback S_5= < 0.5 0.6 1 0 1 0.4 1 >

FIG. 5.4. *An example with five services in service marketing*

agents which are received the message respond by a bid message, if they can compute that message. Initial zone agents choose one of proposes randomly, then sends it the *computing package* or sends bid rejection to other agents. Any agent, which computes contents of the package, sends the result to manager agent, so that manager agent sends it to user agent.

5.6. A Sample Scenario. This section investigates an example and shows the ranking process. Supouse we have five services and only one active user in our ranking system. Each service has at least seven quality attributes which can be as Fig. 5.4. The previous users feedback represented in Fig. 5.4 too.

As shown in Fig. 5.4, we have five services that shown with their quality attributes. It should be noted that quality attributes for services measured by third section and we suppose these values as input our ARank method. Also Fig. 5.4 shows the feedback of services that intered by previously users. Now suppose we have one active user with inputs as follows:

$R_1 = \prec 3 \quad 4 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \succ$

$C_1 = \prec 0.5 \quad 0.1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \succ$

The R_1 shows user's requirments and C_1 shows user's priorities. In ranking process, at first zone agents try to found the services which can satisfy user's requirements (candidated services). In this example, services S_1, S_2 and S_4 can satisfy user's requirments. then the *computing package* includes only candidated services and their feedback and also the active user's inputs (requirments and priorities).

At the second step, one of the zone agents should calculate the *computing package* using Eq. 5.4 and 5.5 and make the results like Fig. 5.1. The result of computation will be as follow:

$S_1 \Rightarrow F_1 = 6.4$ and F_2=0.7

$S_2 \Rightarrow F_1$=3.1 and F_2=0.15

$S_4 \Rightarrow F_1$=4 and F_2=0.35

In the end, Algorithm 2 should be run to Rank the Candidated services. In this Example, service S_1

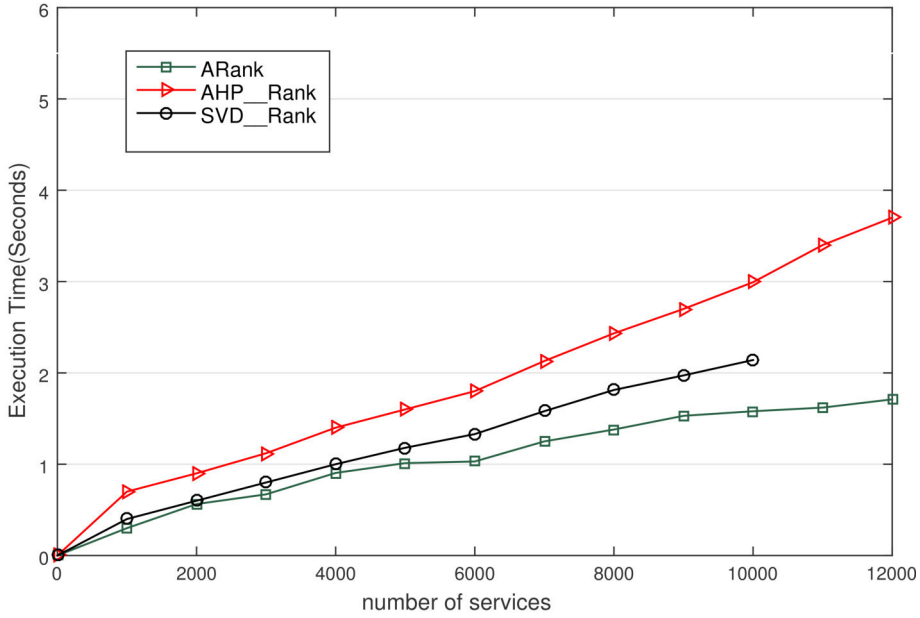

FIG. 6.1. *Response time of system by increasing number of services*

dominated to other two services and receives first rank. Service S_4 received second rank and service S_2 received third rank. This example represented ranking process in ARank.

5.7. Implementation of ARank. The proposed method is implemented on a system with CPU Intel Corei5 Duo 2.53 GHz, 4 GB of RAM, and Windows 7 x86 Enterprise OS using Jade Library (V. 4.3.3) along with earlier methods. In implementing ARank method, we created one manager agent, one user agent, and five zone agents. Values of user required attributes and also values of services attributes are 7 (SMI attributes only). Selection of 7 attributes in implementation is for comparing of proposed method with earlier ones, and also for the existence of 7 features in used dataset.

To analyze the propose method, ARank, version 2 of QWS [23] data set is used. Because QWS is a real dataset, from more than 2,500 web services. This dataset is the only one which includes all 7 SMI attributes. In addition, using these attributes help to rank and compare all proposed method with others.

6. Evaluation Analyses. This section analyzes the required time for getting suggestion from ranking service. Our proposed method is compared with methods based on SVD (Singular Value Decomposition) [25] and based on AHP (Analytic Hierarchical Problem) [28].

The experiment is based on response time of system by increasing number of services. Services are increased from 1 to 12,000 with 1,000 jump. With each increment in number of services, response time from system is measured. Experiment results are shown in Fig. 6.1. The methods based on SVD and based on AHP are respectively shown as SVD_Rank and AHP_Rank.

As shown in Fig. 6.1, by increasing the number of services, SVD_Rank method is capable of responding up to 10,000 services, because it uses Singular Value Decomposition method. As SVD method uses serial and central run, response time increases when the number of services increase. The AHP_Rank also ranks all the existing services; and this method also runs central and serialized. Therefore, it needs more time to respond. In return, the ARank method runs distributed as it uses agents and their cooperation ability has lower response time compared to previous methods.

7. Conclusion and Future Works. In this paper, we proposed a novel method for ranking cloud computing services based on mutiagent systems called ARank. The aim of our system is to decrease user waiting times to get a suggestion from ranking service. The ARank method prevented ranking all of services by choosing candidate services, And it took the advantage of capabilities of agents, by choosing and ranking services in

distributed way which resulted in lower ranking time.

Our proposed method, ARank, prevented ranking all services by choosing some candidate services with zone agents simultaneously. And also this method could use all user requirements using the standard SMI attributes. In addition, ARank is able to use the feedback of users along values for each attribute quality, which inspires a competition between service providers.

For our future works, the requirements of users can be classified into essential and non-essential that will be increased ranking accuracy. In addition, we can create some new services with service composition that are more suitable for users requirements. Furthermore, it is possible to optimize the functions that are finding best matching zone agents to calculate the *computing package*.

REFERENCES

[1] R. Buyya, C. Vecchiola, S. T. Selvi, *Mastering cloud computing, cloud computing architecture*, Foundations and Applications Programming, chapter 4, 2013, pp. 111-140.

[2] S. Ding, S. Yang, Y. Zhang, C. Liang, C. Xia, *Combining QoS prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems*, Knowledge–Based Systems, 56 (2014), pp. 216-225.

[3] S. Garg, S. Versteeg, R. Buyya, *a Framework for ranking of cloud computing services*, Future Generation Computer Systems, 29 (2013), pp. 1012-1023.

[4] A. Quarati, A. Clematis, D. Agostino, *Delivering cloud services with QoS requirements: Business opportunities, architectural solutions and energy-saving aspects*, Future Generation Computer Systems, 55 (2016), pp. 403-427.

[5] B. Rajkumar, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, *Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th Utility*, Future Generation Computer Systems, 25 (2009), pp. 599-616.

[6] Y. Hao, Y. Zhang, J. Cao, *Web services discovery and rank: An information retrieval approach*, Future Generation Computer Systems, 26 (2010), pp. 1053-1062.

[7] L. Chen, Y. Feng, W. Jian, Z. Zheng, *An enhanced QoS prediction approach for service selection*, IEEE International Conference on Services Computing, Washington, DC, 2011, pp. 727-728.

[8] R. Zhang, K. Zettsu, Y. Kidawara, Y. Kiyoki, *Web service ranking based on context*, Second International Conference on Cloud and Green Computing, Xiangtan, 2012, pp. 375-382.

[9] F. Durao, J. Carvalho, A. Fonseka, V. Garcia, *A systematic review on cloud computing*, The Journal of Supercomputing, 68 (2014), pp. 1321-1346.

[10] M. J. Katchabaw, H. L. Lutfiyya, M. A. Bauer, *Usage based service differentiation for end-to-end quality of service management*, Computer Communications, 28 (2005), pp. 2146-2159.

[11] H. Fan, *An integrated personalization framework for SaaS-based cloud services*, Future Generation Computer Systems, 53 (2015), pp. 157-173.

[12] L. Qu, Y. Wang, M. A. Orgun, *Cloud service selection based on the aggregation of user feedback and quantitative performance assessment*, SCC '13 Proceedings of the IEEE International Conference on Services Computing, IEEE Computer Society Washington, DC, USA, 2013, pp. 152-159 .

[13] C. Mao, *Search-based QoS ranking prediction for web services in cloud environments*, Future Generation Computer Systems, 50 (2015), pp. 111-126.

[14] M. Wooldridge, *An introduction to multi-agent systems*, Department of Computer Science, University of Liverpool, 2009, pp. 200-450.

[15] L. Qu, Y. Wang, M. A. Orgun, *Cloud service selection based on the aggregation of user feedback and quantitative performance assessment*, SCC '13 Proceedings of the IEEE International Conference on Services Computing, IEEE Computer Society Washington, DC, USA, 2013, pp. 152-159.

[16] S. Yau, Y. Yin, *QoS-based service ranking and selection for service-based systems*, IEEE International Conference on Services Computing, 2011, pp. 56-63.

[17] M. Almulla, H. Yahyaoui, K. Al-Matori, *A new fuzzy hybrid technique for ranking real world Web services*, Knowledge-Based Systems, 77 (2015), pp. 1-15.

[18] D. Skoutas, D. Sacharidis, A. Simitsis, T. Sellis, *Ranking and clustering web services using multicriteria dominance relationships*, IEEE Transaction on Service Computing, 3 (2010), pp. 163-177.

[19] M. D. Dikaiakos, D. Zeinalipour Yazti, *A distributed middleware infrastructure for personalized services*, Computer Communications, 27 (2004), pp. 1464-1480.

[20] S. S. Yau, Y. Yin, *QoS–based service ranking and selection for service based systems*, IEEE International Conference on Services Computing, Washington, DC, 2011, pp. 56-63.

[21] J. Octavio, A. Ramirez, *Collaborative agents for distributed load management in cloud data centers using Live Migration of virtual machines*, IEEE Transactions on Service Computing, 8 (2015), pp. 916-929.

[22] J. Octavio, A. Ramirez, *Agent-based load balancing in cloud data centers*, Cluster Computing, 18 (2015), pp. 1041-1062.

[23] E. Al-Masri, Q. H. Mahmoud, *Investigating web services on the world wide web*, In Proceedings of the 17th international conference on World Wide Web, ACM, 2008, pp. 795-804.

[24] E. Badidi, *A Framework for software as a service selection and provisioning*, International Journal of Computer Networks & Communications, 5 (2013), pp. 189-200.

[25] H. Chan H, T. Chieu, *Ranking and mapping of applications to cloud computing services by SVD*, Network Operations and Management Symposium Workshops (NOMS Wksps), Osaka, 19-20 April, 2010, pp. 362-369.
[26] P. Choudhury, M. Sharma, K. Vikas, T. Pranshu, V. Satyanarayana, *Service ranking systems for cloud vendors*, Advanced Materials Research, 433 (2012), pp. 3949-3953.
[27] Z. Zheng, X. Wu, Y. Zhang, M. Lyu, J. Wang, *QoS ranking prediction for cloud services*, Journal of IEEE Transactions on Parallel and Distributed Systems, 24 (2012), pp. 1213-1222.
[28] A. Ishizaka, A. Labib, *Analytic hierarchy process and expert choice: benefits and limitations*, OR Insight, 22 (2009), pp. 201-220.
[29] A. Jahani, L. Mohammad Khanli, *Cloud service ranking as a multi objective optimization problem*, Journal of Supercomputing, 72 (2016), pp. 1897-1926.
[30] CSMIC, *CSMIC SMI overview diagram two point one*, Carnegie Mellon University Silicon Valley, Moffett Field, CA USA, 2011, pp. 1-10.
[31] J. H. Chen, *A hybrid model for cloud providers and consumers to agree on QoS of cloud services*, Future Generation Computer Systems, 50 (2015), pp. 38-48.
[32] L. Sun, *Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection*, Future Generation Computer Systems, 57 (2016), pp. 42-55.

Edited by: Pedro Valero Lara
Received: Dec 23, 2016
Accepted: May 25, 2017