International Journal of Engineering

Journal Homepage: www.ije.ir

# Services Composition in Multi-cloud Environments using the Skyline Service Algorithm

M. Heidari, S. Emadi*

*Department of Computer Engineering, Yazd Branch, Islamic Azad University, Yazd, Iran*

*PAPER INFO*

*Paper history:*
Received 08 June 2019
Received in revised form 22 September 2020
Accepted 20 November 2020

*Keywords*:
*Skyline Service*
*Dominant Relationship*
*Web Service*
*Service Composition*
*Multi-cloud Environments*

*ABSTRACT*

The rapid growth of cloud environments has led to the expansion of resources that offer a variety of services. The opertions of the services are usually very simple and may not satisfy the complex needs of the user, hence there is a need for a combination of these services that can fulfill the user's requirements. Most of the service composition methods in cloud environments assume that the involved services came from one cloud, and this is unrealistic because other clouds may provide more relevant services. The challenges in composition services distributed in multi-cloud environments include increased cost and a reduction in its speed due to the increasing number of services, providers, and clouds; so, in order to overcome these challenges, the number of providers and participating clouds must be reduced. This study used the Skyline service algorithm to compose services in multi-cloud environments, which examined all the clouds during the service composition process. The proposed method can provide an applicable composition service to the user with the lowest communication cost by considering the number of clouds and by using fewer providers. The Skyline algorithm involves two steps. In the first one, the best composition in a cloud environment is selected among all the possible providers by considering the number of providers and the communication time. In the second step, the Skyline algorithm is used to create all the possible compositions in a multi-cloud environment. Parameters such as fewer clouds and shorter communication times between the clouds are selected. The results show that the proposed method can find the composition with the least number of clouds, the lowest cost, and has the lowest calculation time. It can be said that the Skyline makes it possible to select a suitable composition of user-requested services in a multi-cloud environment.

***doi**: 10.5829/ije.2021.34.01a.07*

## 1. INTRODUCTION

Web service is a modular and self-described application that is published based on a set of standards such as SOAP, WSDL, and UDDI [1-2]. When a web service is limited to simple features, a set of separated web services must be combined to create a value-added one [3-4]. Service composition problems can be resolved by selecting a set of web services in such a way that their combination meets the functional and non-functional requirements of the user [5]. With the advent and rapid development of cloud computing, more clouds can carry out the existing tasks in the cloud with different functions, and this cloud environment is a natural choice for providing various types of resources as a service. To meet the user's needs, cloud-based systems [6-7] are usually designed by calling up several providers. The service composition in cloud environments allows for the integration of various cloud resources into a set of integrated services for providing cloud-based solutions that meet certain qualitative criteria [8]. Most of the service composition methods that have been proposed for cloud computing consider all the composite services in one cloud, rather than searching services from the various available clouds [9]. Organizations often distribute their services using cloud providers to ensure the availability and quality of the provided services, and also to reduce the risk of data loss [10]. In addition, service composition

*Corresponding Author Institutional Email: emadi@iauyazd.ac.ir (S. Emadi)

Please cite this article as: M. Heidari, S. Emadi, Services Composition in Multi-cloud Environments using the Skyline Service Algorithm , International Journal of Engineering, Transactions A: Basics Vol. 34, No. 01, (2021) 56-65

in multi-cloud environments poses many issues such as the cost of communications within the cloud, increased fiscal costs, and security issues. Hence, challenging tasks include reducing the number of participating clouds and the number of providers due to the limitations of the services. Therefore, the current study seeks to find the best possible service composition in cloud environments using the Skyline service algorithm, which uses both a smaller number of providers and clouds to reduce financial costs.

The Skyline algorithm is based on the concept of Pareto dominance [11]. It has been used to solve research problems such as web service selection, query processing over uncertain data [12-14], effective processing of advanced queries [15], and indexing of time series data. The use of the Skyline algorithm in the proposed method creates all the possible compositions of the providers in a multi-cloud environment. The best composition in a cloud environment is selected by considering the number of providers and the communication time. Parameters such as fewer clouds and a shorter computation time between the clouds are also considered in selecting the most suitable cloud composition.

The innovation of this paper includes modeling the multi-cloud environment using the Skyline in two steps. First, the providers and services were modeled based on user requests. Secondly, the clouds are modeled based on the providers and services selected in the previous step. Then, we introduce the algorithms for the extraction of the candidate services, providers, and clouds based on the Skyline rules.

The rest of this paper is organized as follows. In Section 2, the works related to service composition will be discussed using the Skyline service. In Section 3, the algorithm and the concepts of the Skyline service are expressed. Then the proposed method is outlined is Section 4. Section 5 presents the results and evaluation, and the last section is devoted to conclusion and suggestions.

## 2. RELATED WORKS

Most of the existing approaches to service composition in cloud environments consider all the services in the composition from a single cloud. However, certain algorithms have also been proposed to address this issue. In Section 2.1, other methods will be examined, and in Section 2.2, service composition using the Skyline algorithm will be discussed.

**2. 1. Methods Provided Using Multi-cloud Algorithms** Zou et al. used a tree structure to model a multi-cloud environment (MCB). Then, with the MCB tree search, the minimum request set was created. Accordingly, they proposed three algorithms for selecting the optimal cloud composition. In the first algorithm, they considered all clouds as inputs and evaluated all the possible solutions. This method determined the sequence of the service composition at the time of execution, but with the use of a large number of clouds. The second algorithm recursively defined a service composition in all the cloud compositions. The last algorithm provided an optimal cloud computing approach using an approximate method. However, it was time-consuming and may not be a good cloud computing approach because it used the composition of clouds that utilize service spaces and could impose on some compositions [16]. Gutierrez-Garcia et al. proposed an agent-based multi-cloud service composition approach by using a semi-recursive conventional protocol; however, it has the limitations of agent-based distribution [17]. Jatoth et al. proposed a quality of service (QoS) cloud service composition based on both the modified invasive weed optimization algorithm and an Adaptive Genotype Evolution based Genetic Algorithm (AGEGA) [18-19]. Gavala et al. proposed a QoS aware cloud service composition based on an Eagle Strategy with Whale Optimization Algorithm (ESWOA). However, in these three approaches, they considered multiple QoS parameters for service composition in only one cloud [20]. Yu et al. presented a Greedy-WSC algorithm and an ant colony optimization based algorithm, namely ACO-WSC, to select the service composition in cloud environments with a minimal number of clouds. The Greedy-WSC algorithm selects clouds that offer more services, and the ACO-WSC algorithm is used to combine selected clouds. Their results showed that the ant colony optimization method could efficiently find effective cloud composition with the minimum number of clouds. The disadvantage of this model was its lack of considering semantic information in the composition of web services, especially in a dynamic and distributed environment [21]. Kurdy et al. suggested a composite optimization (COM2) algorithm for cloud services that ensures the selection of clouds with the maximum number of services, which increases the likelihood of completing a service request at a minimum cost. The results of their experiments showed that COM2 was successfully able to compete with previous algorithms in the field of service composition, but it did not consider the interconnecting costs of the clouds [22]. Mezni et al. used formal concept analysis (FCA) and fuzzy formal concept analysis (FFCA) for service composition in a cloud-based environment. The FCA is based on the concept of a network, a powerful tool for classifying cloud information and services. Initially, a cloud computing model was created as a set of formal concepts; then, it extracted and combined the candidate clouds from the formal concepts. Finally, the optimal cloud composition was selected, and the multi-cloud service composition (MCSC) became a classical service

composition problem. In addition to considering the number of clouds in the composition, it also takes into account the cost between the clouds. The tests showed the effectiveness and ability of the FCA-based method to find and group cloud compositions with a minimum number of clouds, the lowest communication cost, and the lowest time to service selection in the nearest cloud or in the same cloud [23-24].

**2. 2. Methods Provided Using the Skyline Algorithm** Yu and Bouguettaya suggested an algorithm that used the dominant relationship between service providers to find a set of the best possible service composition for Skyline services [11]. Instead of examining all the possible composition of services, this algorithm significantly reduces the search space and proposes a low-up computing framework that enables the Skyline algorithm to scale well with a number of services. In their research, three algorithms, namely OPA, DPA and BUA, were developed to select a set of the best possible composition services. The DPA used a parent table and a broad network to achieve enhancement and route ability. The BUA used a powerful low-up computing framework with a linear composite strategy, which improved the performance and the scalability.

Wu et al. provided an algorithm for the composition of services based on service quality. In this way, when a new service comes, the previous service is deleted, and the quality of service is changed. This algorithm reduces the number of selected services through Skyline and chooses the best service using the service quality [25].

In another study, Zhang et al. used the Skyline guaranteed query processing method to build mashup cloud applications and employed similarity tests to achieve an optimal Skyline. Cloud mashup is a composition of several services with a shared data set and integrated functions. This method was used to optimize the composition of web services in large-scale cloud-based mashup applications from the Map-Reduce. Since the choice of Skyline service and hybrid processes were very timely, especially when the data space of the services was very large, a block-based blocking was proposed to shorten the process. After testing 100,000 real websites worldwide in 10 dimensions, it was found that the Map-Reduce based block-removal method was 3.25 times faster than the angular segmentation algorithm, and 1.4 times faster than the network method [26-27].

Liu et al. proposed a dynamic Skyline service selection tool to reduce redundancy. In this method, the process of choosing a service was divided into two stages: the service selection stage and the implementation phase of the selected services. The selection stage used the offline method to calculate the Skyline, and was responsible for updating the Skyline service. Therefore, the offline process never affected the performance of the phases of the service selection. The implementation phase was responsible for selecting the optimal composition of the services, which matched the QoS user limitations. The results showed that this method selected the most appropriate services [28].

Moradi and Emadi presented an algorithm for service composition using the Skyline service in parallel. In this way, the choice of services was based on the quality of service; the use of parallelization techniques had a significant impact on reducing the response time and increasing the speed of the composition of services, as well as reducing the computations [29].

However, most traditional service composition methods regard service composition in a single cloud and consider a balance between the QoS parameters. In this paper, we present an algorithm based on Skyline service, which focuses on reducing the number of clouds and providers.

## 3. THE SKYLINE SERVICE ALGORITHM

The existing approaches in multi-cloud service composition only reduce the number of clouds. This research, like [23], considers modeling the relationship between the providers and the clouds in the selection of optimal clouds, as well as the composition of services by the Skyline service algorithm. The Skyline service algorithm has been used to extract the optimal composition of the providers and clouds. Also, combined services can have sequential, parallel, loop, or conditional structures. In this research, only the sequential structure for combining services and their implementation is considered.

**Definition 1**: A multi-cloud environment is a set in which $C= \{C_1,C_2,...,C_N\}$ where $C_i$ is a cloud and $P=\{P_1,P_2,\ldots,P_N\}$ where $P_i$ is a provider that is hosted by the clouds. A provider also offers a set of services. Every provider may belong to more than one cloud, and every service also may belong to more than one provider.

The multi-cloud service composition problem is given a set of clouds that hosts the services offered by a number of providers. The Skyline service algorithm is designed to select the minimal sub-set of clouds and providers, while reducing the cost of communication between the providers and clouds.

Skyline was originally introduced in the database domain [30]. Given a set of S points in a D-dimensional space, the points in the Skyline are not dominated by any other place in the search space [31].

**Definition 2** (Dominance Service and Skyline service): In service composition, dominance services are better in all parameters of service quality compared to other services. For example, $SA= \{S_1, S_2, S_5\}$ is a set of services that provides task A with QoS= {3, 4, 2}in time and $SB= \{S_2, S_4\}$ that provides task B with QoS= {4, 5}

in time. The Skyline service for SA= $S_5$ and for SB=$S_2$ are not dominated by other services, and it is the best candidate service [28, 32- 33].

The Skyline was introduced for the first time to create a web service and to evaluate its effectiveness [30]. In the service composition, the dominant service is the services that are better than others in all aspects of service quality. To this end, some researchers have proposed different methods for determining the dominant relationship to determine the Skyline service [34-35].

Therefore, if a service is part of Skyline, it is expected to offer better parameters than other services [36]. In the above example for SA and SB, the composition of Skyline services is {S5, S2}, in which a set of services are dominated by none of the services in the other composition [11] as {$S_1$, $S_2$}, {$S_1$, $S_4$}, {$S_2$, $S_2$}, {$S_2$, $S_4$}, {$S_5$, $S_2$}, and {$S_5$, $S_4$}.

One of the algorithms offered by the Skyline service, which is used in this investigation, is a dual progressive algorithm [11] for making composition possible. The root, that is, the parent node, is constructed first, and then the next nodes are constructed**.** The rule to create each node is that the selected services available in composition are different only in one service with its child nodes. For example, the root node in Figure 1 is $a_1$, $b_1$, $c_1$, and its child nodes include ($a_1$, $b_2$, $c_1$), ($a_1$, $b_1$, $c_2$), and ($a_2$, $b_1$, $c_1$). The lattice expansion determines only the sequence of counts between the nodes, and proves that each node is considered after its ancestors, but for nodes that do not have parent-child relationships, an appropriate order must be guaranteed. Since it may have a score of ($a_1$, $b_2$, $c_1$) less than ($a_1$, $b_1$, $c_3$), it should be counted in advance. In order to achieve the progressive counting of the base, the lattice expansion (T) with a heap (H) is used. The lattice expansion ensures that the parent node is counted before the child node. On the other hand, the heap determines the counting of the nodes that do not have a parent-child relationship. The commencement of the manufacturing process starts from the first level. At each step of the count, the lattice expansion is extracted from the heap with the lowest cost and is compared with the existing Skyline. Ultimately, the considered composition is placed in Skyline if it is not lost or eliminated. The progressive algorithm of a node can be generated several times from generating other parent nodes, which creates a replication problem. As shown in Figure 1, the top number of each node shows its parent number. For example, the node ($a_3$, $b_2$, $c_2$) is placed three times in the heap because it has three parents, and each time they develop ($a_3$, $b_2$, $c_2$), they are generated and placed in H. The multiplication of the node has many computational problems since many nodes are processed several times. The same node can be located in Skyline more than once, which causes a false Skyline [9].

The parent table [11] provides a suitable solution for solving a node problem with the least computation. Instead of considering all the ancestors, the parent table only stores information about the number of parents for a given node. The basic rule is that a node can be put in a heap only when all its parents are already processed. The parent table stores the number of parents in each node. Each time the node is compared to another node, the number of parents is reduced by one unit, and the table is updated with new values; eventually, every node in its value reaches zero in the heap. This operation ensures that all the nodes of the child are placed in the heap before the parent nodes [11].

In the next step, the best service in the lattice should be selected taking into account the dominant relationship. Then, the Button-Up Algorithm [11] strategy is to use linear compositions while doing comparisons to select the best composition. A linear composition is to compare the results of the two nodes with the next node, and achieving the best possible composition [11, 29]. Button-Up Algorithm carries out optimization and QoS calculations with positive traits inherited from dual progressive algorithm.

## 4. DETECTING A MULTI-CLOUD ENVIRONMENT USING THE SKYLINE SERVICE ALGORITHM

In this research, the Magnetic Cluster Expansion (MCE) is modeled as a set of lattice expansion, as shown in Figure 2. Each cloud is described as a lattice expansion created to group the providers based on the services they provide, and another lattice expansion has been created to express the relationships between the desirable providers and their hosting clouds.

Since a provider may belong to more than one cloud, so with respect to the given N clouds, the information about the services and their providers is modeled in the N lattice expansion, where each one represents the environment of a cloud. First, a number of the preferred composition of the providers are selected as equal to the

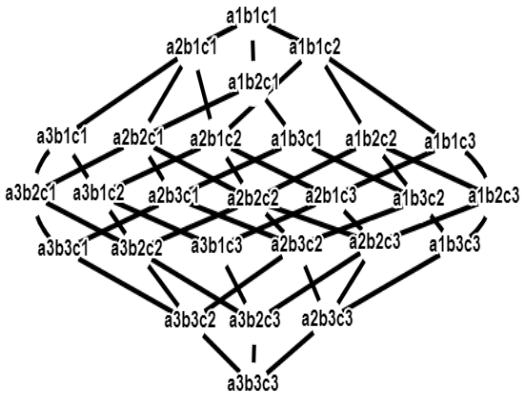


**Figure 1.** Lattice Expansion

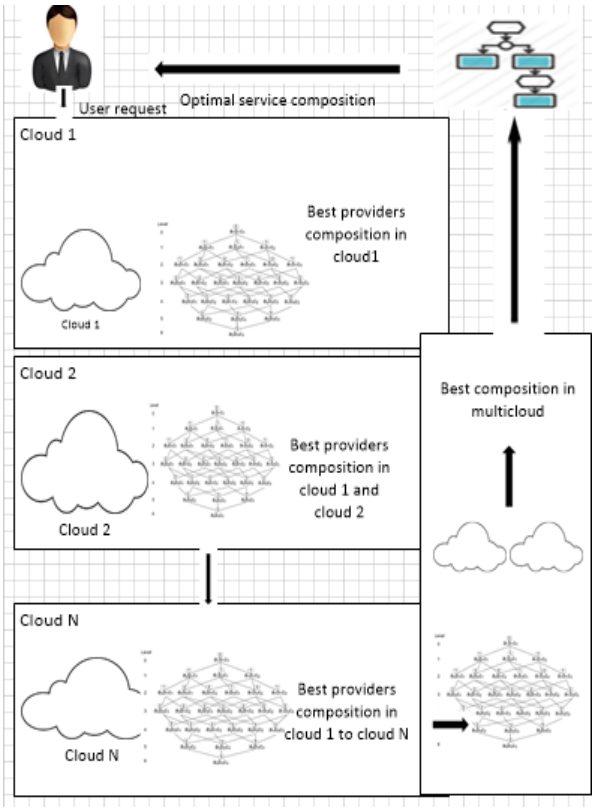


**Figure 2.** The proposed method

number of the available clouds. After comparing and choosing the most suitable composition, a multi-cloud spreading lattice expansion is built, and the optimal composition of the clouds is selected from this lattice expansion.

An example of a multi-cloud environment is shown in Table 1. Thirty services with various QoS functions and capabilities are provided by five providers on three clouds. For example, Cloud $C_1$ hosts three providers, which altogether provide 13 services. Some providers may deploy their services in multiple clouds (e.g., $P_2$, $P_5$).

Based on the example above that shows a cloud environment with three clouds, a lattice is expanded for each cloud, and for a multi-cloud environment, a distributed lattice is modeled. Table 2 describes the relationships between the providers and their host clouds, and Table 3 describes the relationships between the providers and their services in Cloud 1.

This research seeks to find a composition of clouds and providers that hosts the best service and to reduce the

**TABLE 1.** An example of multi-cloud environment

| **Clouds** | **$C_1$** | | | **$C_2$** | | **$C_3$** | | |
|---|---|---|---|---|---|---|---|---|
| Providers | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_1$ | $P_5$ | $P_2$ |
| Services | 5 | 4 | 4 | 2 | 3 | 5 | 4 | 3 |

**TABLE 2.** An example of relationships between the clouds and providers in a multi-cloud environment

| **MCE** | **$C_1$** | **$C_2$** | **$C_3$** | **$C_4$** | **$C_5$** |
|---|---|---|---|---|---|
| **$P_1$** | 0 | 0 | 1 | 0 | 1 |
| **$P_2$** | 0 | 1 | 1 | 0 | 0 |
| **$P_3$** | 0 | 0 | 1 | 1 | 0 |
| **$P_4$** | 1 | 1 | 0 | 0 | 0 |

**TABLE 3.** An example of relationships between the providers and their services in Cloud 1

| **$Cloud_1$** | **$P_1$** | **$P_2$** | **$P_3$** | **$P_4$** |
|---|---|---|---|---|
| $S_1$ | 0 | 5 | 7 | 9 |
| $S_2$ | 5 | 0 | 4 | 6 |
| $S_3$ | 7 | 4 | 0 | 3 |
| $S_4$ | 9 | 6 | 3 | 0 |

cost of communication between the services that come from different clouds. For this purpose, two algorithms are proposed to select a multi-cloud composition that uses the minimum number of providers and clouds. The steps are briefly summarized below:

Step 1- Extracting the optimal composition of providers: In this step, the best composition of providers is extracted in each of the clouds. By comparing the compositions obtained from all the clouds, the optimal composition that meets the user's request is selected and then used as input to determine the optimal cloud composition.

Step 2 - Extracting the optimal composition of the cloud: At this point, the lattice expansion, which shows the relationship between the providers and their host clouds, is used to obtain the optimal composition of clouds according to the providers selected in Step 1. The random composition of the clouds, which hosts the optimal composition of the providers, is selected as the root of the lattice expansion; the lattice expansion is thus complete and is selected based on the dominant relationship of the optimal composition of the clouds.

The following sections give more details about each of the above steps.

**4. 1. Extracting the Optimal Composition of Providers** This step uses the Skyline service algorithm to extract the optimal composition of providers in each cloud. In selecting the optimal composition, none of the existing approaches take into account the number of providers and the cost of communication between the providers. To determine the cost of communication between two providers, each cloud environment uses the information shown in Table 4. In this study, the matrix values are simple, representing the time of communication between two providers (in milliseconds).

**TABLE 4.** Matrix of communication cost between providers in cloud1

| | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|---|
| $P_1$ | 0 | 5 | 7 | 9 |
| $P_2$ | 5 | 0 | 4 | 6 |
| $P_3$ | 7 | 4 | 0 | 3 |
| $P_4$ | 9 | 6 | 3 | 0 |

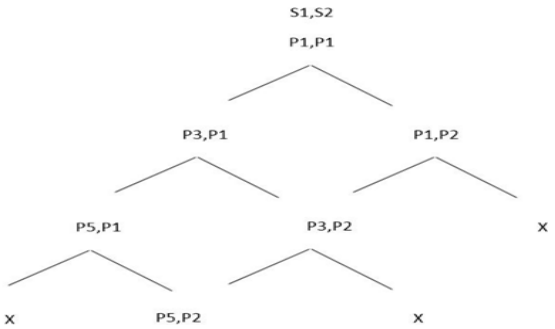


**Figure 3.** An example of lattice expansion of providers

This algorithm considers the user requested services to determine the appropriate composition; Sr is considered as an input to create a lattice expansion in each cloud. For creating a lattice expansion in each cloud (Algorithm 3), the root node is created based on a possible composition of providers (line 4 in Algorithm 1), which satisfies the user's requested services. For example, if a user requests $S_1$, $S_2$ services (Figure 3), the above algorithm will be considered as the root in Cloud 1 of the $P_1$, $P_1$ composition that delivers the services that are being provided; then, the child nodes are constructed.

Rule 1: The child node is a node that differs in the composition of providers with the parent node only in one provider.

So, the child nodes of the above example will be ($P_3$, $P_1$) and ($P_1$, $P_2$); after determining each node, the cost of each node is calculated according to Equation (1).

$$Si \coloneqq \alpha * Ni + \beta \sum_{j=1}^{|E|} cost\ j \quad (1)$$

where E is the set of edges that show the communication between the providers in a composition, cost j denotes the cost of communication between the providers Px and Py in the j relationship link, and Ni is the number of existing providers in the i-th composition. Also, α and β are numeric values representing the number of providers and the communication costs of the providers, respectively. To avoid the presence of providers in dispersed areas and encourage the lowest cost of communication between providers as the most important goal, the amount of α should be smaller than β. Having created the lattice expansion starting from the root node, the root node first appears in the heap and is selected as the Skyline. After removing the root node, its children are added to the heap if all their fathers are examined, and so the cost of each composition is compared with the cost of the composition in the Skyline; then, if the composition is found to be optimal, the Skyline is updated. Hence, the best composition is selected by comparing the cost of the composition. Thereafter, the second cloud's lattice expansion will be created and the optimal composition will be compared with that of the first cloud, and the best composition will be selected. The output of this algorithm is the optimal composition of providers.

### 4. 2. Extracting the Optimal Composition of Clouds

The composition obtained from the algorithm in the previous section is the input of this algorithm. The goal of this stage in a cloud-based environment is to classify the clouds that together provide the equired services. By evaluating all possible compositions, the optimal cloud composition is determined, from which the appropriate services are delivered to the user. Here, to determine the cost of the relationship between the two clouds, the matrix values in Table 5 are simple values that represent the time between the clouds (in milliseconds).

To determine the optimal composition of clouds, the optimal composition of providers from the previous step is considered as input to determine the root of the lattice expansion, and thus the lattice expansion is completed (Algorithm 3). When constructing cloud compositions, the cost of each compound is calculated in accordance with Equation (1). The only difference is that E is the number of edges representing the connections between the clouds in the composition, and j shows the cost of communication between the two clouds, $C_x$ and $C_y$, on the $j_{th}$ communication link. Ni is the number of clouds in the $i_{th}$ composition. The total cost of the composition is calculated by taking into account the total communication costs in the cloud composition according to Equation (1). In this algorithm, α and β are also numerical values representing important factors such as the number of clouds and the cost of cloud communications, respectively. Thus, α should be smaller than β to avoid the presence of clouds in dispersed areas and to encourage the lowest cost of communication between the clouds, which is considered as the most important goal. For example, if the optimal composition obtained from the previous step of composition ($P_3$, $P_1$) is used, the algorithm takes into account in the multi-cloud environment of ($C_3$, $C_3$) compound that hosts the providers in the optimal composition; then, the child nodes are constructed, which are shown in Figure 4.

Rule 2: In creating each child node, the composition is different from that of the provider only with the parent node. So the child nodes of the above example will be ($C_4$, $C_3$) and ($C_3$, $C_5$).

## 5. EXPERIMENTAL RESULTS

This section provides details of the experiments conducted to evaluate the performance of the proposed method. The Java programming language has been used in this approach, and the development environment is NetBeans IDE 8.2.

In this study, Java classes have been used to randomly generate some experimental data sets, including a set of services and relationships between the clouds, providers, and services provided by each provider, as well as α, 0.3, and β, 0.7.

The experiments are conducted in environments with a number of different clouds (between 5 and 100) and services ranging from 1 to 20; since the creation of a multi-cloud environment is a coincidence, the test of each environment is repeated 50 times. The user's request in all the test cases consists of three services.

**5. 1. Estimating the Computation Time** In these experiments, as in similar methods, a concept called density has been considered to determine the impact on the total execution time when the providers are hosted in several clouds; the total execution time is between 20 and 40%, and the number of clouds is between 5 and 100. The composition time results are shown in Figure 5.

**TABLE 5.** Matrix of communication cost between clouds

| | $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|---|
| $C_1$ | 0 | 6 | 8 | 10 |
| $C_2$ | 6 | 0 | 9 | 12 |
| $C_3$ | 8 | 9 | 0 | 4 |
| $C_4$ | 10 | 12 | 4 | 0 |

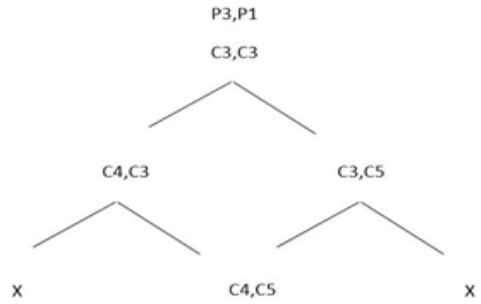


**Figure 4.** Example lattice expansion of clouds

**Algorithm 1:** Extracting optimal composition of providers

**Input:** A user request Sr
**Output:** Best provider composition
1: **Begin**
2: Best provider composition=0;
3: **for each** cloud Ci do
4: Creating expansion lattice based on Sr (Algorithm 3)
5: Best=RootNode; H=RootNode;
6: **While**(! H.isEmpty())
7: Remove the top node from H;
8: **if** n is dominated by Best
9: Best=n;
10: **end if**
11: CN=expand(n,T);
12: **for** all node ni in CN
13: P(ni) --;
14: **if**(P(ni)==0)
15: H.add(ni);
16: **end if**
17: **end for**
18: **end while**
19: **if** Best is dominated by Best provider composition
20: Best provider composition=Best;
21**: end for**
22: **return** Best provider composition;
23: **End**

**Algorithm 2:** Extracting optimal composition of clouds

**Input:** Best provider composition
**Output:** Best cloud composition
1: **Begin**
2: Creating expansion lattice based on
Best provider composition(Algorithm 3)
3: Best=RootNode; H=RootNode;
4: **While**(! H.isEmpty())
5: Remove the top node from H;
6: **if** n is dominated by Best
7: Best=n;
8: **end if**
9: CN=expand(n,T);
10: **for** all node ni in CN
11: P(ni) --;
12: **if**(P(ni)==0)
13: H.add(ni);
14: **end if**
15: **end for**
16: **end while**
17: **return** Best cloud composition;
18: **End**

**Algorithm 3:** Creating Expansion Lattice for providers (or clouds)

**Input:** A provider(cloud) composition that provide user request (or Best provider composition)
**Output:** Expansion Lattice
1: **Begin**
2: **for each** a provider(cloud) composition
3: int num=number of user request (or Best provider composition)
4: **While** (num!=0)
5: change node that number is num based provider(cloud) that is provide same service(provider)
6: num**--;**
7: **end while**
8: **end each**
9: **return** Expansion Lattice;
10: **End**

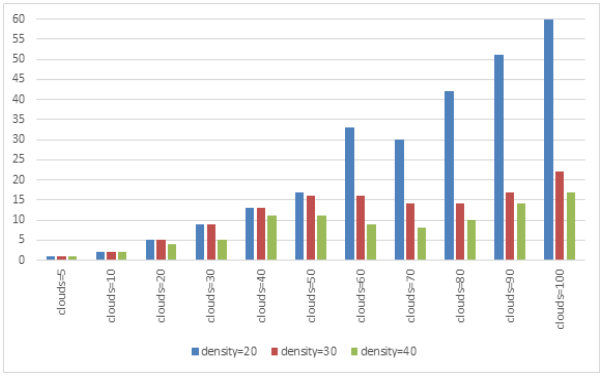


**Figure 5.** Results of the computation time

According to this figure, the computation time at a density of 40 is lower than the other two densities, and especially with a higher number of clouds, this difference is more evident. In general, this algorithm has a low computational time for the cloud environment with a different number of clouds. Also, the execution time is slightly high when a provider is not hosted on several clouds.

**5. 2. Estimating the Cost and Number of Clouds in the Selected Composition** Figure 6 shows that the size of the optimal composition and composition costs are not affected by the changes in density and the number of clouds. The experimental results show that the Skyline-based approach always produces a favorable cloud composition even in a large-scale cloud-based environment, and even when each provider is hosting a small number of clouds.

**5. 3. Comparison of Cloud Communication Costs** In this section, the performance and quality of the proposed solution are compared with the Mezni method [23]. These two methods are compared in a multi-cloudenvironment with 100 clouds and three user-requested services. MCE1 is a cloud environment with a density of 20, MCE2 has a density of 30, and MCE3 is 40.

The overall cost for each cloud compilation generated by the FCA and the Skyline was calculated using defined equation. The results for the FCA are shown in Figure 7, but the value of the Skyline is fixed to be 0.3. It is clear from Figure 7 that for all the MCE settings, the best cost was obtained by Skyline. It also shows that the proposed method always achieves the best cloud composition with the lowest cost.

**5. 4. Comparisons of Run-time** Given the time required to find the optimal cloud composition, the run times in Figure 8 show that Skyline is better than FCA for the three MCE experiments. That is, by changing the density, the proposed algorithm is faster in terms of computational time. This is explained by the dual progressive algorithm, Heap memory and parent table in Skyline algorithm. Also, using a bottom-up algorithm and the linear composition strategy, we can find the optimal combination in the lattice, without needing to run through the whole multi-cloud lattice.

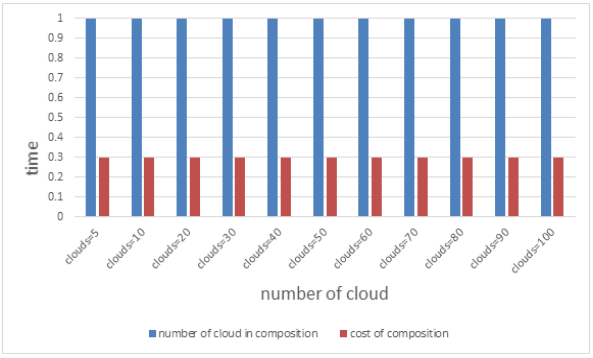


**Figure 6.** Estimating the cost and number of clouds in composition

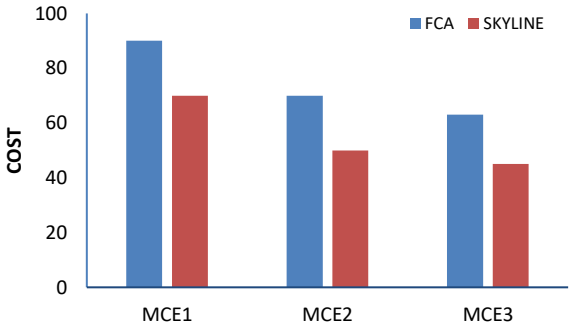


**Figure 7.** Estimating the cost and number of clouds for FCA and Skyline

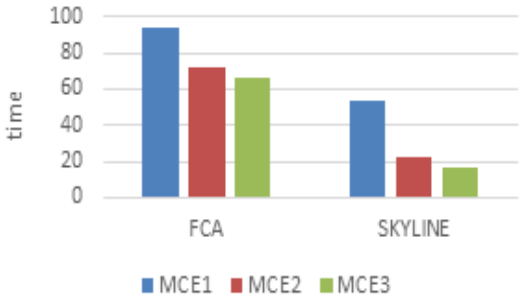


**Figure 8.** Run time in FCA and Skyline

## 6. CONCLUSIONS

With the advent of virtual resource sharing, cloud platforms have created a new paradigm that provides more efficient and convenient services. As stated previously, most of the service composition methods in cloud environments assume that the involved services come from one cloud. This study investigated the use of the Skyline service algorithm to compose services in multi-cloud environments, which examines all the clouds during the service compilation process. Since this algorithm provides the creation of all the possible combinations, the proposed method allows the selection of the optimal composition of user-requested services in a cloud-based environment. In the proposed method, the criteria for choosing the best composition in a cloud environment are fewer providers and a shorter communication time between the providers. Hence, the best composition in a cloud environment is the one that includes these criteria. Overall, the following results have been obtained:

1. The use of the Skyline algorithm makes it possible to review all the possible composition of services offered by providers in a cloud-based environment.

2. The proposed Skyline algorithm always finds the optimal cloud compositions.

3. The proposed algorithm improves the accuracy of the optimal composition and reduces the time of computation.

Also, this study focuses on the sequential structure of a service composition. This is why the total cost of communication between the clouds is calculated based on the order of the services executed as the sum of the communication costs of the provider's composition and the cloud. The sequential structure is one of the four main structures of a service composition in the YAWL model [4], and it is a topic of interest for future studies on other structures.

## 7. REFERENCES


1. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S., "Unraveling the Web Services Web: an Introduction to SOAP, WSDL, and UDDI", ***IEEE Internet Computing***, Vol. 6, No. 2, (2002), 86-93. DOI: 10.1109/4236.991449
2. Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., & Savio, D., "Interacting with the SOA-based Internet of Things: Discovery, Query, Selection, and on-demand Provisioning of Web Services", ***IEEE Transactions on Services Computing***, Vol. 3, No. 3, (2010), 223-235. DOI: 10.1109/TSC.2010.3
3. Du, Y., Hu, H., Song, W., Ding, J., & Lü, J., "Efficient Computing Composite Service Skyline with QoS Correlations", In 2015 IEEE International Conference on Services Computing, (2015), 41-48. DOI: 10.1109/SCC.2015.16
4. Gabrel, V., Manouvrier, M., & Murat, C., "Web Services Composition: Complexity and Models", ***Discrete Applied Mathematics***, Vol. 196, (2015), 100-114. DOI: 10.1016/j.dam.2014.10.020
5. Cui, L., Kumara, S., & Lee, D., "Scenario Analysis of Web Service Composition based on Multi- Criteria Mathematical Goal Programming", ***Service Science***, Vol. 3, No. 4, (2011), 280-303. DOI: 10.1287/serv.3.4.280
6. Bypour, H., Farhadi, M., & Mortazavi, R., "An Efficient Secret Sharing-based Storage System for Cloud-based Internet of Things", ***International Journal of Engineering,*** Vol. 32, No. 8, (2019), 1117-1125. DOI: 10.5829/ije.2019.32.08b.07
7. Jeyanthi, N., Shabeeb, H., Durai, M. S., & Thandeeswaran, R., "Reputation based Service for Cloud User Environment", ***International Journal of Engineering, Transactions B: Applications***, Vol. 27, No. 8, (2014), 1179-1184. DOI: 10.5829/idosi.ije.2014.27.08b.03
8. Jula, A., Sundararajan, E., & Othman, Z., "Cloud Computing Service Composition: A Systematic Literature Review", ***Expert Systems with Applications***, Vol. 41, No. 8, (2014), 3809-3824. DOI: 10.1016/j.eswa.2013.12.017
9. Microsoft Communication & Media Industries, "Multi-Cloud Service Delivery end-to-end Management," Ref.architecture, 2013. https://cloudblogs.microsoft.c om/industry-blog/industry/uncategorized/multi-cloud- service-delivery-and-end-to-end-management-reference- architecture/
10. Venkat, M., 2016. Enterprise cloud strategy: Governance IBM. https://www.ibm.com/blogs/cloud-computing/2016/11/03/enterprise-governance-multi-cloud/
11. Yu, Q., & Bouguettaya, A., "Efficient Service Skyline Computation for Composite Service Selection", ***IEEE Transactions on Knowledge and Data Engineering***, Vol. 5, No. 4, (2013), 776-789. DOI: 10.1109/TKDE.2011.268
12. Belkasmi, D., Hadjali, A., & Azzoune, H., "On Fuzzy Approaches for Enlarging Skyline Query Results", ***Applied Soft Computing***, Vol. 74, (2019), 51-65. DOI: 10.1016/j.asoc.2018.10.013
13. Elmi, S., & Min, J. K., "Spatial Skyline Queries over Incomplete Data for Smart Cities", ***Journal of Systems Architecture***, Vol. 90, (2018), 1-14. DOI: 10.1016/j.sysarc.2018.08.005
14. Lim, J., Li, H., Bok, K., & Yoo, J., "A Continuous Reverse Skyline Query Processing Method in Moving Objects Environments", ***Data & Knowledge Engineering***, Vol. 104, (2016), 45-58. DOI: 10.1016/j.datak.2015.05.003
15. Yang, Z., Li, K., Zhou, X., Mei, J., & Gao, Y., "Top k Probabilistic Skyline Queries on Uncertain Data", ***Neurocomputing***, Vol. 317, (2018), 1-14. DOI: 10.1016/j.neucom.2018.03.052
16. Zou, G., Chen, Y., Yang, Y., Huang, R., & Xu,Y., "AI Planning and Combinatorial Optimization for Web Service Composition in Cloud Computing", In Proccedding of the International Conference on Cloud Computing and Virtualization, (2010), 1-8. DOI: 10.5176/978-981-08-5837-7_166
17. Gutierrez-Garcia, J. O., & Sim, K. M., "Agent-based Cloud Service Composition", ***Applied Intelligence***, Vol. 38, No. 3, (2013), 436-464. DOI: 10.1007/s10489-012-0380-x
18. Jatoth, C., Gangadharan, G.R., & Buyya, R., "Optimal Fitness Aware Cloud Service Composition using an Adaptive Genotypes Evolution based Genetic Algorithm", ***Future Generation Computer Systems***, Vol. 94, (2019), 185-198. DOI: 10.1016/j.future.2018.11.022
19. Jatoth, C., Gangadharan, G. R., & Fiore, U., "Optimal Fitness Aware Cloud Service Composition using Modified Invasive Weed Optimization", ***Swarm and Evolutionary Computation***, Vol. 44, (2019), 1073-1091. DOI: 10.1016/j.swevo.2018.11.001

20. Gavvala, S. K., Jatoth, C., Gangadharan, G. R., & Buyya, R., "QoS-Aware Cloud Service Composition using Eagle Strategy", ***Future Generation Computer Systems***, Vol. 90, (2019), 273-290. DOI: 10.1016/j.future.2018.07.062
21. Yu, Q., Chen, L., & Li, B., "Ant Colony Optimization Applied to Web Service Compositions in Cloud Computing", ***Computers & Electrical Engineering***, Vol. 41, (2015), 18-27. DOI: 10.1016/j.compeleceng.2014.12.004
22. Kurdi, H., Al-Anazi, A., Campbell, C., & Al Faries, A., "A Combinatorial Optimization Algorithm for Multiple Cloud Service Composition", ***Computers & Electrical Engineering***, Vol. 42, (2015), 107-113. DOI: 10.1016/j.compeleceng.2014.11.002
23. Mezni, H., & Sellami, M., "Multi-Cloud Service Composition using Formal Concept Analysis", ***Journal of Systems and Software***, Vol. 134, (2017), 138-152. DOI: 10.1016/j.jss.2017.08.016
24. Mezni, H., & Abdeljaoued, T., "A Cloud Services Recommendation System based on Fuzzy Formal Concept Analysis", ***Data & Knowledge Engineering***, Vol. 116, (2018), 100-123. DOI: 10.1016/j.datak.2018.05.008
25. Wu, J., Chen, L., & Liang, T., "Selecting Dynamic Skyline Services for QoS-based Service Composition", ***Applied Mathematics & Information Sciences***, Vol. 8, No. 5, (2014), 2579. DOI: DOI: 10.1145/1772690.1772693
26. Zhang, F., Hwang, K., Khan, S., & Malluhi, Q., "Skyline Discovery and Composition of Inter-Cloud Mashup Services", ***IEEE Transactions on Services Computing***, Vol. 9, No. 1, (2016), 72-83. DOI: 10.1109/TSC.2015.2449302
27. Zhang, J., Jiang, X., Ku, W. S., & Qin, X., "Efficient Parallel Skyline Evaluation using Mapreduce", ***IEEE Transactions on Parallel and Distributed Systems***, Vol. 27, No. 7, (2016), 1996-2009. DOI: 10.1109/TPDS.2015.2472016
28. Liu, Y., Yang, R., & Zhang, S., "Service Selection Method based on Skyline in Cloud Environment", ***International Journal of Performability Engineering***, Vol. 13, No. 7, (2017). DOI: 10.23940/ijpe.17.07.p5.10391047
29. Moradi, M., & Emadi, S., "Reducing the Calculations of Quality-Aware Web Services Composition Based on Parallel Skyline Service", ***International Journal of Advanced Computer Science and Applications***, Vol. 7, No. 7, (2016). DOI: 10.14569/IJACSA.2016.070744
30. Borzsony, S., Kossmann, D., & Stocker, K., "The skyline Operator", In Proceedings 17th IEEE International Conference on Data Engineering, (2001), 421- 430. DOI: 10.1109/ICDE.2001.914855
31. Papadias, D., Tao, Y., Fu, G., & Seeger, B., "Progressive skyline Computation in Database Systems", ***ACM Transactions on Database Systems***, Vol. 30, No. 1, (2005), 41-82. DOI: 10.1145/1061318.1061320
32. Wang, Y., Song, Y., & Liang, M., "A Skyline-based Efficient Web Service Selection Method Supporting Frequent Requests", In 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD), (2016), 328-333. DOI: 10.1109/CSCWD.2016.7566009
33. Fariss, M., Asaidi, H., & Bellouki, M., "Comparative Study of Skyline Algorithms for Selecting Web Services based on QoS", Procedia Computer Science 127, (2018), 408-415. DOI: 10.1016/j.procs.2018.01.138
34. Alrifai, M., Skoutas, D., & Risse, T., "Selecting Skyline Services for QoS-based Web Service Composition", In Proceedings of the 19th International Conference on World Wide Web, (2010), 11-20. DOI: 10.1145/1772690.1772693
35. Benouaret, K., Benslimane, D., & Hadjali, A., "Ws-Sky: An Efficient and Flexible Framework for QoS-aware Web Service Selection", In IEEE Ninth International Conference on Services Computing, (2012), 146-153. DOI: 10.1109/SCC.2012.83
36. Fekih, H., Mtibaa, S., & Bouamama, S., "Local-Consistency Web Services Composition Approach based on Harmony Search", Procedia Computer Science 112, (2017), 1102-1111. DOI: 10.1016/j.procs.2017.08.135


## Persian Abstract


**چکیده**

رشد سریع بهره‌برداری از محیط‌های ابری موجب ارائه‌ی انواع مختلف منابع به عنوان سرویس در این محیط شده است. از آن‌جا که عملکرد یک سرویس معمولاً بسیار ساده است و پاسخگوی نیاز پیچیده‌ی کاربر نیست، نیاز به ترکیب این سرویسها که قادر به ارضا نیازهای کاربران باشد، ضروری است. بیشتر روشهای ترکیب سرویس در محیط های چند ابری فرض می کنند که سرویس‌های شرکت‌کننده در ترکیب در یک ابر هستند که این رویکرد غیر واقعی است زیرا ممکن است ابرهای دیگر سرویس‌های مناسب‌تری را ارائه دهند. در ترکیب سرویس‌های توزیع شده در محیط‌های چند ابری، یک کار چالش‌برانگیز دیگر کاهش هزینه‌های مالی با کاهش تعداد ارائه‌دهندگان و ابرهای شرکت‌کننده در ترکیب و کاهش هزینه‌ی ارتباطات بین ارائه‌دهندگان و ابرها است. برای رفع این چالش باید تعداد فراهم کنندگان سرویسها در ابرها در فرایند ترکیب کاهش یابد. این تحقیق از الگوریتم Skyline Service برای ترکیب سرویس‌ها در محیط‌های چند ابری استفاده می کند تا تمام ابرها در فرایند ترکیب سرویس بررسی شوند. روش پیشنهادی می تواند یک سرویس ترکیبی قابل استفاده برای کاربر ارائه کند که پارامترهایی همچون کمترین تعداد ارائه‌دهنده و ابر را درنظر می گیرد. الگوریتم Skyline Service در دو مرحله استفاده می شود. در مرحله اول، بهترین ترکیب سرویس در یک ابر از میان تمام فراهم کنندگان با درنظرگرفتن تعداد فراهم کنندگان و زمان ارتباطی انتخاب می شود. در مرحله دوم، الگوریتم Skyline Service برای ایجاد تمام ترکیبات ممکن در محیط چند ابری استفاده می شود. پارامترهایی مثل تعداد ابر کمتر و زمان ارتباطی کمتر بین ابرها در این مرحله اعمال می شود. نتایج نشان می دهد که روش پیشنهادی می تواند ترکیبی با حداقل تعداد ابرها، کمترین هزینه و کمترین زمان محاسباتی را پیدا کند. در نهایت می توان گفت که Skyline Service یک ترکیب مناسب از سرویسهای درخواستی کاربر را در یک محیط چند ابری انتخاب می کند.