Article

Energy-Efficient Cloud Service Selection and Recommendation Based on QoS for Sustainable Smart Cities

Preeti Sirohi [1], Fahd N. Al-Wesabi [2,*] , Haya Mesfer Alshahrani [3], Piyush Maheshwari [4], Amit Agarwal [5], Bhupesh Kumar Dewangan [6], Anwer Mustafa Hilal [7] and Tanupriya Choudhury [6]

[1] Department of Computer Science, Institute of Management Studies, Ghaziabad 201009, India; preetisirohi_10@yahoo.com
[2] Department of Computer Science, College of Science & Art at Mahayil, King Khalid University, Abha 61421, Saudi Arabia
[3] Department of Information Systems, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh 11671, Saudi Arabia; hmalshahrani@pnu.edu.sa
[4] Department of Computer Science, British University in Dubai, Dubai 345015, United Arab Emirates; dr_piyush@yahoo.com
[5] Department of Computer Science, Dr. APJ Abdul Kalam Govt. Institute of Technology, Tanakpur, Uttrakhand 262309, India; coer.info@gmail.com
[6] Informatics Cluster, School of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand 248007, India; bhupesh.dewangan@gmail.com (B.K.D.); tanupriya1986@gmail.com (T.C.)
[7] Department of Computer and Self Development, Preparatory Year Deanship, Prince Sattam Bin Abdulaziz University, Alkharj 11942, Saudi Arabia; a.hilal@psau.edu.sa
* Correspondence: falwesabi@kku.edu.sa

Featured Application: The application of the proposed research is to select service based on optimized results achieved in this research article.

Abstract: The growing demand for cloud technology brings several cloud service providers and their diverse list of services in the market, putting a challenge for the user to select the best service from the inventory of available services. Therefore, a system that understands the user requirements and finds a suitable service according to user-customized requirements is a challenge. In this paper, we propose a new cloud service selection and recommendation system (CS-SR) for finding the optimal service by considering the user's customized requirements. In addition, the service selection and recommendation system will consider both quantitative and qualitative quality of service (QoS) attributes in service selection. The comparison is made between proposed CS-SR with three existing approaches analytical hierarchy process (A.H.P.), efficient non-dominated sorting-sequential search (ENS-SS), and best-worst method (B.W.M.) shows that CR-SR outperforms the above approaches in two ways (i) reduce the total execution time and (ii) energy consumption to find the best service for the user. The proposed cloud service selection mechanism facilitates reduced energy consumption at cloud servers, thereby reducing the overall heat emission from a cloud data center.

Keywords: quality of service (QoS); cloud computing; cloud services; A.H.P.; optimal service; smart buildings

1. Introduction

Cloud computing is based on distributed technology and offers different types of computational resources such as hardware, operating system, and applications to its customers through the Internet on a 24/7 basis and from anywhere in the world [1]. The I.T. giants such as I.B.M., Microsoft, H.P., Google, Amazon, etc., are seen in the cloud market either as a provider or consumer [2]. Cloud technology facilitates its customers to opt for services and reduces the total up-front cost of buying and maintaining the complete infrastructure [3]. There are three main cloud service models: infrastructure as a service (IaaS),

software as a service (SaaS), and platform as a service (PaaS) [4] and the cloud services taken from the provider in the form of "as a service". Through cloud services, databases can be built and accessed for various research areas, e.g., environmental pollution [5], food authentication [6], and building sector [7,8]. In addition, the requirements of each customer are different; therefore, a generalized cloud service selection approach will not be efficient as compared to the customized selection approach, which will incorporate an individual's requirements.

Moreover, the user requirements also depend on multiple criteria, which are, in most cases, often conflicting in nature [9–11]. Therefore, selecting cloud services by considers various conflicting criteria at the same time is also a challenge [12]. Cloud services are a tedious part of adopting cloud technology [13,14].

The performance of any cloud service is measured through the value of its quality of service (QoS) parameters [15]. The Service Measurement Index (S.M.I.) [16] has recognized seven QoS attributes and sub-attributes for each cloud service. Some of the attributes mentioned here include reliability, availability, maintainability, agility, accountability, cost, privacy, security, usability, etc. [17]. The QoS attributes are classified into two types: qualitative and quantitative attributes [18,19], which simplifies the identification and recommendation of the service according to user-customized requirements. The service selection process includes the comparison of the services to each other for finding the best service. If in case there are two or more attributes, then the services should consider all the attributes for comparison. The service selection process is directly proportional to the number of services and the number of attributes. Therefore, if the number of services increases or the number of attributes increases, then it becomes a tedious and time-consuming task to compare the services. Thus, a need for a decision-making system that analyzes the services by considering multiple attributes and recommends the best service to the customer [20]. Evolutionary algorithms [21] are found efficient in finding services in selecting services in case of multi-criteria optimization [22]. Existing work proposed by different researchers using an evolutionary algorithm for selecting cloud services is shown in the work of [23–26]. Some of the papers consider single or limited QoS attributes [27–29], while some articles consider multi-criteria [30–32] for cloud selecting and ranking cloud services.

In the cloud, there is an enormous list of available services, and all of these services might not be useful for consideration in the process of service selection [33]. Therefore, if the user requirements are taken in advance, then unnecessary services can be filtered in the initial step itself. Even in the case of multiple criteria, the candidate services can be filtered [34–37] by considering user requirements for all the mentioned criteria.

To support the decision-making process of service selection and recommendation, the contribution in this paper is as follows:

1. The novel algorithm called cloud service selection and recommendation (CS-SR) is proposed. The algorithm entails four phases, including filtration, evaluation, integration, and last is selection and recommendation. The outcome of CS-SR is two-fold. (a) Offering a QoS-based service selection, (b) reducing overall execution time required to find optimal service;
2. The filtration phase will reduce unnecessary comparison by filtering out candidate services;
3. The proposed approach makes use of quantitative and qualitative attributes that will improve the overall efficiency of our selection and recommendation approach;
4. The proposed CS-SR is compared with the analytical hierarchy process (A.H.P.) [38], efficient non-dominated sorting-sequential search (ENS-SS) [39], and best-worst method (B.W.M.) [40] on the performance parameter: total execution time and the energy consumption used in selecting and recommending the cloud service. The result shows that CS-SR outperforms the compared method. The three existing algorithms are chosen because all three algorithm deals with multi-criteria decision making and finds the optimal solution among the list of available solutions.

The research is based on providing energy-efficient cloud service selection recommendation system. The proposed work, in general, is a step toward saving energy through the cloud servers at large and a step toward efficient cloud service selection and ranking system. The research paper's main objective is to improve the performance of the proposed algorithm CS-SR in two areas; energy consumption and execution time. Multiple studies in the past have presented negative impacts of data centers on climate change. It is believed that the carbon footprint of data centers is equivalent to that of the aviation industry. According to a report, there will be approximately 7.2 million operational data centers by 2021 across the world. Talking in terms of heat emission, data centers could contribute 3.2% of the total worldwide carbon emission by 2025 [41]. Therefore, the energy-efficient mechanism proposed in our manuscript ensures a reduction in carbon emission and supports sustainable smart cities.

This paper is planned as follows: Section 2 briefly discusses different approaches proposed by authors. In Section 3, the CS-SR architecture is proposed. Section 4, the CS-SR algorithm, is presented. In Section 5, an illustration of the working of the CS-SR approach with an example is given. In Section 6, the implementation and analysis of the proposed algorithm are discussed. Finally, Section 7 provides an inference drawn, limitation, and future research ideas.

2. Related Work

The increasing demand for cloud services motivates researchers to explore the research in the area of service selection and ranking. The existing literature discussed below shows the proposed algorithms and approaches for selecting cloud services.

Zheng et al. [42] proposed a ranking framework using QoS in identifying relevant cloud services and also by finding similar users for predicting quality of services values to use time effectively.

Rajkumar et al. [43] designed a SMICloud (Service Measurement Index), and Dewangan et al. proposed WARMS [44], where Ishizaka et al. presented analytic hierarchy process [45] framework using (A.H.P.) [46] to rank services. The ranking process occurs in the following manner: gathered requirements, assigning weightage to attributes, and ranking services.

Jahani and Mohammad Khanli [47] proposed (NSGA SR) to select candidate services and rank them for the multi-objective optimization problem. However, a rank [48] is developed later for improving the ranking process NSGA_SR [49] and also for finding candidate services meeting user's requirements.

Zhang et al. [50] proposed a proficient approach for sorting the solutions for the primary objective function, and the other objective functions can be ignored.

In [51], McClymont et al. use the climbing sorting and deductive approach. In both methods, the deductive sort is found more efficient than the climbing approach.

Roy et al. [52] provided a new approach, known as best order sort (B.O.S.), for selecting and ranking services. The approach depends on two steps for the complete working of the approach. Firstly, all the solutions are sorted for each objective, and secondly, ranks are assigned to the solution. Tang et al. [53] developed a strategy based on the arena's principle for identifying a non-dominated solution to a front. The dominating solution is selected one by one and is placed in the arena, and is known as the arena host. The dominating solution will be the new arena host and will replace the existing solution; otherwise, it is ignored.

In the work of [54], Godse and Mulik talked about some essential parameters required for SaaS selection. The service selection is made using the A.H.P. technique implemented for the case study of salesforce automation. Limam and Boutaba in the work of [55] consider the QoS parameter, such as cost, for designing a trust-based framework to rank SaaS services.

Garg et al. [56] considers cloud services as an MCDM problem and proposed an A.H.P.-based approach for ranking cloud services. In the work of [57], Rehman et al. use the same case study for selecting services in infrastructure as a service (IaaS).

In the work of [58], Sun et al. provided a service selection methodology for MCDM attributes. The author uses a fuzzy-based technique for determining and consolidating relations between the mentioned criteria. In addition, Alabool et al. [59] did a systematic literature survey for different approaches proposed under MCDM.

Whaiduzzaman et al. [60] conducted an extensive literature survey and also compared several multi-criteria decision analysis methods and showed the work through the taxonomic classification of cloud service selection. Mingrui et al. [61] use both the qualitative and quantitative parameters in cloud service selection. The priority related to QoS is also taken into consideration for finding service. The A.H.P. approach is used in the research. The limitation of AHP is that the efficiency of this approach is reduced when this approach is applied for multi-objective optimization problems.

In [62], Jatoth et al. talk about an updated super-efficiency data envelopment analysis for service selection of cloud services, and the evaluation of the ranking process is based on user preferences. In [63], the author extends a gray technique for order of preference and assigns ranks to the services based on the QoS parameter.

The autonomic nature of the cloud [64,65] can result in seamless service selection and address consumer requests efficiently without minimum or no human intervention.

The works illustrated in papers [66–72] discuss the nature of energy-efficient cloud computing and depict ways and architectures for ensuring energy saving in a cloud ecosystem.

Authors Zhang X et al. developed an efficient non-dominated sequential sorting algorithm ENS-SS [40], achieved multi-objective optimization for QoS service selection. The drawback of this approach is that it provides an efficient solution in case of bi-objectives or problems with three objectives, but if the number of services or objectives is increased, this approach does not yield efficiency. Nawaz F. et al. proposed the best-worst method BWM in cloud service selection to achieve multi-objective optimization by Markov-chain technique [39]. The limitation of BWM is that it does not identify an optimal solution in case of multiple objectives as it will produce varying criteria, non-unique weights that may affect the final decision result though this approach is considered better than the AHP approach.

The literature survey shows that the existing methods are efficient in finding services for the users when the number of services or attributes is less. If the existing approach is applied to a large number of services or multiple criteria are taken, then the efficiency in finding optimal service decreases.

3. Proposed Architecture: CS-SR

The proposed architecture of cloud service selection and recommendation system CS-SR is shown in Figure 1. The CS-SR has three main objectives: First, it involves a decision-making system (D.M.S.) that takes the request of the user for services, processes the request, and returns the appropriate service to the user. Secondly, our proposed method prevents unnecessary comparison of the services and ranks them. Thirdly, our approach reduces the overall execution time taken to find the best services and also evaluates the energy consumption used in finding optimal service according to customer requirements.

The working of the proposed architecture CS-SR is as follows: the user will send the request for the required services to the decision-making system [D.M.S.]. The request of the user consists of two parameters: QoS attributes range and priority for QoS attributes. The user requirement helps the decision-making system (D.M.S.) to understand the user requirements expected from the offered services. The D.M.S. will filter out those services that fit into the range from the service list [1 to N]. The service list is made available to the D.M.S. by the cloud service providers [1 to S]. The services offered by the provider also consist of the QoS parameter value and the feedback received from the experience

customers. The D.M.S. in CS-SR architecture helps in reducing the total execution time by removing services that do not fit in the range taken from the user. The Cs-SR architecture also increases the overall accuracy of the service selection and ranking because it considers using both quantitative and qualitative QoS attributes for selecting and ranking of service.
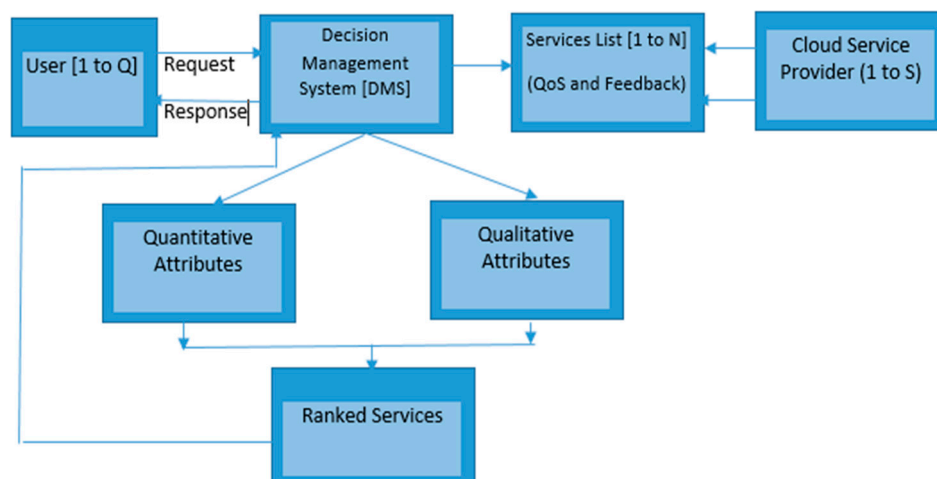

Figure 1. The cloud service selection and recommendation architecture.

The decision-making system (D.M.S.) helps in ranking the service and provide the optimal service to the user, as shown in Figure 2. The D.M.S. system will act as the central coordinating point by maintaining the request of all the users and concurrently finding out the service for each customer. The D.M.S. gets activated as soon as the user request the service. The user requirements are taken through, and the QoS attributes are taken in range (min, max), and the user priority is related to the QoS attributes. The D.M.S. system is comprised of four phases, i.e., (1) filtration, (2) evaluation, (3) integration and ranking, (4) selection and recommendation. Figure 2 shows the internal working of CS-SR, and the detailed discussion of each stage in CS-SR discussed below.

3.1. Filtration Step

The first step where requirements and the priority for the QoS objective taken from the user. The outcome of the filtration step is (1) this step will significantly reduce unnecessary services from the list; (2) this step will help decision-makers in finding optimal service quickly; (3) the filtered candidate service are only those who fit into the user requirements; therefore, it significantly improves the efficiency of the decision maker.

3.2. Evaluation of the Fitness Function

The second step involves identifying and assessing the fitness function separately for both the objective and the subjective attributes [59]. The computation of the fitness function is the crucial step for deciding as it has a direct influence on the ranking of the service. The fitness function is calculated for objective function using the QoS attribute, and the priority of the attribute entered by the user and the fitness function in the case of the subjective attribute is the QoS attribute and the feedback collected from the experienced user regarding the service. The feedback of all the services are taken and stored is shared by the provider to the D.M.S.
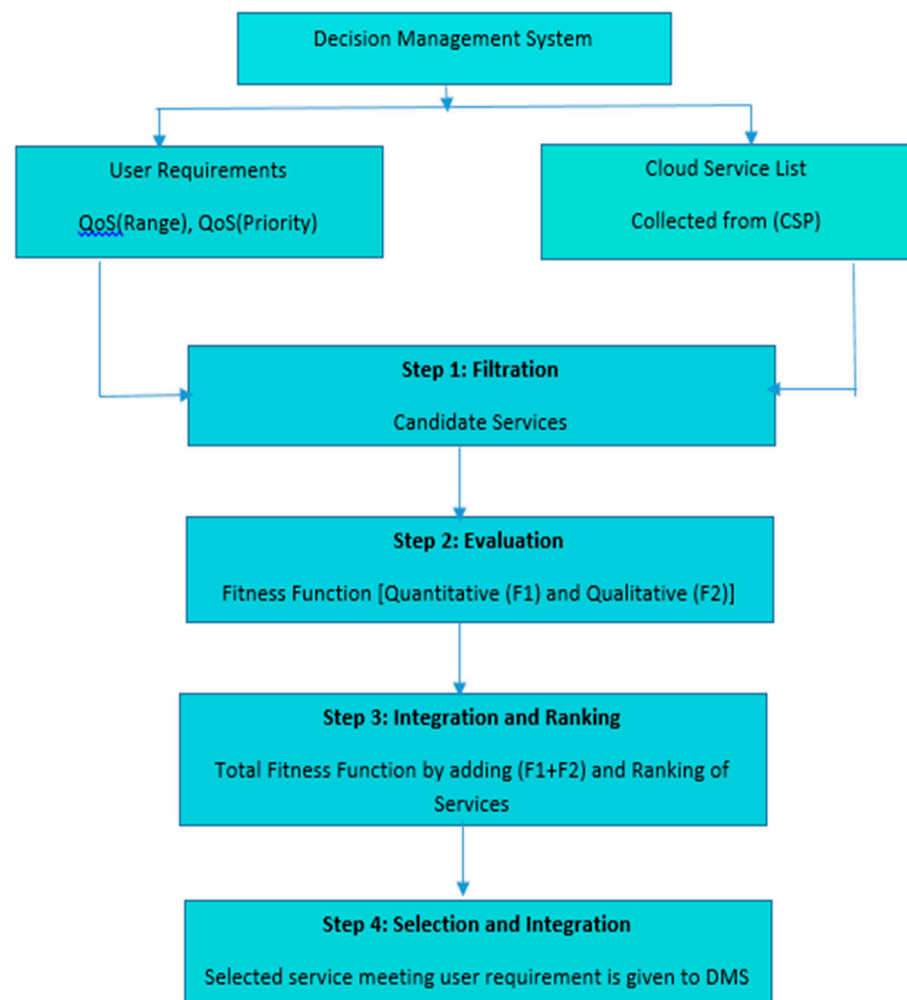
Figure 2. Decision-making system.

3.3. Integration and Ranking

This step integrates the fitness function computed for both objective and subjective attributes for each service. The rank of the service will depend upon the final value received after the integration of both objective fitness function and subjective fitness function.

3.4. Selection

This step helps in selecting the appropriate service from candidate services. The candidate service having the highest rank is taken as an appropriate or the best service according to the user requirements.

The decision-making system (D.M.S.) will pick the highest-ranked service and will pass the service as the execution to the cloud user.

4. Assumption Involved in CS-SR

The entities involved in the proposed algorithm (CS-SR) in Section 3 include users, cloud service providers, and the decision-making system. The primary objective of the algorithm is to filter out the candidate services and rank those services.

4.1. Modeling Variables Used in CS-SR

The variables used in the CS-SR method are the requirements taken from the user in the pre-defined range (with the minimum and the maximum) QoS attributes [23]. The priority of the QoS attributes requested is also given in numeric form.

The number of cloud users is shown in Equation (1):

$$U = < u_1, u_2, \ldots, u_q, \ldots, u_Q >, U \in [1, Q], \tag{1}$$

where U is the set containing the number of users in which u_Q is one of the users from the is the total number of users Q requesting the cloud services.

The requirements raised by one of the cloud users are shown in Equation (2):

$$U_r = < u_{1[min,max]}, u_{2[min,max]}, \ldots, u_{r[min,max]}, \ldots, M > U_r \in [1, M], \tag{2}$$

where U_r refers to the requirements of the user in the given range [min, max]. The $u_{qr[min,max]}$ represents the qth user requirements for the rth attribute taken in range, and M represents the total number of attributes. The QoS priority given by the cloud user is shown in Equation (3).

$$P_u = < p_1, p_2, \ldots, p_i, \ldots, p_M > P_u \in [1, M]. \tag{3}$$

The priority is taken from the user for each attribute because some attributes are more critical to the user. The attributes with the lowest priority are considered more important than others.

The total number of services is N stored in the service set S, and S_j is one of the services from the total services from 1 to N. The service S_j also has QoS attributes associated with the service, and there can be total M QoS attributes for the service S_j. The QoS attribute q_m is one of the QoS attributes for service S_j. The cloud services and their QoS attributes offered by the provider are shown in Equations (4) and (5):

$$S = \{s_1, s_2, \ldots, s_j, \ldots, s_N\}, j \in [1, N], \tag{4}$$

$$s_j = < q_1, q_2, \ldots, q_m, \ldots, q_M >, j \in [1, M], \tag{5}$$

The candidate set C.S. containing the list of candidate services received after the filtration step, and the feedback associated with each candidate service is shown in Equation (6). $s_j q_{jfj}$ represents the jth service with the q_m QoS attribute, and F_j is the feedback for the service taken from an experienced user.

$$CS = \left\{ s_1 q_{1f1}, s_2 q_{2f2}, \ldots, s_j q_{mfj}, \ldots, s_N q_{MfN} \right\} j \in [1, M], \tag{6}$$

4.2. Filtration of Candidate Service in CS-SR

The user sent the requirement and priority to the decision-making system. The D.M.S. will check the user requirements from the available service list and will return the candidate service, and will store their feedback collected from the service provider. The filtration of the required candidate services is shown in Algorithm 1.

Algorithm 1: Filtration of Candidate Services

Input : U_r, S_j, P_u
Ensure : candidate services CS_j, feedback ($f_n S_n$);
1 : for all services in S_j list do
2 : if S_j found in a user-specified range of U_r
3 : add S_j to set of candidate services
4: end if
5: end for
6: return candidate set and feedback to D.M.S.

The above Algorithm 1 shows that every available service is checked to find if the service matches the user requirement, then it is shown in lines 1 and 2. In case if the available service falls in the required range, it is included in the candidate services set

shown in line 3. Lastly, the candidate services with respective feedback are stored in D.M.S. for further action.

4.3. Evaluation of Candidate Service in CS-SR

The candidate service received from the filtration step will go for the evaluation step. Here fitness function is evaluated for candidate services by considering both the quantitative and qualitative assessment.

The quantitative assessment of the fitness function is performed using the QoS value of the service assigned, and the priority given by the provider is shown below:

$$\text{Fitness function } FF_{qn} = f(x) = \sum_1^M \left(CS_j * Pu_j * q_j \right) \tag{7}$$

The qualitative assessment of the fitness function is performed using the feedback collected for the service:

$$\text{Fitness function } FF_{ql} = \sum_1^M \left(CS_j * q_j * f_j \right) \tag{8}$$

4.4. Integration and Ranking of Candidate Service in CS-SR

As mentioned in Algorithm 2 steps, the total fitness function is calculated by adding FF_{qn} and FF_{ql} from Equations (7) and (8). The ranking of the services is performed based on the service value received from the full fitness function:

$$\text{Total fitness function } TFFS_i = FF_{qn} * FF_{ql} = \sum_1^M \left(CS_j * Pu_j * q_j \right) * \sum_1^M \left(CS_j * q_j * f_j \right) \tag{9}$$

Algorithm 2: Integration and Ranking Candidate Services by D.M.S.

Require : FF_{qn}, FF_{ql}
Ensure Total Fitness function $TFFS_i$, Ranked services (RS_i);
1 : Summing up. $\left(TFFS_i = FF_{qn} + FF_{ql} \right)$
2 : $r = 1$;
3 : for services S_i
4 : if $TFFS_i >=$ compared services
5: set Service Rank to R_1 and remove service from comparison
6 : $r = r + 1$
7: repeat step 3–6 for remaining services
8: end if
9: end for
10: return the rank of candidate services

4.5. Selection of Candidate Service in CS-SR

The value of the computed $TFFS_i$ for the service will decide its ranking. The service having the highest $TFFS_i$ is considered to be the appropriate service or the best service according to the customer requirements.

5. Illustration of CS-SR through Example

This section shows the overall working of CS-SR. Suppose we have a total of 10 services (s1, s2, . . . , s10). This service list is provided by different cloud providers and only one active user. Each service has at least two quality attributes, and the feedback stored for all these 10 services, which is shown in Table 1.

The user requirements are taken range from them for their desired attributes; for example, in the above Table 1, the range for Attribute 1 given by the user is in between (96–110), and the range for Attribute 2 mentioned by the user ranges between (6–10). In case the user mentioned the priority for any QoS attributes, he can also mention that in case if there are only two attributes, the user can give only priority high priority to any one of them. In case if we have three attributes, then there should be three priorities represented

in the numerical form (1, 2, and 3). In this example, Attribute 1 is given priority 1, and Attribute 2 is given 2nd priority.

Table 1. List of services with two attributes and their feedback.

S. No.	Services	Attribute 1	Attribute 2	Feedback (1–10) in Range
1.	S1	100	10	7
2.	S2	110	6	3
3.	S3	90	8	5
4.	S4	95	7	6
5.	S5	105	9	4
6.	S6	115	12	9
7.	S7	80	5	8
8.	S8	92	7.5	7
9.	S9	102	8	5
10.	S10	85	6.5	4

Step 1: As shown in Table 2, we have four filtered candidate services from step 1. These candidate services carry their quality attributes and feedback, and later these candidate services will be used as input for further steps and finally ranking the service Attribute 1 (96–110) and for Attribute 2 (6–10).

Table 2. Candidate services.

S. No.	Services	Attribute 1	Attribute 2	Feedback (1–10)
1.	S1	100	10	7
2.	S2	110	6	3
3.	S5	105	9	4
4.	S9	102	8	5

Step 2: Table 3 shows the fitness function applied for qualitative and quantitative assessment on the filtered services shown in Table 2. The feedback is used for the qualitative evaluation, and QoS are used for the quantitative assessment.

Table 3. Fitness function is evaluated for qualitative and quantitative attributes.

S. No.	Services	Attr.1	Attr.2	Fitness Function $FF_{qn} = \sum\limits_{1}^{M}(csi \times pi \times qi)$	Feedback [1–10]	Fitness Function $FF_{ql} = \sum\limits_{1}^{M}(csi \times qifeedback)$
1.	S1	100	10	$100 \times 2 + 10 \times 1 = 210$	7	$100 \times 7 + 10 \times 7 = 770$
2.	S2	110	6	$110 \times 2 + 6 \times 1 = 226$	3	$110 \times 3 + 6 \times 3 = 348$
3.	S5	105	9	$105 \times 2 + 9 \times 1 = 219$	4	$105 \times 4 + 9 \times 4 = 456$
4.	S9	102	8	$102 \times 2 + 8 \times 1 = 212$	5	$102 \times 5 + 8 \times 5 = 550$

Step 3: Table 4 calculates the total fitness value of each candidate service, and rank is assigned to the service. The service that has the highest total fitness value will be ranked 1, and the process continues till all the candidate services are entrusted to their respective rank.

Table 4. Integration and ranking of the services.

S. No.	Services	Attribute 1	Attribute 2	Fitness Function $TFFS_i = FF_{qn} \times FF_{ql}$	Rank ($TFFS_i$)
1.	S1	100	10	$210 \times 770 = 161,700$	Rank 1-S1
2.	S2	110	6	$226 \times 348 = 78,648$	Rank 4-S2
3.	S5	105	9	$219 \times 456 = 99,864$	Rank 3-S5
4.	S9	102	8	$212 \times 550 = 116,600$	Rank 2-S9

Step 4: Based on the total fitness function and ranking of the services, we can see that the S1 has the highest fitness function; therefore, rank 1 is assigned to S1, and S2 has the lowest rank shown in Table 4, and the D.M.S. will send an execution to the user.

6. Experimental Work

6.1. Implementation Details

To validate the performance of the CS-SR algorithm, we compare CS-SR with three known algorithms from the literature: A.H.P., ENS-SS [39], and B.W.M. The experiments were carried out on a Windows 10 P.C. with a 3.30 GHz Intel i5 processor and 4 G.B. of RAM. There are two experiments conducted for all four algorithms: In the first experiment, the number of candidate services is increased from [10–160] with an increment of 10, and the execution time is calculated for bi-objective [m = 2]. In the second experiment, the number of candidate services is [10–160] with an increment of 10, and execution time for multi-objective [m = 5]. The data set used to evaluate the execution time calculated in (milliseconds) is taken from Github. The proposed algorithm CS-SR will find the optimal services even if the number of services or number of objectives is increased.

The performance analysis of the proposed CS-SR approach is performed for two parameters

(a) Execution Time: Execution time is the time the user request for the service and the execution (optimal service) the user gets from the system.
(b) Energy Computation:—The total amount of energy or power consumed for finding services through execution time.

6.2. Analysis of CS-SR through Execution Time

The CS-SR algorithm is compared with three existing approaches A.H.P., ENS-SS, and B.W.M. algorithm [70]. Table 5 shows that the CS-SR algorithm has the lowest execution time as compared to the other three algorithms. The execution time of A.H.P., ENS-SS, and B.W.M. increases with the increase in the number of objectives and number of services. CS-SR shows better performance for execution time when compared with other algorithms.

Tables 5 and 6 represent the execution time for bi-objective [m = 2] and multi-objective [m = 5]. The number of candidate services ranges from [10–160] with an increment of 10.

Table 5 represents the total execution time the CS-SR approach takes in finding the best services according to the user requirements. The execution time increases if the number of candidate services increases. Therefore, we can see from Table 5 and the graph in Figure 3 that for the bi-objective problem, there is a linear increase in the execution time.

Table 6 represents the total execution time for all four approaches A.H.P., ENS-SS, B.W.M., and CS-SR for m = 5. Each approach is executed separately for the same number of candidate services from [10–160], and the execution time is noted for them.

Table 5. Execution time [M = 2].

Candidate Services	AHP	ENS-SS	BWM	CS-SR
10	0.85	0.8	0.7	0.5
20	1.4	1	0.9	0.7
30	1.7	1.2	1	0.8
40	2	1.4	1.2	1
50	2.4	1.7	1.5	1.2
60	2.5	1.9	1.7	1.4
70	2.8	2.1	1.9	1.7
80	3.1	2.4	2.2	1.9
90	3.3	2.7	2.4	2.1
100	3.6	2.9	2.7	2.4
110	3.8	3	2.9	2.6
120	4	3.2	3.1	2.8
130	4.2	3.5	3.3	3.1
140	4.4	3.7	3.6	3.2
150	4.6	4	3.9	3.5
160	4.7	4.2	4.1	3.7

Table 6. Execution time [M = 5].

Candidate Services	AHP	ENS-SS	BWM	CS-SR
10	1.3	0.9	0.7	0.5
20	1.7	1.1	0.9	0.7
30	1.9	1.3	1.1	0.9
40	2.3	1.6	1.4	1.1
50	2.6	1.9	1.7	1.3
60	2.9	2.2	1.9	1.6
70	3.1	2.6	2.1	1.8
80	3.4	2.9	2.5	2.1
90	3.6	3.2	2.8	2.4
100	3.8	3.4	2.9	2.6
110	4.2	3.7	3.1	2.8
120	4.4	3.9	3.3	2.9
130	4.7	4.2	3.6	3.2
140	5	4.6	4	3.6
150	5.3	4.9	4.4	3.8
160	5.6	5.3	4.7	4.1

Table 6 represents the total execution time the CS-SR approach takes in finding the best services according to the user requirements. The execution time increases if the number of candidate services increases. Therefore, we can see from Table 5 and the graph in Figure 3 that for the multi-objective [M = 5] problem, there is a linear increase in the execution time.
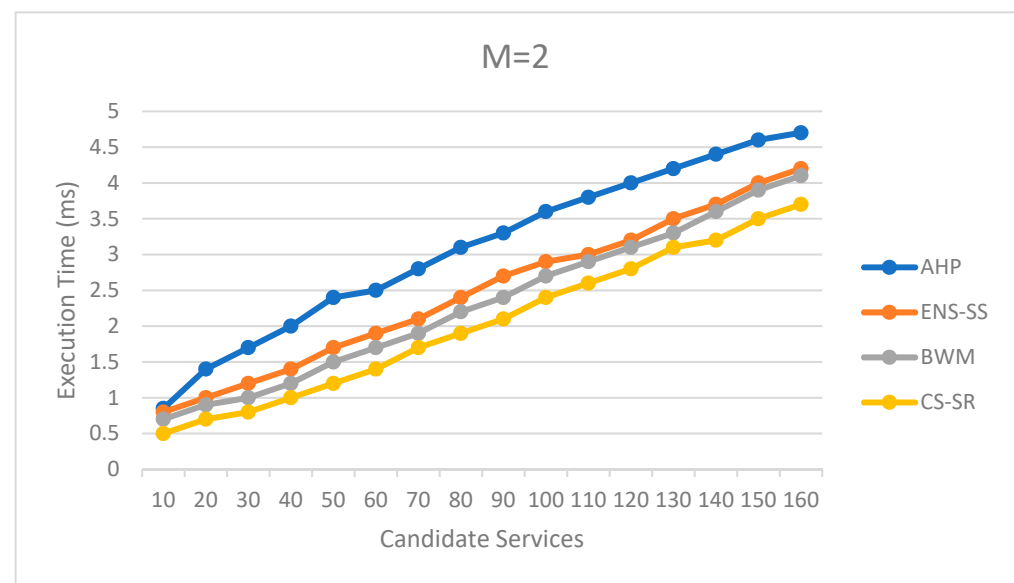
Figure 3. Execution time for bi-objective [M = 2].

6.3. Analysis of CS-SR through Energy Consumption

The energy consumption required for finding the optimal service can be calculated through the execution time. Therefore, the amount of energy consumed using a constant of proportionality calculated using an execution time is obtained in Equation (10). Table 7 shows the energy consumption in (Joules) for each of the algorithms for the service [10–160].

$$\text{E} = \int_{n=1}^{N} \alpha * \text{R.T.} \tag{10}$$

where E is the energy consumption, T is the execution time, and α is the constant of proportionality, in this case, $\alpha = 1$.

Table 7. Energy consumption.

S. No.	M = 2	M = 5
AHP	49.35	55.8
ENS-SS	39.7	47.7
BWM	37.1	41.1
CS-SR	32.6	35.4

The above Tables 5–7 shows that for m = 2, A.H.P. has the highest energy consumption and highest execution time, and CS-SR has the lowest energy consumption and lowest execution time. In addition, for M = 5, the CS-SR approach outperforms the other three existing approaches in terms of energy consumption and execution time.

The following Figure 4 is a graph plotted for the energy consumption for four approaches that shows that A.H.P. consumes the maximum energy for both m = 2 and m = 5. CS-SR approach is found as an efficient approach as compared to the other three approaches.
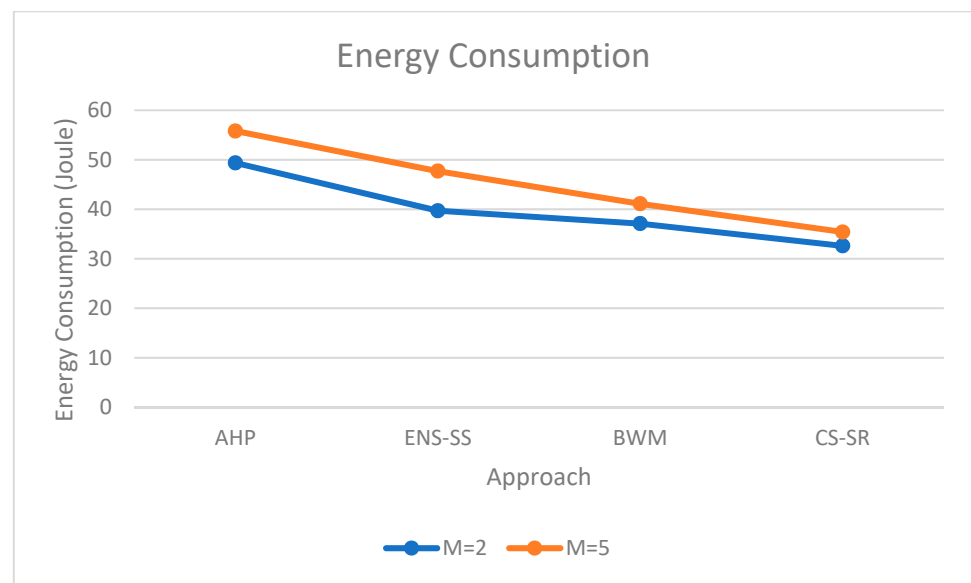
Figure 4. Energy consumption.

The experiment was conducted on CS-SR, A.H.P., and ENS_SS by using a sample size of services from [10–160] with 10 jumps. The total execution time taken by all three algorithms is shown in Figures 3 and 5 and measured against service. The experiments conducted for two attributes in Figure 3 and five attributes in Figure 5, the execution time is evaluated for both the scenario with an increasing number of services. The experiment results in Figure 3 depict the ENS-SS approach is capable of responding initially to the smaller number of services, but the execution time increases later. The A.H.P. approach is also efficient with a lesser number of services but not with a large number of services, and it needs more time to respond and find service. The CS-SR selects are found to be efficient as compared to A.H.P. and ENS-SS in finding an optimal service by reducing execution time.
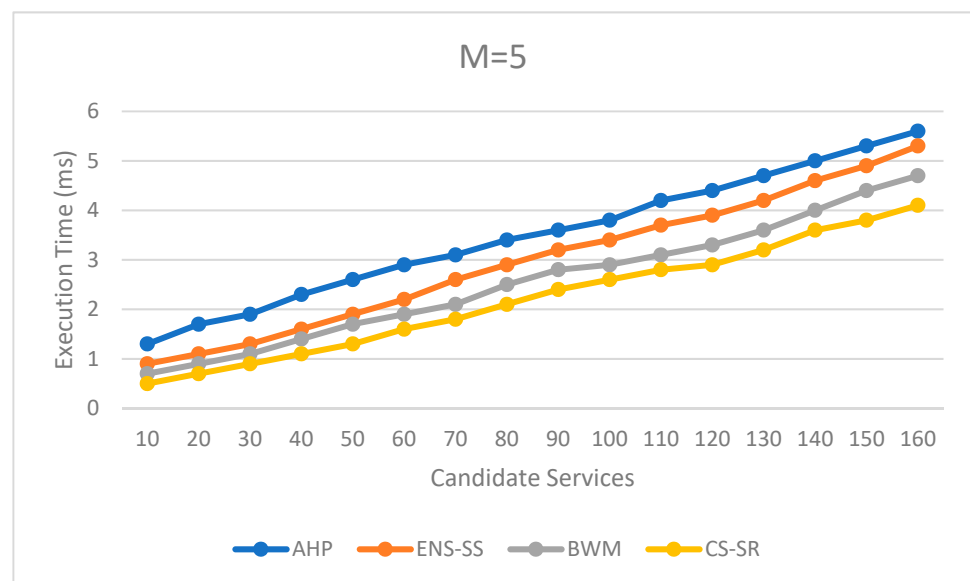

Figure 5. Execution time for multi-objective [M = 5].

Moreover, CS-SR is energy efficient in comparison with A.H.P. and ENS-SS as it reduces energy consumption in context to Jules. The proposed service selection and recommendation model supports QoS along with ensuring reduced execution time and

use of idle cloud servers. Henceforth, the CS-SR model is proficient to total execution time helps in creating an energy-efficient and economically sustainable cloud ecosystem.

7. Conclusions and Future Scope

The research paper proposed a novel algorithm (CS-SR) that is based in D.M.S. for finding appropriate cloud service that meets user requirements. The proposed approach aims to lower down the total execution time in finding the service. The proposed approach CS-SR also reduced the unnecessary comparison using the filtration step. In addition, CS-SR used both the quantitative and qualitative parameters to find the result and turned out to be an efficient approach.

The proposed approach finds the total fitness value by using qualitative and quantitative parameters. The research considers the feedback taken from an experienced user to calculate the fitness value for qualitative parameters. The user feedback can be biased sometimes, and in such cases, the result we obtain by using biased values will not provide an accurate result and can be considered the limitation of the proposed approach. Moreover, the proposed cloud service selection mechanism ensures energy conservation and facilitates reduced carbon emission at cloud data center levels, thus resulting in sustainable smart cities.

The research work can be expanded by classifying the user requirements into essential and non-essential requirements [63–65]. The consideration of different parameters improves the accuracy level and improves the trust between the provider and service user. In addition, the research work can be extended to the optimal composition of service, which will further assist the user is not only in identifying a single service but the group of services.

Author Contributions: All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Buyya, R.; Yeo, S.C.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging I.T. platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [CrossRef]
2. Zhang, Q.; Lu, C.; Raouf, B. Cloud computing: State-of-the-art and research challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18. [CrossRef]
3. Jahani, A.; Leyli, M.K. Cloud service ranking as a multiobjective optimization problem. *J. Supercomput.* **2016**, *72*, 1897–1926. [CrossRef]
4. Vecchiola, C.; Suraj, P.; Rajkumar, B. High-performance cloud computing: A view of scientific applications. In Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, Kaoshiung, Taiwan, 14–16 December 2009.
5. Iordache, M.; Iordache, A.M.; Sandru, C.; Voica, C.; Zgavarogea, R.; Miricioiu, M.G.; Ionete, R.E. Assessment of heavy metals pollution in sediments from reservoirs of the Olt River as tool of environmental risk management. *Rev. Chim.* **2019**, *70*, 4153–4162.
6. Botoran, O.R.; Ionete, R.E.; Miricioiu, M.G.; Costinel, D.; Radu, G.L.; Popescu, R. Amino acid profile of fruits as potential fingerprints of varietal origin. *Molecules* **2019**, *24*, 4500. [CrossRef]
7. Raboaca, M.S. Sustaining the Passive House with Hybrid Energy Photovoltaic Panels—Fuel Cell. *Prog. Cryog. Isot. Sep.* **2015**, *18*.
8. Raboaca, M.S.; Felseghi, R.A. Energy Efficient Stationary Application Supplied with Solar-Wind Hybrid Energy. In Proceedings of the 2019 International Conference on Energy and Environment (CIEM), Timisoara, Romania, 17–18 October 2019; pp. 495–499.
9. Dewangan, B.K.; Jain, A.; Choudhury, T. AP: Hybrid Task Scheduling Algorithm for Cloud. *Rev. d'Intelligence Artif.* **2020**, *34*, 479–485. [CrossRef]

10. Katchabaw, M.J.; Lutfiyya, H.L.; Bauer, M.A. Usage-based service differentiation for end-to-end quality of service management. *Comput. Commun.* **2005**, *28*, 2146–2159. [CrossRef]

11. Fan, H. An integrated personalization framework for SaaS-based cloud services. *Future Gener. Comput. Syst.* **2015**, *53*, 157–173. [CrossRef]

12. Triantaphyllou, E.; Mann, S.H. Using the analytic hierarchy process for decision making in engineering applications: Some challenges. *Int. J. Ind. Eng. Appl. Pract.* **1995**, *2*, 35–44.

13. El-Gazzar, R.; Hustad, E.; Olsen, D.H. Understanding cloud computing adoption issues: A Delphi study approach. *J. Syst. Softw.* **2016**, *118*, 64–84. [CrossRef]

14. Basahel, A.; Yamin, M.; Drijan, A. Barriers to Cloud Computing Adoption for S.M.E.s in Saudi Arabia. *Bvicams Int. J. Inf. Technol.* **2016**, *8*, 1044–1048.

15. Ding, S.; Yang, S.; Zhang, Y.; Liang, C.; Xia, C. Combining QoS prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems. *Knowl. Based Syst.* **2014**, *56*, 216–225. [CrossRef]

16. Garg, S.K.; Versteeg, S.; Buyya, R. Smicloud: A framework for comparing and ranking cloud services. In Proceedings of the Utility and Cloud Computing (U.C.C.), 2011 Fourth IEEE International Conference on IEEE, Victoria, NSW, Australia, 5–8 December 2011.

17. Liu, Y.; Moez, E.; Boulahia, L.M. Evaluation of Parameters Importance in Cloud Service Selection Using Rough Sets. *Appl. Math.* **2016**, *7*, 527. [CrossRef]

18. Stojanovic, M.D.; Rakas, S.V.B.; Acimovic-Raspopovic, V.S. End-to-end quality of service specification and mapping: The third party approach. *Comput. Commun.* **2010**, *33*, 1354–1368. [CrossRef]

19. Qu, L.; Wang, Y.; Orgun, M.A. Cloud service selection based on the aggregation of user feedback and quantitative performance assessment. In Proceedings of the 2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, 28 June–3 July 2013; IEEE Computer Society: Washington, DC, USA, 2013; pp. 152–159.

20. Mao, C. Search-based QoS ranking prediction for web services in cloud environments. *Future Gener. Comput. Syst.* **2015**, *50*, 111–126. [CrossRef]

21. Ardagna, D.; Casale, G.; Ciavotta, M.; Pérez, J.F.; Wang, W. Quality-of-service in cloud computing: Modeling techniques and their applications. *J. Internet Serv. Appl.* **2014**, *5*, 11. [CrossRef]

22. Singh, P.K.; Paprzycki, M.; Bhargava, B.; Chhabra, J.K.; Kaushal, N.C.; Kumar, Y. (Eds.) Futuristic Trends in Network and Communication Technologies. In Proceedings of the First International Conference, FTNCT 2018, Solan, India, 9–10 February 2018.

23. Garg, S.K.; Versteeg, S.; Buyya, R. A framework for ranking of cloud computing services. *Future Gener. Comput. Syst.* **2013**, *29*, 1012–1023. [CrossRef]

24. Chan, H.; Trieu, C. Ranking and mapping of applications to cloud computing services by S.V.D. In Proceedings of the Network Operations and Management Symposium Workshops (NOMS Wksps), Osaka, Japan, 19–23 April 2010; IEEE: Piscataway, NJ, USA, 2010.

25. Fang, H.; Wang, Q.; Tu, Y.-C.; Horstemeyer, M.F. An efficient non-dominated sorting method for evolutionary algorithms. *Evol. Comput.* **2008**, *16*, 355–384. [CrossRef] [PubMed]

26. Baghel, S.K.; Dewangan, B.K. Defense in Depth for Data Storage in Cloud Computing. *Int. J. Technol.* **2012**, *2*, 58–61.

27. Wooldridge, M. *An Introduction to Multi-Agent Systems*; Department of Computer Science, University of Liverpool: Liverpool, UK, 2009; pp. 200–450.

28. Dewangan, B.K.; Agarwal, A.; Choudhury, T.; Pasricha, A. Cloud resource optimization system based on time and cost. *Int. J. Math. Eng. Manage. Sci.* **2020**, *5*, 758–768. [CrossRef]

29. Yau, S.; Yin, Y. QoS-based service ranking and selection for service-based systems. In Proceedings of the IEEE International Conference on Services Computing, Washington, DC, USA, 4 June–9 July 2011; pp. 56–63.

30. Almulla, M.; Yahyaoui, H.; Al-Matori, K. A new fuzzy hybrid technique for ranking real-world Web services. *Knowl. Based Syst.* **2015**, *77*, 1–15. [CrossRef]

31. Skoutas, D.; Sacharidis, D.; Simitsis, A.; Sellis, T. Ranking and clustering web services using multi-criteria dominance relationships. *IEEE Trans. Serv. Comput.* **2010**, *3*, 163–177. [CrossRef]

32. Dikaiakos, M.D.; Yazti, D.Z. A distributed middleware infrastructure for personalized services. *Comput. Commun.* **2004**, *27*, 1464–1480. [CrossRef]

33. Dewangan, B.K.; Agarwal, A.; Venkatadri, M.; Pasricha, A. Design of self-management aware autonomic resource scheduling scheme in cloud. *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.* **2019**, *11*, 170–177.

34. Octavio, J.; Ramirez, A. Collaborative agents for distributed load management in cloud data centres using Live Migration of virtual machines. *IEEE Trans. Serv. Comput.* **2015**, *8*, 916–929.

35. Octavio, J.; Ramirez, A. Agent-based load balancing in cloud data centres. *Cluster Comput.* **2015**, *18*, 1041–1062.

36. Al-Masri, E.; Mahmoud, Q.H. Investigating web services on the world wide web. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008; ACM Press: New York, NY, USA, 2008; pp. 795–804.

37. Zheng, Z.; Wu, X.; Zhang, Y.; Lyu, M.; Wang, J. QoS ranking prediction for cloud services. *J. IEEE Trans. Parallel Distrib. Syst.* **2012**, *24*, 1213–1222. [CrossRef]

38. Khan, I.; Meena, A.; Richhariya, P.; Dewangan, B.K. Optimization in Autonomic Computing and Resource Management. In *Autonomic Computing in Cloud Resource Management in Industry 4.0*; Springer: Cham, Switzerland, 2021; pp. 159–175.

39. Zhang, X.; Tian, Y.; Cheng, R.; Jin, Y. An efficient approach to nondominated sorting for evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* **2014**, *19*, 201–213. [CrossRef]

40. Trueman, C. What Impact Are Data Centres Having on Climate Change? Available online: https://www.computerworld.com/article/3431148/why-data-centres-are-the-new-frontier-in-the-fight-against-climate-change.html (accessed on 9 August 2019).

41. Holst, A. Number of Data Centers Worldwide 2015–2021. Available online: https://www.statista.com/statistics/500458/worldwide-datacenter-and-it-sites/ (accessed on 2 March 2020).

42. Malhotra, R.; Dewangan, B.K.; Chakraborty, P.; Choudhury, T. Self-Protection Approach for Cloud Computing. In *Autonomic Computing in Cloud Resource Management in Industry 4.0*; Springer: Cham, Switzerland, 2021; pp. 213–228.

43. Hao, Y.; Zhang, Y.; Cao, J. Web services discovery and Rank: An information retrieval approach. *Future Gener. Comput. Syst.* **2010**, *26*, 1053–1062. [CrossRef]

44. Dewangan, B.K.; Agarwal, A.; Choudhury, T.; Pasricha, A. Workload aware autonomic resource management scheme using grey wolf optimization in cloud environment. *IET Commun* **2021**, *15*, 1869–1882. [CrossRef]

45. Ishizaka, A.; Labib, A. Analytic hierarchy process and expert choice: Benefits and limitations. *Or Insight* **2009**, *22*, 201–220. [CrossRef]

46. Dewangan, B.K.; Agarwal, A.; Choudhury, T.; Pasricha, A.; Chandra Satapathy, S. Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges. *Softw. Pract. Exp.* **2020**. [CrossRef]

47. Jahani, A.; Farnaz, D.; Leyli, M.K. Arank: A multi-agent-based approach for ranking of cloud computing services. *Scalable Comput. Pract. Exp.* **2017**, *18*, 105–116. [CrossRef]

48. Dewangan, B.K.; Shende, P. The Sliding Window Method: An Environment To Evaluate User Behavior Trust In Cloud Technology. *Int. J. Adv. Res. Comput. Commun. Eng.* **2013**, *2*, 1158–1162.

49. McClymont, K.; Keedwell, E. Deductive sort and climbing sort: New methods for non-dominated sorting. *Evol. Comput.* **2012**, *20*, 1–6. [CrossRef] [PubMed]

50. Roy, P.C.; Islam, M.M.; Deb, K. Best order sort: A new algorithm to non-dominated sorting for evolutionary multiobjective optimization. In Proceedings of the 2016 Genetic and Evolutionary Computation Conference Companion, Denver, CO, USA, 20–24 July 2016; ACM Press: New York, NY, USA, 2016; pp. 1113–1120.

51. Tang, S.; Cai, Z.; Zheng, J. A fast method of constructing the non-dominated set: Arena's principle. In Proceedings of the ICNC'08, Fourth International Conference on Natural Computation, Jinan, China, 18–20 October 2008; IEEE: Piscataway, NJ, USA, 2008; Volume 1.

52. Godse, M.; Mulik, S. An approach for selecting software-as-a-service (saas) product. In *Proceedings of the IEEE International Conference on Cloud Computing, Bangalore, India, 21–25 September 2009*; IEEE Computer Society: Washington, DC, USA, 2009; pp. 155–158.

53. Limam, N.; Boutaba, R. Assessing software service quality and trustworthiness at selection time. *IEEE Trans. Softw. Eng.* **2010**, *36*, 559–574. [CrossRef]

54. Dewangan, B.K.; Agarwal, A.; Venkatadri, M.; Pasricha, A. Resource scheduling in cloud: A comparative study. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 168–173. [CrossRef]

55. Rehman, Z.U.; Hussain, O.K.; Hussain, F.K. Iaas cloud selection using MCDM methods. In Proceedings of the 9th IEEE International Conference on E-Business Engineering, Hangzhou, China, 9–11 September 2012; pp. 246–251.

56. Sun, L.; Dong, H.; Hussain, O.K.; Hussain, F.K.; Liu, A.X. A framework of cloud service selection with criteria interactions. *Future Gener. Comput. Syst.* **2019**, *94*, 749–764. [CrossRef]

57. Tomar, R.; Khanna, A.; Bansal, A.; Fore, V. An architectural view towards autonomic cloud computing. In *Data Engineering and Intelligent Computing*; Springer: Singapore, 2018; pp. 573–582.

58. Kero, A.; Khanna, A.; Kumar, D.; Agarwal, A. An Adaptive Approach Towards Computation Offloading for Mobile Cloud Computing. *Int. J. Inf. Technol. Web Eng. IJITWE* **2019**, *14*, 52–73. [CrossRef]

59. Juarez, F.; Ejarque, J.; Badia, R.M. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Gener. Comput. Syst.* **2018**, *78*, 257–271. [CrossRef]

60. Yaqoob, I.; Ahmed, E.; Gani, A.; Mokhtar, S.; Imran, M. Heterogeneity-aware task allocation in mobile ad hoc cloud. *IEEE Access* **2017**, *5*, 1779–1795. [CrossRef]

61. Hu, B.; Cao, Z.; Zhou, M. Scheduling Real-Time Parallel Applications in Cloud to Minimize Energy Consumption. *IEEE Trans. Cloud Comput.* **2019**. [CrossRef]

62. Mishra, S.K.; Puthal, D.; Sahoo, B.; Jena, S.K.; Obaidat, M.S. An adaptive task allocation technique for green cloud computing. *J. Supercomput.* **2018**, *74*, 370–385. [CrossRef]

63. Xu, M.; Buyya, R. BrownoutCon: A software system based on brownout and containers for energy-efficient cloud computing. *J. Syst. Softw.* **2019**, *155*, 91–103. [CrossRef]

64. Raboaca, M.S.; Dumitrescu, C.; Manta, I. Aircraft Trajectory Tracking Using Radar Equipment with Fuzzy Logic Algorithm. *Mathematics* **2020**, *8*, 207. [CrossRef]

65. Dewangan, B.K.; Agarwal, A.; Venkatadri, M.; Pasricha, A. Sla-based autonomic cloud resource management framework by antlion optimization algorithm. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2019**, *8*, 119–123.

66. Alabool, H.; Kamil, A.; Arshad, N.; Alarabiat, D. Cloud service evaluation method-based Multi-Criteria Decision-Making: A systematic literature review. *J. Syst. Softw.* **2018**, *139*, 161–188. [CrossRef]
67. Singh, P.; Sood, S.; Kumar, Y.; Paprzycki, M.; Pljonkin, A.; Hong, W.C. *Futuristic Trends in Networks and Computing Technologies*; FTNCT 2019 Communications in Computer and Information Science; Springer: Singapore, 2019; Volume 1206, pp. 3–707.
68. Singh, P.K.; Bhargava, B.K.; Paprzycki, M.; Kaushal, N.C.; Hong, W.C. *Handbook of Wireless Sensor Networks: Issues and Challenges in Current Scenario's*; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2020; Volume 1132, pp. 155–437.
69. Whaiduzzaman, M.; Gani, A.; Anuar, N.B.; Shiraz, M.; Haque, M.N.; Haque, I.T. Cloud service selection using multi-criteria decision analysis. *Sci. World J.* **2014**, *2014*, 459375. [CrossRef]
70. Sun, M.; Zang, T.; Xu, X.; Wang, R. Consumer-centered cloud services selection using A.H.P. In Proceedings of the 2013 International Conference on Service Sciences (ICSS), Shenzhen, China, 11–13 April 2013; IEEE: Piscataway, NJ, USA, 2013.
71. Jatoth, C.; Gangadharan, G.R.; Fiore, U. Evaluating the efficiency of cloud services using modified data envelopment analysis and modified super-efficiency data envelopment analysis. *Soft Comput.* **2017**, *21*, 7221–7234. [CrossRef]
72. Jatoth, C.; Gangadharan, G.R.; Fiore, U.; Buyya, R. SELCLOUD: A hybrid multi-criteria decision-making model for selection of cloud services. *Soft Comput.* **2019**, *23*, 4701–4715. [CrossRef]