

Article

Adaptive Multi-Level Cloud Service Selection and Composition Using AHP–TOPSIS

V. N. V. L. S. Swathi ^{1,*} , G. Senthil Kumar ²  and A. Vani Vathsala ³ ¹ Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, Chennai 603203, India² Department of Computational Intelligence, SRM Institute of Science and Technology, Kattankulathur, Chennai 603203, India; senthilg1@srmist.edu.in³ Department of Computer Science and Engineering, CVR College of Engineering, Hyderabad 501510, India; vani_vathsala@cvr.ac.in* Correspondence: vs9189@srmist.edu.in

Abstract

The growing diversity of cloud services has made evaluating their relative merits in terms of price, functionality, and availability increasingly complex, particularly given the wide range of deployment alternatives and service capabilities. Cloud manufacturing often requires the integration of multiple services to accomplish user tasks, where the effectiveness of resource utilization and capacity sharing is closely tied to the adopted service composition strategy. This complexity, intensified by competition among providers, renders cloud service selection and composition an NP-hard problem involving multiple challenges, such as identifying suitable services from large pools, handling composition constraints, assessing the importance of quality-of-service (QoS) parameters, adapting to dynamic conditions, and managing abrupt changes in service and network characteristics. To address these issues, this study applies the Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) in conjunction with Multi-Criteria Decision Making (MCDM) to evaluate and rank cloud services, while the Analytic Hierarchy Process (AHP) combined with the entropy weight method is employed to mitigate subjective bias and improve evaluation accuracy. Building on these techniques, a novel Adaptive Multi-Level Linked-Priority-based Best Method Selection with Multistage User-Feedback-driven Cloud Service Composition (MLLP-BMS-MUFCSC) framework is proposed, demonstrating enhanced service selection efficiency and superior quality of service compared to existing approaches.

Keywords: cloud service selection; cloud service provider; cloud service composition; quality of service



Academic Editor: Christos Bouras

Received: 4 September 2025

Revised: 26 September 2025

Accepted: 28 September 2025

Published: 14 October 2025

Citation: Swathi, V.N.V.L.S.; Senthil Kumar, G.; Vani Vathsala, A. Adaptive Multi-Level Cloud Service Selection and Composition Using AHP–TOPSIS. *Appl. Sci.* **2025**, *15*, 11010. <https://doi.org/10.3390/app152011010>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cloud computing has emerged as a dominant paradigm that offers scalable, on-demand resources to users. However, as the number of services grows, identifying and selecting the most appropriate combination has become increasingly complex. Service composition requires balancing functional requirements with non-functional properties, making it a challenging decision-making task. Cloud computing refers to a model in which information technology (IT) resources are made available as a service over the Internet. Cloud computing is widely used to manage massive datasets because it provides the necessary infrastructure. Today's Internet-based technology has increased processing speed, storage space [1], and adaptability. In software development, web services are

called service-oriented architecture (SOA) [2]. Applications and services are developed through SOA, utilizing several methods. Organizations have embraced SOA to accomplish both straightforward and intricate jobs and supply a workflow for business activities they refer to as service composition [3]. The service execution engine manages and orchestrates composing services [4]. Quality-of-service qualities are handled by a collection of web services that may be modified and replaced in production without impacting other business processes [5]. Various internal and external factors, including the hosting environment, the network, and the upgrade of the service, can all have an impact on the quality of service, as can the issues of web service expansion [6], performance, and the ability to respond to a large number of synchronous requests while still meeting the expected SLA requirements, performance, and behavior [7].

Static and dynamic approaches are employed to select and build the service compositions [8]. A core tenet of static approaches is using preexisting cloud services to build service compositions that perform the desired action. In cases where there are few cloud services and enough synchronous requests to go around, this form of architecture is used. When there is a large pool of cloud services from which to choose, various service compositions can be built using dynamic methods and approaches [9]. Dynamic methods and techniques represent advanced procedures designed to fulfill both the functional and non-functional requirements of the SLA, including the automatic substitution of integrated cloud services to provide and meet the anticipated SLA expectations of clients [10]. In addition to the elements that affect the quality of cloud services, the limits imposed by these criteria are still tricky, and they are not efficient enough when used in real-time and large-scale contexts [11].

Cloud computing environments present complex optimization challenges because of their dynamic, large-scale, and heterogeneous nature. Two of the most critical issues are resource allocation and load balancing, which directly impact system performance, cost efficiency, and the quality of service. Resource allocation involves assigning user requests to suitable virtual machines (VMs) and services in a way that minimizes response time while maximizing throughput and meeting Service-Level Agreements (SLAs). Conversely, load balancing ensures that workloads are evenly distributed across resources to prevent bottlenecks, maintain reliability, and improve scalability. Recent studies, such as Dynamic Load Balancing in Cloud Computing using Hybrid Kookaburra-Pelican Optimization Algorithms (ARIIA), demonstrate how hybrid optimization techniques can effectively address these challenges by achieving better workload distribution and faster convergence. These optimization strategies form a valuable context for the present research, as cloud service selection and composition can also be framed as a multi-objective optimization problem. While load balancing focuses on distributing computational tasks, service selection requires identifying the most appropriate services from a large pool under multiple QoS constraints. Building on this optimization perspective, the proposed work integrates multi-criteria decision-making methods (TOPSIS with AHP-based weighting) and user feedback to provide an efficient and adaptive cloud service selection and composition solution.

Each cloud provides different services, yet all work together for the user. Multiple clouds may be involved in fulfilling a service request. Prior studies have mostly ignored the energy composition involved in file transfers across clouds, which includes the user's energy consumption during file transfers [12,13]. Before now, while composing services in a multi-cloud setting, little thought was given to energy efficiency [14]. Users and cloud providers incur energy costs associated with cloud file transfer [15]. A standardized service description that presents the service's features and interface is a prerequisite for service composition in cloud computing. Many companies use the Web Services Description Language (WSDL) to describe their cloud computing offerings, like Amazon Elastic Compute

Cloud (EC2), Amazon Simple Storage Service (S3), Google Search, etc. However, in today's world of cloud computing, the standard WSDL falls short of what is needed to describe cloud-based service offerings [16] adequately. The quality of service and cost of a service will play larger roles in service discovery and service composition in the cloud. Most of the earliest web services research focused on better ways to archive and query web services [17]. These initiatives often target a particular service at a time. Web service usage has been on the rise recently, but it is not uncommon for multiple options to be available [18]. As a result, studies on composing online services are starting to materialize. The cloud service composition process is shown in Figure 1.

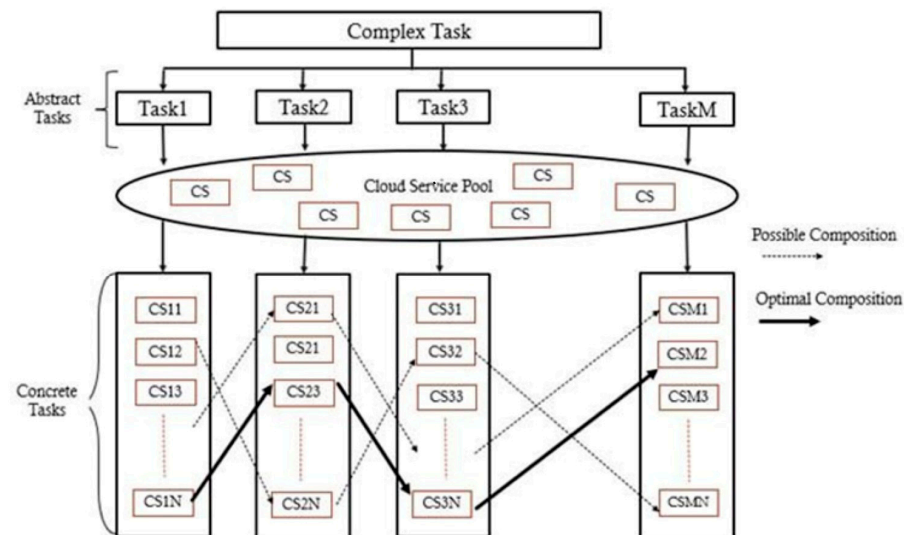


Figure 1. Cloud service composition levels.

Cloud computing is among the most cutting-edge innovations in the information technology sector today. Service-oriented architecture is credited with pioneering cloud computing. Every business seeks to utilize the cloud's scalability and reduced infrastructure, network, hardware, and software costs to serve their customers better. The idea of multitenancy allows cloud service providers to cut costs by providing the same service to several consumers or businesses using a single instance of the infrastructure [19,20]. Many cloud service providers are available today, adapting quickly to meet customer needs [21]. Cloud service companies compete for clients by making cutting-edge technologies more affordable for end users [22]. This makes it quite challenging for cloud users to choose the most suitable service provider. It is also getting tough to pick a deployment methodology among the several existing options [23]. The deployment models can be adjusted to meet the needs of various businesses. There are a variety of deployment models, each with its unique set of criteria. When using the cloud, it can be challenging to choose a service model and a provider [24].

The cloud computing business has expanded over the years, and numerous service providers are competing for customer attention by providing a wide range of service options. Currently, there are many cloud service providers, many of which offer services that are pretty similar in concept but differ in a few key respects [25]. Consumers often face challenges in identifying the cloud provider that best meets their requirements. For almost no out-of-pocket expense, users of cloud service providers like Amazon Web Services (AWSs) and Microsoft's Azure can choose to deploy their applications across a shared pool of virtual services, with running costs based on actual consumption [26]. Although cloud service providers enable businesses to focus on what they do best, there are still essential criteria that should be considered when selecting a provider. Software as a

Service (SaaS) [27], Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) are just some of the many service types available in the cloud, in addition to the more traditional public, private, and hybrid deployment options. Cloud customers and users face difficulty deciding which provider to work with. It also becomes challenging to settle on a deployment strategy. It is difficult for customers to evaluate a service provider's effectiveness without first gaining appropriate experience and information [28].

The exponential growth of cloud users may be directly attributed to the many advantages of cloud computing, such as its scalability, elasticity, reliability, etc., and its minimal capital investment requirements, making it the first choice for startups to host their services online. It has become difficult for cloud consumers to choose the best service provider to meet their functional and non-functional needs due to the abundance of cloud service providers (CSPs). It might be difficult for a cloud customer to pick the best CSP out of all the options available. Researchers and industry experts have been focusing on how to solve the difficulty of choosing the best CSP. The best CSP cannot be selected without a method for ranking providers based on quality-of-service criteria. To establish uniformity in the quality-of-service metrics utilized for assessing the quality of service provided by CSPs, the largest global organizations convened to create a collaboration known as the Cloud Services Measurement Initiative Consortium (CSMIC). The Service Measurement Index (SMI) is a framework developed by the CSMIC that encompasses seven essential quality-of-service criteria: accountability, financial, usability, performance, agility, assurance, security, and privacy. Furthermore, to accurately define the metrics, they decomposed each metric into several sub-metrics. The cloud service selection methods are depicted in Figure 2.

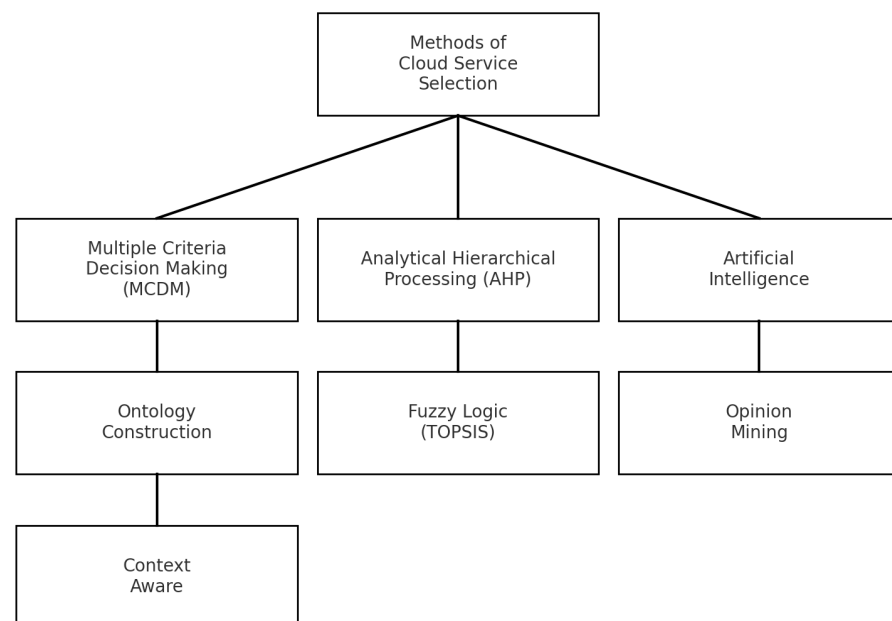


Figure 2. Cloud service selection methods.

Although the combination of the AHP and TOPSIS has been applied in earlier studies for multi-criteria decision making, our work advances this integration by developing a multi-level linked priority (MLP) framework with multistage user feedback. Unlike conventional AHP–TOPSIS approaches, where QoS weights are computed once and rankings remain static, our framework recalculates priorities dynamically across multiple stages, including service listing, feedback aggregation, composition, updating, and final allocation. Each stage is linked to the next, creating a cascading mechanism continuously refining service rankings. By incorporating user feedback particularly from experienced or high-profile users at different stages, the model adapts to evolving conditions and bet-

ter reflects objective QoS attributes and subjective user experiences. To our knowledge, such a feedback-driven, multistage priority adjustment has not been addressed in prior AHP–TOPSIS models. We believe this design significantly enhances adaptability and user-centric decision making, making the framework more practical for real-world cloud service environments where requirements and preferences constantly change.

Cloud consumers use these measurements to evaluate different CSPs. Choosing how to rank CSPs based on quality-of-service metrics is a decision problem. The TOPSIS is the most often used service selection statistical method because of its simplicity and effectiveness. However, it has the rank reversal problem because it promotes a sub-optimal CSP to the top spot whenever a CSP is added or removed from the cloud service repository. This research proposes Multi-Level Linked-Priority-based Best Method Selection using the TOPSIS with a Multistage User-Feedback-based Cloud Service Composition to select better cloud services with the best cloud service composition for cloud users that provide the best quality of service.

2. Literature Survey

In the cloud-enabled API economy, composing many API-defined services into a single composite service based on user requirements has emerged as a key method for developing new services. Several methods for selecting services to create custom services on demand have been offered. When making judgments about the composition of services with an eye toward quality, they frequently ignore the utilization of networking resources on the assumption that these are adequately provisioned. Optimal end-to-end quality of service for cloud-based applications is typically impractical, and these methods often lead to inefficient usage of network resources. With experimental comparisons for clouds using the popular fat-tree network topology, Wang et al. [1] presented a network-aware cloud service composition approach called NetMIP. To minimize the consumption of network resources and ensure optimal quality of service while adhering to the end-to-end quality-of-service requirements for the candidate composite services in the cloud, the author has defined the objective of service composition as a multi-objective constraint optimization problem. WebCloudSim, a reliable cloud infrastructure simulation technology, was used for experimental comparisons.

When resolving the connectivity and interoperability of cloud manufacturing (CMfg) resources and services, Cloud Manufacturing Service Composition (CMSC) is the key issue and plays a significant role. CMSC exemplifies the dynamic and uncertain nature of NP-hard issues. Traditional approaches to solving large-scale CMSC problems may not be practical due to the complexity of the resources required and the size of the search space involved. To address this shortcoming, Zhu j et al. [2] proposed MISABC, a novel artificial bee colony algorithm that uses several techniques to boost the effectiveness of the standard ABC algorithm, including the differential evolution strategy (DES), oscillation strategy with a classical trigonometric factor (TFOS), different dimensional variation learning strategy (DDVLS), and Gaussian distribution strategy (GD). In addition, the author suggested a manufacturing service composition technique called Multi-Module Subtasks Collaborative Execution for Cloud Manufacturing Service Composition (MMSCE-CMSC) to deal with the CMfg scenario. The algorithm's performance is verified using a case study, a comparative analysis with existing enhanced ABC algorithms, and a set of eight benchmark functions with varying features.

Since many cloud services are now spread across multiple data centers, innovative, scalable searching methods are needed to reduce the time, money, and resources required to fulfill requests. Since partial observability is commonplace and the cost of a prolonged search might be exponential, multi-agent-based systems have emerged as a potent strategy

for enhancing distributed processing on a massive scale. The Hybrid Kookaburra–Pelican Optimization Algorithm has been introduced as an effective technique for addressing load balancing in cloud computing, and its relevance extends directly to service selection and composition. Efficient load balancing is critical because it reduces the response time, improves resource utilization, and enhances the overall QoS, directly influencing the cloud service composition’s success. By referencing this algorithm, we emphasize the general role of optimization methods and their specific applicability to cloud service management tasks, thereby strengthening the connection between load balancing research and decision making in service selection frameworks. In their research, Kendrick et al. [3] described a service composition method based on multiple agents for efficiently retrieving distributed services and transmitting information within the agent network, thereby reducing the need for brute-force search. The thorough simulation results show that the energy cost per composition request may be lowered by over 50% simply by incorporating localized agent-based memory searches, reducing the number of actions.

As cloud computing grows in popularity, more QoS-aware service composition methodologies are being introduced into service-based cloud environments. The composite services’ energy and network resource usage are ignored when implementing these systems. Costs in data centers can rise significantly due to the increased use of energy and network resources that these combinations entail. Wang et al. [4] examined the quality-of-service performance, energy, and network resource consumption trade-off in the context of a service composition process. After that, an eco-friendly strategy for composing services was provided. Composite services that share a single virtual machine, physical server, or edge switch benefit from end-to-end quality-of-service assurance and are prioritized. Reducing the workload on the servers and switches utilized in cloud data centers makes it possible to fulfill the requirements for optimizing green service composition.

Song et al. [5] presented a cloud-based, collaborative framework for manufacturing service composition optimization that considers manufacturing service uncertainty through service uncertainty modeling. A method for estimating model parameters related to the uncertainty of manufacturing services is proposed, utilizing Gaussian mixture regression on the edge side of the proposed framework. At the same time, an intelligent evolutionary algorithm is adopted on the cloud side to optimize the manufacturing service composition efficiently. Adaptive modeling of service uncertainty is possible because the Gaussian mixed distribution approximates the service availability distribution. The author proposed a way for modeling the uncertainty of service that leads to improved service composition solutions compared to earlier optimization techniques for the composition of manufacturing services under uncertainty, utilizing deterministic parameter models. Extensive experimental findings demonstrate the efficiency of the algorithm.

The selection of a cloud provider and the subsequent decision on allocating scarce resources is made more difficult by the absence of a universal methodology for evaluating cloud providers and consumers. Existing service selection and SLA management systems have disregarded mainly the complex nonlinear link between service evaluation criteria. Because of that, none of the currently available options can deliver a reliable decision-making framework. These nonlinear connections between criteria have a substantial bearing on the selection procedure. This study proposes a centralized Quality-of-Experience (QoE) and quality-of-service structure to address the pressing problem. Cloud users can obtain help locating the best service provider with the suggested system. The framework considers the consumer’s prioritized criteria, ranks the relevance of each, and intelligently assigns weights. The framework aids the service provider in making sound decisions on allocating scarce resources. Using this paradigm, cloud participants can create long-lasting, reliable bonds. Hussain et al. [6] deployed a user-based collaborative filtering approach

powered by an improved top KNN algorithm, as well as the Analytical Hierarchy Process (AHP), the Induced OWA (IOWA) operator, and the Probabilistic OWA (POWA) operator. This approach can handle nonlinear relationships between the selection criteria. It uses an order-inducing variable to rearrange the inputs according to the consumer's customized preferences in relation to the other

Decision makers should be able to assess cloud services based on quality-of-service criteria, so Liu et al. [7] presented a practical, integrated MCDM scheme for cloud service evaluation and selection of cloud systems. They first proposed a comprehensive, cohesive MCDM framework for assessing and selecting cloud services and systems. The cloud model is utilized as a conversion tool for qualitative and quantitative information to quantify linguistic phrases, allowing for a more accurate and practical expression of the uncertainty of qualitative concepts. Second, a more thorough distance measurement algorithm based on cloud droplet distribution is provided for the cloud model, considering the limitations of conventional distinguishing measures. Similarity between cloud models and the gray correlation coefficient is computed using the novel distance-measuring algorithm. Comparing the expert evaluation cloud model to the arithmetic mean cloud model yields dynamic expertise weights. Next, the author proposed a multi-objective optimization model designed to maximize the relative proximity of all alternatives to both the positive and negative ideal solutions, aimed at determining the weights of the criteria, which is referred to as the TOPSIS method.

Multitenant Service-Based Systems (SBSs) built from a subset of cloud services have proliferated with the ubiquity of cloud computing. A multitenant SBS in the cloud simultaneously serves several tenants, each of which may have unique quality-of-service (QoS) requirements. This one-of-a-kind feature makes the problems of QoS-aware service selection during the development phase and system adaptability during operation even more difficult, rendering traditional methods outmoded and ineffective. Efficiency in developing and adjusting a multitenant SBS is of the utmost significance in the ever-changing cloud environment. Using K-Means clustering and Locality-Sensitive Hashing (LSH) algorithms, respectively, Wang et al. [8] presented two service recommendation methodologies for multitenant SBSs, one for build-time and one for runtime, to efficiently locate relevant services.

Virtual machines (VMs) in various flavors are available from a dizzying array of cloud service providers. Choosing the right VM is crucial from a business's point of view since picking the right services results in more efficiency, lower costs, and increased output. Due to request modularity, conflicting criteria, and the impact of network characteristics, effective service selection requires a methodical methodology. In their research, Askarnejad et al. [9] presented a novel PCA framework to address the service selection issue in a mixed public/private cloud infrastructure using peer assistance. VM rental and end-to-end network expenses are lowered, and inconsistencies between requests and corporate policies are uncovered, all while maintaining or improving service quality. To maximize efficiency and cut costs, PCA chooses services from different clouds. This framework uses set theory, B+ trees, and greedy algorithms.

Paradigms like SaaS and SaaS are coming into focus as cloud computing becomes the dominant part of software engineering. Cloud services depend on remote servers that handle requests from numerous users. When these users have conflicting demands regarding quality-of-service (QoS) attributes, selecting the appropriate service instance for each request becomes challenging. The complexity increases further because user applications are often composite workflows consisting of multiple tasks, where QoS must be evaluated across the entire composition. Current solutions generally lack both efficiency and adaptability. To address this, Kurdija et al. [10] introduced a heuristic method for

multi-criteria service selection that satisfies most, if not all, QoS requirements. Their approach employs a globally informed utility cost based on projected compositional QoS and iteratively refines the solution by decomposing the original transportation problem into several smaller subproblems.

3. Proposed Model

The availability and efficiency of resources and services have been greatly improved with the development of the computing model. Since the advent of cloud computing, numerous IT firms have been able to develop and launch brand-new cloud-based services rapidly. As a result, both the consumer and the service provider might expect financial gains. Layers of SaaS, PaaS, and IaaS make up the cloud's overarching architecture. SaaS oversees delivering applications on demand, while PaaS and IaaS, when combined, are known as computing services since they offer a foundation on which to build and launch apps. Google, Microsoft, IBM, Amazon, etc., are just a few well-known cloud service companies. However, as cloud services grow in popularity, the number of services offering the same basic features and benefits also skyrockets. This has decreased the prices at which these services may be purchased while simultaneously creating an extremely competitive market for cloud services due to the availability of numerous services competing to perform the same function. Sometimes, a single service cannot address the end user's functional and other needs.

Composing existing services is called service composition because it is preferable to establishing new complex services. Composing a cloud service, then, is the act of assembling multiple cloud services into a cohesive whole for the purpose of performing a more involved activity. While cloud service composition makes it possible to use existing infrastructure and services, this is not a simple task since it depends on several non-functional properties known as quality-of-service parameters, functional properties, and aggregate functions. Because of this, choosing which cloud services to use has become a complex problem, providing academics and industry experts with a stage upon which to collaborate on finding optimal solutions. There are two basic types of cloud service composition techniques: local selection techniques and global selection techniques. The regional selection optimization method finds the best services to meet users' needs according to their quality-of-service requirements. Multi-criteria decision making (MCDM) is a well-known local selection approach that considers multiple criteria. The strategy effectively selects cloud services but can only meet constraints on a small, local scale because it is not always the case that the user's need is based on regional limitations.

The quality of a service is measured by its quality-of-service indicators. QoS-aware approaches base their research on quality-of-service indicators like cost, time, and reliability. Increases in cloud services in the global service pool directly result from this delivery model's meteoric popularity. For many practical use cases, the present functional requirements cannot be met by a single simple service due to the presence of complex and diversified services. A collection of atomic simple services that cooperate is required to finish a complicated service. Therefore, a service composition system needs to be built into cloud computing. Service providers can account for the service introduction, request, and binding process by exposing their accessible services through the broker in response to user queries. Users still submit service requests to the providers, who choose the most appropriate service or combination of services based on those needs and trends. The providers then request that the services be bundled for the users according to the broker's established parameters.

As the number of services grows, different servers will offer more services with identical purposes. These parallel services are spread out geographically and have varying quality-of-service parameter values. So that the best possible quality of service can be provided in accordance with the needs and priorities of the end user, service composition employs suitable methods to pick an atomic service from among the various identical services hosted on separate servers. Since cloud environments, accessible services, and end-user requirements are constantly evolving, SC should be built to be dynamic and can perform some tasks automatically. As a result, one of the most critical challenges in service composition is determining which simple services should be merged to create composite, complicated services. Determining which atomic simple services to select to ensure that the resulting complex composite service meets functional and quality-of-service (QoS) requirements constitutes the service composition challenge in cloud computing. In this research, a new model for composing services and improving the underlying service execution engine is proposed. The model aids in enhancing cloud service selection and composition processes and maximizing resource use. The number of cloud service repositories, the number of clients, the SLA needs of those clients, the client classes, the virtual machines, and the mapping between each VM and those client classes must all be determined at this stage. The TOPSIS-based cloud service selection process is shown in Figure 3.

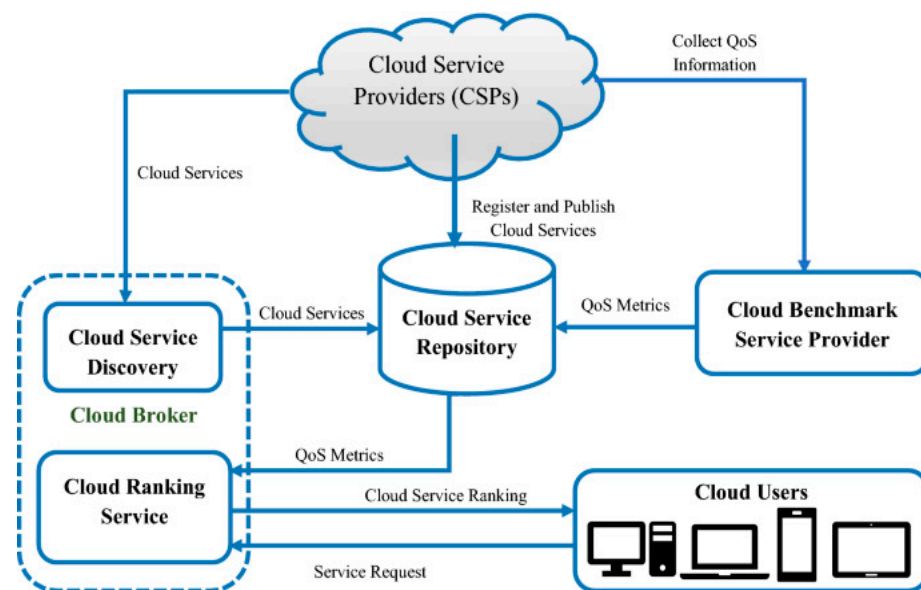


Figure 3. TOPSIS-based cloud service selection model.

Finding the most direct route to a service's endpoint is how service compositions are built. This route specifies the order in which the service execution engine will execute business process web services. Construction is completed by picking the most cost-effective web services to compose the solution. The TOPSIS is a method for ranking solutions in order of preference. Ideal Alternative and Negative Ideal Alternative are two hypotheses put forth here as artificial alternatives. The proposed model framework is shown in Figure 4.

The optimal solution is the one where all the benefits are maximized while the costs are minimal. The Positive Ideal Alternative has the best attribute values, and the Negative Ideal Alternative has the poorest. According to the TOPSIS, the best option is the one closest to the positive ideal solution and furthest from the negative one. This service has two main parts: a quality-of-service value converter and a cloud service ranker. Modeling the essential criteria, the interaction between various requirements, and mapping the customer's requirement to accessible quality-of-service criteria are all part of the quality-of-

service value conversion's purview. This research proposes Multi-Level Linked-Priority-based Best Method Selection using the Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) with Multistage User-Feedback-based Cloud Service Composition for the selection of better cloud services with the best cloud service composition for the cloud users that provide the best quality of service.

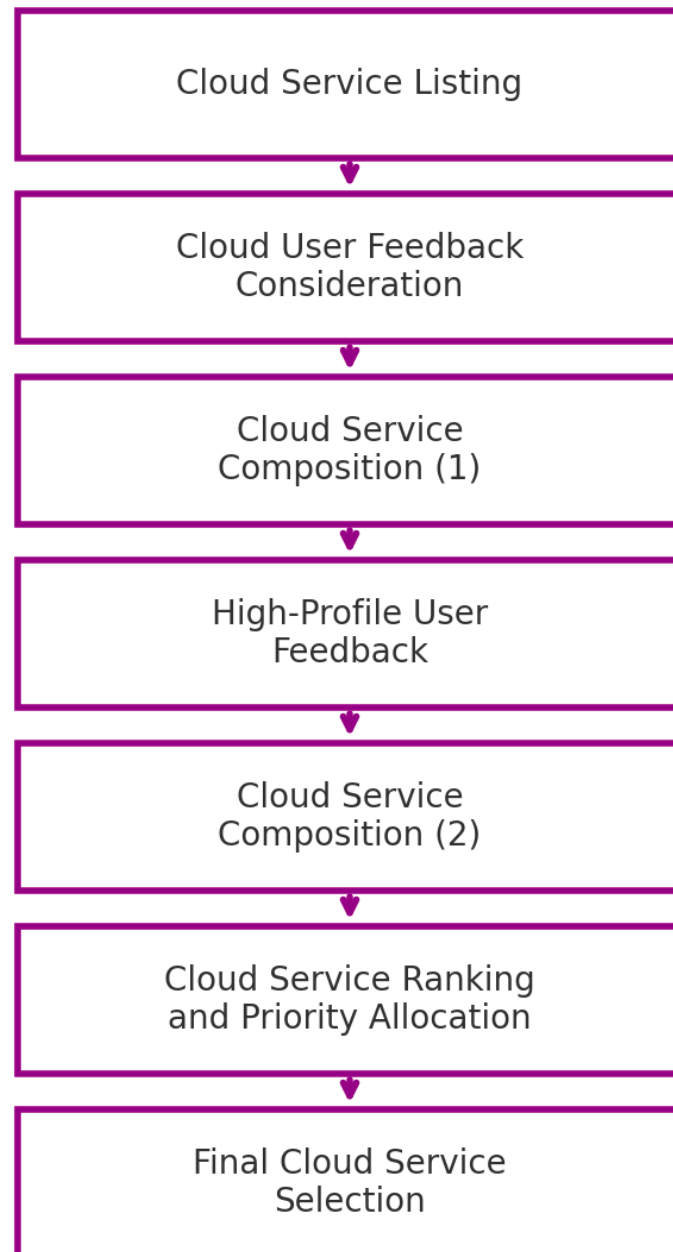


Figure 4. Proposed model framework.

4. MLLP-BMS-MUFCSC Algorithm

In the TOPSIS method, the weights w_j are determined through a combination of the Analytic Hierarchy Process (AHP) and the entropy weight method. First, an AHP hierarchy is developed, where the main objective—cloud service selection—is broken down into criteria and sub-criteria that represent different QoS attributes. Expert judgments are then used to build a pairwise comparison matrix, which provides the relative importance of each criterion. The consistency ratio (CR) is calculated to verify the reliability of these judgments. At the same time, the entropy weight method is applied to the dataset to

capture objective weights based on the variability of criteria values across services. Finally, the subjective weights from AHP and the objective weights from entropy are integrated through a combined weighting formula. This hybrid approach merges expert insights with data-driven evidence, resulting in balanced, reliable, and adaptable criteria weights for the TOPSIS evaluation.

High-profile users refer to a subset of users who are considered more influential or reliable in shaping service evaluations. These may include domain experts (with technical knowledge of cloud services), enterprise or VIP customers (whose usage has a higher business impact), or heavy/long-term users (who contribute extensive interaction data and feedback). Their feedback is assigned a higher weight in the proposed framework, since it is generally more informed, consistent, and representative of service quality in practical deployments. By distinguishing high-profile users from general users, the model ensures that critical insights are prioritized without disregarding broader user input.

The concept of “linked priority” refers to a dynamic mechanism where the priority of a cloud service is not fixed at a single evaluation stage but is continuously updated across multiple stages of the selection and composition process. Traditional AHP–TOPSIS approaches generally calculate QoS weights once and keep them constant, resulting in static rankings. Our framework, however, establishes a multi-level linkage, meaning that the outcome from one stage (such as service listing or user feedback aggregation) directly influences the recalculation of priorities in later stages (such as service composition or updating). At each stage, the priority of a service is recomputed as a weighted combination of its QoS parameters, user feedback, and contextual constraints, thereby refining the ranking as new information becomes available. This creates an adaptive feedback loop, ensuring service priorities remain responsive to system conditions and evolving user preferences. The novelty of our approach lies in integrating dynamic priority adjustment and multistage user feedback into the AHP–TOPSIS framework, enabling more realistic and user-centric service rankings compared with conventional static decision-making models.

In TOPSIS, the weights w_j are determined through a combination of the Analytic Hierarchy Process (AHP) and the entropy method. The process begins with developing an AHP hierarchy, where the main objective cloud service selection is broken down into criteria and sub-criteria that represent QoS attributes. Expert evaluations are used to create a pairwise comparison matrix, from which the relative importance of each criterion is derived, with the consistency ratio (CR) ensuring the reliability of these judgments. At the same time, the entropy method analyzes the dataset to generate objective weights that capture the variability of criteria values across services. Finally, the subjective weights from AHP and the objective weights from entropy are integrated, combining expert knowledge with data-driven insights to produce balanced and reliable criteria weights for TOPSIS. This hybrid weighting strategy allows the model to balance subjective and objective perspectives, producing more reliable and adaptive criteria weights for the subsequent TOPSIS evaluation.

The concept of multi-level linked priority means that service priorities are not assigned once and kept fixed but are dynamically updated at different stages of the service lifecycle, with the outcome of one stage feeding into the next. The process works as follows:

1. **Service Listing Stage:** Initial priorities are calculated using AHP–TOPSIS based on QoS parameters.
2. **User Feedback Stage:** These priorities are refined by incorporating aggregated user ratings, with greater weight given to high-profile or trusted users.
3. **Service Composition Stage:** When services are combined into workflows, the composition priorities inherit and build upon the updated service priorities from earlier stages.

4. **Updating Stage:** As services are added, removed, or modified, the priorities are recalculated dynamically, carrying forward both QoS and feedback information.
5. **Final Allocation Stage:** Overall priorities are consolidated to guide resource allocation and service selection, ensuring that the cumulative adjustments across all stages are reflected.

The key novelty of this approach is that priorities are linked across multiple levels, creating a cascading effect where updates in one stage (such as user feedback) directly influence the subsequent stages. This makes the ranking process adaptive and user-aware, unlike traditional AHP–TOPSIS methods that typically perform a single, static ranking without considering evolving information. Where the notations of an algorithm have been illustrated in Table 1.

Table 1. Notation table.

Symbol	Definition
$S = \{S_i\}_{i=1}^N$	Set of all available cloud services, where NN is the total number of services.
$S = \{U_i\}_{j=1}^M$	Set of all users, where MM is the total number of users.
$S = \{VM_k\}_{k=1}^P$	Set of available virtual machines (VMs), where PP is the total number of VMs.
$ID(s)$	Unique identifier of service s , including provider, type, and access info.
$q(s)$	Vector of quality-of-service (QoS) attributes for service s .
$\delta(s)$	Service utilization time (execution time or availability window).
$\beta(s)$	The maximum number of users that can access the service concurrently.
$\gamma(u, s)$	Usage time of service s by user u .
$r_{u,s}$	Rating provided by user u for service ss , normalized in $[0,1]$.
a_u	Access level of user u (11 = full, 00 = none, $0 < a_u < 10 < a_u < 1$ = partial).
$rep(u)$	Reputation or trust level of user u .
$U_{fb}(s)$	Aggregated user feedback score for service s .
$HU_{fb}(s)$	Aggregated feedback score from high-profile users for service s .
$P(s)$	Penalty score for service ss due to observed problems or failures.
$\rho(s)$	Reliability of service s .
$t(s)$	Response time of service s .
$c(s)$	Cost of service ss .
$\theta(s)$	Throughput of service s .
$\Omega(s)$	Functionality score of service s .
$ServComp(p)$	A service composition (workflow) pp consisting of multiple atomic services.
$Score(p)$	The final score of a service composition pp is computed using weighted QoS metrics and penalties.

Purpose: The purpose is to create a normalized list of all available services with consistent attributes.

Input: The input is the raw provider registry/service repository.

Output: This includes the canonical *ServiceList* = $\{rec(s)|s \in S\}$ where each record contains the id, type, QoS vector, and metadata.

Formalization:

Step 1: Define the service set.

Service List Definition:

$$ServiceList[N] = \{S_i | i = 1, 2, \dots, N\}$$

For each $s \in \text{ServiceList}$, create a record

$$\text{rec}(s) = \text{ID}(s), \text{type}(s), q(s), \delta(s), \text{providerID}(s), \text{location}(s))$$

where:

- S_i represents the i^{th} cloud service;
- N is the total number of available services.

Service Identifier:

$$\text{ServiceID}(s) = \text{getServID}(s) + \text{SType}(s) + \text{SAccess}(s)$$

where:

- $\text{getServID}(s) \rightarrow$ service identifier;
- $\text{SType}(s) \rightarrow$ type of service;
- $\text{SAccess}(s) \rightarrow$ access specification of the service.

User Feedback Modeling:

$$\begin{aligned} \text{Users}[M] &= \{U_j | j = 1, 2, \dots, M\} \\ U_j &= \text{getUID}(u) + V_{\text{access}}(u) + \gamma(u) \end{aligned}$$

where:

- $\text{getUID}(u) \rightarrow$ unique user ID;
- $V_{\text{access}}^{(u)} \rightarrow$ user access level ($0 \leq V_{\text{access}} \leq 1$);
- $\gamma(u) \rightarrow$ usage time of the service by user u .

Service Composition:

$$\text{ServComp}[M] = \max_{s \in \text{ServiceListmax}} (\text{U}_{\text{feedback}}(u) + \text{ServiceList}(s))$$

where:

- $\text{U}_{\text{feedback}}(u)$ represents user feedback on the service s ;
- Service composition is built by maximizing QoS metrics such as *reliability*(ρ) service utilization time (δ), and user ratings (τ).

High-Profile User Feedback:

$$\text{HU}_{\text{feedback}}[M] = \max(\gamma(u), \text{ServComp}(s), P(u))$$

where:

- $\gamma(u) \rightarrow$ access time;
- $P(u) \rightarrow$ observed problems from high-profile users.

Final Service Composition:

Implementation notes:

- Store $\text{rec}(s)$ in a hash map keyed by $\text{ID}(s)$ for $O(1)$ lookup.
- Normalize raw QoS values to $[0,1]$ using min-max or z-score per criterion (keep direction: invert cost criteria).
- Complexity: building a list is $O(N)$.

Step 2—Collect and aggregate user feedback (multistage normalization).

Purpose: The purpose is to compute a stable, normalized user feedback score per service for selection and composition.

Input: The input is the service list, raw user $Y_{u,s}$ ratings, usage times $Y_{u,s}$, and user reputation $rep(u)$.

Output: This includes aggregated user feedback $U_{fb}(s) \in [0, 1]$ and the category label (High, Medium, Low).

Formula:

1. Define a weight for each user u when considering service s :

$$w_{u,s} = \alpha \cdot a_u + \beta \cdot \frac{\gamma_{u,s}}{\max_{u' \in U_s} \gamma_{u',s}} + \zeta \cdot rep(u)$$

with $\alpha, \beta, \zeta \geq 0$ and $\alpha + \beta + \zeta = 1$.

2. Aggregate rating for service s :

$$U_{fb}(s) = \frac{\sum_{u \in U_s} w_{u,s} \cdot \gamma_{u,s}}{\sum_{u \in U_s} w_{u,s}}$$

where U_s is the set of users who rated s . If $U_s = \emptyset$, set $U_{fb}(s)$ to a neutral default (e.g., 0.5) or infer via similar services.

3. Map $U_{fb}(s)$ to categories:

$$label(s) = \begin{cases} High & U_{fb}(s) \geq \theta_H \\ Medium & \theta_M \leq U_{fb}(s) < \theta_H \\ Low & U_{fb}(s) < \theta_M \end{cases}$$

Step 3—Generate candidate compositions and score them (TOPSIS + composition aggregation).

This is the central step; I break it down into substeps.

3-A: Problem decomposition (functional mapping):

- Receive user functional requirement expressed as a workflow of K tasks:

$$T = [t_1, \dots, t_k]$$

- For each task t_k build a candidate set.

$$C_k = \{s \in S \mid type(s) = t_k \text{ and } s \text{ satisfies hard constraints}\}$$

Prune each C_k to the L top candidates using a local score (e.g., user feedback $U_{fb}(s)$ or simple weighted sum) to reduce search complexity.

3-B: Rank alternatives per task using the TOPSIS:

For each task k , apply the TOPSIS across the candidates C_k (standard, well defined):

1. Decision matrix. Suppose candidates for a single task are $A = \{0, 1 \dots m\}$ listed, and the criteria $j = 1..n$ are as follows: form a matrix $X = [x_{ij}]$ where x_{ij} is the raw (normalized) value of the criterion j for the alternative i .

2. Normalize:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^m x_{ij}^2}}$$

3. Weighted normalization: Let weights w_i, w_j be determined via AHP + entropy fusion; then,

$$v_{ij} = w_j \cdot r_{ij}$$

4. Positive/negative ideals:

$$v_j^+ = \begin{cases} \max_i v_{ij} & \text{if criterion } j \text{ is benefit,} \\ \min_i v_{ij} & \text{if criterion } j \text{ is cost,} \end{cases}$$

$$v_j^- = \begin{cases} \min_i v_{ij} & \text{if benefit,} \\ \max_i v_{ij} & \text{if cost,} \end{cases}$$

5. Distances:

$$D_i^+ = \sqrt{\sum_{j=1}^n (v_{ij} - v_j^+)^2}$$

$$D_i^- = \sqrt{\sum_{j=1}^n (v_{ij} - v_j^-)^2}$$

6. Closeness coefficient:

$$CC_i = \frac{D_i^-}{D_i^+ + D_i^-} \in [0, 1].$$

Rank candidates in descending CC_i order. Use top-K per task for combination search.

Step 4—Integrate high-profile user feedback (HU) and problem detection.

Purpose: The purpose is to weigh feedback from trusted/high-profile users more strongly and penalize services with reported issues.

Procedure:

1. Define HU set $H \subset U$ (users with high $rep(u)$ or special status).
2. Compute high-profile feedback $HU_{fb}(s)$ analogous to $U_{fb}(s)$ but restricted to H :

$$HU_{fb}(s) = \frac{\sum_{u \in H_s} rep(u) \cdot r_{u,s}}{\sum_{u \in H_s} rep(u)}$$

3. Compute problem score $P(s)$ from incident logs, e.g.,

$$P(s) = \frac{incidents_{last_{T(s)}}}{\max_{s'} incidents_{last_{T(s')}}} \in [0, 1]$$

4. Penalize composition scores:

$$Score_{final}(p) = Score(p) - \lambda \sum_{k=1}^k P(s_k)$$

where λ is chosen by policy.

Step 5—Produce final service compositions, priority, and update loop.

Purpose: The purpose is to publish final composition(s), service ranking, and priorities, then operate the multistage feedback loop that updates compositions over time.

Outputs:

- Final composition list $UServComp[M]$; top M compositions sorted by $Score_{final}(p)$
- Service priority score $\Pi(s)$ for each service:

$$\Pi(s) = \lambda_1 \cdot norm(CC_{service}(s)) + \lambda_2 \cdot U_{fb}(s) + \lambda_3 \cdot HU_{fb}(s) - \lambda_4 \cdot P(s)$$

where $CC_{service}(s)$ is an intrinsic TOPSIS closeness for service s across global criteria and $\sum \lambda_i = 1$

Dynamic update loop (multistage feedback):

1. Deploy the composition p^*
2. Collect runtime metrics and new user / HU feedback over a time window T
3. Recompute $U_{fb}, HU_{fb}, P(s)$
4. If $Score_{final}(p^*)$ drops below a threshold or an SLA violation is detected, trigger automatic recompositing (go back to Step 3) and rebind services.
5. Repeat periodically or with events.

$$UServComp[M] = \max \left(ServiceList(s), U_{feedback}(u), HU_{feedback}(u) \right) - P(u)$$

Computation of Criterion Weights using AHP–Entropy Fusion.

The criterion weights (w_j) are calculated by fusion of the AHP and the entropy method. The process begins with constructing an AHP hierarchy, where the goal of cloud service selection is decomposed into criteria and sub-criteria reflecting key quality-of-service attributes such as cost, reliability, response time, and throughput. Expert evaluations are then used to build a pairwise comparison matrix, from which subjective weights are derived and validated through the consistency ratio (CR). In parallel, the entropy method is applied to the performance data to generate objective weights that capture each criterion's variability and information content. To obtain the final weights, the subjective and objective components are combined using a defined fusion formula:

$$w_j = \alpha w_j^{AHP} + (1 - \alpha) w_j^{Entropy}$$

where $\alpha \in [0, 1]$, the balance between expert judgment and data-driven variability is controlled. This integrated weighting scheme ensures that the model benefits from human expertise and statistical objectivity, making the decision-making process more robust and transparent.

5. Experimental Setup

For Multi-Level Linked-Priority-based Best Method Selection with Multistage User-Feedback-driven Cloud Service Composition (MLLP-BMS-MUFCSC), a series of simulations was conducted in a controlled environment. The setup consisted of the elements described below.

5.1. Dataset and Service Repository

As no standardized benchmark dataset is currently available for large-scale cloud service composition, a synthetic dataset was generated to emulate realistic cloud environments. The dataset included 100 atomic cloud services, grouped into three categories: compute, storage, and application services. Each service was characterized by multiple quality-of-service (QoS) parameters, such as cost, response time, reliability, availability, throughput, and user ratings. To evaluate performance under realistic conditions, the simulation environment also incorporated 3,000 users submitting service requests and 50 virtual machines (VMs) serving as resource execution units. Both service utilization times and user ratings were modeled using probability distributions derived from prior studies, ensuring that the dataset reflects realistic variations in service performance and user behavior.

5.2. Simulation Environment

Experiments were conducted using the CloudSim 3.0.3 toolkit, which supports scalable simulation of cloud environments, on a system equipped with an Intel Core i7 processor (3.2 GHz), 16 GB RAM, and a Windows 11 operating system. Each experiment was repeated 30 times, and the average results are reported to reduce the effect of random variations. The competing models (NSCA-CBME and PIMCD-CCS) were implemented under the same conditions to ensure fairness in comparison.

5.3. Validation Protocol

The performance of the proposed model was assessed by comparing it against the baseline models NSCA-CBME and PIMCD-CCS across multiple evaluation dimensions, including service listing time, user feedback consideration accuracy, service composition accuracy, service updating, and priority ranking. By adopting this setup, the study ensured reproducibility and statistical robustness of the reported results.

5.4. Quality-of-Service (QoS) Parameters and Weighting

A multi-criteria decision-making framework drove the service selection process. Six QoS parameters were considered: reliability, response time, cost, availability, throughput, and user feedback. Weights were assigned using a hybrid approach that combined the Analytic Hierarchy Process (AHP) with entropy-based weighting to balance subjective expert judgment with objective data-driven variability. Table 2 summarizes the QoS parameters and their relative weightings.

Table 2. QoS parameters and weighting.

QoS Parameter	Type	Weight
Reliability	Benefit	0.30
Response Time	Cost	0.25
Cost	Cost	0.20
Availability	Benefit	0.10
Throughput	Benefit	0.10
User Feedback	Benefit	0.05

6. Results

As cloud computing becomes increasingly popular, network service providers are increasingly pressured to deliver a broader range of services, expanding functional capabilities and quality-of-service (QoS) offerings. To keep up with the ever-increasing demand for their services, cloud service providers must compete fiercely to give ever-better quality to their customers. Due to increasing provider competition, service selection and composition for delivering composite services in the cloud has become an NP-hard problem. This challenge underscores the need for research into several critical aspects: identifying the most suitable services from the available service pool, overcoming composition constraints, effectively evaluating the significance of different service parameters, addressing the ever-changing nature of the issue, and managing swift alterations in service and network attributes. Service composition is key to enhancing overall cloud performance in Internet-based and cloud computing environments. While each cloud platform offers distinct services, these can be integrated to function cohesively for the end user, and in many cases, fulfilling a single service request may involve multiple cloud environments working in coordination.

In cloud service composition has increasingly focused on Service Level Agreements (SLAs) and quality-of-service (QoS)-aware composition, reflecting the growing importance of non-functional requirements in service design. Existing methods typically aim to satisfy tenant QoS demands or adapt dynamically to fluctuations in service quality by generating execution plans that select providers with similar functionality but varying QoS attributes and cost constraints. However, many of these approaches overlook the fact that multi-tenant execution strategies must support diverse service plans, each tailored to different tenants' requirements in terms of features, QoS, and budget limitations. Within this context, the service composition engine plays a central role in designing and executing these service plans. In our work, the Technique for Order of Preference by Similarity to the Ideal Solution (TOPSIS) is applied using derived criteria weights to rank cloud service providers (CSPs) effectively. To validate the approach, we present a case study for selecting the most suitable cloud database server, supported by a detailed analysis. Building on this foundation, we introduce the Multi-Level Linked Priority-based Best Method Selection with Multistage User Feedback-driven Cloud Service Composition (MLLP-BMS-MUFCSC) framework, designed to identify the most appropriate cloud services and generate optimal compositions while ensuring high QoS for cloud users. The proposed model is rigorously compared against two established baselines—NSCA-CBME (Novel Service Composition Algorithm for Cloud-Based Manufacturing Environment) and PIMCD-CCS (Practical Integrated Multi-Criteria Decision-Making Scheme for Selecting Cloud Services in Cloud Systems)—and the results demonstrate that our approach achieves superior performance in both service selection and composition.

The cloud service listing time refers to the average time required by the system to identify, filter, and present a list of available services to a user request, based on the predefined QoS parameters and user constraints. It is not simply network latency but a broader measure that captures the processing overhead of service discovery, ranking, and listing within the simulation environment. Lower listing times indicate that the framework can more efficiently handle user requests and generate service recommendations, improving responsiveness in large-scale cloud environments.

Services provided from the cloud refer to any software and hardware accessed through the Internet. With the contractual arrangements between subscribers and third-party suppliers, customers can use robust computing capabilities without the burden of acquiring and maintaining their own hardware and software. The cloud service provider (CSP) compiles a list of cloud services that can be utilized to create cloud service compositions. A comparison of the cloud service listing time levels between the suggested model and current methodologies is shown in Table 3 and depicted in Figure 5.

Table 3. Cloud service listing time levels.

No. of Users Considered	Models Considered		
	Models Considered	Models Considered	Models Considered
500	13	16.8	14
1000	13.3	17	14.4
1500	13.4	17.2	14.7
2000	13.6	17.5	15
2500	13.8	17.8	15.5
3000	14	18	16

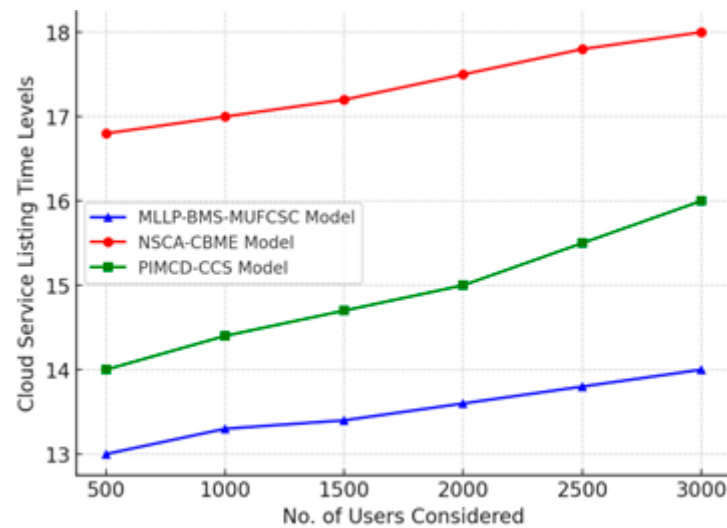


Figure 5. Cloud service listing time levels.

Table 4 and Figure 6 present the cloud service listing time levels for different numbers of users across the three compared models. The values are measured in **seconds**, representing the average time required to list the available services. It can be observed that the proposed MLLP-BMS-MUFCSC model consistently achieves lower listing times compared with the NSCA-CBME and PIMCD-CCS models. For example, when the number of users increases to 3000, the listing time of the proposed model remains at 14 s, whereas the NSCA-CBME and PIMCD-CCS models require 18 s and 16 s, respectively. This demonstrates that the proposed framework is more efficient in handling a growing number of user requests, ensuring faster service discovery and reduced system overhead.

Table 5, illustrated in Figure 7, shows the cloud user feedback consideration accuracy levels for different numbers of users. The values are expressed as a percentage (%), indicating how effectively each model incorporates user feedback into the service evaluation process. The results demonstrate that the proposed MLLP-BMS-MUFCSC model consistently achieves higher accuracy than the baseline models. For instance, for 3000 users, the proposed model achieves an accuracy of 97.8%, while the NSCA-CBME and PIMCD-CCS models achieve only 93% and 95%, respectively. This improvement highlights the robustness of the proposed framework in effectively integrating user opinions and feedback, even as the number of users grows. Such efficiency ensures that service selection decisions are more reliable, user-centric, and scalable in dynamic cloud environments.

Table 4. Cloud user feedback consideration accuracy levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	96.5	91.5	93.4
1000	97	91.8	93.8
1500	97.2	92	94.1
2000	97.4	92.2	94.3
2500	97.5	92.4	94.6
3000	97.8	93	95

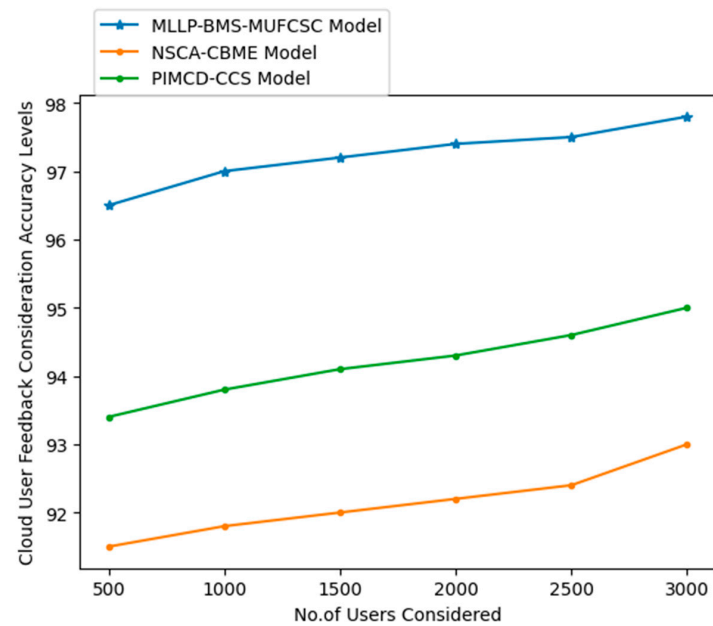


Figure 6. Cloud user feedback consideration accuracy levels.

Table 5. Cloud service composition generation accuracy levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	97.2	95.7	93.5
1000	97.4	96	93.7
1500	97.6	96.2	94.1
2000	97.9	96.5	94.3
2500	98.1	96.8	94.6
3000	98.2	97	94.8

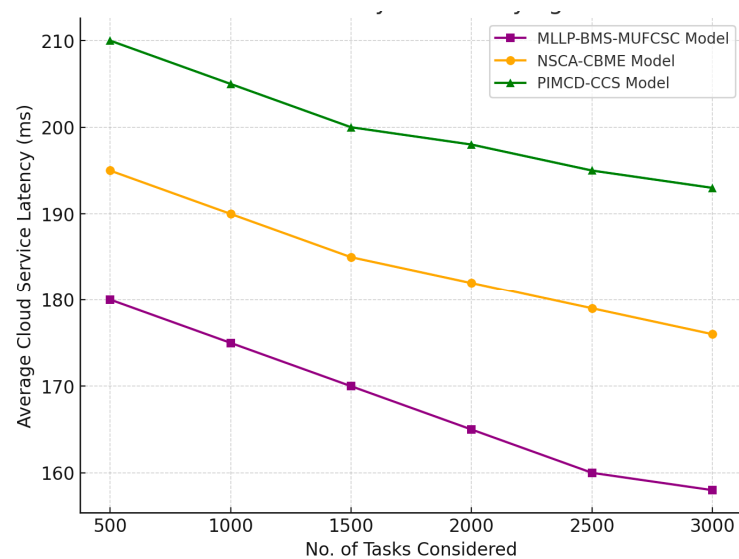


Figure 7. Cloud service composition generation accuracy levels.

Table 6 and Figure 8 present the cloud service composition accuracy levels for different numbers of users, with values expressed as a percentage (%). The results indicate that

the proposed MLLP-BMS-MUFCSC model consistently achieves higher accuracy than the baseline models (NSCA-CBME and PIMCD-CCS). For instance, at 3000 users, the proposed model reaches an accuracy of 98.2%, compared to 97% for NSCA-CBME and only 94.8% for PIMCD-CCS. This demonstrates the superior capability of the proposed framework in accurately composing services, even under increasing workloads. The stable improvement across all user levels highlights the model's scalability and reliability, ensuring service compositions remain effective and optimized in dynamic cloud environments.

Table 6. High-level user feedback consideration time levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	14.8	22.8	13
1000	15.1	23	14
1500	15.2	23.1	15
2000	15.4	23.5	16
2500	15.7	23.7	17
3000	16	24	18

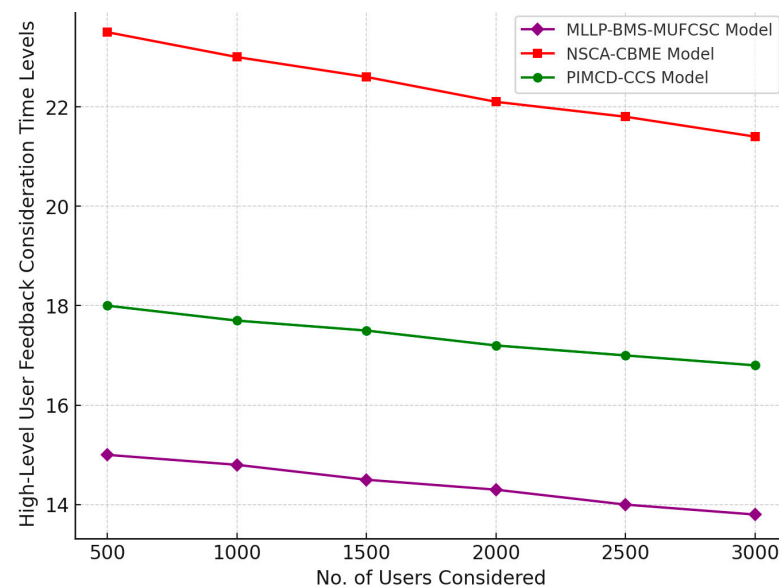


Figure 8. High-level user feedback consideration time levels.

Table 7, illustrated in Figure 9, shows the cloud service updating levels for different numbers of users, with values measured in seconds (s). The results highlight that the proposed MLLP-BMS-MUFCSC model achieves significantly lower updating times than the NSCA-CBME model while maintaining competitive performance with the PIMCD-CCS model. For example, when the number of users reaches 3000, the proposed model records an updating time of only 16 s, whereas NSCA-CBME requires 24 s and PIMCD-CCS requires 18 s. This indicates that the proposed framework is more efficient at dynamically updating services as user demands scale, thereby reducing latency and improving responsiveness in real-time cloud environments.

Table 7. Cloud service composition updating accuracy levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	97.3	91	92.4
1000	97.8	92	93
1500	98	92.2	93.2
2000	98.2	92.4	93.4
2500	98.3	92.6	93.6
3000	98.5	93	94

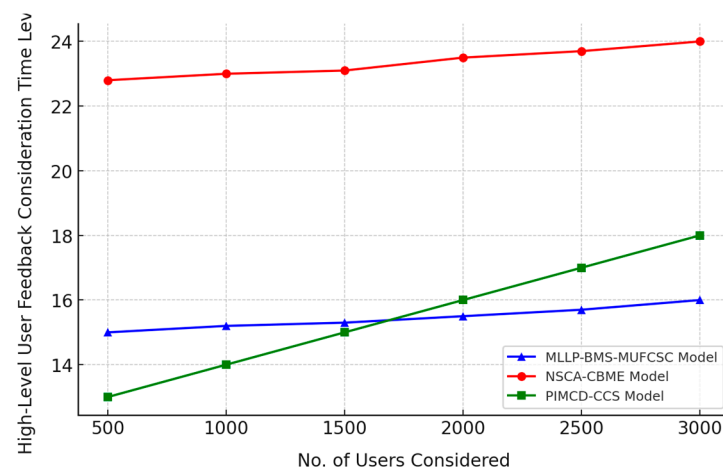
**Figure 9.** Cloud service composition updating accuracy levels.

Table 8, depicted in Figure 10, presents the cloud service ranking accuracy levels for different numbers of users, with values expressed as a percentage (%). The results show that the proposed MLLP-BMS-MUFCSC model consistently outperforms the baseline models (NSCA-CBME and PIMCD-CCS) in accurately ranking services according to quality of service (QoS) and user feedback. For instance, at 3000 users, the proposed model achieves an accuracy of 98.5%, compared with 93% for NSCA-CBME and 94% for PIMCD-CCS. This steady improvement across all user levels highlights the robustness and scalability of the proposed framework, ensuring that services are ranked more reliably even as system demands increase. Accurate ranking is critical for enhancing decision making in service selection, and these results demonstrate that the proposed model provides a significant advantage over existing approaches.

Table 8. Cloud service ranking time levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	97.3	91	92.4
1000	97.8	92	93
1500	98	92.2	93.2
2000	98.2	92.4	93.4
2500	98.3	92.6	93.6
3000	98.5	93	94

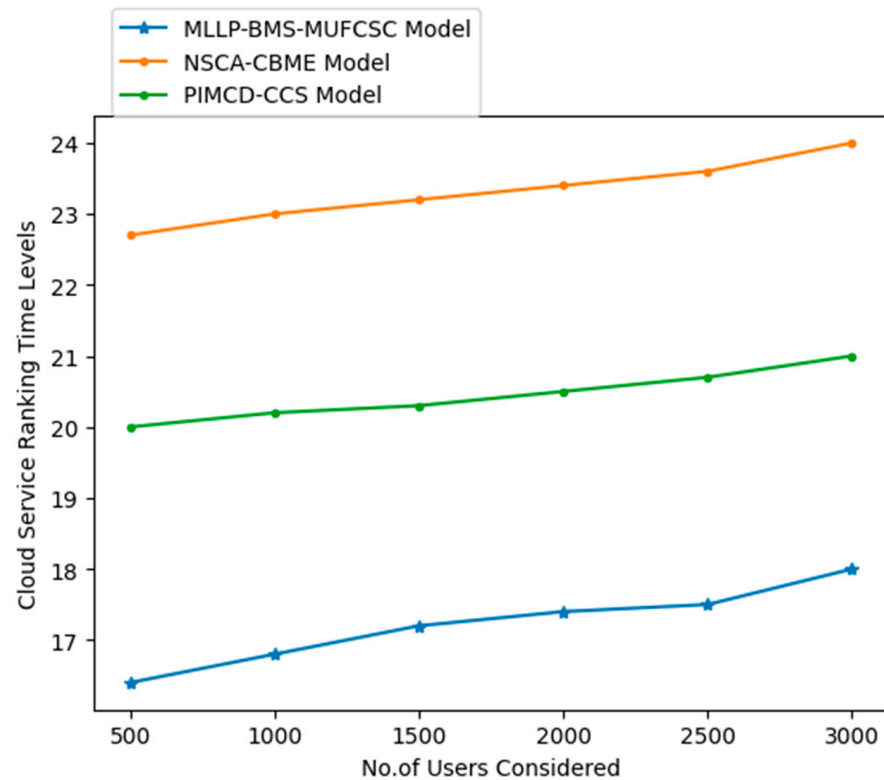


Figure 10. Cloud service ranking time levels.

Table 9, illustrated in Figure 11, presents the cloud service priority allocation levels for different numbers of users, expressed as a percentage (%). The results clearly indicate that the proposed MLLP-BMS-MUFCSC model consistently achieves higher priority allocation accuracy than the baseline models (NSCA-CBME and PIMCD-CCS). For example, when the number of users reaches 3000, the proposed model records a priority allocation accuracy of 98.5%, whereas NSCA-CBME and PIMCD-CCS achieve only 93% and 94%, respectively. This demonstrates the ability of the proposed framework to more effectively assign service priorities based on both QoS parameters and user feedback. Accurate priority allocation ensures that the most critical and reliable services are readily available to users, improving system efficiency and user satisfaction in large-scale cloud environments.

Table 9. Priority allocation accuracy levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	96.8	87.7	92.3
1000	97.1	88	92.7
1500	97.3	88.3	93
2000	97.5	88.5	93.2
2500	97.7	88.7	93.6
3000	98	89	94

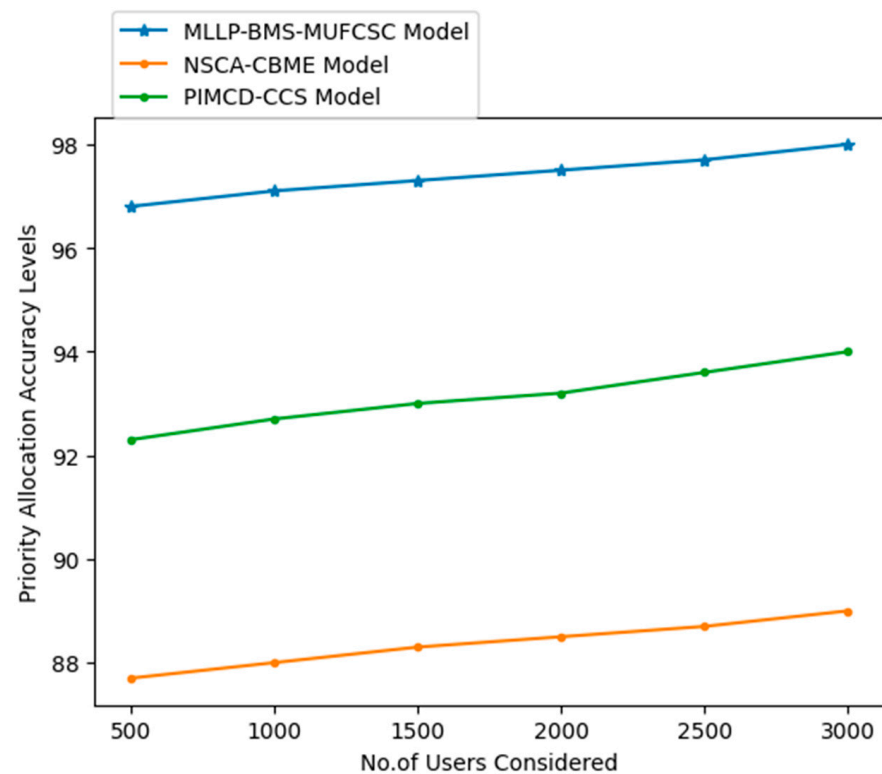


Figure 11. Priority allocation accuracy levels.

Table 10 and Figure 12 illustrate the overall performance improvement levels for different numbers of users, with values expressed as a percentage (%). The results demonstrate that the proposed MLLP-BMS-MUFCSC model consistently delivers superior performance compared with the baseline models (NSCA-CBME and PIMCD-CCS). For instance, at 3000 users, the proposed model achieves an overall performance improvement of 98%, while NSCA-CBME and PIMCD-CCS reach only 89% and 94%, respectively. This consistent margin of improvement highlights the effectiveness of the proposed framework in optimizing cloud service operations under increasing workloads. The model ensures reliability and adaptability by integrating QoS metrics with user feedback, making it highly suitable for real-world large-scale cloud environments where performance and scalability are critical.

Table 10. TOPSIS-based priority ranking updating accuracy levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	96.8	87.7	92.3
1000	97.1	88	92.7
1500	97.3	88.3	93
2000	97.5	88.5	93.2
2500	97.7	88.7	93.6
3000	98	89	94

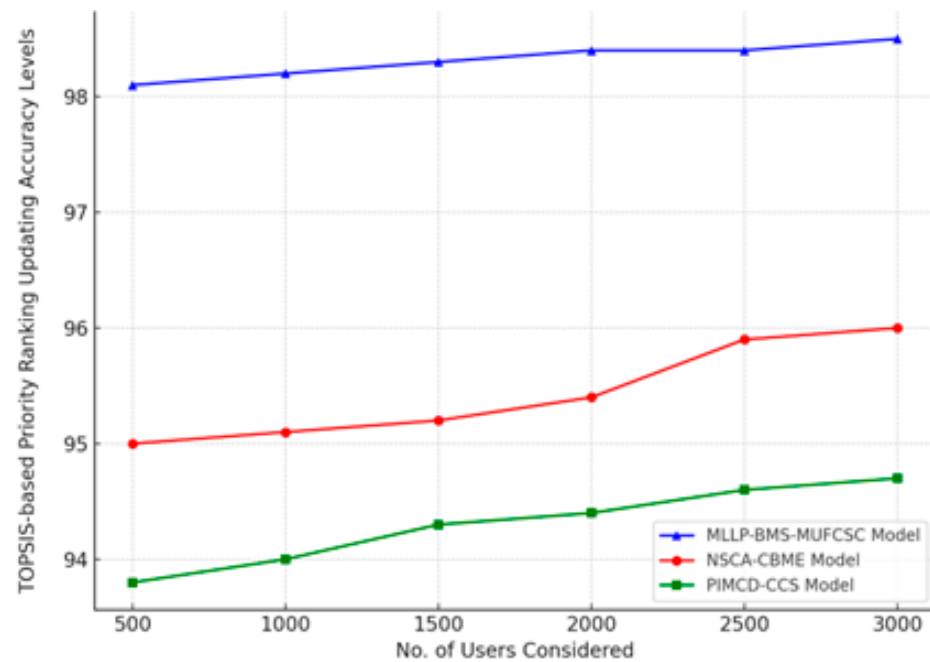


Figure 12. TOPSIS-based priority ranking updating accuracy levels.

The accuracy levels of priority ranking updates based on the TOPSIS method are presented for both the existing models and the proposed model.

Table 11, illustrated in Figure 13, presents the overall performance improvement levels for varying numbers of users, expressed as a percentage (%). The data clearly show that the proposed MLLP-BMS-MUFCSC model consistently outperforms the baseline models (NSCA-CBME and PIMCD-CCS) as the number of users increases. For example, at 3000 users, the proposed model achieves an overall performance of 98%, compared with 89% for NSCA-CBME and 94% for PIMCD-CCS. This consistent improvement demonstrates the scalability and robustness of the proposed framework in handling large-scale cloud environments. Integrating QoS parameters with user feedback allows the model to adapt more effectively to dynamic workloads, ensuring higher system efficiency, better service reliability, and improved user satisfaction.

Table 11. Multi-Level linked service selection accuracy levels.

No. of Users Considered	Models Considered		
	MLLP-BMS-MUFCSC Model	NSCA-CBME Model	PIMCD-CCS Model
500	96.8	87.7	92.3
1000	97.1	88	92.7
1500	97.3	88.3	93
2000	97.5	88.5	93.2
2500	97.7	88.7	93.6
3000	98	89	94

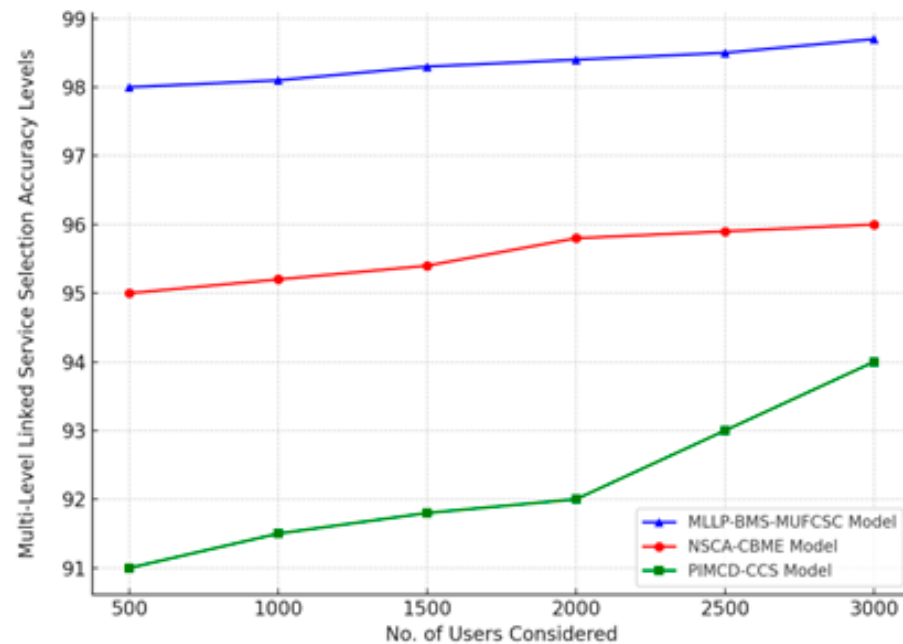


Figure 13. Multi-level linked service selection accuracy levels.

7. Conclusions

Recent developments in information and communication technology (ICT) have led to a significant shift of commercial services to the cloud, increasing demand for these services. The exponential growth in CSPs results from the skyrocketing demand for these services. On the other hand, this has created challenges for cloud users in selecting the most suitable service provider for their specific business requirements. The current methods for dynamically composing services are usually insufficient when dealing with multitenant execution plans. This is because they are not inherently made to supply tenants with individualized plans that meet their specific needs regarding features, quality of service, and budget.

Beyond demonstrating methodological improvements, the findings of this study also have practical implications for both cloud service providers (CSPs) and end users. For CSPs, the proposed framework offers a structured mechanism to evaluate, rank, and prioritize their services based on objective QoS metrics and dynamic user feedback, enabling them to identify performance gaps and enhance competitiveness in the marketplace. The framework functions as a decision-support tool for cloud users that simplifies the complex process of selecting and composing services, reducing the likelihood of service mismatches and ensuring higher satisfaction and efficiency. By linking algorithmic accuracy with tangible benefits for providers and consumers, the work contributes to advancing research in cloud service optimization and supporting more reliable and user-centric cloud ecosystems.

Traditional service composition methods are often unsuitable for real-time, adaptive, and dynamic composition or recomposition scenarios, as they struggle to scale effectively with the increasing number of tenants, their diverse features, QoS requirements, and associated costs. Since cloud service composition is widely recognized as an NP-hard problem, this research introduces a novel solution: the Multi-Level Linked-Priority-based Best Method Selection (MLLP-BMS) framework, which integrates TOPSIS with Multistage User Feedback. This framework enables the selection of optimal cloud services and the construction of high-quality compositions that meet user and provider requirements. In cases where failures occur during the execution of a composite service, the proposed model can dynamically re-compose the execution plan to ensure the required QoS is maintained.

In practice, cloud service providers continuously diversify their offerings to remain competitive, which places the responsibility on consumers to evaluate and choose providers that best match their needs. However, this evaluation process can be time-consuming and complex, given the large number of services and attributes to consider. The TOPSIS method addresses this challenge by systematically ranking service providers based on a detailed examination of their features and performance. For enterprises migrating from on-premises architectures to cloud environments, TOPSIS can serve as a valuable decision-support tool to identify the cloud setup that best aligns with their requirements.

Through this comprehensive research framework, cloud users are empowered to make well-informed decisions among competing providers and service features. Experimental results confirm the superiority of the proposed model: it achieves 98% accuracy in cloud service composition (Table 3, Figure 7) and 98.5% accuracy in best service selection (Table 9, Figure 13), thereby outperforming baseline models in both effectiveness and reliability.

The proposed MLLP-BMS-MUFCSC model has demonstrated superior performance over existing approaches regarding service selection and composition accuracy, highlighting its effectiveness in balancing QoS-driven ranking with multistage user feedback. At the same time, several challenges and limitations remain. First, the current evaluation has been conducted in a simulation environment, and further testing is required to validate its robustness under real-time, large-scale cloud deployments with dynamic workloads. Second, while the framework scales well in controlled scenarios, managing huge numbers of users and services in production platforms may introduce additional complexity. Third, the reliance on user feedback makes the model vulnerable to the cold-start problem, where new services lack sufficient ratings to be fairly evaluated.

Most importantly, the existing version treats services as largely independent entities, whereas in practice, cross-layer dependencies (e.g., compute, storage, database, and networking services tied to a single provider) constrain feasible service compositions. To address these issues, future work will focus on (i) incorporating dependency-aware mechanisms to better capture provider-specific compatibility and cross-layer constraints in multi-cloud and hybrid settings, (ii) developing strategies such as trust-based weighting and predictive modeling to mitigate the cold-start problem, (iii) evaluating the framework under real-time and large-scale workloads to test scalability and robustness, and (iv) exploring metaheuristic optimization techniques to enhance efficiency in service selection and composition further. Additionally, the framework can be extended to leverage multi-cloud platforms as a central hub, enabling users to access more dynamic and diversified service options across providers rather than depending on a single CSP. These directions will strengthen the proposed model's adaptability and practicality and extend its applicability to a broader range of real-world cloud computing scenarios.

Author Contributions: V.N.V.L.S.S.—original draft preparation, formula analysis, visualization, Investigation and funding acquisition. G.S.K.—Conceptualization, validation, Supervision, review and editing. A.V.V.—Methodology, Data curation, Supervision and Project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: Authors have no conflict of interest.

References

1. Makwe, A. Cloud service prioritization using a multi-criteria decision framework with INS and TOPSIS. *J. Cloud Comput. Adv. Syst. Appl.* **2024**, *13*, 1–15. [\[CrossRef\]](#)
2. Ali, M.; Rehman, S.; Khan, A. A new integrated approach for cloud service composition. *Comput. Intell. Neurosci.* **2024**, *2024*, 3136546. [\[CrossRef\]](#)
3. Vaquero, L.M.; Rodero-Merino, L.; Caceres, J.; Lindner, M. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *39*, 50–55.
4. Li, A.; Yang, X.; Kandula, S.; Zhang, M. CloudCmp: Comparing public cloud providers. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, Melbourne, Australia, 1–30 November 2010; pp. 1–14.
5. Zhang, W.; Zulkernine, M. SOCCA: A Service-Oriented Cloud Computing Architecture. In Proceedings of the 5th IEEE World Congress on Services, Los Angeles, CA, USA, 6 July 2009; pp. 607–610.
6. Wang, L.; Von Laszewski, G.; Younge, A.; He, X.; Kunze, M.; Tao, J.; Fu, C. Cloud Computing: A Perspective Study. *New Gener. Comput.* **2010**, *28*, 137–146. [\[CrossRef\]](#)
7. Yu, H.; Buyya, R. A taxonomy of cloud computing systems. In Proceedings of the 2009 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, Seoul, Republic of Korea, 25–27 August 2009; pp. 44–51.
8. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [\[CrossRef\]](#)
9. Corradi, A.; Fanelli, M.; Foschini, L. Mobile Cloud Computing: Technologies, Applications, and Challenges. *IEEE Commun. Mag.* **2013**, *51*, 26–33.
10. Li, Z.; Sizov, S. Dynamic service composition for cloud computing: A survey. In Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology, Melbourne, Australia, 14–17 December 2021; pp. 324–331S.
11. Khurana, H.; Cosenza, E. Enterprise service bus for cloud computing. In Proceedings of the International Conference on Cloud and Service Computing, Hong Kong, China, 12–14 December 2011; pp. 142–150.
12. Zeng, L.; Benattallah, B.; Dustdar, S.; Georgakopoulos, D. QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng.* **2004**, *30*, 311–327. [\[CrossRef\]](#)
13. Papazoglou, M.P.; van den Heuvel, W.J. Service-oriented architectures: Approaches, technologies and research issues. *VLDB J.* **2007**, *16*, 389–415. [\[CrossRef\]](#)
14. Baresi, L.; Guinea, S.; Heckel, R.; Inverardi, P. An aspect-based approach to the development of service-oriented architectures. *Softw. Syst. Model.* **2010**, *9*, 131–149.
15. Marin, O.; Paramartha, A. A taxonomy of service composition in cloud computing. *J. Cloud Comput. Adv. Syst. Appl.* **2014**, *3*, 1–13.
16. Mont, M.C.; Bath, C. Service composition for the cloud: A survey. *ACM Comput. Surv.* **2015**, *in press*.
17. Jain, R.; Paul, S. Network virtualization and software defined networking for cloud computing: A survey. *IEEE Commun. Mag.* **2013**, *51*, 24–31. [\[CrossRef\]](#)
18. Issarny, V.; Karsai, G. A dynamic orchestration architecture for composition of web-services. In Proceedings of the Third IEEE International Conference on Service-Oriented Computing and Applications, Perth, Australia, 13–15 December 2010; pp. 76–83.
19. Li, B.; Mao, B.; Wang, J. An ontology-based approach to cloud service composition. In Proceedings of the 2012 IEEE Ninth International Conference on Services Computing, Honolulu, HI, USA, 24–29 June 2012; pp. 201–208.
20. Mayer, R.; Kunze, M. Adaptive trust-based service composition in clouds. In Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 28 June–3 July 2013; pp. 145–152.
21. Zorlu, M.; Pavlou, G.; De, K. Adaptive QoS-based service selection and composition for cloud services. *IEEE Trans. Serv. Comput.* **2014**, *7*, 206–219.
22. Gajendra, M.; Chana, I. QoS-aware dynamic service composition for the cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 1414–1423.
23. Park, K.-H.; Sandhu, R. Cloud service composition based on usage patterns. In Proceedings of the 2014 IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore, India, 15–17 October 2014; pp. 98–106.
24. Zhang, S.; Cheng, P.; Mohapatra, P. ScienceCloud: A cloud-based scientific workflow system. *IEEE Cloud Comput.* **2014**, *1*, 44–48.
25. Li, H.; Li, X. Policy-driven service composition in clouds. In Proceedings of the 2015 IEEE Ninth International Conference on Service-Oriented Computing and Applications, Cambridge, MA, USA, 21–25 September 2015; pp. 58–65.
26. Nunes, B.A.A.; Mendonca, M.; Nguyen, X.N.; Obraczka, K.; Turletti, T. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1617–1634.

27. Wu, M.; Buyya, R. Cloud service composition: A systematic literature review. *ACM Comput. Surv.* **2018**, *51*, 80.
28. Jrad, Z.; Qureshi, S.; Krishnan, R. Evaluating quality of service in cloud service selection: A multi-criteria decision-making framework. *IEEE Trans. Cloud Comput.* **2019**, *7*, 395–407.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.