# A Service Replacement Method Integrating Service Clustering and Collaborative Similarity (Postprint)

**Authors:** Tian Yuqing, Peng Fei, Wang Huadong, Hu Qiang

**Date:** 2022-05-10T00:00:00+00:00

## Abstract

Current service replacement methods predominantly focus on matching replacement services with failed services at the functional and quality levels, while lacking consideration of their collaborative relationships. To address this issue, this paper proposes a service replacement method that integrates service clustering and collaborative relationships. First, a service collaboration graph is constructed using the compositional relationships between services, sequence sampling is performed based on Node2Vec to generate collaboration vectors, and collaboration similarity between services is subsequently calculated. Then, by introducing service clusters to reduce the service search space during replacement, a candidate replacement service set is rapidly constructed from functional and quality perspectives. Finally, within the candidate replacement service set, optimal recommendation of replaceable services is achieved by comprehensively integrating service quality scores and collaboration similarity. Experimental results demonstrate that the proposed method can effectively improve the efficiency and rationality of service replacement.

## Full Text

## Preamble

**Vol. 39 No. 9**
**Application Research of Computers**
**Accepted Paper**

**A Service Substitution Method Combining Service Clustering and Collaboration Similarity**

**Tian Yuqing, Peng Fei, Wang Huadong†, Hu Qiang**
(College of Information Science & Technology, Qingdao University of Science &

Technology, Qingdao, Shandong 266061, China)

**Abstract:** Current service substitution methods primarily focus on functional and quality matching between substitutive and invalid services, yet lack consideration of their collaborative relationships. To address this limitation, this paper proposes a service substitution method that integrates service clustering and collaboration relationships. First, we construct a service collaboration graph based on service composition relationships, perform sequence sampling using Node2Vec to generate collaboration vectors, and subsequently compute collaboration similarity between services. Then, by introducing service clusters to reduce the search space during substitution, we rapidly construct candidate substitutive service sets from functional and quality perspectives. Finally, within the candidate set, we achieve optimal substitutive service recommendation by comprehensively evaluating service quality scores and collaboration similarity. Experiments demonstrate that the proposed method effectively improves both the efficiency and rationality of service substitution.

**Keywords:** service clustering; collaboration similarity; service substitution; process co-occurrence

## 0 Introduction

With the maturation of next-generation communication technologies such as cloud computing, IoT, and 5G, the number of services on networks has increased rapidly, leading to widespread adoption of Service-Oriented Architecture (SOA) in software development and deployment. Different types of services are integrated into various business systems. To facilitate invocation, services are typically designed as network applications with fine-grained functionality. For complex service requests, service composition is employed to construct service processes from groups of functionally related services that collaborate to complete responses.

Services across the internet, IoT, and various ad-hoc networks operate in dynamic and volatile network environments that may interrupt their working states. To provide higher quality and more competitive services, platforms employ numerous incentive mechanisms to drive continuous functional evolution and version updates. Consequently, both the complex objective network environment and subjective functional evolution needs can cause service failures, creating demands for service substitution and migration in existing business systems. How to efficiently and accurately achieve service substitution represents a critical challenge in SOA-based business system development and deployment.

Existing research on service substitution primarily considers service matching from functional and quality perspectives, with contributions concentrated on computing functional similarity and QoS-based optimization. Due to the vast number of services on networks, current methods face enormous search

spaces when finding functionally similar services, resulting in complex and time-consuming substitution processes. Moreover, existing service discovery methods neglect collaborative factors. After obtaining candidate services that can functionally replace invalid services, they typically select the service with optimal QoS as the substitute. Such approaches focus on local optimization of service quality at the replacement position while ignoring collaborative relationships between the substitute service and other services in the business process context, lacking global evaluation of collaborative compatibility between the substitute and the business environment.

Given that current service substitution methods generally suffer from large search spaces, low efficiency, and insufficient consideration of collaborative adaptation between substitute services and business process contexts, this paper proposes a service substitution method integrating service clustering and collaboration similarity. The main contributions are:

1) We construct a service collaboration graph and propose a Node2Vec-based service collaboration similarity computation method, which enhances substitution rationality by incorporating collaboration similarity between substitute and invalid services.

2) We establish a service cluster-oriented request response pattern that reduces search space through service clusters, improving the efficiency of candidate service set discovery during substitution.

3) We conduct simulation experiments to validate the proposed method's effectiveness from two perspectives: substitution discovery time and co-occurrence rate between substitute services and other services in the business process containing the invalid service.

---

## 1 Related Work

Service substitution is an important research problem in service computing. Researchers first consider functional compatibility between substitute and invalid services, then incorporate factors such as quality evaluation, discovery efficiency, interface data consistency, and composition preferences to propose various substitution methods.

For instance, Liang et al. used term co-occurrence in service description markup to classify services and established service operation interface compatibility and substitution methods, achieving 85% substitution accuracy. However, this method still faces significant challenges in identifying semantic heterogeneity of alternative services. Zhang mined Web service execution context characteristic rules to qualitatively represent different Web service performance under varying execution contexts, employing knowledge discovery-based methods to select optimal candidate substitute services according to current conditions. Gao proposed a data consistency checking method for dynamic service substitution that

uses data replacement patterns to regulate different substitution behaviors, validating its effectiveness through case analysis and experiments.

Santhanam used preference networks to represent and reason about non-functional attribute preferences in service requirements for substitute selection. Sara treated Web services as sets of operations, transforming service substitution into operation-level substitution. By constructing a network of existing service operations and defining four types of operation interface matching degrees, this approach significantly improved substitution efficiency compared to direct Web service substitution. Du et al. proposed a service substitution method based on service cluster nets that improved discovery efficiency through service clusters, though the requirement for consistent parameter interfaces limited search space reduction and substitution flexibility. Hu et al. proposed a process collaboration-aware service substitution method that enhanced substitution rationality through process collaboration, but suffered from incomplete collection of process collaboration dependencies, limiting the accuracy of computed collaboration strength.

Service substitution essentially represents service rediscovery based on the invalid service's functionality. Therefore, service discovery achievements provide valuable references. Service discovery efficiency and accuracy are two key metrics for evaluation and improvement. Researchers typically employ clustering, bipartite graphs, and indexing methods to enhance retrieval mechanisms and discovery speed, with clustering being most widely used for effectively reducing search space. Service discovery accuracy primarily depends on similarity determination between requirements and services. Early research extracted keywords representing service features from description documents (e.g., WSDL) and computed similarity through term frequency or semantic similarity.

Recently, as natural language text becomes increasingly common for service descriptions, topic models and deep learning methods have become mainstream for extracting service topic features from description text and determining similarity based on topic feature vectors. For example, Nabli et al. combined TF-IDF with LDA models to propose a topic vector generation model for cloud service similarity computation, improving discovery accuracy during service crawling. Hu et al. used the GSDMM model to generate service representation vectors with probability correction factors, enhancing similarity computation precision in service discovery and clustering, and experimentally demonstrated GSDMM's superior topic expression capability compared to LDA, BTM, and other models. Lizarralde used variational autoencoders to learn features from service descriptions, constraining encoding representations to model latent variables. This method outperformed traditional topic models like LDA and LSA but had significantly higher complexity for feature learning and model training.

To further improve service substitution efficiency and rationality, this paper introduces service clustering, the GSDMM topic model, and collaboration similarity into service substitution. Service clustering reduces search space during substitution, while GSDMM generates topic vectors from service descriptions to

improve similarity computation accuracy and substitution matching precision. By modeling process transition dependencies between services, we construct a Node2Vec-based service collaboration similarity computation method that comprehensively considers service quality and collaboration similarity to enhance substitution recommendation rationality.

## 2 Service Collaboration Similarity

Social network theory research shows that when multiple participants collaborate on a task, they prefer to work with familiar partners or those with previous collaboration experience. In service composition scenarios, users similarly prefer services with historical composition relationships to build new business processes, as these services demonstrate higher reliability in interface compatibility and business deployment. Therefore, among candidate substitute services, selecting services with similar collaboration partners to the invalid service increases substitution rationality.

To measure whether services share similar collaboration relationships, we propose the concept of service collaboration similarity, which uses similarity magnitude to measure the probability of two services appearing in similar composition service process contexts. Collaboration similarity depends on two factors: service co-occurrence rate and process distance. The more frequently two services appear in similar service processes, the greater their collaboration similarity; the closer their process distance to the same service, the greater their collaboration similarity.

To compute collaboration similarity between two services, we construct a service collaboration graph as an undirected graph where nodes represent services and edges represent collaboration relationships. Numerous service process models are accumulated on cloud platforms. By traversing these models, services are abstracted as nodes in the collaboration graph, and service transition dependencies are mapped as edges to complete graph construction.

**Definition 1 (Collaboration Dependency):** In a service composition process model $sp$, if service $s_i$ and service $s_j$ have business logic transitions, they are called collaboration dependencies, denoted as $s_i \rightarrow s_j$.

**Definition 2 (Service Collaboration Graph):** The service collaboration graph is defined as an undirected weighted graph $SCG = (V, E)$ where: - $V = \{v_1, v_2, v_3, \dots, v_n\}$ is the set of service nodes, with each node $v_i$ representing a service; - $E = \{e = (v_i, v_j) \mid 1 \leq i, j \leq n\}$ is the set of collaboration edges, where $e = (v_i, v_j)$ indicates that services $s_i$ and $s_j$ corresponding to nodes $v_i$ and $v_j$ satisfy $s_i \rightarrow s_j$.

[Figure 1: see original paper] shows a sample fragment of a service collaboration graph constructed using Mashup services registered on ProgrammableWeb. Mashup services are composite services typically containing 2-6 Web services

that collaboratively complete complex business functions. After constructing the service collaboration graph, Node2Vec is employed to sample node sequences and generate vectors for each node-represented service, called service collaboration vectors.

**Algorithm 1** generates the service collaboration similarity matrix, from which collaboration similarity between any two services can be obtained. Lines 1-6 construct the collaboration graph by initializing an empty SCG, then traversing service processes in the process library $SP$, adding nodes $v(s)$ and $v(s')$ and edge $\langle v(s), v(s')\rangle$ for services $s$ and $s'$ with collaboration dependencies.

Lines 7-20 use Node2Vec to generate service collaboration vectors. Line 7 initializes an empty set *walks* for storing node sequences from random walks. Line 8 loops to generate $r$ random walks. For each service node $u$ in the collaboration graph, lines 9-10 initialize a sequence array $walk[u]$. Lines 11-17 generate node sequences of length $l$ through random neighbor selection and negative sampling, appending *walk* to *walks*. Line 19 uses SGD to train on *walks* and generate collaboration vectors $scv[i]$ for each node. Lines 21-23 compute collaboration similarity between any two services using cosine similarity of vectors and construct matrix $CSM$, which is returned in line 24.

---

## 3.1 Service Cluster-Oriented Request Response Pattern

In traditional SOA architecture, service requests are fulfilled by individual services or service composition processes published on the network, facing massive service comparisons and low discovery efficiency. Service substitution requires finding alternative services for existing responses, incurring enormous search costs.

To improve discovery efficiency during substitution, we introduce service clusters into SOA architecture, establishing the response pattern shown in [Figure 2: see original paper]. A virtual resource layer is added to traditional SOA, containing service clusters. Unlike conventional patterns, service requests from the business model layer are no longer responded to by atomic services directly but first by service clusters in the virtual resource layer, which then bind optimal response services that subsequently bind to atomic services or composition processes in the physical resource layer.

Introducing the virtual resource layer and service clusters increases response granularity, enabling a "service cluster-service" two-level discovery pattern. Since similar services are grouped into clusters, the search space is substantially reduced, improving discovery efficiency.

---

## 3.2 Service Substitution Algorithm

The service substitution process comprises two phases: candidate service set generation, which obtains services that can replace the invalid service functionally; and comprehensive optimization based on service quality and collaboration similarity.

**Definition 3 (Service):** A service is defined as a 6-tuple $s = (Id, D, I, O, Q, L)$, where $Id$ is the service identifier, $D$ is the functional description text, $I$ and $O$ are input and output parameter sets, $Q$ is the quality set, and $L$ is the service address identifier.

For example, the GeocodingApi service from Google Maps registered on ProgrammableWeb has description $D$: "Geocoding is the process of converting addresses (like '1600 Amphitheatre Parkway, Mountain View, CA') into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map. Reverse geocoding is the process of converting geographic coordinates into a human-readable address." Its input parameter set is $I = \{(\text{string address})\}$ and output parameter set is $O = \{(\text{float latitude}), (\text{float longitude})\}$.

The quality set $Q = \{q_i\}$, where $q_i = \{N, C, V, U\}$ with $N$ as the quality parameter name, $C$ as the comparison operator, $V$ as the parameter value, and $U$ as the unit. For instance, $q = (\text{ResponseTime}, <, 0.5, \text{ms})$ indicates response time less than 0.5 milliseconds.

**Definition 4 (Service Cluster):** A service cluster is defined as a 7-tuple $sc = (CId, V, I, O, Sw, Qc)$, where $CId$ is the cluster identifier, $V$ is the functional vector, $I$ and $O$ are input and output parameter sets, $Sw$ is the set of services contained in the cluster, and $Qc$ is the quality parameter set.

Service clusters are obtained by clustering services and do not have functional description text, using functional vector $V$ to represent cluster functionality, computed as the mean of all service vectors in the cluster. Input and output parameter sets are the unions of all services' parameters. Quality parameter values should cover all services' quality ranges, represented using interval notation $q_i = \{N, C, [V_{min}, V_{max}], U\}$. For example, $Qc = \{(\text{ResponseTime}, <, [0.1, 0.5], \text{ms}), (\text{ResponseRate}, >, [98, 100], \%)\}$ indicates response times between 0.1-0.5ms and response rates between 98-100%.

We employ the GSDMM model, suitable for short text topic extraction, to generate functional vectors from service descriptions. Previous work validated that GSDMM generates significantly higher-quality vectors for Web service descriptions than other topic models. GSDMM is based on the Dirichlet Multinomial Mixture (DMM) model with Gibbs Sampling for approximate solution. The DMM model is shown in [Figure 3: see original paper]. The Gibbs sampling process repeatedly samples a word across all topics to obtain the word-topic distribution matrix, yielding document-topic and word-topic matrices. The proba-

bility formula for a description belonging to a topic during sampling is:

$$p(z_i = z \mid \mathbf{w}_i, \mathbf{z}_{\neg i}) \propto \frac{m_z + \alpha}{K \cdot \alpha + D - 1} \cdot \frac{n_{z,w_i} + \beta}{n_z + V \cdot \beta}$$

where $K$ is the initial topic count, $D$ is the total number of descriptions, $m_z$ is the document count under topic $z$, $n_z$ is the word count under topic $z$, $n_{z,w_i}$ is the occurrence count of word $w_i$ under topic $z$, and $\mathbf{z}_{\neg i}$ represents the current document.

**Definition 5 (Subset Parameters):** Parameter sets $P_a$ and $P_b$ are defined such that $P_a$ is a subset parameter of $P_b$ if and only if: - $\text{Num}(P_a) \leq \text{Num}(P_b)$, where $\text{Num}(p)$ denotes the number of parameters; - $\forall m_i \in P_a, \exists n_j \in P_b$ satisfying $m_i \propto n_j$ and $\text{Type}(m_i) \cong \text{Type}(n_j)$, where $\text{Type}(m)$ denotes parameter type, $\propto$ denotes semantic equivalence, and $\cong$ denotes parameter type compatibility. This relationship is denoted as $P_a \Leftarrow P_b$.

During service substitution, we first find candidate services matching the invalid service functionally and at the interface level, then compute optimal substitutes based on quality and collaboration similarity. We use functional similarity of service descriptions as the functional similarity between services. For services $s_1$ and $s_2$, let $v(s_i)$ denote the functional vector generated from description text $s_i.D$ using GSDMM. Functional similarity is computed as the cosine similarity:

$$\text{FucSim}(s_1, s_2) = \frac{v(s_1) \cdot v(s_2)}{\|v(s_1)\| \cdot \|v(s_2)\|}$$

For quality quantification, given candidate service set $S = \{s_1, s_2, \ldots, s_m\}$ with $n$ quality parameters each, where $[q_{i1}, q_{i2}, \ldots, q_{in}]$ are service $s_i$'s quality values, we compute weighted quality scores using:

$$\text{Qscore}(s_i) = \sum_{j=1}^{n} w_j \cdot q'_{ij}$$

where $w_j$ represents quality parameter weights. Equation (4) handles negative quality parameters (e.g., response time, cost) where smaller values indicate better quality, while equation (5) handles positive parameters (e.g., stability, throughput) where larger values are better.

**Algorithm 2** presents the service substitution algorithm under the service cluster model incorporating collaboration similarity. Line 1 initializes empty sets $CSR$ for candidate service clusters and $cs_r$ for candidate substitute services. Lines 2-4 use functional similarity FucSim to filter clusters that can substitute invalid service $se$, adding functionally equivalent clusters to $CSR$. Lines 5-9 traverse services in $CSR$, using subset parameter matching to add services with compatible interfaces to $cs_r$.

Lines 10-12 compute recommendation scores for each service in $cs_r$. The recommendation score combines quality score and collaboration similarity with the invalid service using weights $\alpha$ and $\beta$ that can be adjusted based on their relative importance. Line 13 selects the service with maximum recommendation score as the substitute $rst$, returned in line 14.

---

## 4 Simulation Experiments

This section validates the proposed method's effectiveness through experiments evaluating both substitution discovery efficiency and recommendation rationality. The experimental environment uses an i5-10210U CPU, 8GB RAM, Windows 10 OS, with Python implementation.

Two datasets were constructed: Dataset 1 is a real dataset crawled from ProgrammableWeb, retaining 5,327 valid Mashup services containing 786 WebAPI services after cleaning, clustered into 34 service clusters based on service tags. To improve substitution feasibility and facilitate interface matching, we manually corrected input/output parameters of the 786 services to unify synonymous data types and parameter names.

Dataset 2 is a synthetic dataset generated from 21,425 valid Web API services across 156 categories crawled from ProgrammableWeb. We randomly selected 80 categories, extracting 1-10 services per category to obtain 400 original services. These were processed to generate 3,600 new services, totaling 4,000 services:

1) **Service description generation:** For each original service description, 1-20 new descriptions were generated by randomly replacing 10-70% of words.

2) **Service interface parameter generation:** For each original parameter, two synonyms were obtained to create a candidate parameter name set, from which new services randomly selected parameters.

3) **Quality setting:** Since ProgrammableWeb lacks quality parameters, we selected response time ($rt$), availability ($av$), and cost ($cs$), randomly generating values from intervals $\{rt[0.01, 0.1], av[97\%, 100\%], cs[0, 20]\}$ for all services.

4) **Collaboration relationship generation:** 40,000 service processes with 8 service nodes each were randomly generated. The 4,000 services were divided into 8 categories of approximately 500 services each, corresponding to process nodes. During process generation, one service was randomly selected from each category to form an 8-service process.

### 4.2 Service Discovery and Substitution Efficiency

Service substitution essentially rediscovers services based on invalid service functionality, making discovery and substitution efficiency critical validation met-

rics. We compare our method (SM_CC) against four recent service discovery methods [22, 27, 28, 29] and four substitution methods [6, 11, 12, 30].

For Dataset 1, we use the 34 service categories as cluster count. For Dataset 2, we test clustering with 40, 80, 120, 160, and 200 categories. [Figure 4: see original paper] shows the discovery time trend, revealing that efficiency relates to cluster size, with time first decreasing then increasing as cluster granularity becomes finer. Based on the trend, we select 120 clusters for optimal discovery speed.

[Figure 5: see original paper] and [Figure 6: see original paper] compare discovery times across methods in both datasets. The results show our method outperforms others, with service clusters providing more significant search space reduction for larger datasets. In Dataset 1, our method's average time ratio to other methods is 1:1.79, improving to 1:1.96 in Dataset 2 as service count increases, representing an 8.67% average reduction. Methods using pre-trained topic models for service vectors also outperform others.

[Figure 7: see original paper] and [Figure 8: see original paper] compare substitution times. Our method demonstrates lower substitution time than alternatives. Methods incorporating service clusters show lower times than non-clustered approaches. Sara's method incurs highest time due to complex interface matching. In Dataset 1, our method's average time ratio to four alternatives is 1:1.75, improving to 1:1.93 in Dataset 2, confirming significant efficiency gains from service clusters.

### 4.3 Substitution Rationality

Assuming functional adequacy, our algorithm recommends optimal substitutes by comprehensively considering service quality and process collaboration effects. We propose the concept of service co-occurrence rate to evaluate rationality, measuring the probability of recommended services co-occurring with other services in the invalid service's process context.

Service co-occurrence rate is the ratio of two services appearing together in processes to their total individual occurrences. Let $\mathrm{OccuNum}(s_i)$ denote $s_i$'s occurrence count across all processes. The co-occurrence rate between $s_i$ and $s_j$ is:

$$\mathrm{ServiceCo\_Occu}(s_i, s_j) = \frac{\mathrm{OccuNum}(s_i \cap s_j)}{\mathrm{OccuNum}(s_i) + \mathrm{OccuNum}(s_j)}$$

The process co-occurrence rate between substitute $st$ and invalid service $se$ is defined as the sum of co-occurrence rates between $st$ and all other services in $se$'s process. If $S$ is the service set in $se$'s process, then:

$$\mathrm{Proc\_SerCo\_}(st, se) = \mathrm{ServiceCo\_Occu}(S \quad \{se\}, st)$$

[Figure 9: see original paper] and [Figure 10: see original paper] show process co-occurrence rates for different methods. Our method achieves rates 2-6 times higher than alternatives in both datasets, validating that incorporating collaboration similarity improves substitution rationality.

---

## 5 Conclusion

To improve service substitution efficiency and rationality, this paper proposes a service cluster-based method integrating process collaboration similarity. Service clusters reduce search space and improve discovery efficiency. Modeling and measuring process collaboration relationships, combined with service quality in optimal recommendation, enhances substitution rationality. Experimental results validate the method's effectiveness in improving both efficiency and rationality.

Future work will investigate the impact of clustering granularity on substitution efficiency and improve collaboration effect modeling and measurement methods to further enhance efficiency and rationality.

## References

[1] Akbar M A, Khan A A, Mahmood S, et al. A robust framework for cloud-based software development outsourcing factors using analytical hierarchy process [J]. Journal of Software: Evolution and Process, 2021, 33(2): e2275.

[2] Cheng H, Zhong M, Wang J. Diversified keyword search based web service composition [J]. Journal of Systems and Software, 2020, 163: 110542.

[3] Sefati S, Navimipour N J. A QoS-Aware Service Composition Mechanism in the Internet of Things Using a Hidden-Markov-Model-Based Optimization Algorithm [J]. IEEE Internet of Things Journal, 2021, 8(20): 15620-15627.

[4] Wang Xianqing, Huang Changqin, Luo Xuan, et al. Determining substitutability of cloud services supported by semantically extended type theory [J]. Journal on Communications, 2016, 37(2): 20-31.

[5] Al-Sayed M M, Hassan H A, Omara F A. An intelligent cloud service discovery framework [J]. Future Generation Computer Systems, 2020, 106: 438-466.

[6] Hu Q, Shen J. A Cluster and Process Collaboration-Aware Method to Achieve Service Substitution in Cloud Service Processes [J]. Scientific Programming, 2020: 1298513.

[7] Liang Q, Lee B S, Hung P C K. A rule-based approach for availability of service by automated service substitution [M]// Software: Practice and Experience, 2014, 44(1): 47-76.

[8] Zhang M W, Zhu Z L, Li D C, et al. An execution context aware approach for Web service substitution [J]. In 7th International Conference on Advanced Information Management and Service. IEEE, 2011: 13-18.

[9] Gao H, Duan Y, Miao H, et al. An approach to data consistency checking for the dynamic replacement of service process [J]. IEEE Access, 2017, 5: 11700-11711.

[10] Santhanam G R, Basu S, Honavar V. In IEEE International Conference on Services Computing [C]// IEEE, 2009: 210-217.

[11] R. Sara, A. Bakhta, and L. Lakhdar, A similarity network for web services operations substitution [J]. Journal of King Saud University-Computer and Information Sciences, 2020, 32(9): 1055-1062.

[12] Du Y Y, Gai J J, Zhou M C. A Web service substitution method based on service cluster nets [J]. Enterprise Information Systems, 2017, 11(10): 1523-1542.

[13] Natarajan B, Obaidat M S, Sadoun B, et al. New clustering-based semantic service selection and user preferential model [J]. IEEE Systems Journal, 2020, 15(4): 4980-4988.

[14] Obidallah W J, Raahemi B, Ruhi U. Clustering and association rules for web service discovery and recommendation: a systematic literature review [J]. SN Computer Science, 2020, 1(1): 1-33.

[15] Chen K, Kuang C. Web service discovery based on maximum weighted bipartite graphs [J]. Computer Communications, 2021, 171: 54-60.

[16] Hafsi A, Gamha Y, Ben Njima C, et al. BIG-SWSDM: BIpartite graph based social web service discovery model [C]// International conference on business information systems. Springer, Cham, 2020: 307-318.

[17] Abdelli A, Serrai W, Mokdad L. A novel and efficient index based web service discovery approach [J]. Computer Standards & Interfaces, 2022, 80: 103586.

[18] Cassar G, Barnaghi P, Moessner K. Probabilistic matchmaking methods for automated service discovery [J]. IEEE Transactions on Services Computing, 2013, 7(4): 654-666.

[19] Agarwal N, Sikka G, Awasthi L K. Enhancing web service clustering using Length Feature Weight Method for service description document vector space representation [J]. Expert Systems with Applications, 2020, 161: 113682.

[20] Cheng B, Li C, Zhao S, et al. Semantics mining & indexing-based rapid web services discovery framework [J]. IEEE Transactions on Services Computing, 2021, 14(3): 864-875.

[21] Abid A, Rouached M, Messai N. Semantic web service composition using semantic similarity measures and formal concept analysis [J]. Multimedia Tools and Applications, 2020, 79(9): 6569-6597.

[22] Nabli H, Djemaa R B, Amor I A B. Efficient cloud service discovery approach based on LDA topic modeling [J]. Journal of Systems and Software, 2018, 146: 233-248.

[23] Hu Q, Shen J, Wang K, et al. A Web service clustering method based on topic enhanced Gibbs sampling algorithm for the Dirichlet Multinomial Mixture model and service collaboration graph [J]. Information Sciences, 2022, 586: 239-260.

[24] Lizarralde I, Mateos C, Zunino A, et al. Discovering web services in social web service repositories using deep variational autoencoders [J]. Information Processing & Management, 2020, 57(4): 102231.

[25] Dania A, Griffin L L. Using social network theory to explore a participatory action research collaboration through social media [J]. Qualitative research in sport, exercise and health, 2021, 13(1): 41-58.

[26] Grover A, Leskovec J. node2vec: Scalable feature learning for networks [C]// Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

[27] Lu C. A Novel Web Service Discovery Method Combining Semantic Interface Similarity and Context Similarity [C]// IOP Conference Series: Earth and Environmental Science. IOP Publishing, 2021, 693(1): 012006.

[28] Zhang F, Zeng Q, Duan H, et al. Composition context-based web services similarity measure [J]. IEEE Access, 2019, 7: 65195-65206.

[29] Abdelli A, Serrai W, Mokdad L. A novel and efficient index based web service discovery approach [J]. Computer Standards & Interfaces, 2022, 80: 103586.

[30] Zhang zhaoyang. Research on service substitution to group services [D]. Nanjing University of Posts and Telecommunications, 2018.

*Note: Figure translations are in progress. See original paper for figures.*