Diverse Top-k Service Composition for Consumer Electronics with Digital Twin in MEC

Jintao Wu, Jinxiang Sha, Muhammad Bilal, Yiwen Zhang, and Xiaolong Xu

***Abstract*—Mobile Edge Computing (MEC) stands as an indispensable technology in the facilitation of 5G networks, enabling the deployment of widely-used services on edge servers situated in close proximity to consumer electronics. Within the MEC framework, a central role is attributed to edge service composition (ESC), pivotal in bolstering functionality and ameliorating user experiences. Presently, prevailing methods for ESC predominantly center on the prioritization of Quality of Service (QoS) optimization, presenting a solitary optimal composite service for consumer electronics invocation. Regrettably, this approach sidelines the significance of solution diversity within composite services, potentially resulting in service overload and the suboptimal utilization of edge resources. To surmount these challenges, this study integrates digital twin (DT) technology and diversified search mechanisms into the MEC domain, offering an innovative diversified top-*k* service composition methodology known as DSC-DT. By harnessing the capabilities of DT technology, DSC-DT enables the emulation and assessment of diverse composite service solutions within a virtual space. Specifically, the proposed methodology models the procedure of service composition within a DT environment as an issue of subgraph isomorphism. This is succeeded by the configuration of the diversification process as an independent set predicament within an undirected graph, efficiently resolved through a greedy algorithmic paradigm. It is noteworthy that DSC-DT accommodates a gamut of query types, including normal queries, constraint queries, and optimal queries. The efficacy and efficiency of the proposed approach are corroborated through comprehensive experiments conducted upon authentic datasets.**

***Index Terms*—Mobile Edge Computing, Consumer Electronics, Diversified Top-*k* Service Composition, Digital Twin**

Manuscript received XXX 2023; revised XXX 2023 and XXX 2023; accepted XXX 2023. This research was supported by the Natural Science Foundation of Jiangsu Province of China (No. BK20211284), the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps (No. 2020DB005), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 23KJB520020), and the Construction and Application Demonstration of city-oriented Epidemic Prevention and Control Decision Support Platform (No. 2021AB034).

(Corresponding author: Muhammad Bilal)

Jintao Wu, Jinxiang Sha and Xiaolong Xu are with the Department of School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: wjt@nuist.edu.cn; jinxiangs@126.com; xlxu@ieee.org).

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, Bailrigg, Lancaster LA1 4WA, United Kingdom (e-mail: m.bilal@ieee.ac.kr).

Yiwen Zhang is with the School of Computer Science and Technology, Anhui University, Anhui, China (e-mail: zhangyiwen@ahu.edu.cn).

Digital Object Identifier XXXX

I. INTRODUCTION

With the growing intelligence of consumer electronics and the progression of mobile edge computing (MEC) technology [1, 2], an increasing cadre of service providers is disseminating lightweight, loosely-interlinked, and self-contained services across edge servers [3, 4]. This dissemination empowers mobile users to beckon forth these services via consumer electronics such as smartphones, laptops, and wearable devices, characterized by minimal latency, thus adroitly fulfilling their real-time imperatives [5]. In light of the burgeoning assortment and intricacy of mobile user requisites, the provision of a solitary service to meet a user's needs has become an increasingly onerous undertaking [6]. Consequently, Edge Service Composition (ESC) mechanisms have been posited, orchestrating the automated assemblage of multiple edge services to engender a value-added service, thereby satiating user queries (i.e., a request for a composite edge service) [7].

Employing Quality of Service (QoS) as a quantifiable metric for non-functional attributes, composition systems are empowered to identify appropriate component services from a spectrum of functionally akin services at the edge [8, 9]. The preponderance of contemporary ESC techniques are designed to embody QoS-consciousness, seeking to proffer mobile users with the quintessential composite service [7, 10]. Nonetheless, this breed of methodologies presents a conundrum in scenarios where any element service within the optimal composite service becomes inaccessible due to uncontrollable exogenous factors. In such instances, the MEC infrastructure necessitates the reconstitution of a novel composite solution, thereby fostering an uptick in service latency and system energy consumption. A viable stratagem entails the incorporation of a top-*k* search modality, proffering users multiple composite service options-a paradigm entrenched in traditional cloud service composition. To illustrate, Benouaret et al. [11] propound a service composition technique tailored to accommodate fuzzy preference queries. This approach capitalizes upon fuzzy sets to conceptualize user preferences, delivering the top-*k* solutions that duly factor in these preferences. In consideration of QoS, Jiang et al. [12] present an innovative top-*k* service composition technique christened KPL, distinguished by its commendable scalability, efficiency, and accuracy. However, it is prudent to acknowledge that the KPL approach pivots upon centralized computation, necessitating a potent and dependable central server, alongside considerable bandwidth allotment for computation, data warehousing, and communication. These requisites constrain the applicability of KPL within the edge domain. Deng et al. [13] contrive a service composition strategy underpinned by backtracking search and depth-first search

tactics, fostering streamlined parallel processing of user appeals. This methodology orchestrates top-*k* service composition searches with a decentralized manner, thus mitigating the burden on central nodes and uplifting overall system availability. Similarly, Li et al. [14] postulate a novel approach employing a relational database to automate service composition. This schema involves a premeditated generation of all conceivable service compositions, archived within a relational database. Upon receipt of a user solicitation, this relational database approach integrates SQL queries to peruse the database, retrieving the top-*k* valid composite services embodying optimal QoS—effectively addressing both the functional and non-functional prerequisites of users.

The advent of top-*k* service composition methodologies has extended users' access to alternative solutions. However, these methodologies have predominantly focused on refining or devising efficient algorithms for top-*k* searches, relegating the study of result set diversity to the background. This oversight can lead to service redundancy, wherein all *k* composite services offered by the system may include identical sets of constituent services, i.e., high-QoS edge services. In scenarios where these services are rendered inaccessible due to external perturbations or unforeseen incidents, multiple composite service solutions may concurrently become unattainable, rendering the top-*k* search futile. Additionally, these solutions characterized by limited diversity often lead to the repetitive invocation of high-QoS services, with low-QoS services being underexploited, thus resulting in wastage of precious edge resources. Therefore, the top-*k* service composition paradigm strives not merely to identify a cohort of composite services at the edge, exhibiting the highest feasible objective score (such as QoS), but also mandates the attainment of considerable diversity among these solutions. Recent times have witnessed the ascent of diversity within the scope of service computing. For instance, in countering the quandary of service redundancy within traditional QoS-conscious service selection, Gou et al. [15] introduce an innovative diversity-infused graph-based QoS prediction model—DSSN (Diverse Service Selection Network). This model augments selection diversity to mitigate redundancy and exhibits greater aptitude for diversity-aware service selection amidst user requirements characterized by ambiguity. Concerning service recommendation, Kang et al. [16] proffer an original approach tailored for diversified QoS-centric service recommendations, with a specific focus on uncertain QoS preferences. The effort here seeks to produce a comprehensive roster of services that not only satiate stipulated QoS requisites but also showcase a melange of diversity in their offerings. To aid system engineers in the expedient assembly of service-oriented software systems based on given keywords, Cheng et al. [17] introduce the quandary of top-*k* diversified service composition and propound an innovative diversified keyword search method predicated on service connection graphs to redress this quandary. Notwithstanding the strides in addressing the intricacies of diversified service composition, the realization of diverse top-*k* service composition within MEC remains challenging, underscored by the strictures imposed by finite edge resources, the heterogeneity of edge devices, and the capriciousness of candidate services.

Considering the exigencies and constraints including edge service composition, digital twin (DT) technology emerges as a prospective panacea. DT represents an attractive technology that charts the mapping of physical systems onto a mirrored virtual space [18, 19]. Employing an assemblage of high-performance sensors and swift communication, DT integrates data from multifaceted physical entities, supplemented by data analysis and simulation, to present an almost real-time facsimile of the actual state of physical entities. This mapping between the realms of actuality and virtuality offers stakeholders a holistic insight into the subject system, thereby fundamentally reshaping the landscape of design and engineering [20]. Presently, a plethora of efforts is underway to enable real-time and efficacious monitoring of system status information within DT-facilitated MEC systems. Prominently featured in these efforts is the paradigm of edge service offloading, wherein specific computational tasks are transposed from end devices onto MEC servers [21-23]. This not only lightens the burden on end devices but also amplifies computational efficiency. These research initiatives offer substantial sustenance and guidance for the integration of DT within MEC, grappling with pivotal domains such as the optimization of system resource allocation [24, 25], the curbing of energy consumption and latency [18, 26], and the augmentation of user experiences [27, 28].

Diverging from extant works, this paper centers on a DT-enabled MEC architecture geared toward offering diversified alternatives for service composition to consumer electronics. We proffer a diversified edge service composition methodology, designed to enrich solution diversity, minimize service failure rates, and abate computational expenditures. The ultimate aim resides in enhancing the user experience of mobile users harnessing consumer electronics. In specific terms, we map the relationships between MEC and service invocation into a virtual construct, generating diverse top-*k* composite services within this virtual space. Subsequently, the fruits of this construction are projected back into the real world, proffering mobile users with a plethora of alternative solutions marked by minimal service overlap when interfacing with consumer electronics. The primary contributions of this paper unfold as follows.

- We propound an innovative ESC framework that augments edge resource utilization through the integration of digital twin technology and enhances the diversity of composite services through the instantiation of diversified search strategies.
- We unveil a novel approach, christened DSC-DT, for top-*k* service composition that strives to amplify the diversity of composite service resolutions accessible to mobile users. In essence, we conceptualize the process of service composition as a conundrum of independent sets within an undirected graph, subsequently addressed through an expeditious greedy algorithm.
- DSC-DT operates on the foundation of an edge service data graph and employs a proficient subgraph search algorithm to address user-issued inquiries. Currently, DSC-DT accommodates three distinct query categories: normal queries, constraint queries (i.e., queries with quality constraints), and optimal queries (i.e., queries with both quality constraints and optimization objectives).

The subsequent sections of this article are structured as follows. The ensuing segment provides an overview of pertinent definitions germane to diversified top-*k* edge service

composition, serving to contextualize our proposed approach. Subsequently, Section III offers an exhaustive description of the DSC-DT algorithm and its practical instantiation. Section IV includes the presentation of results derived from diverse experimental scenarios aimed at substantiating the efficacy of our proposed methodology. Finally, the article culminates with concluding remarks and an exploration of prospective trajectories for future research in Section V.

II. System Model and Problem Definition

Fig.1. shows the framework of a digital twin empowered MEC system, including three main components: M base stations equipped with edge servers, N edge services deployed on these servers, and C mobile devices (i.e. consumer electronics) capable of invoking the available edge services. For each base station, edge service, and mobile device, a digital twin is created based on collected information such as position, velocity, status, and energy consumption. These digital twins serve as accurate simulations of their corresponding physical entities, capturing their real-time operational states and characteristics. Through data synchronization and interaction with the actual system, the digital twins facilitate precise real-time monitoring, analysis, and optimization, thereby enabling effective performance management and decision-making in the MEC system[29].

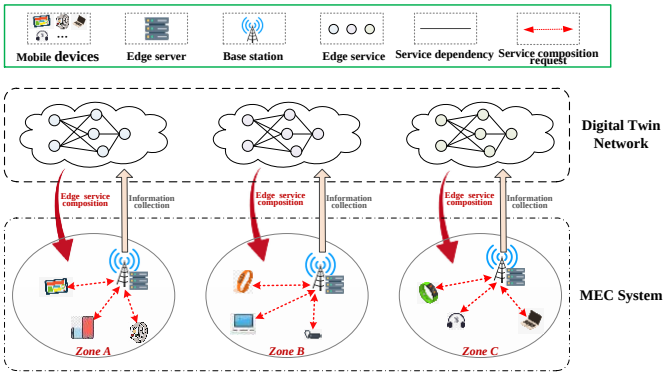


Fig. 1. The architecture of a DT-empowered MEC system.

In our DT-empowered framework, when a mobile user submits a request to a local edge server, the MEC system generates a top-k solution set for the request. Before the top-k method is proposed, we first define the basic concepts used in diversified top-k service composition.

Definition 1 (Mobile Device): The umbrella term "mobile devices" includes all consumer electronics vested with the capability to summon edge services. Let $d = (id_d, l_d, vs, bs)$ represent a mobile device, where id_d represents its unique identifier, l_d signifies its present location, vs denotes its velocity, and bs indicates the associated base station.

Definition 2 (Base Station): An entity fitted with an edge server is termed a "base station," succinctly represented as a 4-tuple $bs = (id_b, lb, r, e)$, where id_b corresponds to the distinctive identifier of the base station, lb denotes the geographic coordinates of the base station, r signifies the coverage radius, and e signifies the presence of an edge server.

In our DT-empowered MEC system, each base station corresponds to an attached edge server which serves a local area called a zone, whereas a mobile device is associated with only one zone. Multiple edge services, as defined below, are deployed on each edge server.

Definition 3 (Edge Service): An "edge service" is defined as a 4-tuple $s = <id_s, f, bs, \text{QoS}>$, with id_s connoting a unique service identifier, f representing the functional defining inclusive of input, output, preconditions, and result, bs indicating the base station housing the edge service, and $\text{QoS} = \{q_1, q_2, ...\}$ denoting the assortment of quality attributes associated with s, as collected from service level agreements (SLAs), involving factors such as cost, response time, reliability, throughput, and more.

In Definition 3, the input and output parameters of an edge service encapsulate its functional traits, while the QoS component embraces its non-functional attributes. The repository of edge services, accessible on edge servers, is encapsulated by $S = \{s_1, s_2, ...\}$. In our context, we conceptualize an "edge service data graph" in alignment with the ensuing definition, orchestrating the organizational framework for these services:

Definition 4 (Edge Service Data Graph): An "edge service data graph" is defined as a directed, vertex-labeled, edge-labeled graph $G = <S, A, LS, FS, LA, FA>$. Here, S denotes the set of vertices within G, signifying the reservoir of edge services; $A \subseteq S \times S$ encapsulates a set of directed edges in G, symbolizing service composability. The edge $a(s_i, s_j) \in A$ signifies that s_j can succeed s_i in the composition sequence of s_i , s_j; LS embodies the ensemble of vertex labels, embodying a series of tasks; For each vertex $s \in S$, $FS(s) \in LS$ denotes the label of s; LA embodies the cohort of edge labels, designating task dependencies. The edge $la(s_i, s_j) \in LA$ denotes that $FS(s_j) = T_j$ can follow $FS(s_i) = T_i$ within the business process transitioning from T_i , T_j; for each edge $a \in A$, $FA(a) \in LA$ designates the label of a.

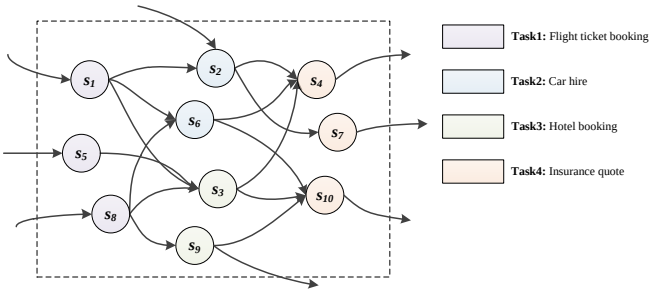


Fig. 2. Part of the edge service data graph, G

It should be noted that each $a \in A$ is obtained from the historical interactions between edge services utilizing data mining methodologies [30, 31]. In this undertaking, we leverage the concept of DT to simulate this data graph, enabling real-time emulation and reflection of the dynamic states and mutations of edge services. Through the deployment of DT technology, precision and stability in data graph depiction can be upheld. Following the graph's construction, its stability remains relatively intact, with updates necessitating minimal

overhead, particularly for events (e.g., the integration of novel services or the obsolescence services leaving). Fig. 2. proffers a partial illustration of an edge service data graph, including 10 edge services and 4 tasks. This graph predominantly serves the sphere of classical tourism.

Definition 5 (Business Process Request): The business process request is formalized as a 4-tuple $Q = < LS_Q, LA_Q, C, k >$, where $LS_Q = \{T_1, T_2, …., T_n\}$ designates an assemblage of tasks, with each task T_i corresponding to a component functionality within the business process; n, an integer, signifies the aggregate count of tasks necessary for the query; $LA_Q = \{a(T_i, T_j)|T_i \in T, T_j \in T\}$ conveys dependencies among tasks in T, where $a(T_i, T_j)$ denotes that the outputs of T_j underpin the inputs of T_i (i.e., T_j can be the succeeding task of T_i); A is invoked to depict the structural architecture of the query; $C=\{c_1, c_2, …, c_i, …\}$ encapsulates the QoS constraints formulated by mobile users; k, an integer, embodies the upper limit for the count of solution options for the query.

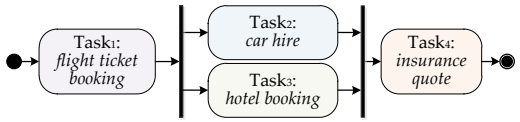


Fig. 3. Example of business process query

Fig. 3. is an example of a business process request Q containing four tasks sent by a mobile user to the travel booking platform. In our DT-empowered MEC system, upon receiving a request from a mobile device, the system constructs a set of service solutions tailored to fulfill the specific requirements of the request. These service solutions are defined as follows:

Definition 6 (Service Solution): A service solution constitutes an edge service composition, explicated as $sc = <S_R, A_R, LS_R, FS_R, OQ>$, where $S_R = \{s_1, s_2, … s_n\}$ denotes the compendium of edge services composing the service solution; ; $A_R = \{a(s_i, s_j)|s_i \in S_R, s_j \in S_R\}$ conveys dependencies between edge services within SR, with $a(s_i, s_j)$ denoting that the inputs of s_j are contingent on the outputs of s_i (i.e., s_j can be the succeeding service of s_i); and A_R endows the structural schema of the service solution; LS_R involves service labels, each concomitant with a task or functionality; for each service $s \in S_R$, $FS_R(s) \in LS_R$ designates its label of s; OQ expresses the holistic QoS of the service solution.

As aforementioned, the service solutions aimed at satisfying system responses to business requests ought to be marked by diversity. To quantify the diversity inherent in these solutions, we introduce the concept of "edge service coverage."

Definition 7 (Edge Service Coverage): Given a solution set $R = \{sc_1, ..., sc_i, ..., sc_k\}$, where $sc_i = < S_i, A_i, LS_i, FS_i, OQ_i >$, the cover set of R, denoted by $C(R)$, is defined as the union of edge service sets: $C(R) = S_1 \cup S_2 \cup ... \cup S_k$. The edge service coverage of R is given by $|C(R)|$.

According to the above description, in DT-empowered MEC, the diversified top-k service composition problem is defined as follows:

Definition 8 (Diversified Top-k Service Composition in DT-empowered MEC): Considering an edge service data graph G, a business process request Q the *diversified top-k service composition* problem (DSCQ) in of DT-empowered MEC pertains to the selection of a solution ensemble R, comprising no more than $Q.k$ service compositions from G, which comprehensively include all tasks and dependencies specified in Q, and such that R exhibits the most extensive edge service coverage among all possible selections of $Q.k$ or fewer service compositions.

Based on different quality requirements, business process requests can be categorized into normal requests, constraint requests, and optimal requests. Each type of request yields different result sets of diversified top-k service solutions. For normal requests, the results focus on identifying k service solutions that offer maximum service coverage without considering the user's service quality constraints. Constraint requests, on the other hand, exclude service solutions that fail to meet the user's service quality requirements. From the remaining candidate solutions, k service solutions with the highest service coverage are selected. Optimal requests share similarities with constraint requests, but they differ in that the results must include at least one service solution that satisfies an optimal QoS requirement.

III. Diversified top-k Service Composition

This section provides an in-depth exploration of the proposed DSC-DT approach, focusing on its application in diversified top-k service composition queries within the context of MEC. We commence by presenting a foundational framework that outlines the key components and interactions of the approach. Subsequently, we discuss each phase of the approach, offering comprehensive insights into their functionalities and contributions.

A. Framework of proposed approach

Our formulated DSC-DT methodology is structured as a five-step procedure:

Step 1: Map the business process request formulated by the user into the context of the DT-empowered MEC system;

Step 2: Prune the data graph predicated on the specifics of the business process request, leading to the culling of edge services and dependencies that are unfeasible within the definitive service solutions;

Step 3: Seek all conceivable candidate solutions within the pruned data graph that align with the user's functional and non-functional requisites. Subsequently, assemble a solution graph predicated on the extent of service overlap notable among these candidate solutions;

Step 4: Infuse diversification into the retrieved service solutions, premised upon the orchestrated solution graph;

Step 5: Return k service solutions endowed with diversity back to the user, emanating from the DT-empowered MEC system.

Detailed explanations of the pruning, construction, and diversification operations will be proffered in ensuing sections.

B. Pruning of Data Graph

The increase in the number of edge services has led to a commensurate expansion in the magnitude of the edge service data graph instituted within the purview of DT-empowered MEC. This increase, while affording mobile users a plethora of

alternate resolutions, concomitantly augments the exploration space during the process of service composition. To optimize the efficiency of edge service composition, this section introduces a pruning strategy aimed at sieving out services and their interdependencies from the service data graph, which are unlikely to appear in the final service solution.

Firstly, two data index structures, the vertex index and the dependency index, are introduced based on the constructed edge service data graph G. These indices are defined as follows:

Definition 9 (Vertex Index): The vertex index represents the mapping from tasks to candidate service sets, which can be formalized as $V\text{-}map=\{T_1 \rightarrow S_1, T_2 \rightarrow S_2, \ldots, T_i \rightarrow S_i, \ldots\}$, where

1) $T_i \in G.LS$ is the task label contained in the edge service data graph G;
2) $S_i \subseteq G.S$ is the set of candidate edge services for task T_i in the data graph G. For each $s \in S_i$, it satisfies that $G.FS(s) = T_i$.

Definition 10 (Dependency Index): The dependency index represents the mapping from task dependencies to service dependency sets and can be formalized as $D\text{-}map=\{la(T_i, T_j) \rightarrow A_{ij}\}$, where

1) $la(T_i, T_j) \in G.LA$ is the task dependency contained in the edge service data graph G;
2) $A_{ij} \subseteq G.A$ represents the set of dependencies between edge services in the data graph G. For each $a(s_i, s_j) \in A_{ij}$, it satisfies that $G.FS(s_i) = T_i$ and $G.FS(s_j) = T_j$.

In congruence with the development of the edge service data graph, the establishment of the vertex index and dependency index can be executed offline. Post-construction, the stability of the vertex/dependency index remains relatively undisturbed and can be updated with nominal overhead in response to specific events.

For the user-defined business process request $Q = <LS_Q, LA_Q, C, k>$, the set of candidate services and their correlated service dependencies linked to LS_Q and LA_Q can be expediently retrieved through the utilization of the vertex/dependency index. Employing the service data graph portrayed in Fig. 2 and the request exhibited in Fig. 3, the intermediary result depicted in Fig. 4 can be swiftly garnered through recourse to the vertex/dependency index. This result can be instantiated sans direct access to the physical edge server, thereby expediting the construction process. Moreover, this operation effectually sieves out an appreciable number of unrelated services, thereby curtailing the exploration space in subsequent phases.

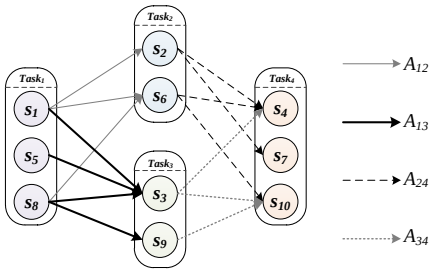


Fig. 4. Illustration of intermediate results

However, it is necessary to acknowledge that the derived intermediary result still encompasses a considerable number of candidate services and service dependencies that are destined to be excluded from the ultimate result. For instance, concerning s_5, while it is positioned as a candidate service for T_1, there is an absence of any element within $D\text{-}map$ $[(T_1, T_2)]$ that designates s_5 as its precursor service. In essence, opting for s_5 to execute T_1 would lead to the infeasibility of T_2 to locate an appropriate service for execution, thereby compromising the capacity to provide a service solution consistent with the user's exigencies. Hence, the necessity emerges to further refine this intermediary result through the filtration of services and their dependencies that do not factor into the ultimate service solution. To counter this conundrum, an innovative forward-backward pruning strategy is postulated, oriented towards refining the intermediate results based on the user's predefined requisites.

This strategy consists of two distinct stages. The first stage is *Forward-Pruning*, involving a breadth-first traversal (BFS) of the business process requests, leading to the elimination of candidate services that fall short of fulfilling their dependencies and corresponding cascading dependencies. The BFS algorithm, noted for its simplicity, serves as the bedrock for myriad significant graph algorithms. Algorithm 1 offers a detailed description of this algorithm.

Algorithm 1: ***Forward-Pruning***

Input: Business process request Q,
Vertex index $V\text{-}map$,
Dependency index $D\text{-}map$.
Output: $V\text{-}map$, $D\text{-}map$.

```
1:  Q_T ← Ø
2:  For each Task T ∈ Q.LS_Q
3:      Do Staus[T] = 0
4:  Staus[T_start] = 1
5:  ENQUEUE(Q_T, T_start)
6:  while Q_T ≠ Ø
7:      Do T ← ENQUEUE(Q_T)
8:          For each T_succ ∈ Succeed(T)
9:            If Staus[T_succ] = = 0
10:             Then
11:               Staus[T_succ] = 1
12:           ENQUEUE(Q_T, T_succ)
13:           S_pre= Precursor(D-map[(T, T_succ)])
14:           S_de= V-map[T] - S_pre
15:           V-map[T]= V-map[T] & S_pre
16:               Update(D-map, S_de)
17: Return V-map, D-map
```

The *Forward-Pruning* stage is underpinned by the input of the business process request Q, the vertex index, and the dependency index, resulting in the revised vertex/dependency index as output. The initialization operations for *Forward-Pruning* are enacted in lines 1-4, with the *Status* variable denoting the visitation status of a task. A value of 0 denotes unvisited status, while 1 signifies visited status. Line 5 enqueues the initiating task of Q into the Q_T queue. The while loop in lines 6-16 is iteratively executed until the Q_T queue is depleted. In each iteration, the leading task T of Q_T is dequeued (line 7), followed by the marking of all successor tasks in Q as visited through the changes of their visitation status (lines 8-12). The candidate services for task T are extracted from the $D\text{-}map$,

premised on the prevailing task dependency (T, T_{succ}) in line 13. Line 14 identifies the disqualified candidate services for task T, while the valid candidate services for T are obtained in line 15. Finally, line 16 revises the edge index of the *D-map* by discarding all dependencies affiliated with the invalidated candidate services.

In summary, the *Forward-Pruning* algorithm efficaciously obviates candidate services and their associated service dependencies that suffer from incompleteness concerning subsequent task/service dependencies, ensuing in the *V-map* and *D-map*. While invalid services and dependencies are excised, a measure of redundancy such as s_7 still persists. This can be attributed to the *Forward-Pruning*'s exclusive emphasis on successor task/service dependencies, thereby inadvertently neglecting constraints arising from precursor tasks/services. To surmount this lacuna, an inverse exploration approach known as *Back-Pruning* becomes indispensable to refine the results stemming from *Forward-Pruning*. *Back-Pruning* essentially inverts the search process, harnessing the output of *Forward-Pruning* in conjunction with the business process request as input. Importantly, the *D-map* necessitates reverse processing, including the inversion of the order of predecessor and successor task/service. The output of *Back-Pruning* is a more refined refined vertex/dependency index. The pruning methodologies of *Back-Pruning* and *Forward-Pruning* synergistically ensure that the remaining G exclusively involves service solutions aligned with Q. Relative to the original G, this substantively truncates the exploration space for edge service composition.

C. Construction of Solution Graph

In light of a business process request Q and the refined vertex index *V-map* and dependency index *D-map*, the process of edge service composition can be streamlined by seeking subgraphs within *V-map* and *D-map* that include all the stipulated tasks and dependencies of Q. This problem aligns with the contours of isomorphic subgraph problem. In this research, we leverage the VF2 algorithm [32], a well-recognized isomorphic subgraph algorithm, to manage varying query types of queries. Within the context of a user request, there might exist numerous candidate service solutions within the domain of DT-enabled MEC, designated as R. The primary objective is to offer the user with an array of service solutions, thereby maximizing edge service coverage.

To capture the interrelationships among diverse service solutions, we introduce a structural entity termed the solution graph, which orchestrates the array of solutions.

Definition 11 (**Solution Graph**): The solution graph is essentially an undirected weighted graph, which can be formalized as the tuple $SG = <R, OP>$, where

1) $R = \{r_1, r_2, ..., r_i, ...\}$ represents the set of candidate service solutions;
2) $OP \subseteq R \times R$ denotes the overlap relationship between solutions. For each element $op = <r_i, r_j, n> \in OP$, r_i and r_j are the i-th and j-th service solutions in R, and n ($n > 0$) signifies the weight between r_i and r_j, indicating the service coverage between the two composite services.

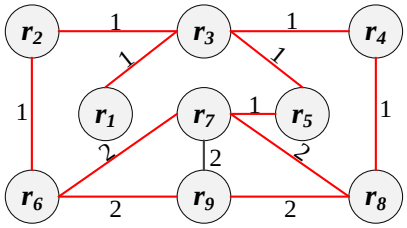


Fig. 5. An example of solution graph

Illustrated in Fig. 5, a sample solution graph integrates 9 distinct service solutions and 12 overlay relationships. The weight assigned to each overlay relationship signifies the extent of service overlap between two solutions. As exemplified by the relationship between r_6 and r_7, which shares 2 overlapping services, the preference is for returning r_1 and r_6 over r_6 and r_7 while offering $k = 2$ solutions to the user. The rationale behind this choice lies in the fact that the number of overlapping services ($|C(R)| = 0$) between r_1 and r_6 is lesser than that ($|C(R)| = 2$) between r_6 and r_7. Nevertheless, as k marginally augments, the task of intuitively selecting k diverse service solutions from the solution graph becomes intricate. For instance, in the pursuit of the top-3 diverse solutions among a solution graph accommodating 15 service solutions, evaluating the diversity of 455 result sets ensues, entailing considerable temporal overhead for successive searches. Therefore, users necessitate a more efficient modus operandi to secure the top-k diverse service solutions during the progression of edge service composition. The subsequent section will proffer an exhaustive explanation of diversity management hinged on the solution graph.

D. Top-k Diversification Processing

In a solution graph *SG*, the effort to obtain a set of diverse service solutions translates into the acquisition of a cluster of non-adjacent nodes, commonly identified as an independent set of *SG*. The notion of an independent set is defined as follows:

Definition 12 (Independent Set): Considering SG, a vertex subset I is deemed an independent set if there exists no edge linking any two vertices within I. Should the addition of a new vertex to I result in the loss of its independence, the set I is then classified as a maximum independent set. The quantity denoting the maximum independent set's size is commonly referred to as the independence number.

Correspondingly, an independent set denotes a constellation of vertices within graph *SG*, where no pair of vertices shares an edge. While pursuing the maximum independent set of *SG* guarantees maximal service coverage within the service solution set, the identification of a maximum independent set within a graph is acknowledged to be an NP-hard undertaking. Moreover, in this study, it is conceivable that a maximum independent set, comprising k service solutions, might prove elusive within a given solution graph *SG*. This limitation arises upon k exceeding the independence number of *SG*. For instance, the discernment of an independent set of $k = 6$ vertices (service solutions) within the solution graph depicted in Fig. 5 is unfeasible, given the independence number of *SG* being 5.

In order to secure a repertoire of service solutions within *SG*

featuring the utmost $|C(R)|$, a diverse search algorithm (*DiverseSearch*) is propounded herein. This approach leverages a greedy mechanism during each iteration, incorporating a service solution with the minimal degree of association to the result set R, until no additional service solutions remain. To effectuate this, we employ $\sum_{i=1}^{|R|} weight(a(r_i, v))$ to measure the degree of association, with $|R|$ denoting the count of service solutions in the prevailing result set R, r_i representing the i-th service solution, v signifying the service solution to be included, and $a(r_i, v)$ designating the edge between r_i and v. The *weight* function serves to determine the weight of a specific edge within *SG*.

Algorithm 2 offers the pseudocode encapsulating the mechanism for securing diverse service solutions, executed within the framework of the solution graph.

Algorithm 2: *DiverseSearch*

Input: Solution Graph *SG*,
Diversified Result Set $R = \emptyset$,
Maximum solution counts k ($k<|SG|$).
Output: Diversified Result Set *R*

```
1:  Do:
2:      G = Hierarchy (SG, R);
3:  While |G| > 0:
4:      v = minAd(G, R);
5:      R.add(v)
6:      if |R| == k:
7:          Return R
8:      else:
9:          deleted(G, v)
10: diverseSearch (SG, R)
```

Line 2 generates a subgraph *G* grounded in *SG* and *R*, aimed at precluding the redundant selection of service solutions. Lines 4-7 are committed to identifying the node *v* within *G*, characterized by the least degree of association to *R*. The underlying objective is to ensure that the freshly incorporated service solution *v* curtails service overlap with extant *R*, thereby heightening the aggregate $|C(R)|$. Line 9 undertakes an update to *G*, guaranteeing that the service solutions generated within the same iteration layer are non-overlapping, resulting in an independent set. Effectively, lines 3-9 encapsulate the procedure of deploying a greedy mechanism for the exploration of an independent set within *G*. Line 10 triggers a new iteration.

Crucially, the initialization of *R* as an empty set precludes the estimation of nodes' association degrees in *SG* solely through *R*, rendering the selection of the first service solution unfeasible. Therefore, for normal and constraint queries, the first service solution is determined as the node possessing the minimum association degree. However, when confronted with an optimal query, the service solution characterized by the most optimal overall QoS assumes the role of the initial service solution. This deliberate stratagem ensures the inclusion of highly optimized composite services within the ultimate result set.

The computational complexity of Algorithm 2 hinges on the magnitude of *SG*. In a worst-case scenario, each layer necessitates the calculation of association degrees among all $|G|$ nodes, yielding an independent set of size merely 1. As such, the time complexity of Algorithm 2 can be succinctly expressed as $O(|SG|^2)$. Nonetheless, it is pertinent to note that the abundance of nodes in *SG* tends to be relatively restrained. This observation holds true, particularly in scenarios involving constraint queries and optimal queries, wherein the node count remains notably limited. Therefore, the efficacy of the *DiverseSearch* algorithm is abundantly evident in DT-empowered MEC systems.

V. Experimental Evaluation

This section evaluates the effectiveness (measured by edge service coverage), and efficiency (measured by computational overhead) of DSC-DT in response to three types of queries in MEC.

A. Experiment setup

All experiments conducted in this study were underpinned by the QWS dataset. The QWS dataset includes over 2500 real-world services, complete with functional attributes and corresponding nine-dimensional service quality information , sourced from public registries, search engines, and service portal websites [33, 34]. In recent years, this dataset has been fervently adopted by researchers in service-oriented software engineering, attesting to its prominence and relevance in the discipline [35].

To investigate the effectiveness and efficiency of the proposed approach across diverse circumstances, a series of experiments were performed in this paper. The key factors considered include the number of tasks in business process requests, the number of quality constraints, the size of the edge service data graph, the density of the data graph (the ratio of edges to nodes), the constraint intensity, and the value of *k*. By systematically varying these factors (as outlined in TABLE I), an exhaustive evaluation of DSC-DT's performance was effectuated. For each trial, multiple services were extracted from the QWS dataset via the classical Erdős-Rényi model [36] , thereby generating a stochastic edge service data graph. The adoption of the Erdős-Rényi model ensured the random linkage of service composability within the data graph. Subsequently, 1000 business process requests were randomly generated from the synthesized data graph, following the random surfer methodology [37].

In order to respond to normal queries, constraint queries, and optimal queries, three versions of the DSC-DT algorithm were customarily designed based on the elucidation furnished in Section III: DSC-DT-N, DSC-DT-C, and DSC-DT-O.

- DSC-DT-N: Serving as a reference baseline, DSC-DT-N solely considers only functional attributes without any constraints pertainingto QoS. It returns *k* diverse service solutions, operating within the solution graph of candidate service solutions.
- DSC-DT-C: This variant simultaneously considers functional attributes and constraints on QoS. It returns *k* diverse service solutions that fulfill system quality constraints. DSC-DT-C operates on the solution graph consisting of candidate service solutions compliant with the specified constraints.

- DSC-DT-O: Akin to DSC-DT-C, DSC-DT-O takes into cognizance both functional attributes and QoS constraints. Operating on a solution graph featuring candidate service solutions that adhere to the designated constraints, DSC-DT-O diverges in returning a set of k diverse service solutions, one among which excels in QoS performance.

TABLE I
EXPERIMENT CONFIGURATION

Factor	Experiment Set					
	#1	#2	#3	#4	#5	#6
Query Size	2-10	5	5	5	5	5
Number of Quality Constraints	2	1-9	2	2	2	2
Graph Size (Number of Nodes)	2000	2000	2000-10000	2000	2000	2000
Graph Density (Edges/Nodes)	6	6	6	1-10	6	6
Stringency of Quality Constraints	50	50	50	50	10-100	50
Value of k	10	10	10	10	10	5-50

In this study, for constraint queries and optimal queries, QoS constraints were spontaneously generated with the application of a predetermined constraint intensity. The efficacy of the DSC-DT methodology was quantitatively assessed through edge service coverage (ESC), while the computational efficiency was appraised by the runtime (expressed in milliseconds). Each experimental results represent the average value obtained from 1000 queries.

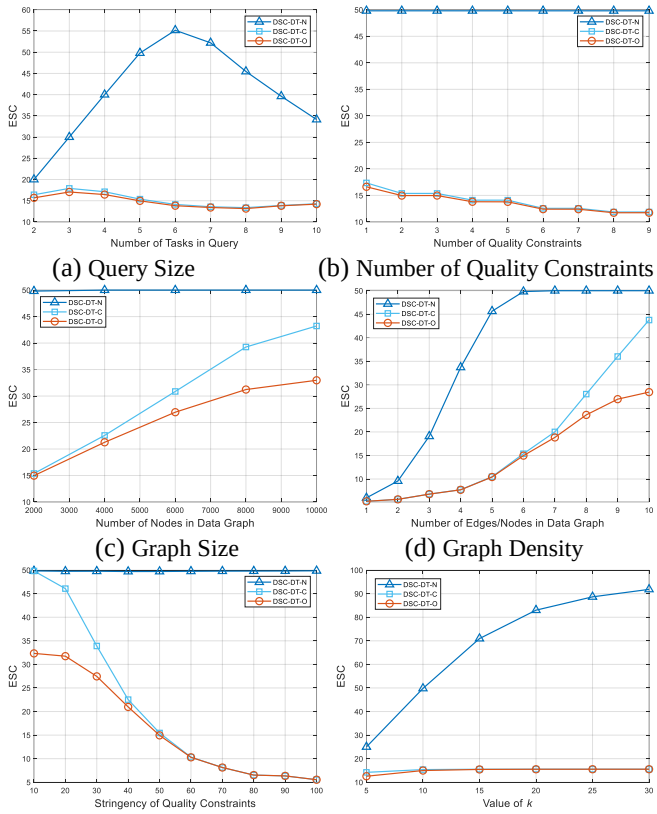
All algorithms were implemented using Python 2.7 and executed on an Ubuntu 18.04 LTS system housing an Intel i5-7300HQ CPU, clocked at 2.50 GHz, and equipped with 8 GB of RAM, thereby offering apt suitability for deployment as an edge server.

B. Effectiveness evaluation

Fig. 6. illustrates the diversity of the proposed DSC-DT method in returning the top-k service solutions under various factors. The term "ESC" denotes Edge Service Coverage, abbreviated to underscore the quantification of result set diversity, as per the definition in Definition 7. Elevated ESC values denote an augmented diversity within the array of solutions.

Fig. 6(a) illustrates the experimental results pertaining to scenario #1, where the influence of request size on the DSC-DT algorithm is investigated. The findings highlight that both the DSC-DT-C and DSC-DT-O approaches exhibit comparable ESC metrics. As the tally of tasks gradually expands, a slight initial increment followed by a modest decrement and eventual stabilization characterizes their ESC values. Moreover, Fig. 6(a) reveals that the ESC exhibited by DSC-DT-N surpasses those of the DSC-DT-C and DSC-DT-O variants. This divergence is attributable to the extensive scope of diversified exploration that DSC-DT-N embarks upon, traversing a broader solution graph including the entirety of candidate service solutions, thus yielding a more diverse ensemble of top-k service solutions for end users.

Fig. 6(b) presents the ESC within experimental scenario #2. Notably, as the number of quality constraints mounts, the ESC of both DSC-DT-C and DSC-DT-O experiences a gradual attenuation. In contrast, the ESC for the DSC-DT-N method remains impervious to fluctuations in the number of quality constraints, considering its exemption from the integration of system quality constraints. Moreover, as displayed from Fig. 6(b), the DSC-DT-N method exhibits an ESC value of 50, indicative of the absence of service overlap amid any two service solutions in the result set. This compellingly underscores the pronounced diversity characterizing the search results achieved through the DSC-DT-N methodology within experimental scenario #2.



(a) Query Size (b) Number of Quality Constraints

(c) Graph Size (d) Graph Density

(e) Stringency of Quality Constraints (f) Value of k

Fig. 6. Impact of factors on ESC

Fig. 6(c) showcases the results for scenario #3, which delves into the impacts of the count of nodes (i.e., edge services) within the data graph upon the DSC-DT algorithm. The findings unveil a linear increase in ESC metrics for all three methods, concomitant with the amplification of node numbers. For

instance, when confronted with 10,000 nodes, the DSC-DT-O returns a result set with an ESC of 33 for a business process request comprising two quality constraints and five tasks. This figure alludes to the fact that over 60% of the service solutions within the ultimate result sets are characterized by overlapping attributes, thereby accentuating the performance of the DSC-DT algorithm in orchestrating diversified exploration.

Fig. 6(d) unveils the findings pertinent to scenario #4, which studies the repercussions of density within the edge service data graph upon the DSC-DT algorithm. Akin to the observations in Fig. 6(c), the ESC values for all three methodologies exhibit an upswing as the data graph density increments.

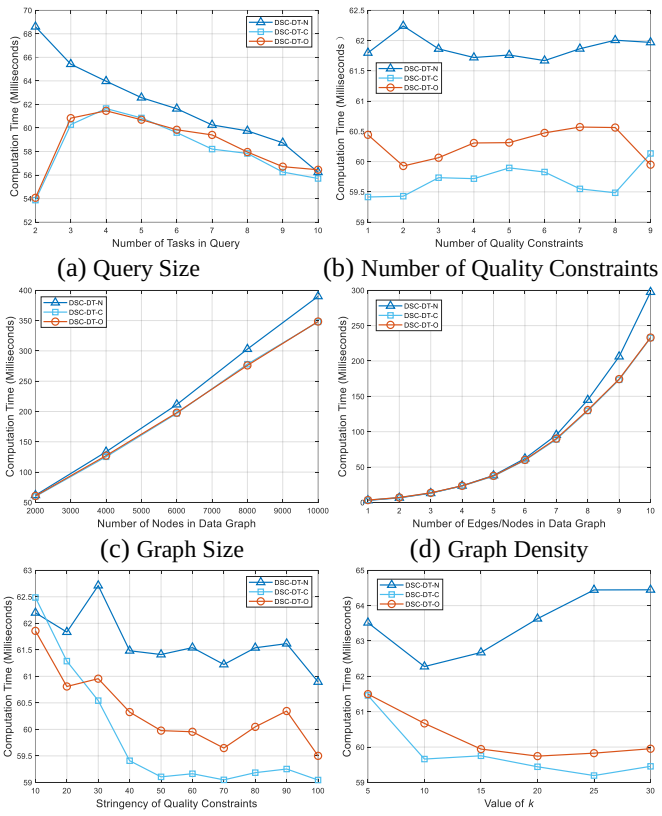
Fig. 6(e) presents the findings stemming from scenario #5, where the influence of the quality constraints' intensity on diversified results is explored. In aggregate, the DSC-DT-N method demonstrates the most robust ESC, trailed by DSC-DT-C. A notable observation unfolds as the intensity value ascends: initially, the results procured by DSC-DT-C closely mirror those of DSC-DT-N. However, with the continued elevation of intensity value, DSC-DT-C's performance gradually aligns with that of DSC-DT-O. This evolution is attributed to the intersection of low-intensity scenarios, wherein user-bound QoS constraints are lenient or, in certain instances, devoid of system QoS constraints. Under such circumstances, the solution graphs fashioned by DSC-DT-C and DSC-DT-N exhibit striking resemblance. Yet, as intensity value increases, the roster of candidate service solutions satisfying QoS prerequisites reduces considerably, potentially plummeting below the threshold of k.

Fig. 6(f) imparts the ESC results including scenario #6. In broad strokes, an augmented k corresponds to an increased ESC, corroborating the disclosures in Fig. 6(f). In tandem with incrementing k, the ESC for DSC-DT-N registers pronounced amplification. Remarkably, DSC-DT-C and DSC-DT-O exhibit nearly identical ESC values, barring an exception when $k = 5$. This divergence emerges owing to the potential excess of k in relation to the number of service solutions conforming to user-defined QoS prerequisites within the data graph, i.e., $k > |SG|$.

C. Efficiency evaluation

Fig. 7 illustrates the comprehensive efficiency of the proposed DSC-DT method in returning diversified service solutions to users across diverse factors. Notably, an intriguing observation is produced: the runtime of the DSC-DT-N methodology, which abstains from QoS constraints, generally surpasses that of the DSC-DT-C and DSC-DT-O methods, which factor in QoS constraints. This discrepancy emanates from the markedly augmented size of the solution graph caused by DSC-DT-N relative to the solution graphs fashioned by DSC-DT-C and DSC-DT-O. Elaborate detailed results is expounded below.

Fig. 7(a) outlines the runtime of the three variant methodologies across diverse counts of tasks in scenario #1. The runtime of the DSC-DT-N approach de-escalates in tandem with augmenting task counts, owing to the precipitous reduction in candidate service solutions capable of fulfilling user requisites. Conversely, the other two methodologies experience an upswing in runtime when confronted with relatively modest task counts, as they confront the task of selecting QoS-constrained service solutions from a profuse assortment of candidates. As task counts burgeon, parallel to DSC-DT-N, the contingent of candidate service solutions satisfying user prerequisites undergoes drastic diminution, prompting a decline in runtime. The results in Fig. 7(b) once again corroborate that DSC-DT-N incurs the most extended runtime. Fig. 7(c) illustrates the efficiency of the three variant methodologies across varying node counts. The results indicate that the runtime of DSC-DT increases concomitantly with the proliferation of nodes. Notwithstanding, even under extreme scenarios, such as those characterized by 10,000 nodes, both the DSC-DT-C and DSC-DT-O methodologies manage to provide users with top-k diversified service solutions within the confines of 400ms. Similar to Fig. 7(c), the results showcased in Fig. 7(d) expound the increased runtime of DSC-DT in response to increased data graph densities. The results in Fig. 7(e) indicate that the DSC-DT-N method, unburdened by quality constraints, exhibits negligible changes in runtime with fluctuations in intensity values. Generally, a higher k value corresponds to lengthier runtimes. Surprisingly, the empirical results from Fig. 7(f) reveal a contrary trend: the runtime of the DSC-DT-C and DSC-DT-O methodologies ebbs as k increases. This counterintuitive behavior can be attributed to the phenomenon where k surpasses the cardinality of the solution graph (|SG|) in these methodologies.



(a) Query Size (b) Number of Quality Constraints

(c) Graph Size (d) Graph Density

(e) Stringency of Quality Constraints (f) Value of k

Fig. 7. Impact of factors on computation time in milliseconds

Predicated on the comprehensive analysis of the experimental results, it is deduced that the proposed DSC-DT method excels

in efficaciously generating diversified service solutions. Its runtime remains relatively unperturbed by amplifications in task counts, variations in constraint metrics, or elevated *k* values. Hence, the efficiency exhibited by the DSC-DT method is judiciously suited for a myriad of practical applications.

V. Conclusion

In this study, we introduced an innovative edge service composition methodology termed DSC-DT, which harnesses digital twin technology and diverse search strategies to surmount the challenges entrenched within the MEC landscape. By simulating and evaluating an array of composite service solutions within a virtual space, DSC-DT proffers diversified options for consumer electronics. The empirical results corroborate the efficacy of DSC-DT in producing diverse composite service solutions, concurrently enhancing edge resource utilization and augmenting user experiences.

To summarize, this research contributes to the innovative exploration of edge service composition through DSC-DT, offering an effective solution to meet the diverse service requirements in MEC. However, there are still areas for improvement. In the field of edge computing, deep learning and reinforcement learning currently stand out as advanced technical solutions[38-40]. Future research directions include incorporating edge resource constraints and designing robust AI algorithms for scalability. Additionally, there exists potential for probing the pragmatic implementation and optimization of DSC-DT, delving into its deployment nuances and performance refinement. Additionally, the exploration of domain-specific applications such as industrial automation remains promising, inviting the prospect of interdisciplinary collaboration.

References

1. Cui, G.M., et al., Demand Response in NOMA-Based Mobile Edge Computing: A Two-Phase Game-Theoretical Approach. *IEEE Transactions on Mobile Computing*, 2023. **22**(3): p. 1449-1463.
2. Cui, G.M., et al., Location Privacy Protection via Delocalization in 5G Mobile Edge Computing Environment. *IEEE Transactions on Services Computing*, 2023. **16**(1): p. 412-423.
3. Wu, J.T., et al., Constraint-aware and multi-objective optimization for micro-service composition in mobile edge computing. *Software-Practice & Experience*, 2023. DOI: org/10.1002/spe.3217
4. Tian, H., et al., DIMA: Distributed cooperative microservice caching for internet of things in edge computing by deep reinforcement learning. *World Wide Web*, 2022. **25**(5): p. 1769-1792.
5. Yan, H.Z., et al., Edge server deployment for health monitoring with reinforcement learning in internet of medical things. *IEEE Transactions on Computational Social Systems*, 2022. DOI: 10.1109/TCSS.2022.3161996.
6. Jiang, Y., et al. TSC-ECFA: A Trusted Service Composition Scheme for Edge Cloud. *IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. 2021. p. 58-65.
7. Long, T., et al. A novel fault-tolerant approach to web service composition upon the edge computing environment. *In International Conference on Web Services*. 2021. p. 15-31.
8. Gao, B., et al., An Online Framework for Joint Network Selection and Service Placement in Mobile Edge Computing. *IEEE Transactions on Mobile Computing*, 2022. **21**(11): p. 3836-3851.
9. Zhang, Y., et al., Covering-based Web Service Quality Prediction via Neighborhood-aware Matrix Factorization. *IEEE Transactions on Services Computing*, 2019. **14**(5): p. 1333-1344.
10. Al Ridhawi, I., et al., A collaborative mobile edge computing and user solution for service composition in 5G systems. *Transactions on Emerging Telecommunications Technologies*, 2018. **29**(11).
11. Benouaret, K., et al. Top-k web service compositions using fuzzy dominance relationship. *IEEE International Conference on Services Computing*. 2011. p. 144-151.
12. Jiang, W., S.L. Hu, and Z.Y. Liu, Top K Query for QoS-Aware Automatic Service Composition. *IEEE Transactions on Services Computing*, 2014. **7**(4): p. 681-695.
13. Deng, S.G., et al., *Top-k Automatic Service Composition:* A Parallel Method for Large-Scale Service Sets. *IEEE Transactions on Automation Science and Engineering*, 2014. **11**(3): p. 891-905.
14. Li, J., et al., Full Solution Indexing for Top-K Web Service Composition. *IEEE Transactions on Services Computing*, 2018. **11**(3): p. 521-533.
15. Guo, C., et al., QoS-aware diversified service selection. *IEEE Transactions on Services Computing*, 2023. **16**(3): p. 2085-2099
16. Kang, G., et al. Diversified QoS-centric service recommendation for uncertain QoS preferences. *IEEE International Conference on Services Computing (SCC)*. 2020. p. 288-295.
17. Cheng, H., et al. Diversified keyword search based web service composition. *Journal of Systems and Software*, 2020. **163**: p. 110540.
18. Sun, W., et al., Reducing offloading latency for digital twin edge networks in 6G. *IEEE Transactions on Vehicular Technology*, 2020. **69**(10): p. 12240-12251.
19. Zhang, K., et al., Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks. *IEEE Transactions on Industrial Informatics*, 2021. **18**(2): p. 1405-1413.
20. Lu, Y., et al., Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks. *IEEE Transactions on Industrial Informatics*, 2020. **17**(7): p. 5098-5107.
21. Zhang, L., et al., Digital twin-assisted edge computation offloading in industrial Internet of Things with NOMA. *IEEE Transactions on Vehicular Technology*, 2023. DOI: 10.1109/TVT.2023.3270859
22. Li, B., et al., Digital twin assisted task offloading for aerial edge computing and networks. *IEEE Transactions on Vehicular Technology*, 2022. **71**(10): p. 10863-10877.
23. Xu, X., et al., Computation offloading and service caching for intelligent transportation systems with digital twin. *IEEE Transactions on Intelligent Transportation Systems*, 2022. **23**(11): p. 20757-20772.
24. Van Huynh, D., et al., URLLC edge networks with joint optimal user association, task offloading and resource

allocation: A digital twin approach. *IEEE Transactions on Communications*, 2022. **70**(11): p. 7669-7682.
25. Xu, X., et al., Service offloading with deep Q-network for digital twinning-empowered Internet of Vehicles in edge computing. *IEEE Transactions on Industrial Informatics*, 2020. **18**(2): p. 1414-1423.
26. Liu, W., et al., Energy Efficient Computation Offloading in Aerial Edge Networks With Multi-Agent Cooperation. *IEEE Transactions on Wireless Communications*, 2023. DOI: 10.1109/TWC.2023.3235997.
27. Yu, J., et al., Attention-based QoE-aware Digital Twin Empowered Edge Computing for Immersive Virtual Reality. *arXiv preprint arXiv:2305.08569*, 2023.
28. Guo, J., et al., Survey on digital twins for Internet of Vehicles: Fundamentals, challenges, and opportunities. *Digital Communications and Networks*, 2022. DOI: org/10.1016/j.dcan.2022.05.023.
29. Jiang, Y., et al., Digital twin-enabled real-time synchronization for planning, scheduling, and execution in precast on-site assembly. *Automation in Construction*, 2022. **141**: p. 104397.
30. Ni, Y., et al., NCSR: Negative-connection-aware service recommendation for large sparse service network. *IEEE Transactions on Automation Science and Engineering*, 2015. **13**(2): p. 579-590.
31. Huang, K., et al., An empirical study of programmable web: A network analysis on a service-mashup system. *IEEE 19th International Conference on Web Services*. 2012. p. 552-559.
32. Cordella, L.P., et al., A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 2004. **26**(10): p. 1367-1372.
33. Zeng, L., et al., QoS-aware middleware for web services composition. *IEEE Transactions on software engineering*, 2004. **30**(5): p. 311-327.
34. Zeng, L., et al. Quality driven web services composition. *Proceedings of the 12th international conference on World Wide Web*. 2003. p. 411-421
35. Wu, J., et al., Substructure similarity search for engineering service-based systems. *Journal of Systems and Software*, 2020. **165**: p. 110569.
36. Durrett, R., Random graph dynamics. *Cambridge: Cambridge university press*. 2007. **200**(7)
37. Blum, A., et al., A random-surfer web-graph model. *Proceedings of the Third Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, 2006. p. 238-246.
38. Yao, L., et al., Dynamic Edge Computation Offloading for Internet of Vehicles With Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022. **24**(11), p. 12991-12999.
39. Xu, X.L., et al., Safe: Synergic Data Filtering for Federated Learning in Cloud-Edge Computing. *IEEE Transactions on Industrial Informatics*, 2023. 19(2): p. 1655-1665.
40. Li, Z., et al., Federated Learning-Based Cross-Enterprise Recommendation With Graph Neural Networks. *IEEE Transactions on Industrial Informatics*, 2023. 19: p. undefined.