A Graph Neural Network and Pointer Network-Based Approach for QoS-Aware Service Composition

Xiao Wang, Hanchuan Xu, Xianzhi Wang, *Member, IEEE*, Xiaofei Xu, *Member, IEEE*, and Zhongjie Wang, *Member, IEEE*

Abstract—QoS-aware service composition aims to aggregate multiple existing services to meet users' complex functional and nonfunctional requirements that cannot be met by simple services. The accumulation of user tasks and service composition solutions makes it possible to mine empirical rules from those historical compositions to reduce the search space and thus improve the composition efficiency. Traditional empirical rule-based methods focus on mining with well-designed rules, ignoring the underlying correlations between tasks and between services. At the same time, infrequently used services are not valued by these methods, but these services may still be used in constructing optimal service solutions. In addition, many methods use reinforcement learning to compose services to efficiently construct service solutions, but they do not achieve the same effect as traditional meta-heuristic methods. In view of the above shortcomings, considering the ability of graphs to express relationships, we first construct tasks and services as graphs and then use a graph neural network (GNN) to mine underlying correlations and predict the probability that each service will be used to construct the solution corresponding to the task. Next, based on these high-probability services, we utilize pointer network-based reinforcement learning to efficiently construct the initial service solution. The pointer network is often used to solve combinatorial optimization problems and is noninferior to meta-heuristics in small-scale data. To increase its generalization, we superimpose another layer on the pointer network. Finally, to take advantage of infrequently used services, we use the local whale optimization algorithm to fine-tune the initial service solution to obtain an improved service solution. Experimental results show that our approach outperforms several existing methods in terms of both composition efficiency and solution quality.

Index Terms—Service composition, deep learning, graph neural network, pointer network, empirical rule.

1 Introduction

The rapid development of big data, cloud computing, and the Internet of things has increased the availability of software services on the Internet. Many services offer similar functions with different quality of service (QoS). Meanwhile, user tasks are becoming complex and include a workflow of many subtasks. This complexity makes it difficult to fulfill tasks with a single service. As a countermeasure, QoS-aware service composition enables the aggregation of multiple services into a QoS-optimal solution that not only fulfills the complex functional requirements but also optimizes the nonfunctional requirements of users [1]. Constructing a QoS-optimal service solution is a typical nondeterministic polynomial (NP) problem [2].

Service composition is one of the hot topics in service computing. Many methods have been developed for service composition [3]. We classify them into several categories according to the algorithms they use: exact algorithms [4], [5], meta-heuristic-based methods [6], [7], [8], reinforcement learning-based methods [9], [10] and graph-based methods [11], [12]. Although these "from-scratch" approaches have yielded good results on various datasets, they do not take advantage of historically successful service composition solutions.

Existing experience demonstrates that maximizing the use of empirical rules from historical service composition solutions can improve composition efficiency and solution quality [13], [14]. Some studies have explored mining empirical rules from historical service solutions and using them for service composition, such as the service domain-oriented artificial bee colony algorithm (S-ABC) [15], [16] and the service domain features-oriented genetic algorithm (SDF-GA) [17]. However, these methods usually have the following shortcomings:

- Their mining of empirical rules is relatively simple, such as reusing successful partial solutions, reusing frequently proposed task fragments, and designing rules to classify services; they lack deeper and more comprehensive mining.
- Although some algorithms explore the use of reinforcement learning to solve service composition problems, the effect is far from the effect of meta-heuristic algorithms. More algorithms use traditional meta-heuristic algorithms to construct service solutions, but these algorithms only converge after many iterations, resulting in low efficiency.
- They do not pay sufficient attention to infrequently used services. Although these services have a low probability of use, they may still be used in the

- *Xiao Wang, H. Xu, X. Xu and Z. Wang are with the Faculty of Computing, Harbin Institute of Technology, Harbin, China, 150001. E-mail: {wxlxq, xhc, xiaofei, rainy}@hit.edu.cn*
- *Xianzhi Wang is with the School of Computer Science, University of Technology Sydney, Australia. E-mail: XIANZHI.WANG@uts.edu.au*

Manuscript received ; revised .

optimal service solution.

For the first shortcoming, if we want to fully utilize the experience in historical compositions, we must mine the underlying correlations between tasks, between services, and between tasks and services. To capture the underlying correlation between services, a service relationship graph is constructed based on the service history usage records. Considering the ability of graphs to represent relationships, using GNNs on the service relationship graph, information between related services can be propagated to learn interdependent service features. To relate the subtasks to the corresponding constraints in the task, we construct the constraint graph. Using GNN on the constraint graph allows sharing features between subtasks and constraints and between subtasks. As a result, the learned task features contain subtask features, constraint features, and their relationships. Further, to mine the correspondence between tasks and services, we transform the mining problem into a multilabel classification problem. Combined with the representations of tasks and services, a multilabel classifier is trained to predict the probability that each service is used to construct the service solution corresponding to the task. Through the above two steps, we eliminate low-probability services and reduce the search space of the subsequent service composition.

To address the second shortcoming, since the search space has been reduced by GNN, a pre-trained model is considered to improve the composition speed while avoiding degrading the quality of the solution. Since most deep Q-learning-based service composition methods do not perform well, we use pointer network-based reinforcement learning to construct service solutions efficiently. The pointer network is not inferior to the meta-heuristic algorithm in small-scale data and is superior to other machine learning methods in solving combinatorial optimization problems. Meanwhile, it can also handle variable-length inputs [18]. Since the generalization of the network is weak in solving constrained optimization problems, we use a two-layer pointer network to increase the generalization.

To address the third shortcoming, we leverage all services to fine-tune the initial service solution through a local meta-heuristic algorithm. Here, we take full advantage of infrequently used services, treat all services equally (the frequently used services cannot be ignored, after all, they are more likely to be used), and avoid overusing empirical rules. After fine-tuning, we can obtain an improved service solution, and since it is only a partial fine-tuning, the efficiency can be guaranteed. Since there are numerous meta-heuristic algorithms that can be employed, it is difficult to find the most suitable one for this problem, so we pay more attention to the solution idea, that is, whether the use of low-probability services will further improve the quality of the solution. We compare several meta-heuristic algorithms and choose the shrink-wrapped strategy and spiral update strategy of the whale optimization algorithm, which performed the best in our experiments, as the fine-tuning algorithm.

Based on the above improvements for the three shortcomings, we propose a framework for solving the service composition problem, as shown in Figure 1, which includes three main approaches: GNN-based candidate service reduction, pointer network-based initial service solution construction, and whale optimization algorithm-based service solution fine-tuning.

The main contributions of this paper are as follows:

- We represent the historical tasks and service usages as graphs and pioneer the use of GNNs to mine empirical rules in historical compositions.
- To improve the composition efficiency on the premise of ensuring the solution quality, we use pointer network-based reinforcement learning to construct the initial service solution and add another layer to the pointer network to increase its generalization.
- We fine-tune the initial service solution by taking advantage of infrequently used services, avoiding the inadequacy of using empirical rules — potentially missing out on more optimal solutions.

The rest of this paper is organized as follows. In Section II, we discuss the related works. In Section III, we introduce the problem definition and the research framework. In Section IV, we describe our method. In Section V, the focus is on evaluating our method in comparison with others. Finally, Section VI offers some concluding remarks.

2 Related Work

In this section, we provide a review of service composition and its empirical rules, including three different perspectives: the empirical rules and their mining approaches in service composition, the empirical rule-based service composition approaches and traditional service composition approaches without empirical rules.

2.1 Empirical Rules and Their Mining in Service Composition

With the rapidly increasing number of available services and service solutions, discovering and exploiting empirical rules to improve composition efficiency from large historical compositions is a critical and time-consuming task [19], [20], [21]. Some studies utilize historical service solutions to generate reusable and flexible service solution fragments and classify them for subsequent extraction and reuse [22], [23]. Furthermore, some studies consider the problem from a task perspective, mining frequently proposed similar tasks and exploiting them in the modeling and service composition of new tasks [13], [24]. Although discovering and mining empirical rules has received substantial attention from researchers, these approaches are not sufficiently comprehensive. A complete empirical rule should cover the correlations between tasks, between services and between tasks and services. Moreover, the research depth is slightly insufficient: mining with rules alone is not adequate to mine all the underlying correlations.

2.2 Empirical Rule-Based Service Composition Approaches

Using the mined empirical rules, researchers design corresponding algorithms and complete service composition. Liu et al. [23] mine valuable fragments from historical service

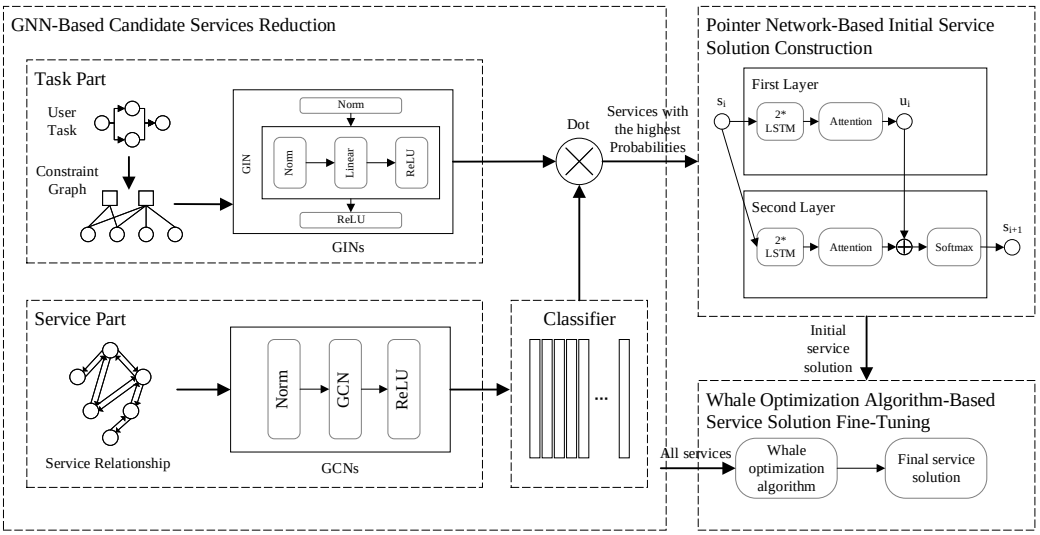

Fig. 1. Research Framework

solutions and incorporate these reusable service solution fragments into the global planning optimization algorithm (GP) and artificial bee colony algorithm (ABC) to improve the composition efficiency. Xu et al. [15] propose a service-domain-oriented ABC (S-ABC) that utilizes the *a priori* correlation and similarity of service. Liu et al. [16] improve S-ABC and use the C4.5 algorithm and neural network for parameter tuning and dependency establishment between problem features and parameters. Liu et al. [17] divide the candidate services into multiple service spaces according to the service domain characteristics and use the genetic algorithm to select services in these service spaces. Although these methods improve the composition efficiency and solution quality, as mentioned in Section 1, these methods are typically characterized by overuse of the empirical rules or insufficient research on algorithms.

2.3 Service Composition Approaches without Empirical Rules

For the service composition approaches without empirical rules, algorithmic innovation is especially important. Meta-heuristic-based algorithms and Q-learning-based algorithms are the two main approaches to solve the service composition problem. Gavvala et al. [7] propose an eagle strategy with the whale optimization algorithm (ESWOA) that ensures a proper balance between exploration and exploitation. Yang et al. [8] propose a dynamic ant-colony genetic hybrid algorithm, and the execution time of the two algorithms can be controlled dynamically based on the current solution quality. Wang et al. [9] adopt a recurrent neural network to improve Q-learning. The algorithm employs the heuristic behavior selection strategy to perform targeted behavior selection when facing different types of states. Liang et al. [10] propose a deep reinforcement learning algorithm that combines basic deep Q-learning, the dueling architecture, and the prioritized replay mechanism. Although these approaches have made considerable improvements, they are still limited to these two types of algorithms. The composition efficiency and solution quality cannot be balanced: meta-heuristic-based algorithms generate solutions of high quality but low efficiency, while Q-learning-based algorithms generate solutions of high efficiency but much lower quality than meta-heuristic-based algorithms.

Based on these previous works, we use GNN to mine empirical rules, use the pointer network for effective and efficient composition, and fine-tune the initial service solution with infrequently used services to further improve the performance.

3 Problem Definition

We divide the service composition problem into four components - user task, service solution, objective function, and constraints. The goal of the problem is to find a service solution that satisfies all subtasks (functional attributes) and constraints (nonfunctional attributes) with the optimal objective function value. These four components are defined below.

3.1 User Tasks

A user task is usually decomposed into several subtasks and the corresponding workflow in the service composition. Let $Task =< T, workflow >$, where $T = \{t_1, t_2, ..., t_n\}$ is the subtask set, $workflow$ is the priority between subtasks, and n denotes the number of subtasks for a $Task$.

3.2 Services and Service Solutions

Each subtask t_i is performed by the abstract service S_i, which is the collection of candidate services $S_i = \{s_{i,1}, s_{i,2}, ..., s_{i,n_i}\}$, where n_i refers to the number of candidate services for S_i. Thus, the service solution of $Task$ is $sol = \{s_1, s_2, ..., s_n\} \subseteq \bigcup_{i=1}^{n} S_i$, where s_i is a candidate service from S_i.

3.3 Objective Function

We use the objective function F to evaluate the fitness of sol. Each service s_i corresponds to a set of QoS values, $QoS_i = \{q_{i,1}, q_{i,2}, ...\}$, where $q_{i,l}$ denotes the l-th QoS value of service s_i. Since the minimization and maximization problems can be transformed into each other, we take the minimization problem as an example. The service composition for minimization of F is defined as follows:

$$\min F(sol) = \sum_{l=1}^{|QoS|} AGG(q_l) \times w_l \tag{1}$$

where $|QoS|$ is the number of QoS attributes, w_l is the weight of the l-th QoS value, which is fixed by the user based on his or her interest, $\Sigma_l w_l = 1$. $AGG(q_l)$ is the l-th QoS aggregation function value of sol. The calculation of $AGG(q_l)$ is based on the $workflow$ composed of sequential, parallel, conditional, and loop. Some common QoS aggregation functions are shown in Table 1, where p_i is the probability of s_i in the conditional structure. In addition, the QoS aggregation functions of the loop structure are calculated by multiplying the number of loops by the value of the aggregation function of the internal structure.

TABLE 1
QoS aggregation functions for different workflow structures

QoS attributes	Sequential	Parallel	Conditional(p_i)
Response Time	$\sum_{i=1}^{n} q_{i,rt}$	$\max q_{i,rt}$	$\sum_{i=1}^{n} q_{i,rt} \times p_i$
Throughput	$\min q_{i,tp}$	$\min q_{i,tp}$	$\sum_{i=1}^{n} q_{i,tp} \times p_i$
Reliability	$\prod_{i=1}^{n} q_{i,rel}$	$\prod_{i=1}^{n} q_{i,rel}$	$\prod_{i=1}^{n} q_{i,rel} \times p_i$
Availability	$\prod_{i=1}^{n} q_{i,avl}$	$\prod_{i=1}^{n} q_{i,avl}$	$\prod_{i=1}^{n} q_{i,avl} \times p_i$

3.4 Constraints

Constraints are the user QoS limits for one or more subtasks. Let the constraint set $C = \{c_1, c_2, ...\}$, where c_i can be an equation or inequality. We define the inequality constraint as an example:

$$c_i = AGG_{T_{c_i}}(q_l) < k_i \tag{2}$$

where T_{c_i} is the subtasks constraint by c_i, $AGG_{T_{c_i}}(q_l)$ is the l-th QoS aggregation function value of the subsolution corresponding to T_{c_i} and k_i is the constraint value of c_i. For example, the user has a constraint c_1 that requires the overall response time of the three subtasks t_1, t_2, t_3 to be less than 100ms. If they are sequential, $c_1 = \sum_{i=1}^{3} q_{i,rt} < 100$ms, where $q_{i,rt}$ is the response time of the service corresponding to t_i in the service solution.

4 METHODS

In this section, we first provide a general overview of the research framework, followed by a detailed description of the approaches used in each part of the framework.

4.1 Overview

The research framework shown in Figure 1 is divided into three main approaches.

GNN-Based Candidate Service Reduction: In this section, we use GNN to mine the empirical rules in the historical compositions and further predict the probability that each service is used to construct the service solution corresponding to the task. Subsequently, selecting the services with high probability for composition can effectively reduce the candidate service space and improve the composition efficiency. The approach mines the empirical rules from three different perspectives:

- The empirical rules between tasks: We transform the functional and nonfunctional attributes of the user task into a constraint graph (see Section 4.2.1) and then use a graph isomorphism network (GIN) [25] to learn the representation for the transformed task.
- The empirical rules between services: We construct a service relationship graph (see Section 4.2.3) to model the correlation between services and then use a graph convolutional network (GCN) [26] to mine the underlying correlation between services in the graph. Finally, we obtain a representation of each service.
- The empirical rules between tasks and services: We map the service representation into a multilabel classifier and train it with the correspondence between historical tasks and services. Application of the trained classifier directly to the task representation yields the probability that each service is used to construct the service solution corresponding to the task.

Pointer Network-Based Initial Service Solution Construction: In this section, we construct the initial service solution using the two-layer pointer network-based reinforcement learning on the top-k high-probability services obtained in the previous section. The first layer optimizes the constraints and finds a service solution that meets the constraints. The second layer optimizes the objective function to find an improved service solution while meeting the constraints.

Whale Optimization Algorithm-Based Service Solution Fine-tuning: In this section, we use the local meta-heuristic algorithm to fine-tune the initial service solution constructed in the previous section to take advantage of infrequently used services. After experiments, we choose to use the shrinking encircling strategy and spiral updating strategy in the whale optimization algorithm as the local meta-heuristic algorithm.

We summarize the main symbols in a table of notations, as shown in Table 2.

To make our framework easier for the reader to understand, we give a qualitative running example in Appendix A.

4.2 GNN-Based Candidate Service Reduction

In this section, we use GNN to mine the empirical rules in the historical compositions and obtain the probability that each service can be used to construct the service solution corresponding to the task. To make the presentation clearer, we present each part separately.

4.2.1 Constraint Graph

To relate the functional and nonfunctional attributes of the task, we propose the constraint graph $G =< V, E >$ as

TABLE 2
Summary of the Notations

Approaches	Notations	Descriptions
GNN-Based Candidate Service Reduction	G	The constraint graph constructed from the user task
	V	The node set consisting of the subtasks T and constraints C in G
	E	The edge set connecting subtasks and constraints in G
	D	The hidden dimension of each model in the candidate service reduction
	$\mathbf{N}$	The node feature consisting of the node category and node constraints
	$\mathbf{N_G}$	The node features of the constraint graph, i.e., the input of the task representation learning
	$\mathbf{V}$	The node embeddings of the constraint graph, i.e., the output of the task representation learning
	$\mathbf{v_G}$	The constraint graph embedding accumulated by node embeddings
	$\mathbf{S}$	The service feature consisting of the service category and the QoS
	$\mathbf{S_G}$	The service features of the service relationship graph, i.e., the input of the service representation learning
	$\mathbf{W}$	The service embeddings of the service relationship graph, i.e., the output of the service representation learning
	$\hat{\mathbf{y}}$	The service predicted probabilities multiplied by $\mathbf{S_G}$ and $\mathbf{v_G}$
Pointer Network-Based Initial Service Solution Construction	k	The top-k services with the highest probability corresponding to each subtask based on the results of the candidate service reduction
	d	The hidden dimension of each layer in the initial service solution construction
	$\mathbf{S}^{'}$	The service feature consisting of the QoS and constraints
	$\mathbf{S}_{PN}$	The service matrix including $k\|T\|$ service features, i.e., the input of the pointer network
	$\mathbf{S}^{'}{}_{PN}$	The input matrix after a linear layer
	$\mathbf{ENC}$	The latent memory state matrix of the encoder
	$\mathbf{enc}_i$	The i-th latent memory state of the encoder
	$\mathbf{dec}_i$	The i-th latent memory state of the decoder
	$\mathbf{u}_i^j$	The output of the pointer mechanism to $\mathbf{dec}_i$ in j-th layer pointer network
Whale Optimization Algorithm-Based Service Solution Fine-tuning	pop	The population size
	t_max	The maximum number of iterations
	$\mathbf{sol_i}$	The service solution corresponding to the i-th individual
	$\mathbf{sol_{init}}$	The initial service solution constructed by the pointer network
	$\mathbf{sol}^*$	The current optimal solution
	$\mathbf{D}, \mathbf{D}^{'}, A, a, c, r$	Other parameters used in the whale optimization algorithm

the input of the task model, where $V = T + C$ is the node set consisting of the subtask and constraints and $E = \{e_1, e_2, ...\}$ is the edge set. Edge $e_k =< c_i, t_j >$ indicates that there is an edge e_k between c_i and t_j when $t_j \in T_{c_i}$.

In addition, the constraint graph consists of two special merging rules.

- If c_i and c_j constrain the same subtask set, i.e., $T_{c_i} = T_{c_j}$, they are merged into one node.
- If c_i constrains only one subtask, i.e., $T_{c_i} = \{t_j\}$, c_i and t_j are combined into one node.

For example, in Figure 2, there are 5 subtasks $T = \{t_1, t_2, t_3, t_4, t_5\}$ and 4 constraints $C = \{c_1, c_2, c_3, c_4\}$, where the subtask sets corresponding to each constraint are $T_{c_1} = T_{c_2} = \{t_1, t_2, t_3, t_4, t_5\}, T_{c_3} = \{t_1, t_2\}$ and $T_{c_4} = \{t_5\}$. Since $T_{c_1} = T_{c_2}$, according to merging rule 1, c_1 and c_2 are merged into one node, which is represented as a dashed box containing c_1 and c_2 in Figure 2. Since c_4 constrains only one subtask t_5, according to merging rule 2, c_4 and t_5 are merged into one node, which is represented as a dashed box containing c_4 and t_5 in Figure 2.

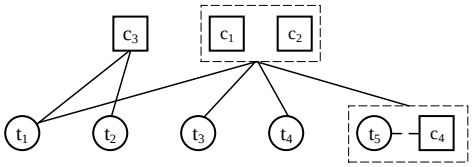

Fig. 2. Constraint graph

In the constraint graph, the node feature $\mathbf{N} = \text{concat}(\mathbf{n}_1, \mathbf{n}_2) \in \mathbb{R}^{1+2|QoS|}$ is represented by two parts, where concat means concatenating two vectors. The first part is for the node category, $\mathbf{n}_1 \in \mathbb{R}^1$. For constraint nodes, the node category is 0. For subtask nodes, the node category is from 1 to n. The second part is for the node constraints, $\mathbf{n}_2 \in \mathbb{R}^{2|QoS|}$. Each QoS occupies two dimensions: the former dimension is the constraint minimum value and the latter dimension is the constraint maximum value. If there is no constraint minimum value for a QoS, set it to 0 (before this, Min-Max normalization is performed for each QoS value). Similarly, when there is no constraint maximum value, set it to 1.

4.2.2 Task Representation Learning

Graph representation learning is a hot topic of recent research. It can convert graph data into a vector that machine learning can process to support subsequent optimization and analysis. Researchers have proposed a number of graph representation learning approaches [25], [27]. In this section, we use the GIN [25] as the representation learning model for constraint graphs. Let $\mathbf{v}_i \in \mathbb{R}^D$ be the node representation vector, where D is the dimension of the node representation. For the first layer, the input is a $\mathbf{N}_G \in \mathbb{R}^{|T| \times (1+2|QoS|)}$ matrix. For the last layer, the output is $\mathbf{V} \in \mathbb{R}^{|T| \times D}$. GIN updates $\mathbf{v}_i$ as

$$\mathbf{v}_i' = h_\Theta\left((1+\epsilon) \cdot \mathbf{v}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{v}_j\right) \tag{3}$$

where $\mathcal{N}(i)$ is the set of the nodes directly connected to node i and h_Θ denotes a neural network, for which we use a multilayer perceptron (MLP). ϵ is a trainable parameter.

After obtaining the output of the last layer, we use summation to aggregate the node features and obtain the graph representation vector $\mathbf{v}_G \in \mathbb{R}^D$:

$$\mathbf{v}_G = \sum_{i \in V} \mathbf{v}_i \tag{4}$$

The GNNs will be trained together with the classifier and more details will be provided in Section 4.2.4.

4.2.3 Service Relationship Graph

In this section, we define the correlation between services by mining the co-occurrence of services in the historical compositions, i.e., the service relationship graph.

Before introducing the service relationship graph, we first define the service feature. As for the constraint graph node feature, the service feature $\mathbf{S} = \text{concat}(\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{R}^{1+|QoS|}$ contains two parts. The first part $\mathbf{s}_1$ is the service category, where $\mathbf{s}_1 \in \mathbb{R}^1$ is from 1 to $|T|$. The second part $\mathbf{s}_2$ is the QoS, where $\mathbf{s}_2 \in \mathbb{R}^{|QoS|}$. Each standardized QoS occupies one dimension.

In the service relationship graph, nodes are services and edges are conditional probabilities between services. An edge pointing from s_i to s_j denotes the probability that service s_j is used under the condition that service s_i is used. To construct the service relationship graph, we first calculate the number of occurrences n_i for each service s_i and the number of co-occurrences $n_{i,j}$ for each two services s_i and s_j in the historical service solutions. Then, we approximate

the conditional probability $p(s_j|s_i) \cong \frac{n_{i,j}}{n_i}$, which is the edge pointing from s_i to s_j in the graph, and the edge weight is the value of the conditional probability.

Figure 3 shows a subgraph of the service relationship graph that contains three services: s_1, s_2 and s_3. The figure shows that the edge weight from s_1 to s_2 is different from that from s_2 to s_1, indicating that when s_2 appears in the service solution, s_1 will also occur with a high probability ($p(s_1|s_2) = 0.7$). However, when s_1 appears in the service solution, s_2 will not necessarily occur ($p(s_2|s_1) = 0.2$).

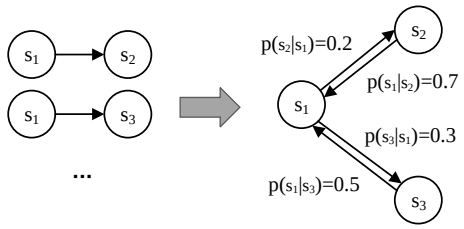

Fig. 3. Service relationship graph

4.2.4 Classifier Learning

GNNs are widely used in multilabel classification problems [28], [29]. In this section, we learn the service representation in the service relationship graph via a GCN [26] and map the service representation as a multilabel classifier. The trained classifier is applied to the task representation to obtain the selection probability of each service.

Let $\mathbf{w}_i \in \mathbb{R}^D$ be the service representation vector. For the first layer, the input is the matrix of the service relationship graph $\mathbf{S}_G \in \mathbb{R}^{|S|\times(|1+|QoS|)}$. For the last layer, the output is $\mathbf{W} \in \mathbb{R}^{|S|\times D}$. GCN updates $\mathbf{w}_i$ as

$$\mathbf{w}_i' = \Theta \sum_{s_j \in \mathcal{N}(s_i)\cup\{s_i\}} \frac{p(s_j|s_i)}{\sqrt{d_j d_i}} \mathbf{w}_j \tag{5}$$

with $d_i = 1 + \sum_{s_j \in \mathcal{N}(s_i)} p(s_j|s_i)$. $\Theta \in \mathbb{R}^{D\times D}$ is a trainable parameter matrix. By applying the learned classifier to the output $\mathbf{v}_G$ of the task part, we can obtain the predicted probabilities $\hat{\mathbf{y}} \in \mathbb{R}^{|S|}$ as

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{v}_G) \tag{6}$$

where σ is a sigmoid function.

We assume that the correct classifications is $\mathbf{y}$. Our network is trained using binary cross-entropy loss as follows:

$$loss = -\frac{1}{|S|}\sum_{i=1}^{|S|}(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)) \tag{7}$$

where $y_i = \{0, 1\}$ indicates whether s_i is used to construct the service solution corresponding to the task and $\hat{y}_i$ is the predicted probability of s_i.

4.3 Pointer Network-Based Initial Service Solution Construction

In this section, we first find the top-k services with the highest probability corresponding to each subtask based on the results of the candidate service reduction section. These $|T| \times k$ services are then used as input and optimized using two-layer pointer network-based reinforcement learning to obtain an initial service solution sol_{init}.

4.3.1 Service Feature

The service feature $\mathbf{S}' \in \mathbb{R}^{|QoS|+2|C|}$ consists of the QoS and constraints, where the QoS part is the same as in Section 4.2.3, and the constraint part is constraint $\mathbf{s}_2 \in \mathbb{R}^{2|C|}$. Each constraint c_i occupies two dimensions. If subtask t_j corresponding to service s_j is in T_{c_i}, the values of the service in these two dimensions are the minimum and maximum constraint values; otherwise, both values are set to 0.

4.3.2 Pointer Network

The pointer network (PN) [30] is a novel policy model for solving combinatorial optimization problems. The trained model performs well on many problems, such as the traveling salesman problem (TSP) [18] and the vehicle routing problem (VRP) [31]. Another advantage of PN is that it can handle variable-length inputs [30], making the approach much more adaptable. It uses a pointing mechanism, resembling the attention mechanism [32], to make the model point to a specific position in the input sequence (i.e., the input services).

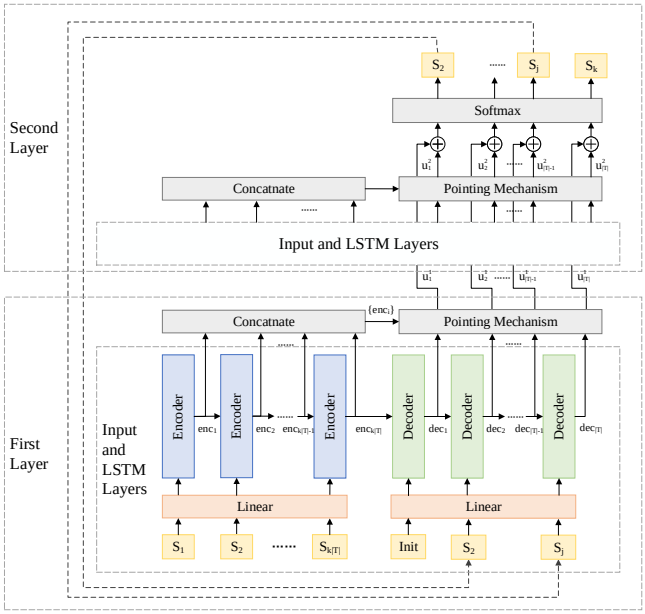

Fig. 4. Two-layer pointer network

A one-layer PN consists of two long short-term memory neural networks (LSTM) [33], i.e., an encoder and a decoder, a point mechanism, a d-dimensional vector of trainable parameters, and a linear layer. It consists of three main steps as follows.

- Input: After the candidate service reduction, we find the top-k services with the highest probabilities based on each subtask and encode them according to the rules in Section 4.3.1 to obtain the service matrix $\mathbf{S}_{PN} \in \mathbb{R}^{k|T|\times(|QoS|+2|C|)}$. This matrix is passed through a linear layer to obtain the input to the model $\mathbf{S}'_{PN} \in \mathbb{R}^{k|T|\times d}$, where d is the output dimension of the linear layer.

- Encoder: The encoder of PN is a d-dimensional LSTM. We transform $\mathbf{S}'_{PN}$ into an input sequence containing $k|T|$ vectors. At this time, each vector in the sequence is a service. These services are input to the LSTM in turn to get latent memory state (hidden state) matrix $\mathbf{ENC} = \mathrm{LSTM}(\mathbf{S}'_{PN}) \in \mathbb{R}^{k|T| \times d}$, where the row vector $\mathbf{enc}_i \in \mathbb{R}^d$ is the i-th hidden state of the encoder.
- Decoder: Like the encoder, the decoder is also an LSTM. We define a d-dimensional vector of trainable parameters $\mathbf{Init}$ as the initial input to the decoder. Meanwhile, $\mathbf{enc}_{k|T|} \in \mathbb{R}^d$ from the encoder (i.e., the last hidden state) is also input as the initial hidden state to the decoder. The first hidden state output from the decoder is saved as $\mathbf{dec}_1 = \mathrm{LSTM}(\mathbf{Init}, \mathbf{enc}_{k|T|}) \in \mathbb{R}^d$. Then, $\mathbf{dec}_1$ is input as the query with $\mathbf{ENC}$ from the encoder into the pointer mechanism to get $\mathbf{u}_1 = \mathbf{ENC} \times \mathbf{dec}_1 \in \mathbb{R}^{k|T|}$. Here, we choose the most straightforward dot product as our pointer mechanism. However, other mechanisms such as the multi-head attention mechanism can also be used. In the service composition problem, each subtask corresponds to only one service. Therefore, to avoid the duplication of the same category of services in the final solution, we also need to adjust $\mathbf{u}_1$ by modifying the weights of the latter $(k-1)|T|$ dimensions to $-\infty$ so that the service corresponding to the first subtask can be correctly selected. The modified $\mathbf{u}_1$ is passed through a softmax layer to obtain the selection probabilities of the services. Finally, the selected service is input into the decoder as the next vector. Thus, after $|T|$ selections, we can construct the initial service solution based on the selected services.

Since there are many constraints in the task, we use a two-layer pointer network to increase the generalization of the model. With the two-layer pointer network, we restrict the results of the first layer to feasible service solutions so that the optimization of the objective function F at the second layer can find a more optimal solution among these service solutions.

Our two-layer PN depicted in Figure 4 differs from the above steps in only the encoder step. The output of the first layer pointer mechanism $\mathbf{u}_i^1$ is directly added to the output of the second layer pointer mechanism $\mathbf{u}_i^2$. Thus, the training results of the first layer can be used as a guide for the second layer to select services that are more likely to meet the constraints.

4.3.3 Optimization Strategies

The pointer network can be trained by both supervised learning with labels and reinforcement learning (samples without labels). Two reasons motivate us to choose reinforcement learning to optimize PN over supervised learning with labels.

1) In supervised learning with labels, the model's performance is closely related to the quality of the supervised labels. However, in practice, the cost (i.e., label generation time) of obtaining high-quality labels is very high.
2) Combinatorial optimization problems usually have multiple feasible solutions, and other solutions can guide the model to converge to the optimal solution. However, labels can usually only record the optimal solution. Limited by their generalization ability, training with labels cannot predict the results completely and accurately, resulting in poor training performances. To solve this problem, further processing of the labels is required, bringing more preprocessing costs.

Therefore, we use a policy-based reinforcement learning algorithm [34] to optimize the parameters of the pointer network, denoted as $\boldsymbol{\theta}$. The training objectives are different in the two-layer PN. For the first layer, the training objective is the number of constraint violations. For the second layer, the training objective is the objective function value F. We take the second layer as an example; given the input service sequence seq, the training objective is defined as

$$J(\boldsymbol{\theta}|seq) = \mathbb{E}_{\pi \sim p_\theta(.|seq)} F(\pi|seq) \tag{8}$$

where we replace sol_{init} with the generic state notation π in reinforcement learning. $p_\theta(\pi|seq)$ is the selection probability of π, which can be decomposed as

$$p(\pi|seq) = \prod_{i=1}^{|T|} p(\pi(i)|\pi(<i), seq) \tag{9}$$

The gradient of Equation 8 with the policy gradient method is:

$$\nabla_\theta J(\boldsymbol{\theta}|seq) = \mathbb{E}_{\pi \sim p_\theta(.|seq)} \Big[\Big(F(\pi|seq) - b(seq) \Big) \nabla_\theta \log p_\theta(\pi|seq) \Big] \tag{10}$$

In the actual training process, each batch will have B samples. This gradient can be approximated via Monte Carlo sampling as:

$$\nabla_\theta J(\boldsymbol{\theta}|seq) \approx \frac{1}{B} \sum_{i=1}^{B} \Big(F(\pi_i|seq_i) - b(seq_i) \Big) \nabla_\theta \log p_\theta(\pi_i|seq_i) \tag{11}$$

where $b(seq_i)$ is a baseline function.

We use active search [18] for training. In the active search, $b(seq_i)$ is unified into a parameter b, and this parameter is continuously updated during training. The training pseudo-code for a batch B is shown in Algorithm 1.

Algorithm 1 Active search training

Input: Batch size B, Parameter β.
Output: Network parameters θ.

```
for i ∈ {1..B} do
    π_i ~ SAMPLESOLUTION(p_θ(·|seq_i));
    Calculate F(π_i|seq_i);
end for
b = b × β + (1 − β) × ( (1/B) Σ_{i=1}^{B} F(π_i|seq_i) )
g_θ = (1/B) Σ_{i=1}^{B} ( F(π_i|seq_i) − b ) ∇_θ log p_θ(π_i|seq_i)
θ = ADAM(θ, g_θ)
return θ
```

For our two-layer PN, the first layer is trained first. Then, after the first layer results have converged, the two layers

are trained together. The service sequence from the second layer's output is the initial service solution obtained.

4.4 Whale Optimization Algorithm-Based Service Solution Fine-tuning

Since the initial service solution is constructed of only services with high probability, in this section, we fine-tune the initial service solution in the global service repository to take advantage of infrequently used services.

We compare several local strategies of classical evolutionary algorithms and finally choose the shrinkage envelope strategy and spiral update strategy from the whale optimization algorithm (WOA) for local optimization (see Section 5.3.6 for comparison results). WOA mimics the behavior of humpback whales swimming towards their prey and performs well in classical optimization problems [35].

In this section, we use an integer encoding scheme. The service solution is defined as a vector $\mathbf{sol}_i = \{p_{s_1}, p_{s_2}, ..., p_{s_{|T|}}\}$, where s_j denotes the p_{s_j}-th candidate service of S_j.

The pseudo-code of the algorithm is presented in Algorithm 2.

Algorithm 2 Service solution fine-tuning

Input: $\mathbf{sol}_{init}$, S
Output: $\mathbf{sol}^*$

```
Initialize pop, t_max; sol* = sol_init; t = 0;
Randomly generate pop service solutions sol_i;
while t < t_max do
    calculate parameters by Equation 13;
    for each sol_i do
        update sol_i by Equation 12;
        if F(sol_i) < F(sol*) then
            sol* = sol_i;
        end if
    end for
end while
return sol*
```

First, the population size pop, the maximum number of iterations t_{max}, and the current optimal solution $\mathbf{sol}^* = \mathbf{sol}_{init}$ are initialized. Then, pop service solutions $\mathbf{sol}_i$ are randomly generated.

Each $\mathbf{sol}_i$ is the updated using the following equations [36], where the first equation corresponds to the shrinking encircling strategy and the second equation corresponds to the spiral updating strategy.

$$\mathbf{sol}_i^t = \begin{cases} \mathbf{sol}^* - A \cdot \mathbf{D}, & p < 0.5, |A| < 1 \\ \mathbf{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \mathbf{sol}^*, & p > 0.5 \end{cases} \tag{12}$$

where p is a random number from 0 to 1, l is a random number from -1 to 1, and b is a parameter to ensure the logarithmic shape. The other parameters in the equation are defined as follows:

$$\begin{cases} \mathbf{D} = c \cdot \mathbf{sol}^* - \mathbf{sol}_i^{t-1} \\ \mathbf{D}' = \mathbf{sol}^* - \mathbf{sol}_i^{t-1} \\ A = 2a \cdot r - a \\ c = 2r \end{cases} \tag{13}$$

where a is decreased from 2 to 0 linearly in every iteration and r is a random number from 0 to 1.

After t_{max} iterations, the algorithm terminates. The services corresponding to each dimension in $\mathbf{sol}^*$ are composed to obtain the final service solution.

5 Experimental Analysis

In this section, we first introduce the experimental setup and the dataset and then calibrate the parameters in the model. Finally, we compare the calibrated approach with other novel approaches to demonstrate the effectiveness of our approach.

5.1 Experimental Setup and Dataset

Since there is no publicly available dataset with tasks and services on the service composition problem, as the majority of experiments in service composition [7], [8], [37], we use synthetic tasks with publicly available service QoS datasets for our experiments.

We synthesize two datasets; the first one uses the services from the normalized QWS dataset [38], including various QoS attributes. To improve the composition complexity, we remove the service categories with the number of services less than 5. The cleaned service dataset has a total of 2507 services in 48 categories.

In addition, to demonstrate the generalizability of our approach, we synthesize a more extensive and differently distributed dataset. We first count the QoS in the QWS dataset, and the result shows that it follows a long-tailed distribution. Then, we synthesize a dataset (Normal dataset) with 50 abstract services, each with 500 services. The QoS in our synthetic dataset is shown to obey a normal distribution by the Shapiro–Wilk test [39]. Except for the difference in the number of abstract services for the tasks, the rest of the design of the Normal dataset is similar to the QWS dataset.

We randomly select from 20 to 30 (from 40 to 50 for the Normal dataset) abstract services (subtasks) from the service dataset to form the user task and randomly add global and local constraints. According to such rules, four thousand tasks are generated as a task dataset, of which 3000 are used for model training and 1000 are used for algorithm performance testing.

We use a pruning search algorithm (an exact algorithm with a longer time) to generate the optimal service solution as the historical service solution dataset. These historical service solutions are combined with the corresponding tasks to train multilabel classifiers or to test the performance of the algorithm.

All algorithms are evaluated in terms of the composition time and solution quality. The definition of solution quality is as follows:

$$sq = \frac{F(sol^*)}{F(sol)} \tag{14}$$

where sol^* denotes the optimal service solution. The optimal solution has a solution quality of 1. All other solutions have values between 0 and 1. The greater the solution quality, the better the solution.

We use response time and throughput as the objective functions and use availability and reliability as the constraints. The objective function is defined as follows.

$$\min F(sol) = \frac{1}{2}\Big[\frac{\sum_{i=1}^{n} q_{i,rt}}{n} + \max(1 - q_{i,tp})\Big] + v \tag{15}$$

TABLE 3
The performance of the multilabel classification approach with different numbers of GIN and GCN layers

Layers		Performance Evaluation			
GINs	GCNs	QWS		Normal	
		p@1	p@5	p@1	p@5
2	2	**0.9190**	**0.5152**	0.9330	0.7648
2	3	0.9050	0.5084	0.9730	0.8148
2	4	0.9120	0.5134	**0.9930**	**0.8360**
3	2	0.9130	0.5142	0.9170	0.7700
3	3	0.8900	0.5032	0.9700	0.8254
3	4	0.8920	0.4974	0.9930	0.8328
4	2	0.8940	0.4928	0.9000	0.7634
4	3	0.8910	0.5052	0.9480	0.7960
4	4	0.8940	0.5152	0.9750	0.8360

where $q_{i,rt}$ and $q_{i,tp}$ are the response time and throughput of the i-th service in sol and v is the number of constraint violations.

Since availability and reliability are calculated in the same way, we take availability as an example and define the constraint c_i as shown below.

$$c_i = \prod_{s_j \in S_{c_i}} q_{j,avl} > k_i \tag{16}$$

where $q_{j,avl}$ is the availability of the j-th service in sol, S_{c_i} is the service constraint by c_i, and k_i is the constraint value of c_i.

Due to space limitations, the statistics for the distribution of the data are shown in the Appendix B.

All experiments are executed on a PC with an 8-core CPU, 32.0 GB RAM, and a 10.0 GB GPU.

5.2 Parameter Calibration

We calibrate the parameters in the model to obtain better results for our approach.

5.2.1 Calibration of the number of GIN and GCN layers

In the GNN-based candidate service reduction, with GIN and GCN as the main components of the approach, the number of layers is essential for the model testing performance. Therefore, we calibrate the number of GIN and GCN layers in this section.

We evaluate the performance of the multilabel classification in the GNN-based candidate service reduction with 2 to 4 GIN layers and 2 to 4 GCN layers using the classic $p@k$ in the extreme multilabel classification problem. $p@k$ is defined as follows:

$$p@k = \frac{1}{k} \sum_{l \in \text{rank}_k} y_l \tag{17}$$

where rank_k denotes the predicted highest probability k labels and y_l denotes the actual value of the l-th label. In this experiment, we set $k = 1, 5$.

We run 20 epochs for each set of parameters and record the best $p@k$. The results are shown in Table 3.

For the QWS dataset, the best performance is obtained when the number of GIN and GCN layers is 2, and for the Normal dataset, the best performance is obtained when the number of GIN layers is 2 and the number of GCN layers is 4. A greater number of model layers does not necessarily lead to better results. A reasonable explanation for the result is that as the number of model layers increases, the propagation between nodes will accumulate and lead to oversmoothing.

Therefore, in the subsequent experiments, we choose a 2-layer GIN and a 2-layer GCN for the QWS dataset and a 2-layer GIN and a 4-layer GCN for the Normal dataset.

5.2.2 Calibration of top-k between the candidate service reduction and the initial service solution construction

Before the initial service solution construction, it is necessary to choose a suitable k and find the top-k services based on the results of the candidate service reduction.

Since the composition time is essentially the same, we only compare the solution quality for each value. We experiment for k from 2 to 10, and the results are shown in Figure 5.

Figure 5 indicates that when $k < 6$, the solution quality increases roughly with increasing k, but when $k \geq 6$, the solution quality does not change substantially as k increases. As k increases, the number of services available for optimization increases, as does the quality of the solution. However, when k exceeds a certain threshold (5 in the experiments), the limitations of the two-part model for the candidate service reduction and the initial service solution construction lead to an inability to continue increasing the solution quality. The reasons are as follows. First, the candidate service reduction approach has already selected the high-probability services. The vast majority of the remaining services with small probability are not suitable for this task, and increasing the value of k does not cause much change in the quality of the solution. Second, when the pointer network has an excessive number of inputs, the quality of the solution output from the pointer network is limited by the network's generalization ability. Although the training dataset results will be better, they will not have much effect on the results of the test dataset.

Therefore, we set $k = 5$ for the QWS dataset experiment. For the Normal dataset, we do the same experiment and set $k = 10$.

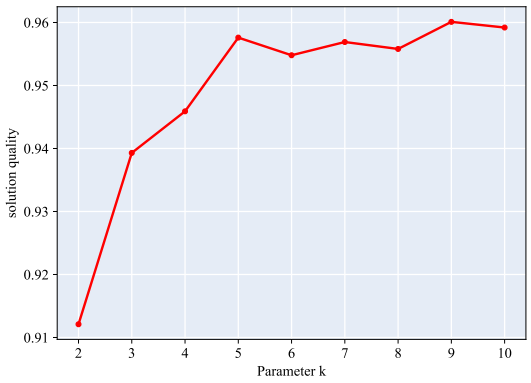

Fig. 5. The performance of the initial service solution construction approach with different k in the QWS dataset

TABLE 4
Comparison of performance for different deep learning approaches

Algorithms		ML	C-Tran	SCoNE	Q2L
QWS	p@1	**0.9190**	0.9151	0.8090	0.9184
	p@5	0.5152	**0.5179**	0.3990	0.5125
	Training Time(min)	5.833	8.043	5.297	8.111
Normal	p@1	**0.9930**	0.8820	0.9530	0.9890
	p@5	**0.8360**	0.8126	0.8200	0.8280
	Training Time(min)	31.81	28.31	26.96	35.38

5.3 Ablation Experiments

To demonstrate that each module in the network has a positive impact on the algorithm's performance, we decompose each module and design ablation experiments.

The results shown in Sections 5.3.2, 5.3.4, 5.3.5 and 5.3.6 are for the QWS dataset, and the results for the Normal dataset are similar. However, due to space limitations, the results of the Normal dataset are omitted.

5.3.1 Comparison of Approaches with and without GNN in terms of Candidate Service Reduction

To demonstrate the effectiveness of using GNN, in this section, we compare our GNN-based candidate service reduction approach (ML) with three state-of-the-art multi-label classification approaches without using GNN: C-Tran [40] (CNN and Transformer-Based), SCoNE [41] (CNN and Attention-Based), and Q2L [42] (Transformer-Based). We modify the necessary parts, such as the input and output formats and some preprocessing models that do not fit the problem (e.g., Bert, for text representation of labels). As described in Section 5.2.1, we evaluate the performance with $p@k$. The comparison results are shown in Table 4.

For the performances of the approaches, our approach performs better than the other approaches in most of the indicators, and only p@5 in the QWS dataset is slightly lower than C-Tran. For the generalizations of the approaches, Q2L and our approach are applicable to all datasets, while C-Tran is more applicable to small-scale datasets (QWS, 2.5k labels) and SCoNE is more applicable to large-scale datasets (Normal, 25k labels).

Therefore, after balancing the performance and generalization, we conclude that the architecture with GNN is better than the architectures without GNN in this problem.

5.3.2 Impact of the Service Relationship Graph on the Candidate Service Reduction

The service relationship graph (SRG) defines the correlation between services by mining the conditional probability of their co-occurrence with other services in the historical compositions. In this section, we demonstrate the effectiveness of SRG via experiments.

We evaluate the performance of our multilabel classification in the GNN-based candidate service reduction (ML) with SRG and without SRG using p@5 defined in Equation 17, where ML without SRG replaces the GCN layer with a linear layer and the other parts are unchanged. The results are shown in Figure 6.

By comparing the two curves with different epochs, we can find that ML with SRG outperforms ML without SRG in terms of convergence speed and algorithm performance. Therefore, SRG proves to be effective.

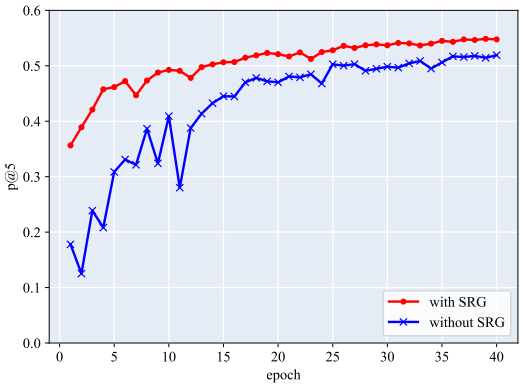

Fig. 6. Comparison of p@5 with SRG and without SRG in the QWS dataset

TABLE 5
Comparison of the performance for PN training approaches

Training Approaches		Supervised Learning with Labels		Reinforcement learning
Label Generation Rule		Optimal	ML+ESWOA	-
QWS	Solution Quality	0.8698	0.8670	**0.9576**
	Label Generation Time (min)	1000+	43.76	-
	Training Time (min)	1.388	1.405	48.18
Normal	Solution Quality	0.8909	0.8698	**0.9482**
	Label Generation Time (min)	1000+	165.9	-
	Training Time (min)	1.572	1.536	72.99

5.3.3 Comparison of Different Training Approaches for the Initial Service Solution Construction

To demonstrate the effectiveness of using reinforcement learning to optimize PN, we compare it with supervised learning with labels.

We generate labels of different qualities with the following two different rules:

1) Generate labels with a pruned search algorithm, obtaining optimal solution labels.
2) Generate labels with ML+ESWOA, the best-performing baseline in the experiments (see Section 5.4).

The results are shown in Table 5. For the solution quality, regardless of which label generation rules are compared, reinforcement learning is far better than other approaches. Meanwhile, although their training times are shorter, the total time is closer to or even exceeds that of reinforcement learning after adding the label generation time because reinforcement learning does not need to generate labels in advance. Therefore, we can conclude that using reinforcement learning to train pointer networks is superior to using supervised learning with labels.

5.3.4 Impact of Modules on the Initial Service Solution Construction

For the two-layer pointer network-based initial service solution construction, we first use the ML module to reduce

the search space and then add a layer to PN to improve the generalization. In this section, we illustrate the effect of these two modules on the result of the initial service solution construction with experiments.

We add a layer to PN, ML module to PN, and ML module and a layer to PN (the model used in this paper), respectively, named 2PN, ML+PN, and ML+2PN. Then, we compare the solution qualities of the initial service solution construction based on the three models with different epochs. The results are shown in Figure 7.

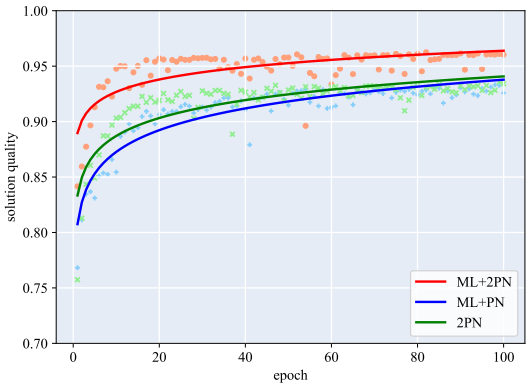

Fig. 7. Comparison of solution qualities of ML+2PN, ML+PN and 2PN in the QWS dataset

Since the results of each epoch are relatively scattered, we fit the results of each algorithm as a curve.

First, observing the ML+PN curve, we can see that although we add the ML to reduce the search space, ML+PN still has a gap with 2PN. This result proves that we need a model with stronger generalization ability.

Then, by comparing the results of 2PN and ML+2PN, we can find that the convergence of ML+2PN is faster than that of 2PN, and the solution quality is better. Although the range of services available is larger in 2PN, most services are not suitable for this task. The excessive search space reduces the convergence speed of 2PN.

Finally, by comparing the three curves, we demonstrate that the two modules positively impact the initial service solution construction result.

5.3.5 Impact of Modules on the Service Solution Fine-Tuning

The trained ML+2PN module can obtain a near-optimal service solution with very high efficiency. However, if the user is still unsatisfied with this service solution, we use the WOA-based service solution fine-tuning (WOA(local)) to optimize it further. In this section, we illustrate the effect of the previous modules (ML and 2PN) on the result of the service solution fine-tuning via experiments. Since using the local meta-heuristic algorithm alone results in low effectiveness, we use its corresponding global meta-heuristic algorithm (WOA) instead.

We add the ML module and ML+2PN module before WOA(local), respectively, named ML+WOA(local) and ML+2PN+WOA(local), and compare the solution quality of these three approaches. ML+WOA(local) means we choose the services with the highest probability from ML as the initial service solution and execute WOA(local). The results are shown in Figure 8.

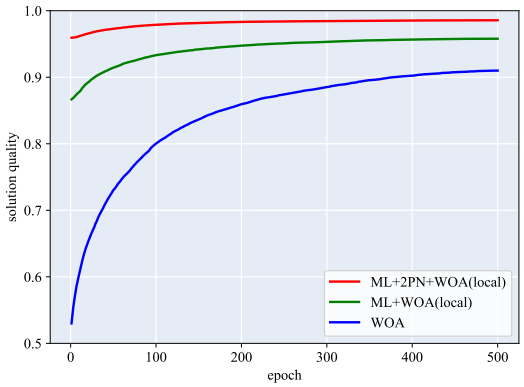

Fig. 8. Comparison of solution qualities of WOA, ML+WOA(local) and ML+2PN+WOA(local) in the QWS dataset

By comparing the curves of WOA with those of the other two algorithms, we can find that in terms of both the convergence speed and the solution quality, ML+2PN+WOA(local) is significantly better than the other two algorithms. The solution quality of ML+WOA(local) at 500 epoch is only close to the initial solution quality of ML+2PN+WOA(local), and the solution quality of WOA is even worse. Therefore, we can conclude that each module added before the service solution fine-tuning positively impacts the service solution.

5.3.6 Selection of the Local Meta-Heuristic Algorithm in the Service Solution Fine-Tuning

As mentioned in Section 4.4, we compare the meta-heuristic algorithms used in the service solution fine-tuning to demonstrate that our chosen local strategy in WOA outperforms the other algorithms.

We compare the local policies in three algorithms commonly used in service compositions, genetic algorithm (GA) [43], artificial bee colony algorithm (ABC) [15], and ant colony optimization algorithm (ACO) [8].

To satisfy the requirements of local fine-tuning, we must adapt the algorithms. For GA, we always keep the initial service solution in the population. If there is no better solution, the other individuals in the population gradually approach the initial solution and perform a local search. For ABC, we initialize the leader bee on the initial service solution, and the follower bee will search locally in the neighborhood of that solution. For ACO, since ACO in DAAGA [8] is responsible for the local search and GA is responsible for the global search, we directly use the local ACO as the local search strategy.

The parameters of the three baselines and WOA are set according to those in previous papers [7], [8], [15], [43]. To reduce the effect of search time, we fix the algorithm running time and proportionally increase or decrease the parameters that affect that time.

The results shown in Table 6 demonstrate that WOA outperforms the other three algorithms in both the QWS and Normal datasets. Therefore, we choose the local strategy in WOA as the service solution fine-tuning approach.

TABLE 6
Comparison of solution qualities of different local meta-heuristic algorithms

Algorithms	Datasets	
	QWS	Normal
GA	0.9712	0.9601
ABC	0.9789	0.9630
ACO	0.9752	0.9589
WOA	**0.9831**	**0.9653**

5.4 Algorithm Comparisons

To evaluate the performance of our approach, we compare it with recently proposed approaches.

Empirical rule-based service composition approaches are the focus of the comparison. We choose SDFGA [17] and DPKSD [24] as the baselines. These methods mine empirical rules in two different ways: designing rules to classify services and reusing successful partial solutions and frequently proposed task fragments.

Additionally, we compare our approach with algorithms without empirical rules. We choose the recently proposed algorithms ESWOA [7], DAAGA [8] (two meta-heuristic algorithms) and PDDQN [10] (a reinforcement learning algorithm) as the baselines. Due to the weak performance of PDDQN alone and the fact that this approach and our pointer network-based initial service solution construction approach are both reinforcement learning algorithms, we add an ML module in front of this algorithm to improve its performance.

To demonstrate the effect of 2PN, we replace 2PN in ML+2PN with DAAGA and ESWOA, named ML+DAAGA and ML+ESWOA, and involved in the comparison.

The parameters of the above baselines are either adopted from the original paper or increased proportionally to ensure convergence.

We compare the average solution quality (demonstrating the performance of the approaches), the standard deviation (demonstrating the stability of the approaches), composition time and training time obtained by ML, ML+2PN, and ML+2PN+WOA(local) with that of the five baselines. The results are shown in Table 7.

First, we compare the solution quality of the approaches. The performance of ML is the poorest, not only in terms of its average solution quality but also in terms of its stability. Due to the limitation of the model generalization, some of the solutions are not of high quality (manifested by a high number of violated constraints), which reduces the overall performance of the method.

To solve this problem, we use 2PN to optimize the services with higher probabilities obtained by ML. ML+2PN has outperformed all approaches without ML. Compared with ML+metaheuristics (i.e., ML+ESWOA and ML+DAAGA), ML+2PN is superior to the two comparison methods on the Normal dataset but superior to ML+DAAGA and inferior to ML+ESWOA on the QWS dataset. Considering the composition time, using a two-layer pointer network after ML achieves the desired result of increasing the composition efficiency without degrading the solution quality.

Further, we add WOA(local) to ML+2PN. Compared with other approaches, we can find that ML+2PN+WOA(local) performs much better. Therefore, our approach can construct an improved service solution in less time than the other approaches.

Then, we compare the composition time and training time of the approaches and give a trade-off for them. Since our methods are based on deep learning and reinforcement learning, the composition times are shorter than other methods, but there is an extra training time. For ML, although the composition time and training time are short, the solution is with lower quality and less stable, so the method is not suitable for using alone. ML+metaheuristics (i.e., ML+ESWOA and ML+DAAGA) trade a longer composition time for better solution quality. Therefore, they are suitable for providing better service solutions for users who do not care about the composition time when the training time is insufficient. ML+2PN trades a longer training time for a very short composition time and better solution quality. Therefore, it is suitable for providing better service solutions for users who need immediate access to service solutions when the training time is sufficient. ML+2PN+WOA(local) gives the best solution quality but requires a longer training time and some composition time. Therefore, it is suitable for providing service solutions for users who prefer the solution quality when the training time is sufficient.

In addition, to demonstrate that our method is not completely dependent on WOA(local), we add WOA(local) to the baselines and compare our method with them.

The results for the QWS dataset are shown in Table 8 and those for the Normal dataset are similar. Although the addition of the local algorithm improves the quality of the baselines' solution qualities, the results are still worse than those of our method. We omit the results of ML+PDDQN because this method has already utilized the results of ML.

Due to space limitations, the results of the Friedman test [44] and Nemenyi test [45] which are often used to compare the performance of multiple models [45] are shown in the Appendix C.

6 Conclusion and Future Work

With the massive increase in candidate services and user tasks, the service composition problem has become more complex. Researchers have found that mining empirical rules from historical compositions can effectively reduce the search space and improve composition efficiency. Existing methods do not mine the empirical rules comprehensively and deeply, and they do not pay attention to services that are not frequently used. Therefore, in this paper, we propose a graph neural network and pointer network-based approach to address the shortcomings of existing methods. Our approach consists of three methods: GNN-based candidate service reduction, pointer network-based initial service solution construction, and whale optimization algorithm-based service solution fine-tuning. The candidate service reduction approach first mines empirical rules using graph neural networks and predicts the probability that each service is used to construct the service solution corresponding to the task. The initial service solution construction approach uses a two-layer pointer network to construct an initial

TABLE 7
Comparison of the performances of different approaches. * indicates that ML+2PN+WOA(local) significantly outperforms the best baseline based on the Nemenyi test ($p - value < 0.005$). - indicates that the time is very short or meaningless.

Approaches	QWS			Normal		
	Solution Quality	Composition Time(s)	Training Time(min)	Solution Quality	Composition Time(s)	Training Time(min)
SDFGA	0.8809±0.1428	1.183	<1	0.8581±0.1669	3.751	3.340
DPKSD	0.9015±0.1453	0.9524	<1	0.8770±0.1489	3.287	<1
ESWOA	0.8901±0.1512	1.334	-	0.8707±0.1648	4.197	-
DAAGA	0.8923±0.1387	1.822	-	0.8756±0.1524	4.369	-
ML	0.8520±0.2305	-	5.833	0.8049±0.2267	-	31.81
ML+PDDQN	0.8856±0.1375	1.568	5.833	0.8355±0.1621	3.239	31.81
ML+ESWOA	0.9609±0.0699	0.9691	5.833	0.9364±0.0443	3.295	31.81
ML+DAAGA	0.9475±0.0724	1.464	5.833	0.8994±0.0509	3.925	31.81
ML+2PN	0.9576±0.0574	-	54.01	0.9482±0.0229	-	104.8
ML+2PN+ WOA(local)	**0.9831*±0.0199**	0.6848	54.01	**0.9653*±0.0180**	2.149	104.8

TABLE 8
Comparison of the performances after adding WOA(local) to the baselines in the QWS dataset

Approaches	Solution Quality	Composition Time(s)
SDFGA+WOA(local)	0.9352	1.842
DPKSD+WOA(local)	0.9417	1.576
ESWOA+WOA(local)	0.9158	1.995
DAAGA+WOA(local)	0.9383	2.427
ML+2PN+WOA(local)	**0.9831**	**0.6848**

service composition solution. Finally, to take advantage of infrequently used services, the service solution fine-tuning approach fine-tunes the initial service solution to construct an improved service solution. Comparative experiments with five other related methods are used to evaluate the performance of our method.

Since the datasets used in the experiments are synthetic, one of our future works is to construct a real task and service dataset to support service compositions. Simultaneously, we will explore new and better network structures used for each part of the framework to improve the composition efficiency and solution quality.

ACKNOWLEDGMENTS

Research in this paper is supported by the National Key Research and Development Program of China (No.2021YFB3300700) and the National Natural Science Foundation (NSF) of China (Nos.61832004, 61832014).

REFERENCES

[1] Q. Z. Sheng, X. Qiao, A. V. Vasilakos *et al.*, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218–238, 2014.

[2] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on software engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[3] A. S. da Silva, H. Ma, Y. Mei *et al.*, "A survey of evolutionary computation for web service composition: A technical perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 538–554, 2020.

[4] V. Gabrel, M. Manouvrier, and C. Murat, "Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming," in *ICSOC*. Springer, 2014, pp. 108–122.

[5] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Hybrid optimization algorithm for large-scale qos-aware service composition," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 547–559, 2015.

[6] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for qos-aware cloud service composition," *Journal of Intelligent Manufacturing*, vol. 29, no. 8, pp. 1773–1792, 2018.

[7] S. K. Gavvala, C. Jatoth, G. Gangadharan *et al.*, "Qos-aware cloud service composition using eagle strategy," *Future Generation Computer Systems*, vol. 90, pp. 273–290, 2019.

[8] Y. Yang, B. Yang, S. Wang *et al.*, "A dynamic ant-colony genetic algorithm for cloud service composition optimization," *The International Journal of Advanced Manufacturing Technology*, vol. 102, no. 1, pp. 355–368, 2019.

[9] H. Wang, M. Gu, Q. Yu *et al.*, "Adaptive and large-scale service composition based on deep reinforcement learning," *Knowledge-Based Systems*, vol. 180, pp. 75–90, 2019.

[10] H. Liang, X. Wen, Y. Liu *et al.*, "Logistics-involved qos-aware service composition in cloud manufacturing with deep reinforcement learning," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 101991, 2021.

[11] P. Wang, Z. Ding, C. Jiang *et al.*, "Automatic web service composition based on uncertainty execution effects," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 551–565, 2015.

[12] P. Hennig and W.-T. Balke, "Highly scalable web service composition using binary tree-based parallelization," in *ICWS*. IEEE, 2010, pp. 123–130.

[13] X. Xu, R. Liu, Z. Wang *et al.*, "Re2sep: A two-phases pattern-based paradigm for software service engineering," in *2017 IEEE World Congress on Services (SERVICES)*. IEEE, 2017, pp. 67–70.

[14] X. Xu, G. Motta, Z. Tu *et al.*, "A new paradigm of software service engineering in big data and big service era," *Computing*, vol. 100, no. 4, pp. 353–368, 2018.

[15] X. Xu, Z. Liu, Z. Wang *et al.*, "S-abc: A paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition," *Future Generation Computer Systems*, vol. 68, pp. 304–319, 2017.

[16] R. Liu, Z. Wang, and X. Xu, "Parameter tuning for s-abcpk: An improved service composition algorithm considering priori knowledge," *IJWSR*, vol. 16, no. 2, pp. 88–109, 2019.

[17] T. Li, T. He, Z. Wang *et al.*, "Sdf-ga: a service domain feature-oriented approach for manufacturing cloud service composition," *Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 681–702, 2020.

[18] I. Bello, H. Pham, Q. V. Le *et al.*, "Neural combinatorial optimization with reinforcement learning," in *The 5th International Conference on Learning Representations (ICLR), Toulon, France, April 24-26, 2017*, 2017.

[19] J. Liu, J. Jiang, X. Cui *et al.*, "Power consumption prediction of web services for energy-efficient service selection," *Personal and ubiquitous computing*, vol. 19, no. 7, pp. 1063–1073, 2015.

[20] H. Wang, X. Xu, Z. Wang *et al.*, "Analyzing the influence of domain features on the optimality of service composition algorithm," in *SCC*. IEEE, 2015, pp. 427–434.

[21] S. Wang, Z. Wang, and X. Xu, "Mining bilateral patterns as priori

knowledge for efficient service composition," in *ICWS*. IEEE, 2016, pp. 65–72.
[22] R. Yang, B. Li, J. Wang *et al.*, "Scky: A method for reusing service process fragments," in *ICWS*. IEEE, 2014, pp. 209–216.
[23] R. Liu, X. Xu, Z. Wang *et al.*, "Probability matrix of request-solution mapping for efficient service selection," in *ICWS*. IEEE, 2017, pp. 444–451.
[24] H. Xu, X. Wang, Y. Wang *et al.*, "Domain priori knowledge based integrated solution design for internet of services," in *SCC*. IEEE, 2020, pp. 446–453.
[25] K. Xu, W. Hu, J. Leskovec *et al.*, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.
[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[27] F. M. Bianchi, D. Grattarola, L. Livi *et al.*, "Graph neural networks with convolutional arma filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
[28] Z.-M. Chen, X.-S. Wei, P. Wang *et al.*, "Multi-label image recognition with graph convolutional networks," in *CVPR*, 2019, pp. 5177–5186.
[29] R. You, Z. Guo, L. Cui *et al.*, "Cross-modality attention with semantic graph embedding for multi-label classification," in *AAAI*, vol. 34, no. 07, 2020, pp. 12 709–12 716.
[30] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *NIPS*, vol. 28, 06 2015.
[31] M. Nazari, A. Oroojlooy, M. Takáč *et al.*, "Reinforcement learning for solving the vehicle routing problem," in *NIPS*, 2018, pp. 9861–9871.
[32] S. Chaudhari, V. Mithal, G. Polatkan *et al.*, "An attentive survey of attention models," *arXiv preprint arXiv:1904.02874*, 2019.
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] R. S. Sutton, D. A. McAllester, S. P. Singh *et al.*, "Policy gradient methods for reinforcement learning with function approximation," in *NIPS*, 2000, pp. 1057–1063.
[35] F. S. Gharehchopogh and H. Gholizadeh, "A comprehensive survey: Whale optimization algorithm and its applications," *Swarm and Evolutionary Computation*, vol. 48, pp. 1–24, 2019.
[36] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.
[37] C. Jatoth, G. Gangadharan, U. Fiore *et al.*, "Qos-aware big service composition using mapreduce based evolutionary algorithm with guided mutation," *Future Generation Computer Systems*, vol. 86, pp. 1008–1018, 2018.
[38] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *WWW*, 2008, pp. 795–804.
[39] J. A. Villasenor Alva and E. G. Estrada, "A generalization of shapiro–wilk's test for multivariate normality," *Communications in Statistics—Theory and Methods*, vol. 38, no. 11, pp. 1870–1883, 2009.
[40] J. Lanchantin, T. Wang, V. Ordonez *et al.*, "General multi-label image classification with transformers," in *CVPR*, 2021, pp. 16 478–16 488.
[41] K. Pham, K. Kafle, Z. Lin *et al.*, "Learning to predict visual attributes in the wild," in *CVPR*, 2021, pp. 13 018–13 028.
[42] S. Liu, L. Zhang, X. Yang *et al.*, "Query2label: A simple transformer way to multi-label classification," *arXiv preprint arXiv:2107.10834*, 2021.
[43] A. E. Yilmaz and P. Karagoz, "Improved genetic algorithm based approach for qos aware web service composition," in *ICWS*, 2014, pp. 463–470.
[44] P. B. Nemenyi, *Distribution-free multiple comparisons*. Princeton University, 1963.
[45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

Xiao Wang received his B.SC., M.SC. in mechanical engineering, computer science and technology from Harbin Institute of Technology, Harbin, China, in 2017 and 2020. He is currently pursuing his Ph.D. in Computer Science and Technology at Harbin Institute of Technology, Harbin, China. His research interests include service computing, service compositions, graph neural network, etc.

Hanchuan Xu received his B.SC., M.SC. and Ph.D. degrees in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 1999, 2003 and 2011, respectively. He is currently a lecturer at the Faculty of Computing, Harbin Institute of Technology, Harbin, China. He is the author or co-author over more than 20 academic papers. His current research interests include service computing, software service engineering, enterprises computing.

Xianzhi Wang is a lecturer at School of Computer Science, University of Technology Sydney. His research lies in the fields of Internet of Things (IoT), data mining, machine learning, recommender systems, and cybersecurity, with a focus on misinformation detection, cognitive analysis, and domain big data analytics (health, transport, e-commerce). His work has been published in top-tier journals and conferences such as IEEE TNNLS, IEEE MC, IEEE TSC, ACM TIST, ACM TOIT, ICDM, KDD, AAAI, IJCAI, UbiComp, SIGIR, CIKM, ER, PAKDD, IJCNN, ICSOC, ICWS.

Xiaofei Xu is a professor at Faculty of Computing, and vice president of the Harbin Institute of Technology. He received the Ph.D. degree in computer science from Harbin Institute of Technology in 1988. His research interests include enterprise intelligent computing, services computing, internet of services, and data mining. He is the associate chair of IFIP TC5 WG5.8, chair of INTEROP-VLab China Pole, fellow of China Computer Federation (CCF), and the vice director of the technical committee of service computing of CCF. He is the author of more than 300 publications. He is member of the IEEE and ACM.

Zhongjie Wang is a professor at Faculty of Computing, Harbin Institute of Technology (HIT). He received the Ph.D. degree in computer science from Harbin Institute of Technology in 2006. His research interests include services computing, mobile and social networking services, and software architecture. He is the author of more than 80 publications. He is a member of the IEEE.