Interaction-Oriented Service Entity Placement in Edge Computing

Yu Liang, Jidong Ge, *Member, IEEE,* Sheng Zhang, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* Lingwei Pan, Tengfei Zhang, and Bin Luo, *Member, IEEE*

Abstract—Distributed Interactive Applications (DIAs) such as virtual reality and multiplayer online game usually require fast processing of tremendous data and timely exchange of delay-sensitive action data and metadata. This makes traditional mobile-based or cloud-based solutions no longer effective. Thanks to edge computing, DIA Service Providers (DSPs) can rent resources from Edge Infrastructure Providers (EIPs) to place service entities that store user states and run computation-intensive tasks. One fundamental problem for a DSP is to decide *where* to place service entities to achieve low-delay pairwise interactions between DIA users, under the constraint that the total placement cost is no more than a specified budget threshold. In this paper, we formally model the service entity placement problem and prove that it is NP-complete by a polynomial reduction from the set cover problem. We present GPA, an efficient algorithm for service entity placement, and theoretically analyze its performance. We evaluated GPA with both real-world data trace-driven simulations, and observed that GPA performs close to the optimal algorithm and generally outperforms the baseline algorithm. We also output a curve showing the trade-off between the weighted average interaction delay and the budget threshold, so that a DSP can choose the right balance.

Index Terms—Edge computing, distributed interactive applications, interaction delay, service entity placement.

1 Introduction

Distributed Interactive Applications (DIAs), e.g. multiplayer online game [1], virtual or augmented reality [2], and collaborative computer-aided design [3], are becoming popular, as they allow a group of geographically-distributed users to interact with each other synchronously via their mobile devices or computers. A DIA usually consists of two components: service entity and client. A service entity maintains application metadata (including user state and application state) [4], while a client (user) is only responsible for sending user-initiated operations to the service entities and receiving updates from the service entities. A service entity can serve a certain number of clients, depending on the service capacity of the entity. In a traditional DIA architecture, service entities are placed in the remote cloud, while clients are computers or mobile devices.

A typical interaction between two DIA users consists of three phases. Firstly, a user u sends an operation to u's service entity; secondly, after necessary computations, u's service entity sends the results to v's service entity; lastly, after necessary computations, v's service entity sends the results to user v. Since DIAs are human-in-the-loop applications, it is important to improve the interactivity of DIAs by minimizing pairwise interaction delays. However, in a traditional DIA architecture, the wide area network (WAN) latency between a DIA user and its service entity is very unlikely to improve in the foreseeable future, since the primary targets of WAN networking today are security and manageability, which are at odds with delay [5].

Edge computing is one of the emerging technologies aiming to enable fast computation at the network edge [6–8]. Small-scale edge servers are placed at the network edge, geographically near users and data. Thanks to edge computing, the interaction delay in DIAs can be reduced by moving service entities from remote clouds to nearby edge servers. To achieve this, DIA Service Providers (DSPs) can rent resources from Edge Infrastructure Providers (EIPs) to place service entities that store user states and run computation-intensive tasks, and DIA users can reach these service entities via local wireless connections. It is easy to see that, the interaction delay is directly affected by where the DSP places service entities. Usually, placing a service entity at different edge servers may have different placement costs. Therefore, one fundamental problem for a DSP is to *decide where to place service entities to minimize the weighted average interaction delay between users, under the constraint that the total placement cost is no more than a specified budget threshold*. We refer to this problem as the Interaction-oriented Service Entity Placement (ISEP) problem.

Fig. 1 shows an example. Multiple edge servers are connected through a metropolitan-area network. Suppose the service entities of u_1 and u_2 are located at edge servers s_2 and s_4, respectively, then the dashed

- *Y. Liang, J.D. Ge, S. Zhang, L.W. Pan, T.F. Zhang, and B. Luo are with the State Key Lab. for Novel Software Technology, Nanjing University, China. Y. Liang, J.D. Ge, L.W. Pan, T.F. Zhang, and B. Luo are also with the Software Institute, Nanjing University. S. Zhang is also with the Department of Compute Science and Technology, Nanjing University. E-mail: {dg1832003, mg1832005, mf1832244}@smail.nju.edu.cn, {gjd, sheng, luobin}@nju.edu.cn.*
- *J. Wu is with the Center for Networked Computing, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.*

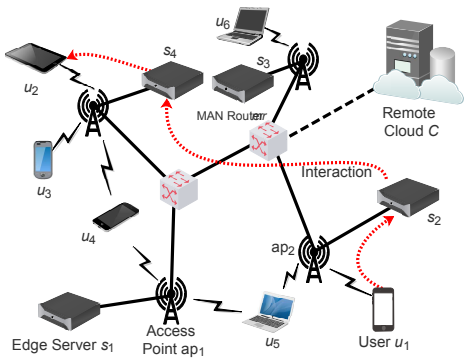


Fig. 1: An example scenario to illustrate the edge computing-enabled DIA architecture. Edge servers are connected through a metropolitan-area network. Suppose the service entities of u_1 and u_2 are located at s_2 and s_4, respectively, then the dashed red line shows the interaction path between u_1 and u_2.

red line shows the interaction path between u_1 and u_2. From the perspective of a DSP, it should carefully decide the placement of service entities under a given budget threshold, so as to minimize the weighted average pairwise interaction delay.

The ISEP problem is non-trivial due to the following intertwined challenges. First, when multiple service entities are placed, each user has to choose one as its service entity. The way each user chooses its service entity also affects the pairwise interaction delay. Second, the placement costs at different edge servers are heterogeneous, and making the placement decisions should respect such heterogeneity. Last but not least, the interaction path between any two users consists of three parts, and it is impossible to give a closed-form expression for an interaction delay. These intrinsically intertwined challenges together complicate the ISEP problem.

In this paper, we formally model the ISEP problem and prove that it is NP-complete by a polynomial reduction from the set cover problem. We present GPA, an efficient algorithm for placing service entities, and theoretically analyze its performance. We evaluated GPA with both real-world data traces and large-scale simulations, and observed that GPA performs close to the optimal algorithm and generally outperforms the baseline algorithm. Our main contributions are three-fold: (1) To the best of our knowledge, we are the first to present the interaction-oriented edge-enabled service entity placement problem and prove that it is NP-complete. (2) We design an efficient algorithm for service entity placement and theoretically give the performance gap between GPA and the optimum. (3) Both real-world data traces and large-scale simulations show that the proposed algorithm performs close to the optimal algorithm and generally outperforms the baseline algorithm.

The rest of the paper is organized as follows. We survey related work in Section 2. We introduce the background and problem in Section 3. The NP-completeness results are presented in Section 4. We then present an efficient algorithm in Section 5. Evaluation is given in Section 6. We conclude the paper and discuss limitations in Section 7.

2 Related Work

There are many works considering efficient offloading in edge computing. The multi-user computation partition problem with the objective of minimizing the average completion time for all users was studied in [9, 10]. Time slot assignment for energy-efficient mobile offloading is investigated in [11]. Chen et al. [12] adopted a game theoretic approach to solve the multi-user multi-channel computation offloading problem. Zhang et al. [13] focused on minimizing the completion time of mobile workloads. Some other works proposed offloading mobile workloads to nearby edges or remote clouds from the architecture perspective [10, 11, 14–18].

Some studies focused on job dispatching in edge computing. Tong et al. [19] proposed a hierarchical edge architecture and designed a heuristic workload dispatching algorithm to minimize average job execution delay. Given multiple jobs and multiple edge servers, Tan et al. [20] proposed to greedily dispatch jobs and schedule jobs using the Highest Residual Density First rule. Given an application that contains dependent tasks, Sundar and Liang [21] investigated the problem of dispatching tasks to multiple edges with deadline constraints, so as to minimize application execution cost. Liu et al. [22] focused on edge server assignment and frame resolution selection to minimize service latency for mobile augmented reality applications. Gao et al. [23] considered both edge workload and access network congestion when placing service entities. Distributed support for machine learning jobs was discussed in [24, 25] .

Service entity placement was investigated in some recent works. Jia et al. [26] studied the load balancing between multiple edge clouds. Yu et al. [27] investigated the problem of joint edge server provisioning and routing path selection from the perspective of networking. Xu et al. [28] considered the caching and offloading problem in resource-limited edge servers to minimize computation latency. Zhang and Tang [3] studied the client assignment problem for DIAs. Liang et al. [4] proposed a utility-based entity placement framework. Wang et al. [29] studied a similar service entity placement problem that resembles the uncapacitated facility location problem [30] while ours can be reduced from the set cover problem.

In summary, none of existing studies has investigated the *interaction*-oriented edge-enabled service entity placement problem, in which we focus on reducing the weighted average user-to-user interaction delay to improve DIA interactivity. We provide a non-trivial NP-complete result and design an efficient heuristic. We also reveal the trade-off between delay and budget.

3 Problem Formulation

The interaction delay in DIAs can be reduced by moving service entities from remote clouds to nearby edge servers. In edge computing environments, edge servers are usually deployed on a business premise such as

in a doctor office or a coffee shop [5]. According to the Open Edge Computing initiative [31], edge server resources tend to be virtualized and can be allocated at a fine granularity by the aid of lightweight virtualization techniques. Therefore, DSPs can rent resources from EIPs to place service entities, and DIA users can reach these service entities via local wireless connections. However, where DSPs place their service entities to optimize interaction delay is not trivial.

We consider a metropolitan-area edge computing scenario which contains a set of n edge servers, denoted by $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. These servers are dispersed within a city, e.g. inside restaurants and in schools. Given a DIA, its provider can leverage historical data analysis and market investigate to find a set of stable and regular users (e.g., faithful fans of an online multi-player game). The provider wants to optimize the quality of experience indicated by the interaction delay of these users, since they are the majority of all users. For the other users that irregularly visit the DIA, the provider can reserve a fixed number of service entities that provide service for them. Therefore, in this paper, we assume the set of users is stable and denote it by $\mathcal{U} = \{u_1, u_2, ..., u_m\}$. The remote cloud, denoted by C, serves as the default server that runs service entities. Compared with edge servers, the cloud is far more powerful and we can place any number of service entities in it, and thus, the cloud is assumed to be able to serve all users simultaneously.

Network and delay. Edge servers and users are usually connected to a metropolitan-area network (MAN) through access points (APs) and MAN routers, as shown in Fig. 1. We can model the network by a graph $G = (V, E, d)$, in which V consists of edge servers, users, APs, MAN routers, and the remote cloud, while E contains the communication links[1]. The delay function d: $E \rightarrow \mathbf{R}$ gives the delay of each link in E. For example, $d(u_4, ap_1)$ represents the delay between user u_4 and AP ap_1. These delays can be obtained from EIPs or by measuring historical delays and updating them over time.

For a pair of a user and an edge server that are not directly connected by a link, we use $p(u_i, s_j)$ to denote the delay of the shortest path between user u_i and edge server s_j. By the definition of $p(\cdot, \cdot)$, we know it satisfies the *triangular inequality*, i.e., for any u_i, s_j, and u_k, we have $p(u_i, s_j) + p(s_j, u_k) \leq p(u_i, u_k)$.

Placement cost. DIAs are distributed networked systems that contain service entities and thin clients. A client or user can be thought of as an I/O device that sends user-initiated operations to the service entities and receives state updates from the service entities, while a service entity can be thought of as a daemon run in an edge server. The service capacity of each service entity is K, i.e. each entity can serve at most K users. Each edge server s_i is associated with a placement cost w_i, i.e. it costs w_i to place a service entity at server s_i. We use x_i to represent the number of service entities placed at edge server s_i. The placement budget threshold is denoted by Q. We have the following cost constraint:

$$\sum_{i \in \{1,2,...,n\}} w_i x_i \leq Q. \tag{1}$$

Running a service entity at an edge server requires a certain amount of physical resources. Suppose a service entity requires b units of resources and the resource capacity of edge server s_i is B_i. We have the following cost constraint:

$$b x_i \leq B_i, \quad \forall i \in \{1, 2, ..., n\}. \tag{2}$$

For simplicity, let $\boldsymbol{W} = [w_1, w_2, ..., w_n]$ be the cost vector, let $\boldsymbol{B} = [B_1, B_2, ..., B_n]$ be the resource capacity vector, and let $\boldsymbol{X} = [x_1, x_2, ..., x_n]$ be a placement vector.

Service entity association. As we mentioned earlier, when multiple service entities are placed, each user has to choose one as its service entity. The way each user chooses its service entity also affects the pairwise interaction delay. In this paper, for simplicity, we assume that each user chooses the nearest service entity among all placed service entities under a placement $\boldsymbol{X}$, while respecting the constraint that each service entity can serve at most K users; for a service entity, when more than K users choose it as their service entity, it selects the top K nearest users.

We let $s(u_i, \boldsymbol{X})$ be the server where the service entity of u_i is located under a placement $\boldsymbol{X}$. If no service entity can accommodate more users, i.e., every service entity is full, and a user u_i does not have a service entity yet, then its service entity would be placed in the cloud C, that is, $s(u_i, \boldsymbol{X}) = C$.

Take Fig. 1 for example; suppose that $K = 2$, the placement $\boldsymbol{X} = [1, 0, 0, 1]$, $p(u_1, s_1) < p(u_1, s_4)$, $p(u_2, s_1) > p(u_2, s_4)$, $p(u_3, s_1) > p(u_3, s_4)$, $p(u_4, s_1) < p(u_4, s_4)$, $p(u_5, s_1) < p(u_5, s_4)$, $p(u_6, s_1) > p(u_6, s_4)$, therefore, u_1, u_4, and u_5 would choose the entity placed at s_1 as their service entity, while the other users would choose the entity placed at s_4 as their service entity. However, since $K = 2$, $p(u_4, s_1) < p(u_1, s_1)$, $p(u_5, s_1) < p(u_1, s_1)$, $p(u_2, s_4) < p(u_6, s_4)$, and $p(u_3, s_4) < p(u_6, s_4)$, s_1 can only serve u_4 and u_5 while s_4 can only serve u_2 and u_3. In summary, $s(u_4, \boldsymbol{X}) = s(u_5, \boldsymbol{X}) = s_1$, $s(u_2, \boldsymbol{X}) = s(u_3, \boldsymbol{X}) = s_4$, and $s(u_1, \boldsymbol{X}) = s(u_6, \boldsymbol{X}) = C$.

Interaction delay and weight. Given a placement $\boldsymbol{X}$, the interaction path between u_i and u_j consists of three parts: the path from u_i to $s(u_i, \boldsymbol{X})$, the path from $s(u_i, \boldsymbol{X})$ to $s(u_j, \boldsymbol{X})$, and the path from $s(u_j, \boldsymbol{X})$ to u_j. Therefore, the interaction delay $D(u_i, u_j, \boldsymbol{X})$ between between u_i and u_j under a placement $\boldsymbol{X}$ is

$$\begin{aligned} D(u_i,u_j, \boldsymbol{X}) &= p(u_i, s(u_i, \boldsymbol{X})) \\ &\quad + p(s(u_i, \boldsymbol{X}), s(u_j, \boldsymbol{X})) + p(u_j, s(u_j, \boldsymbol{X})). \end{aligned} \tag{3}$$

Different pairs of users may have different interaction frequencies. We use f_{ij} to represent the weight of the interaction between u_i and u_j, where a larger f_{ij} means

1. We use "link" instead of "edge" in a graph for clarity, as "edge" means edge server in this paper.

more interactions between them. Without loss of generality, we have $f_{ij} = f_{ji}$ for any i, $j \in \{1, 2, ..., m\}$, and $f_{ii} = 0$ for any $i \in \{1, 2, ..., m\}$. We make two notes. On one hand, these frequencies capture the heterogeneous interaction preferences of users and they only affect the weighted average interaction delay; without these frequencies, the proposed algorithm still works. On the other hand, some DIAs may have stable interaction patterns, for example, several friends form a team that makes explorations together in the World of Warcraft [1]. In this case, the historical interaction information can be used to infer future interaction frequency.

Note that, there are a total of $\frac{m(m-1)}{2}$ pairs of users. Without loss of generality, we assume the sum of the weights of these $\frac{m(m-1)}{2}$ pairs is 1, that is,

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} f_{ij} = 1. \tag{4}$$

We let $\boldsymbol{F} = [f_{ij}]_{m \times m}$ denote the weight matrix.

Objective. The objective of ISEP is to minimize the weighted average pairwise interaction delay, which is defined as follows:

$$E(\boldsymbol{X}) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} D(u_i, u_j, \boldsymbol{X}) f_{ij}. \tag{5}$$

Main notations are summarized in Table 1 for quick reference. We have the ISEP problem:

$$\min \quad E(\boldsymbol{X}) = \sum_{i=1}^{m} \sum_{j=i+1}^{m} D(u_i, u_j, \boldsymbol{X}) f_{ij} \tag{6a}$$

$$\text{s.t.} \quad \sum_{i \in \{1,2,...,n\}} w_i x_i \leq Q \tag{6b}$$

$$b x_i \leq B_i, \quad \forall i \in \{1, 2, ..., n\} \tag{6c}$$

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} f_{ij} = 1 \tag{6d}$$

$$x_i \in \{0, 1, 2, ...\}, \quad \forall i \in \{1, 2, ..., n\} \tag{6e}$$

4 NP-COMPLETENESS RESULTS

By reducing the NP-complete Set Cover (SC) problem [30] to ISEP, we can prove that ISEP is NP-complete.

Theorem 1: The decision version of ISEP is NP-complete.

Proof: We provide the descriptions on the decision version of SC and ISEP as follows.

- *Decision version of SC*: Given a universe $\mathcal{H} = \{e_1, e_2, ..., e_M\}$ of M elements and an integer q, a collection of subsets of $\mathcal{H}$, $\mathcal{R}_1$, $\mathcal{R}_2$, ..., and $\mathcal{R}_N$, does there exist a sub-collection of these subsets with size no more than q that covers all elements of $\mathcal{H}$?
- *Decision version of ISEP*: Given a set of users $\mathcal{U}$, a set of edge servers $\mathcal{S}$, a remote cloud C, a topology graph $G = (V, E, d)$, the resource requirement b of a single service entity, a placement cost vector $\boldsymbol{W}$, a resource capacity vector $\boldsymbol{B}$, an interaction weight matrix $\boldsymbol{F}$, a budget threshold Q, a service capacity K, does there exist a placement $\boldsymbol{X}$ that makes $E(\boldsymbol{X})$ defined in Eq. (5) no larger than a threshold Z?

TABLE 1: Main notations for quick reference.

Symbol	Meaning
s_i, $\mathcal{S}$, n	i-th server, set of servers, # of servers
u_i, $\mathcal{U}$, m	i-th user, set of users, # of users
C	remote cloud
K	# of users that can be served by one entity
Q	budget threshold
b	# of required resources to place one entity
$d(\cdot, \cdot)$	delay of each direct link
$p(\cdot, \cdot)$	delay of the shortest path
w_i, $\boldsymbol{W}$	placement cost at s_i, cost vector
B_i, $\boldsymbol{B}$	resource capacity of s_i, capacity vector
f_{ij}, $\boldsymbol{F}$	weight of the interaction between u_i and u_j, weight matrix
x_i, $\boldsymbol{X}$	# of entities placed at s_i, placement vector
$s(u_i, \boldsymbol{X})$	the server at which the entity of u_i is located
$D(u_i, u_j, \boldsymbol{X})$	interaction delay between u_i and u_j under $\boldsymbol{X}$
$E(\boldsymbol{X})$	weighted average interaction delay under $\boldsymbol{X}$

Without loss of generality, let

$$< M, N, q, \mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_N >$$

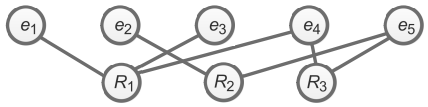
denote an instance of SC; let

$$< \mathcal{U}, \mathcal{S}, G = (V, E, d), \boldsymbol{W}, b, \boldsymbol{B}, \boldsymbol{F}, Q, K, Z >$$

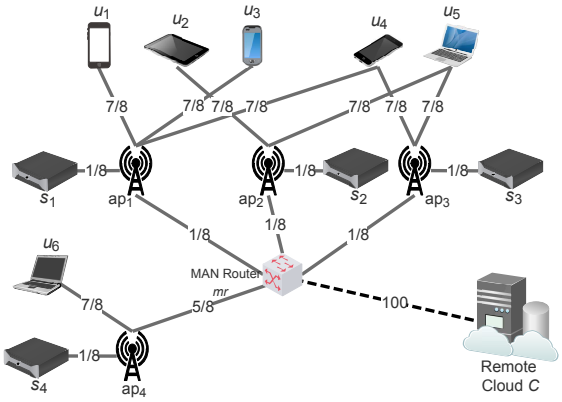
denote an instance of ISEP.

In the following, we show that, any instance of SC can be polynomially reduced to an instance of ISEP. The reduction maps an instance of SC into an instance of ISEP using the following *rules*:

(1°) $|\mathcal{U}| = m \leftarrow M + 1$;
(2°) $|\mathcal{S}| = n \leftarrow N + 1$;
(3°) (topology) add $(N + 1)$ APs and let s_i directly connect to ap_i; add one MAN router mr and let every AP directly connect to the router; add one remote cloud and let it directly connect to the router; for every $i \in \{1, 2, ..., M\}$ and every $j \in \{1, 2, ..., N\}$, let u_i directly connect to ap_j *if and only if* $e_i \in R_j$ in SC; let u_{M+1} directly connect to s_{N+1};
(4°) (delay between users and APs) if there is a link between user u_i and AP ap_j, then $d(u_i, ap_j) \leftarrow \frac{7}{8}$;
(5°) (delay between servers and APs) for every $i \in \{1, 2, ..., N + 1\}$, $d(s_i, ap_i) \leftarrow \frac{1}{8}$;
(6°) (delay between the MAN router and APs) for every $i \in \{1, 2, ..., N\}$, $d(mr, ap_i) \leftarrow \frac{1}{8}$; besides, $d(mr, ap_{N+1}) \leftarrow \frac{5}{8}$;
(7°) (delay between the MAN router and the remote cloud) $d(mr, C) \leftarrow 100$;
(8°) $w_i \leftarrow 1$ for every $i \in \{1, 2, ..., N + 1\}$;
(9°) $b \leftarrow 1$;
(10°) $B_i \leftarrow 1$ for every $i \in \{1, 2, ..., N + 1\}$;
(11°) $f_{i,M+1} \leftarrow \frac{1}{M}$ for every $i \in \{1, 2, ..., M\}$; and $f_{ij} \leftarrow 0$ for all other pairs of users;
(12°) $Q \leftarrow q + 1$;



(a) An instance of SC: $M = 5$ and $N = 3$



(b) The corresponding ISEP instance: $m = 6$ and $n = 4$

Fig. 2: An example of the polynomial reduction from SC to ISEP, where servers and users can be seen as subsets and elements, respectively.

(13°) $K \leftarrow M$;
(14°) $Z \leftarrow 3$.

Rules (1°) and (2°) specify the number of users and edge servers, respectively. Note that, the number of users is one more than that of elements in SC, and the number of edge servers is one more than that of subsets in SC.

Rule (3°) specifies the network topology in the instance of ISEP, in which servers can be seen as subsets and users can be seen as elements. We manually add $(N + 1)$ APs, one MAN router, and one remote cloud. It should be noted that, for every $i \in \{1, 2, ..., M\}$ and every $j \in \{1, 2, ..., N\}$, we let u_i directly connect to ap_j in ISEP *if and only if* $e_i \in R_j$ in SC. Fig. 2 shows an example of the reduction from SC to ISEP.

Rules (4°) to (7°) specify various delays. Rule (4°) means the delay of a link between a user and an AP is always $\frac{7}{8}$, e.g., $d(u_4, s_1) = \frac{7}{8}$ in Fig. 2(b). Rule (5°) specifies the delay between servers and APs. In Rule (6°), the delay between ap_{N+1} and the router is different from that between ap_i $(1 \le i \le N)$ and the router. Rule (7°) specifies the delay between the router and the cloud. See Fig. 2 for an example of these delays.

Rule (8°) means the placement cost at any edge server is 1. Rules (9°) and (10°) specify the resource requirement of a single service entity and the resource capacity of each server. Given these two rules, we can place at most one service entity at a server.

Rule (11°) specifies the interaction weights. There are two types of weights: the weight between u_{M+1} and any other user is $\frac{1}{M}$, and the weight between any two users other than u_{M+1} is 0. Note that, in doing so, the sum of weights is 1.

Rules (12°) to (14°) specify the values of Q, K, and Z.

To confirm the validity of this reduction, we have to show that it works in the case of either outcome depicted. We have the following lemmas, the proofs of which can be found in Sections 4.1 and 4.2.

Lemma 1: If the instance of SC has a yes solution, then the corresponding ISEP instance has a yes placement.

Lemma 2: If the instance of SC does not have a yes solution, then the corresponding ISEP instance does not have a yes placement, either.

Finally, it is easy to see that, the reduction from SC to ISEP (i.e., 14 rules) can be finished in polynomial time. Hence, the theorem is proven. □

4.1 Proof of Lemma 1

If the instance of SC has a yes solution, then there are indeed q subsets that can cover all elements. Without loss of generality, we assume that these q subsets are $\mathcal{R}_{i_1}$, $\mathcal{R}_{i_2}$, ..., $\mathcal{R}_{i_q}$. We then prove that the following placement $\boldsymbol{X}^* = [x_1^*, x_2^*, ..., x_{N+1}^*]$ in which

$$x_i^* = \begin{cases} 1 & \text{if } i \in \{i_1, i_2, ..., i_q, N+1\}, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

is a yes placement that makes $E(\boldsymbol{X}^*)$ no larger than the delay threshold $Z = 3$.

Given such a placement $\boldsymbol{X}^*$ in Eq. (7), since $\mathcal{R}_{i_1}$, $\mathcal{R}_{i_2}$, ..., $\mathcal{R}_{i_q}$ cover all elements in the SC instance, we know for any $i \in \{1, 2, ..., M\}$, there exists at least one service entity placed at server s_j such that both u_i and s_j are directly connected to the same AP. Besides, a service entity can serve $K = M$ users. Therefore, the delay between any user u_i and its service entity is $\frac{1}{8} + \frac{7}{8} = 1$ (according to Rules (4°)(5°)). In other words, we have

$$p(u_i, s(u_i, \boldsymbol{X}^*)) = 1, \text{ for any } i \in \{1, 2, ..., M\}. \tag{8}$$

Since $x_{N+1} = 1$ according to Eq. (7), and s_{N+1} is the nearest server from u_{M+1} according to Rule (3°), we have $s(u_{M+1}, \boldsymbol{X}^*) = s_{N+1}$, thus,

$$p(u_{M+1}, s(u_{M+1}, \boldsymbol{X}^*)) = 1. \tag{9}$$

Combining them together, we have

$$\begin{aligned} E(\boldsymbol{X}^*) &= \sum_{i=1}^{M} D(u_i, u_{M+1}, \boldsymbol{X}^*) f_{i,M+1} \\ &+ \sum_{i=1}^{M} \sum_{j=i+1}^{M} D(u_i, u_j, \boldsymbol{X}^*) f_{ij} \\ &= \sum_{i=1}^{M} (p(u_i, s(u_i, \boldsymbol{X}^*)) + p(s(u_i, \boldsymbol{X}^*), \\ &\quad s(u_{M+1}, \boldsymbol{X}^*)) + p(u_{M+1}, s(u_{M+1}, \boldsymbol{X}^*))) \cdot \frac{1}{M} \\ &+ \sum_{i=1}^{M} \sum_{j=i+1}^{M} D(u_i, u_j, \boldsymbol{X}^*) \cdot 0 \\ &= \sum_{i=1}^{M} (1 + p(s(u_i, \boldsymbol{X}^*), s_{N+1}) + 1) \cdot \frac{1}{M} \\ &= \sum_{i=1}^{M} (1 + 1 + 1) \cdot \frac{1}{M} \text{ (due to Rules (5°)(6°))} = 3, \end{aligned} \tag{10}$$

which indicates $\boldsymbol{X}^*$ in Eq. (7) is a yes placement to the ISEP instance. The lemma holds immediately.

4.2 Proof of Lemma 2

Without loss of generality, let $\boldsymbol{X}' = [x'_1, x'_2, ..., x'_{N+1}]$ denote a yes placement to the instance of ISEP. Since $Q = q + 1$ and we can place at most one service entity at a server (due to Rules (9°)(10°)), the number of 1's in $\boldsymbol{X}'$ is at most $q + 1$.

To prove Lemma 2, we first provide two properties that a yes placement to ISEP must have.

***Property** 1:* If $\boldsymbol{X}' = [x'_1, x'_2, ..., x'_{N+1}]$ is a yes placement to the instance of ISEP, then x'_{N+1} must be 1, i.e., we must place a service entity at s_{N+1}.

Proof: We prove this by contradiction and assume the opposite, i.e., $x'_{N+1} = 0$.

If $x'_{N+1} = 0$, then $s(u_{M+1}, \boldsymbol{X}')$ must be some edge server other than s_{N+1}. Based on Rules (4°)(5°)(6°), for any $i \in \{1, 2, ..., N\}$, the smallest delay between u_{M+1} and s_i is $\frac{7}{8} + \frac{5}{8} + \frac{1}{8} + \frac{1}{8} = \frac{7}{4}$. Therefore, no matter what $s(u_{M+1}, \boldsymbol{X}')$ is, the delay between u_{M+1} and $s(u_{M+1}, \boldsymbol{X}')$ is at least $\frac{7}{4}$, i.e.,

$$p(u_{M+1}, s(u_{M+1}, \boldsymbol{X}')) \geq \frac{7}{4}. \tag{11}$$

For every $i \in \{1, 2, ..., M\}$, due to Rules (4°)(5°), the delay between a user and the nearest server from the user is at least 1, that is,

$$\min_{j \in \{1,2,...,N+1\}} p(u_i, s_j) = 1, \tag{12}$$

which implies, for every $i \in \{1, 2, ..., M\}$,

$$p(u_i, s(u_i, \boldsymbol{X}')) \geq 1. \tag{13}$$

Combining Eq. (11) and Eq. (13) together, due to Rules (5°)(6°), for each $i \in \{1, 2, ..., M\}$, we have

$$\begin{aligned} &D(u_i, u_{M+1}, \boldsymbol{X}') \\ =&p(u_i, s(u_i, \boldsymbol{X}')) + p(s(u_i, \boldsymbol{X}'), s(u_{M+1}, \boldsymbol{X}')) \\ &+p(u_{M+1}, s(u_{M+1}, \boldsymbol{X}')) \\ \geq&1 + (\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}) + \frac{7}{4} = \frac{13}{4}. \end{aligned} \tag{14}$$

Therefore, the interaction delay under $x'_{N+1} = 0$ is

$$\begin{aligned} E(\boldsymbol{X}') &= \sum_{i=1}^{M} D(u_i, u_{M+1}, \boldsymbol{X}') \cdot \frac{1}{M} \\ &\geq \frac{13}{4} \sum_{i=1}^{M} \frac{1}{M} = \frac{13}{4} > Z = 3. \end{aligned} \tag{15}$$

A contradiction! Therefore, we proved that, given a yes placement $\boldsymbol{X}' = [x'_1, x'_2, ..., x'_{N+1}]$ to the instance of ISEP, x'_{N+1} must be 1. □

***Property** 2:* If $\boldsymbol{X}' = [x'_1, x'_2, ..., x'_{N+1}]$ is a yes placement to the instance of ISEP, then $\sum_{i \in \{1,2,...,N\}} x'_i \geq 1$, i.e., at least one of x'_1, x'_2, ..., and x'_N is 1.

Proof: If $\boldsymbol{X}' = [x'_1, x'_2, ..., x'_{N+1}]$ is a yes placement, due to Property 1, $x'_{N+1} = 1$. Since the service capacity K of each entity is M (due to Rule (13°)) and we can place at most one service entity at a server (due to Rules (9°)(10°)), the service entity placed at s_{N+1} cannot serve more than M users. Remember that there are $M+1$ users, therefore, at least one of x'_1, x'_2, ..., and x'_N must be 1. □

With the above two properties, we can prove Lemma 2 now. If a placement does not have the two properties stated in Property 1 and Property 2, then it cannot be a yes placement and Lemma 2 holds immediately. Therefore, in the following, we just need to prove "if the instance of SC does not have a yes solution, a placement—which has the two properties stated in Property 1 and Property 2—cannot be a yes placement to the instance of ISEP".

Let $\boldsymbol{X}' = [x'_1, x'_2, ..., x'_{N+1}]$ denote such a placement.

Due to Lemma 1, x'_{N+1} must be 1 in a yes placement to the instance of ISEP; since the budget threshold is $q + 1$ (due to Rule (12°)), we can place at most q service entities at q edge servers among s_1, s_2, ..., and s_N in a yes placement to the instance of ISEP.

However, if the instance of SC does not have a yes solution, that is, we cannot use q subsets to cover all M elements in the instance of SC, then we cannot place q service entities at q edge servers among s_1, s_2, ..., and s_N to make each user be directly connected to some AP which is directly connected with one of these q service entities. In other words, there exists at least one user, say u_k, who cannot be directly connected with any AP that is directly connected to any one of these q entities.

Due to Lemma 2, at least one of s_1, s_2, ..., and s_N has a service entity in a yes placement to the instance of ISEP. Without loss of generality, we assume s_t has a service entity. From the view of u_k, the delay between u_k and s_t is smaller than that between u_k and s_{N+1}, because $p(u_k, s_t) = \frac{7}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{5}{4}$ (due to Rules (4°)(5°)(6°)) and $p(u_k, s_{N+1}) = \frac{7}{4}$. Hence, $s(u_k, \boldsymbol{X}') = s_t$, and we have

$$p(u_k, s(u_k, \boldsymbol{X}')) = \frac{5}{4}. \tag{16}$$

Combining them together, we have

$$\begin{aligned} E(\boldsymbol{X}') &= \sum_{i=1}^{M} D(u_i, u_{M+1}, \boldsymbol{X}') \cdot \frac{1}{M} \\ &= \sum_{i=1, i \neq k}^{M} D(u_i, u_{M+1}, \boldsymbol{X}') \cdot \frac{1}{M} \\ &+ D(u_k, u_{M+1}, \boldsymbol{X}') \cdot \frac{1}{M} \\ &\geq (M-1) \cdot (1+1+1) \cdot \frac{1}{M} + (\frac{5}{4} + 1 + 1) \cdot \frac{1}{M} \\ &= 3 + \frac{1}{4M} > Z = 3, \end{aligned} \tag{17}$$

implying that $\boldsymbol{X}'$ is not a yes placement. The lemma holds immediately.

5 Algorithm and Analysis

In this section, we present an efficient Greedy Placement Algorithm (GPA) for ISEP and prove that the weighted average interaction delay achieved by GPA is not far away from the optimal delay in some cases.

Algorithm 1: Greedy Placement Algorithm (GPA)

Input: $\mathcal{U}, \mathcal{S}, G=(V,E,d), \boldsymbol{W}, \boldsymbol{F}, Q, K, Z$
Output: $\boldsymbol{X}$

1 initialize $P \leftarrow$ the non-decreasing order of the shortest delay between all users and all servers;
2 initialize $\boldsymbol{X}=[x_1,x_2,...,x_n] \leftarrow [0,0,...,0]$;
3 $(s(u_1,\boldsymbol{X}),...,s(u_m,\boldsymbol{X})) \leftarrow (C,...,C)$;
4 use $s(u_1,\boldsymbol{X}),...,s(u_m,\boldsymbol{X})$ to compute $E(\boldsymbol{X})$;
5 **while** $\sum_{i\in\{1,2,...,n\}} w_i x_i < Q$ **do**
6 $previousE \leftarrow E(\boldsymbol{X})$; $minE \leftarrow E(\boldsymbol{X})$; $minServer \leftarrow -1$;
7 **for** $k=1$; $k \le n$; $k++$ **do**
8 **if** $bx_k+b \le B_k$ *and* $w_k + \sum_{i\in\{1,2,...,n\}} w_i x_i \le Q$ **then**
9 construct a new placement $\boldsymbol{X}^\dagger=[x_1^\dagger,x_2^\dagger,...,x_n^\dagger]$ where (1) $x_k^\dagger \leftarrow x_k^\dagger+1$, and (2) $x_j^\dagger \leftarrow x_j$ for all $j \ne k$;
10 $(s(u_1,\boldsymbol{X}^\dagger),...,s(u_m,\boldsymbol{X}^\dagger)) \leftarrow$ ComputeAssoc$(P,\mathcal{U},\mathcal{S},G,K,\boldsymbol{X}^\dagger)$;
11 Use $s(u_1,\boldsymbol{X}^\dagger),...,s(u_m,\boldsymbol{X}^\dagger)$ to compute $E(\boldsymbol{X}^\dagger)$;
12 **if** $E(\boldsymbol{X}^\dagger) < minE$ **then**
13 $minE \leftarrow E(\boldsymbol{X}^\dagger)$; $minServer \leftarrow k$;
14 **if** $minServer \ne -1$ *and* $minE < previousE$ **then**
15 $x_{minServer} \leftarrow x_{minServer}+1$;
16 return $\boldsymbol{X}$

Algorithm 2: ComputeAssoc$(P,\mathcal{U},\mathcal{S},G,K,\boldsymbol{X})$

Input: $P,\mathcal{U},\mathcal{S},G=(V,E,d),K,\boldsymbol{X}$
Output: $s(u_1,\boldsymbol{X}),...,s(u_m,\boldsymbol{X})$

1 initialize $y_j \leftarrow 0$ for each $s_j \in \mathcal{S}$;
2 initialize $z_i \leftarrow 0$ and $s(u_i,\boldsymbol{X}) \leftarrow C$ for each $u_i \in \mathcal{U}$;
3 **for** *each* $p(u_i,s_j)$ *in the sorted order* P **do**
4 **if** $z_i=0$ *and* $y_j < Kx_j$ **then**
5 $z_i \leftarrow 1$; $y_j \leftarrow y_j+1$; $s(u_i,\boldsymbol{X}) \leftarrow s_j$;
6 return $(s(u_1,\boldsymbol{X}),...,s(u_m,\boldsymbol{X}))$

5.1 Algorithm

The main idea of GPA is as follows. GPA consists of multiple iterations. In each iteration, we consider each edge server in which its residual resource is sufficient to place a new service entity, and compute the weighted average interaction delay if we would place a service entity in this server; we then select the edge server to place a new service entity that reduces the weighted average interaction delay and leads to the smallest one. The details are shown in Alg. 1.

Initially, no service entity is placed (line 2), and all users choose the remote cloud as the place for their service entities at this time (line 3). With these associations, we can easily compute the weighted average interaction delay $E(\boldsymbol{X})$ (line 4).

In the while loop, for each server s_k, if its residual resource can afford a new service entity and the total placement cost does not exceed the budget threshold if we place a service entity in s_k (line 8), we then temporarily increase x_k by one and compute the new service entity associations and the corresponding interaction delay $E(\boldsymbol{X}^\dagger)$ (lines 9-11). After we consider all possible servers, we select the server to place a new service entity that reduces the weighted average interaction delay and leads to the smallest one (lines 12-15).

In the GPA algorithm, an important sub-procedure is ComputeAssoc$(P,\mathcal{U},\mathcal{S},G,K,\boldsymbol{X})$, which is shown in Alg. 2. Note that, P stores the non-decreasing order of the shortest delay between all users and servers. Given a delay $p(u_i,s_j)$, if the user u_i does not have a service entity and the number of users that choose s_j as their service entity is less than Kx_j, then we associate u_i with s_j, i.e., $s(u_i,\boldsymbol{X})=s_j$ (lines 3-5).

The time complexity of GPA is dominated by the while loop. Denote by $\min\{w_i\}$ the minimal placement cost, then there are at most $\frac{Q}{\min\{w_i\}}$ iterations. In each iteration, we may have to compute the associations and weighted interaction delay for at most n times. It is not hard to see that, computing the weighted interaction delay costs $O(m^2)$ time, since there are m users. ComputeAssoc checks each pair of user and server, and it costs $O(mn)$ time. Thus, the time complexity of GPA is $O(\frac{Q}{\min\{w_i\}}n\cdot(m^2+mn)=O(\frac{Q}{\min\{w_i\}}(m^2n+mn^2))$.

5.2 Analysis

The analysis of GPA is hard in general, thus, we consider some special cases of ISEP in which (1) the interaction frequency is homogeneous, (2) each edge server is sufficient to serve all users, and (3) it costs one unit of budget to place a service entity at any server. That is, we consider the ISEP problem in which the following three constraints are satisfied:

$$f_{ij}=\frac{2}{m(m-1)},\quad \forall i,j\in\{1,2,...,m\} \text{ and } i<j, \tag{18}$$

$$\min_{i\in\{1,2,...,n\}} \frac{B_i}{b}\cdot K \ge m, \tag{19}$$

and

$$w_i=1,\quad \forall i\in\{1,2,...,n\}. \tag{20}$$

We now show some application examples that have these properties. Take multiplayer online game for example. Based on market investigation and historical information, the DSP (i.e., the game provider) knows the number of users and fortunately these users have uniform interaction frequency. From the EIP perspective, it may obtain the same revenue from leasing the same amount of resources at each server it owns, thus, it may cost the same amount of budget for the DSP to rent a

fixed amount of resources to place a service entity. The capacity of each physical server is so large that it can accommodate all users[2].

5.2.1 The Optimal Weighted Average Interaction Delay

Without loss of generality, suppose that a placement selects edge servers s_1, s_2, ..., and s_g to deploy service entities (note that an edge server may host more than one service entity). Such a placement can be denoted by $\boldsymbol{X'} = [x'_1 \geq 1, x'_2 \geq 1, ..., x'_g \geq 1, x'_{g+1} = 0, ..., x'_n = 0]$. For each edge server s_i, we denote the set of users that choose one of the entities placed on s_i as its entity by $\mathcal{U}_i$. According to this definition, $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_g$ form a partition of $\mathcal{U}$, that is, $\cup_{i=1}^{g}\mathcal{U}_i = \mathcal{U}$ and $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset, \forall i \neq j$.

Consider two users u_i and u_j that select one entity placed on s_i and one entity placed on s_j as its entity, respectively, then the interaction delay between them is

$$D(u_i, u_j, \boldsymbol{X'}) = p(u_i, s_i) + p(s_i, s_j) + p(s_j, u_j). \quad (21)$$

Then, the total interaction delay between the users in $\mathcal{U}_i$ and the users in $\mathcal{U}_j$ is

$$\begin{aligned}&\sum_{u_i \in \mathcal{U}_i}\sum_{u_j \in \mathcal{U}_j} D(u_i, u_j, \boldsymbol{X'}) = \sum_{u_i \in \mathcal{U}_i}\sum_{u_j \in \mathcal{U}_j} p(u_i, s_i)\\ &+ \sum_{u_i \in \mathcal{U}_i}\sum_{u_j \in \mathcal{U}_j} p(s_i, s_j) + \sum_{u_i \in \mathcal{U}_i}\sum_{u_j \in \mathcal{U}_j} p(s_j, u_j)\\ &= \sum_{u_i \in \mathcal{U}_i} |\mathcal{U}_j| p(u_i, s_i) + |\mathcal{U}_i||\mathcal{U}_j| p(s_i, s_j) + |\mathcal{U}_i| \sum_{u_j \in \mathcal{U}_j} p(s_j, u_j).\end{aligned} \quad (22)$$

The total interaction delay between all $\frac{m(m-1)}{2}$ pairs of users consists of two parts: the delay between two users that belong to different $\mathcal{U}_i$'s and the delay between two users that belong to the same $\mathcal{U}_i$. The first part is

$$\begin{aligned}&\sum_{(i<j)\&\&(i,j\in\{1,2,...,g\})} \sum_{u_i \in \mathcal{U}_i}\sum_{u_j \in \mathcal{U}_j} D(u_i, u_j, \boldsymbol{X'})\\ &= \sum_{i=1}^{n}\sum_{j=i+1}^{n} \Big(\sum_{u_i \in \mathcal{U}_i} |\mathcal{U}_j| p(u_i, s_i)\\ &+ |\mathcal{U}_i||\mathcal{U}_j| p(s_i, s_j) + |\mathcal{U}_i| \sum_{u_j \in \mathcal{U}_j} p(s_j, u_j)\Big)\\ &= \sum_{k=1}^{g}\sum_{u_h \in \mathcal{U}_k} (p(u_h, s_k) \sum_{h \neq k} |\mathcal{U}_h|) + \sum_{i=1}^{n}\sum_{j=i+1}^{n} |\mathcal{U}_i||\mathcal{U}_j| p(s_i, s_j).\end{aligned} \quad (23)$$

The second part is

$$\begin{aligned}&\sum_{k=1}^{g} \sum_{(u_i, u_j \in \mathcal{U}_k)\&\&(i \neq j)} D(u_i, u_j, \boldsymbol{X'})\\ &= \sum_{k=1}^{g} \sum_{(u_i, u_j \in \mathcal{U}_k)\&\&(i \neq j)} (p(u_i, s_k) + p(u_j, s_k))\\ &= \sum_{k=1}^{g} \sum_{u_h \in \mathcal{U}_k} (p(u_h, s_k) \cdot (|\mathcal{U}_k| - 1))\end{aligned} \quad (24)$$

2. Although a single physical server can accommodate all users, it may not be the server that has the shortest delay to all users. Hence, the DSP should not place all entities at one server.

Combining Eqs. (23) and (24), we know the total interaction delay between all $\frac{m(m-1)}{2}$ pairs of users is

$$\begin{aligned}&\sum_{k=1}^{g}\sum_{u_h \in \mathcal{U}_k} (p(u_h, s_k)(\sum_{h \neq k} |\mathcal{U}_h| + |\mathcal{U}_k| - 1))\\ &+ \sum_{i=1}^{n}\sum_{j=i+1}^{n} |\mathcal{U}_i||\mathcal{U}_j| p(s_i, s_j)\\ &= \sum_{k=1}^{g}\sum_{u_h \in \mathcal{U}_k} (p(u_h, s_k)(m-1))\\ &+ \sum_{i=1}^{n}\sum_{j=i+1}^{n} |\mathcal{U}_i||\mathcal{U}_j| p(s_i, s_j).\end{aligned} \quad (25)$$

Therefore, the average interaction delay under $\boldsymbol{X'} = [x'_1 \geq 1, x'_2 \geq 1, ..., x'_g \geq 1, x'_{g+1} = 0, ..., x'_n = 0]$ is

$$\begin{aligned}E(\boldsymbol{X'}) = \frac{2}{m(m-1)}\{&\sum_{k=1}^{g}\sum_{u_h \in \mathcal{U}_k} (p(u_h, s_k)(m-1))\\ &+ \sum_{i=1}^{n}\sum_{j=i+1}^{n} |\mathcal{U}_i||\mathcal{U}_j| p(s_i, s_j)\}.\end{aligned} \quad (26)$$

Without loss of generality, we assume that $\boldsymbol{X'} = [x'_1 \geq 1, x'_2 \geq 1, ..., x'_g \geq 1, x'_{g+1} = 0, ..., x'_n = 0]$ is the optimal placement, then OPT is equal to Eq. (26).

5.2.2 Performance Gap

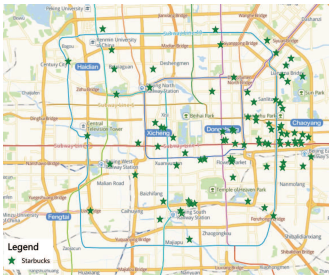
Suppose that s_z is the first edge server selected by GPA to place a service entity. If s_z is the only server (i.e., the placement $\boldsymbol{X}_z$ is $[x_1 = 0, ..., x_{z-1} = 0, x_z \geq 1, x_{z+1} = 0, ..., x_n = 0]$), since s_z is sufficient to serve all users (due to Eq. (19)), all users would choose the entities at s_z as their service entities. In this case, the average interaction delay is

$$E(\boldsymbol{X}_z) = \frac{2(m-1)}{m(m-1)} \sum_{u_i \in \mathcal{U}} p(u_i, s_z) = \frac{2}{m} \sum_{u_i \in \mathcal{U}} p(u_i, s_z). \quad (27)$$

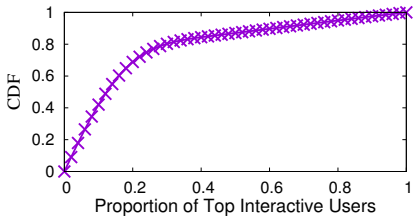
Remember that $p(\cdot, \cdot)$ denotes the smallest delay between two users or servers, therefore, $p(\cdot, \cdot)$ satisfies the triangular inequality. Hence, we have

$$\begin{aligned}&\frac{2}{m}\sum_{u_i \in \mathcal{U}} p(u_i, s_z) = \frac{2}{m}(\sum_{u_i \in \mathcal{U}_1} p(u_i, s_z) + \sum_{u_i \notin \mathcal{U}_1} p(u_i, s_z))\\ &\leq \frac{2}{m}(\sum_{u_i \in \mathcal{U}_1} p(u_i, s_z) + \sum_{k=2}^{g}\sum_{u_i \in \mathcal{U}_k} (p(u_i, s_k) + p(s_k, s_z)))\\ &= \frac{2}{m}(\sum_{k=1}^{g}\sum_{u_i \in \mathcal{U}_k} p(u_i, s_k) + \sum_{k=2}^{g}\sum_{u_i \in \mathcal{U}_k} p(s_k, s_z)).\end{aligned} \quad (28)$$

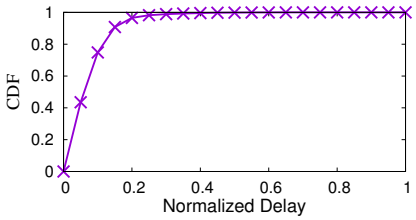
Given any placement $\boldsymbol{X}$ generated by GPA, since GPA selects a new server to place a new service entity only if the new entity reduces the weighted average interaction delay, we know $E(\boldsymbol{X}) \leq E(\boldsymbol{X}_z)$.



(a) Locations of 92 Starbucks within the 4th ring road of Beijing, China.

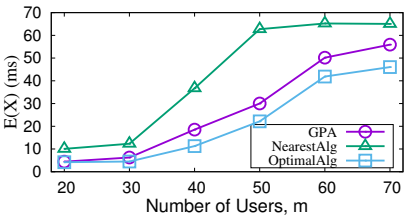


(b) $y \times 100\%$ interactions are from $x \times \%$ most interactive users.

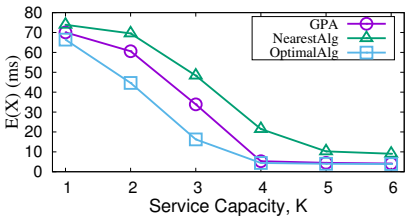


(c) $y \times 100\%$ connections are associated with $x \times \%$ smallest delays.

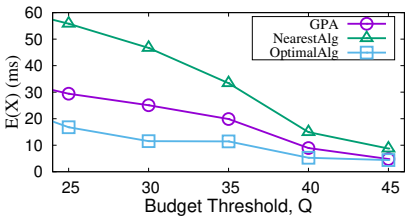
Fig. 3: Simulation settings.



(a) Impact of the number of users



(b) Impact of the service capacity



(c) Impact of the budget threshold

Fig. 4: Edge servers at synthetical locations. Randomized Delay. The default setting is $n = 18$, $m = 40$, $K = 3$, and $Q = 30$.

Therefore, the performance gap between GPA and the optimal solution is

$$\begin{aligned} & E(\boldsymbol{X}) - OPT \leq E(\boldsymbol{X}_z) - OPT \\ & \leq \frac{2}{m}(\sum_{k=1}^{g}\sum_{u_i \in \mathcal{U}_k} p(u_i, s_k) + \sum_{k=2}^{g}\sum_{u_i \in \mathcal{U}_k} p(s_k, s_z)) \\ & - \frac{2}{m(m-1)}\{\sum_{k=1}^{g}\sum_{u_h \in \mathcal{U}_k}(p(u_h, s_k)(m-1)) \\ & + \sum_{i=1}^{n}\sum_{j=i+1}^{n}|\mathcal{U}_i||\mathcal{U}_j|p(s_i, s_j)\} \\ & = \frac{2}{m(m-1)}[(m-1)\sum_{k=2}^{g}\sum_{u_i \in \mathcal{U}_k} p(s_k, s_z) \\ & - \sum_{i=1}^{n}\sum_{j=i+1}^{n}|\mathcal{U}_i||\mathcal{U}_j|p(s_i, s_j)]. \end{aligned} \tag{29}$$

6 Performance Evaluation

We answer the following questions in our evaluation: (1) How effective is GPA's placement? (2) How well does GPA approximate the optimal placement in terms of interaction delay? (3) What is the effect of the number of users and the service capacity? (4) Can GPA provide any suggestions on choosing the budget threshold?

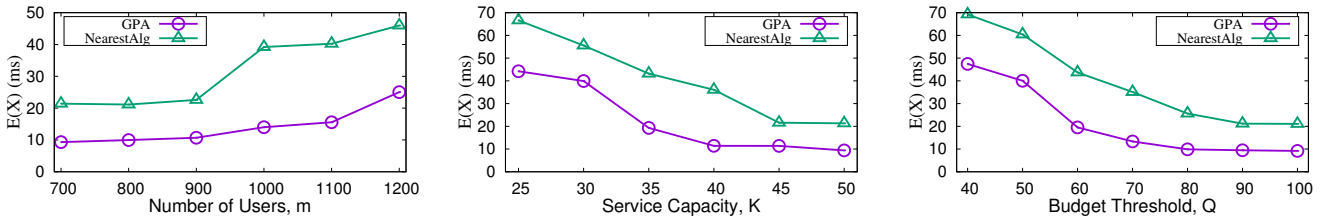
6.1 Setup

We consider a metropolitan area that contains edge servers and users in this paper. For locations of edge servers, we use the locations of Starbucks within the 4th ring road of Beijing, as shown in Fig. 3(a). Similar to a previous study [29], we use Starbucks' locations as the locations of edge servers, because the distribution of them in a city usually achieves a decent coverage of users, making them very suitable for placing edges. We calculate the minimum bounding rectangle of these 92 Starbucks with two sides parallel to a meridian. Then, we extend this rectangle by adding 5km to each side, so as to form the area of interest, within which we randomly generate user locations. The placement cost at each edge server is uniformly generated from the range $[1, 5]$.

The interaction weights among users are synthesized following realistic distributions revealed in [32]. Fig. 3(b) shows $y \times 100\%$ interactions are from $x \times \%$ most interactive users, where the interactivity of user u_i is measured by $\sum_{j \neq i} f_{ij}$. For example, 80% interactions are from nearly 26% most interactive users in Fig. 3(b).

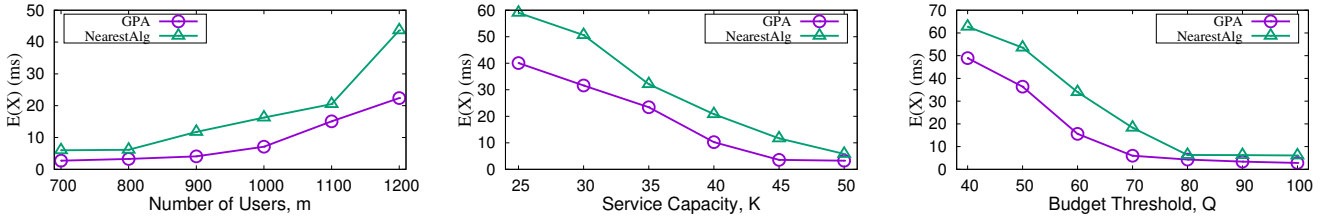
Delays are generated in two ways: Proportional Delay and Randomized Delay. In the former, the delay between any two objects (a user or an edge server) is proportional to the Euclidean distance between them [4, 29]. Let $AvgDelay$ denote the average delay generated in this way. In the latter, we first assume there is a connection between two objects if the Euclidean distance between them is not larger than a pre-defined threshold, e.g., 5km; then, the delays of these connections are synthesized following realistic distributions disclosed in [33], making sure the average delay of these connections is $AvgDelay$. The CDF of the synthetically generated delays is shown in Fig. 3(c), in which $y \times 100\%$ connections are associated with $x \times \%$ smallest delays. For example, 80% connections are associated with nearly 14% smallest delays in Fig. 3(c). In both ways, the delay between a user or an edge server and the cloud is uniformly generated from the range $[30ms, 50ms]$.

We compare GPA with OptimalAlg and NearestAlg. OptimalAlg is a brute-force algorithm that searches the best from a total of $2^{|\mathcal{S}|}$ different placements, and NearestAlg greedily places a service entity at the edge server which has the largest number of users that choose it as



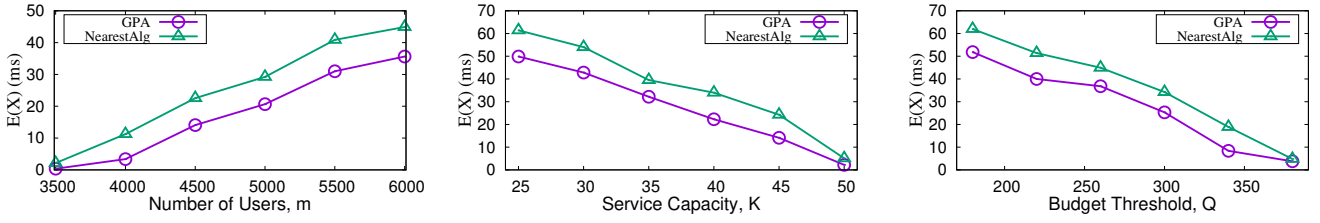
(a) Impact of the number of users (b) Impact of the service capacity (c) Impact of the budget threshold

Fig. 5: Edge servers at Starbucks' locations. Proportional Delay. The default setting is $m = 1,000$, $K = 40$, and $Q = 70$.



(a) Impact of the number of users (b) Impact of the service capacity (c) Impact of the budget threshold

Fig. 6: Edge servers at Starbucks' locations. Randomized Delay. The default setting is $m = 1,000$, $K = 40$, and $Q = 70$.



(a) Impact of the number of users (b) Impact of the service capacity (c) Impact of the budget threshold

Fig. 7: Edge servers at synthetical locations. Randomized Delay. The default setting is $n = 400$, $m = 5,000$, $K = 40$, and $Q = 300$.

the nearest edge server.

Since the number of edge servers is fixed, i.e., 92, in the above setting, we cannot evaluate the proposed algorithm with more edge servers. We also conduct another set of simulations where the locations of both edge servers and users are randomly generated within an area. In the following, all the results are obtained by averaging five independent runs.

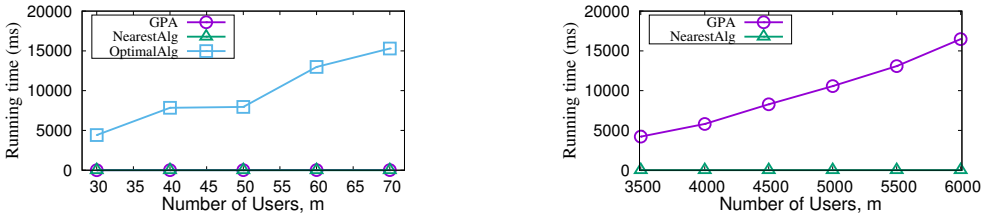
6.2 Results

Since it is impractical to run the brute-force OptimalAlg in general, we evaluate the performance of GPA, OptimalAlg, and NearestAlg under a smaller setting, in which we randomly decrease the number of Starbucks into 18 and we generate delays using the randomized way. Fig. 4 shows the results. In general, GPA achieves a near optimal placement when m is small, K is large, or Q is large; GPA outperforms NearestAlg at all times. Specifically, throughout this set of simulations, the weighted average interaction delay achieved by GPA is 217% at most (170% on average) as large as that achieved by OptimalAlg, while the $E(X)$ achieved by Nearest is 486% at most (313% on average) as large as that achieved by OptimalAlg.

Figs. 5 and 6 show the comparison results of GPA and NearestAlg when we generate delays using Proportion Delay and Randomized Delay, respectively. In Fig. 5(a) and Fig. 6(a), when the number of users increases, the weighted average interaction delay increases in both algorithms. The main reason is, users compete for service entities; when more users are involved, many of them cannot choose the nearest service entities from them as their entities, leading to a slightly larger pairwise interaction delay. In Figs. 5(b) and 6(b), when the service capacity increases, more users can choose better service entities as their entities; hence, the pairwise interaction delay decreases in both algorithms. In Figs. 5(c) and 6(c), when the budget threshold increases, more service entities can be placed, leading to a smaller interaction delay.

Fig. 7 shows the comparison results under a large setting in which the number of edge servers is 400. We see that, even in this large scale, GPA achieves a much smaller interaction delay than NearestAlg. Most of the findings from the previous figures still hold here. We would like to highlight here that, the weighted average interaction delays in Fig. 4 and Fig. 7 are of the same order of magnitude, given that most of the settings in Fig. 7 are the same as those in Fig. 4 except for n, m, K, and Q. That is, even if both of the number of users and the number of edge servers increase significantly, the achieved delay remains the same as long as we can increase service capacity (by forming a server cluster [3]) and budget threshold.

In Fig. 8(a), we see the running time of OptimalAlg



(a) Running time in the Starbucks' trace

(b) Running time in the synthetical trace

Fig. 8: Running time comparison.

is extremely high, since it is a brute-force algorithm. Fig. 8(b) shows the running time of GPA is higher than that of NearestAlg, since in each iteration of GPA, it needs to find the best edge server to place a new service entity that incurs the least interaction delay. It is worth mentioning that, although the running time of GPA is not small, it only runs when the set of users change significantly. In practice, given a scenario, GPA can output a curve (e.g., Fig. 7(c)) for the DIA service provider. Since placing more service entities incurs more monetary costs while the decrease of interaction delay brings more monetary benefits, the provider can choose the right balance based on the curve to maximize its monetary revenue.

7 Conclusion and Future Work

In this paper, we study the service entity placement at the network edge for enhancing the interactivity of DIAs. We present the interaction-oriented edge-enabled service entity placement problem and show it is NP-complete. We design an efficient placement algorithm and show its performance bound. We evaluated GPA with both real-world data traces and large-scale simulations, and observed that GPA performs close to the optimal algorithm and generally outperforms the baseline.

We discuss the limitations of our work which may inspire the future work. Firstly, our work does not consider processing delay at service entities as we believe the network latency is more difficult to improve than the processing delay [3]. A busy service entity can always be better provisioned to meet the delay requirements.

Secondly, we assume that the wireless connections between users and access points are fixed, e.g., u_5 connects with ap_1 and ap_2 at all times in Fig. 1(a). In reality, the connection between a user and an access point may change over time, due to the physical motion of the user. A simple way to adapt to this changing situation is to collect the connection statistics of each user and then use a connection distribution to represent a user.

Lastly, the service entity placement is optimized for the pairwise interaction delay in the ISEP problem. However, more and more interactions involve more than two users, e.g., an area-of-effect spell in DOTA can affect an area around a point and any user within the area would be affected. Under such a scenario, network latency has to be comprehensively considered for improving user-perceived quality-of-service. Extending GPA to handling the multi-user interaction delay is part of future work.

Acknowledgments

This work was supported in part by National Key R&D Program of China (2016YFC0800803), NSFC (61872175), NSF of Jiangsu Province (BK20181252), CCF-Tencent Open Fund, and Collaborative Innovation Center of Novel Software Technology and Industrialization. Jidong Ge is the corresponding author.

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proc. of Mobidata 2015*, pp. 37–42.
[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. of ACM MobiSys 2014*, pp. 68–81.
[3] L. Zhang and X. Tang, "Client assignment for improving interactivity in distributed interactive applications," in *Proc. of IEEE INFOCOM 2011*, pp. 3227–3235.
[4] Y. Liang, J. Ge, S. Zhang, J. Wu, Z. Tang, and B. Luo, "A utility-based optimization framework for edge service entity caching," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–12, 2019.
[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
[7] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 14–22, June 2013.
[8] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *Proc. of IEEE INFOCOM 2019*, pp. 2566–2574.
[9] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, 2015.
[10] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Prof. of IEEE INFOCOM 2016*, pp. 1–9.
[11] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *arXiv preprint arXiv:1605.08518*, 2016.
[12] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.
[13] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware vnf placement for service-customized 5g network slices," in *Proc. of IEEE INFOCOM 2019*, pp. 1–9.
[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *Proc. of ACM MobiSys 2010*, pp. 49–62.
[15] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. of ACM EuroSys 2011*, pp. 301–314.
[16] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. of INFOCOM 2012*, pp. 945–953.

[17] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "COMET: code offload by migrating execution transparently," in *Proc. of USENIX ODSI 2012*, pp. 93–106.
[18] F. Liu, P. Shu, and J. C. S. Lui, "AppATP: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3051–3063, Nov 2015.
[19] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Prof. of IEEE INFOCOM 2016*, pp. 1–9.
[20] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. of INFOCOM 2017*, 2017, pp. 1–9.
[21] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *Prof. of IEEE INFOCOM 2018*, pp. 1–9.
[22] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Prof. of IEEE INFOCOM 2018*, pp. 1–9.
[23] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE INFOCOM 2019*, pp. 1–9.
[24] Y. Bao, Y. Peng, C. Wu, and Z. Li, "Online job scheduling in distributed machine learning clusters," in *Prof. of IEEE INFOCOM 2018*, pp. 1–9.
[25] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *Proc. of IEEE ICDCS 2019*, pp. 1–11.
[26] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. of IEEE INFOCOM 2016*, pp. 1–9.
[27] R. Yu, G. Xue, and X. Zhang, "Application provisioning in fog computing-enabled internet-of-things: A network perspective," in *Prof. of IEEE INFOCOM 2018*, pp. 1–9.
[28] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Prof. of IEEE INFOCOM 2018*, pp. 1–9.
[29] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. of IEEE INFOCOM 2018*, pp. 1–9.
[30] V. Vazirani, *Approximation algorithms*. Springer, 2004.
[31] "Open Edge Computing," http://openedgecomputing.org/.
[32] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proc. of ACM EuroSys 2009*, pp. 205–218.
[33] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring latency variation in the internet," in *Proc. of ACM CoNEXT 2016*, pp. 473–480.

Yu Liang is a PhD candiate in Nanjing University. She received her MS degree from Nanjing University in 2011. She was a senior software engineer in Trend Micro China Development Center between 2011 and 2017. Her research interests include resource allocation in cloud and edge computing. Her publications include those appeared in COMNET, COMCOM, ICDCS, and Globecom.

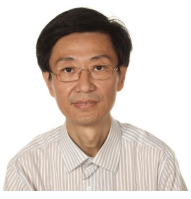
Jidong Ge is an Associate Professor at Software Institute, Nanjing University. He received his PhD degree in Computer Science from Nanjing University in 2007. His current research interests include cloud computing, distributed computing, workflow scheduling, workflow modeling, process mining. His research results have been published in more than 60 papers in international journals and conference proceedings including IEEE TSC, JASE, FGCS, JSS, Inf. Sci., ESA, ICSE, APSEC, ICSSP, HPCC, SEKE etc.

Sheng Zhang (M'14) is an assistant professor in the Department of Computer Science and Technology, Nanjing University. He is also a member of the State Key Lab. for Novel Software Technology. He received the BS and PhD degrees from Nanjing University in 2008 and 2014, respectively. His research interests include cloudedge computing and edge computing. To date, he has published more than 60 papers, including those appeared in *TMC*, *TPDS*, *TC*, *MobiHoc*, *ICDCS*, *INFOCOM*, *IWQoS*, and *ICPP*. He is a member of the IEEE and a senior member of the CCF.

Jie Wu (F'09) is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Mobile Computing, IEEE Transactions on Service Computing, Journal of Parallel and Distributed Computing, and Journal of Computer Science and Technology. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Lingwei Pan received the BS degree at the Software Institute, Nanjing University. He is currently working toward the MS degree at the Software Institute, Nanjing University, under the supervision of Prof. Jidong Ge. His research interests include cloud computing and NLP.

Tengfei Zhang received the BS degree at the Software Institute, Northeastern university, China. He is currently working toward the MS degree at the Software Institute, Nanjing University, under the supervision of Prof. Jidong Ge. His research interests include NLP and Data Mining.



Bin Luo is a full Professor at the Software Institute, Nanjing University. His main research interests include cloud computing, computer network, workflow scheduling, software engineering. His research results have been published in more than 60 papers in international journals and conference proceedings including IEEE TSC, ACM TOIST, FGCS, JSS, Inf. Sci., ESA etc. He is leading the institute of applied software engineering at Nanjing University.