Cost-Effective Edge Server Network Design in Mobile Edge Computing Environment

Ruikun Luo, Hai Jin, *Fellow, IEEE*, Qiang He, *Senior Member, IEEE*, Song Wu, *Member, IEEE*, and Xiaoyu Xia

Abstract—*Mobile edge computing* (MEC) deploys edge servers at the base station in the proximity of users to provide cloud computing-like computing and storage functionalities, which can achieve applications' low latency requirement at the network edge. The *edge server network* (ESN), constituted by edge servers in an area and the links between them, can host app vendors' services for serving nearby users. Many existing studies have demonstrated that a high ESN density allows for high service performance because edge servers can communicate and share resources with each other effectively over the ESN. However, in the real-world MEC environment, constructing a high-density ESN may incur high construction costs. The trade-off between construction cost and network density plays a vital role in the design of an ESN. Unfortunately, existing studies of MEC have commonly and simply assumed the densities of the ESNs in their experiments. In this paper, we make the first attempt to study the design of cost-effective ESNs with the aim to trade off between the network construction cost and the network density. We model this novel *Edge Server Network Design* (ESND) problem as a constrained optimization problem and prove its $\mathcal{NP}$-hardness. ESND-O as an optimal approach is proposed based on integer programming to solve small-scale ESND problems. Another approximation approach named ESND-A is designed to solve large-scale ESND problems efficiently. We conduct extensive experiments to test the performance of ESND-O and ESND-A on a real-world dataset, and the experimental results demonstrate their effectiveness and efficiency against four representative approaches.

Index Terms—Mobile edge computing, edge server network, network density, optimization algorithm

1 Introduction

In recent years, mobile traffic has rapidly increased with the fast growth of mobile and *internet-of-things* (IoT) services. CISCO's report [1] predicts that the world's mobile traffic is expected to increase at an average annual rate of 46%, and will reach 77 exabytes per month by 2022. Transmitting such a large amount of data can consume excessive network resources and incur heavy network traffic. Meanwhile, the cloud computing paradigm constrained by the heavy network transmitting overhead is failing to guarantee applications' low latency demands. To tackle these challenges, *mobile edge computing* (MEC) as a key enabler technology that facilitates the 5G mobile network, which pushes the cloud computing-like storage and computing functionalities to the edge of the network close to end-users and end-devices [2], [3].

In the MEC environment, adjacent edge servers can communicate with each other via high-speed links between them [4], [5]. The edge servers in a specific area and the links between constitute an *edge server network* (ESN) [6], [7]. App vendors can hire the computation and storage resources (together referred to as *resources* hereafter) on edge servers for deploying their services over the ESN, so as to serve nearby users with low latency [8]. This can significantly reduce the expensive cloud-to-edge data transmission costs [6], [9], [10] and the end-to-end service latency [11]. Over the ESN, geographically adjacent edge servers can share their resources to alleviate the issues raised by their constrained resources [12]. Compared with the cloud-edge architecture, the collaboration ESN can overcome the problem of single point of failure and the performance bottleneck caused by the backhaul network [13], [14].

Existing studies of MEC infrastructure focus on the placement of edge servers across optional locations to achieve various optimization objectives, e.g., minimum edge server deployment cost [15], maximum user coverage [16], maximum edge network robustness [17], etc. However, the importance of ESN design is neglected. Edge network density, measured by the average number of links per edge server, significantly impacts edge servers' ability to collaborate, and consequently the performance of the services deployed on edge servers. In general, a high network density connects edge servers to many other edge servers and allows them to utilize more system resources through collaboration. This allows for high service performance because it enables low-latency messaging and data transmissions between edge servers. This has been

- *Ruikun Luo, Hai Jin, and Song Wu are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: {rkluo, hjin, wusong}@hust.edu.cn.*
- *Qiang He is with the Department of Computing Technologies, Swinburne University of Technology, Melbourne, VIC 300, Australia. E-mail: qhe@swin.edu.au.*
- *Xiaoyu Xia is with the School of Information Technology, Deakin University, Geelong, VIC 3220, Australia. E-mail: xiaoyu.xia@deakin.edu.au.*

Manuscript received 10 Aug. 2021; revised 20 May 2022; accepted 23 May 2022. Date of publication 30 May 2022; date of current version 8 Dec. 2022.
This work was supported by the National Science Foundation of China (NSFC) under Grant 62032008.
(Corresponding author: Hai Jin.)
Recommended for acceptance by J. Li.
Digital Object Identifier no. 10.1109/TSUSC.2022.3178661

confirmed by the experimental results of many existing studies in the field of MEC, such as edge user allocation [18], edge data caching [19], [20], edge data distribution [6], edge data deduplication [21], and edge data synchronization [22]. Take the edge data caching problem [19], [20] as an example. It caches popular data on edge servers with the aims to guarantee the low data retrieval latency over the ESN for mobile users at minimum data caching costs. Collaboration among edge servers plays a vital role in this problem. As demonstrated by the experimental results presented in [19], [20], [23], a high network density allows for low data caching costs by enabling efficient collaboration between edge servers - it is easier for app vendors to ensure low data retrieval latency for their users with minimum cache data replicas over the ESN.

From the perspective of the *edge infrastructure provider* (EIP), e.g., Amazon and T-Mobile, a high ESN density often leads to a high network construction cost. It includes the hardware and labour costs of deploying networking devices like network cables, optical fibers, routers, switches, etc. These metrics are usually billed region-specifically. Thus, the geographic distances between edge servers are adopted in our model to measure the construction cost in a generic manner because it is usually costly to build a link between two far away edge servers and vice versa. A straightforward solution to constructing an ESN with a high network density in an area is to deploy a high-speed link between every pair of adjacent edge servers in the area. However, this can easily incur an excessive network construction cost - the density of 5G base stations is up to 50 per km^2 [24]. In practice, an EIP must ensure that the network construction cost does not exceed its budget. Within this budget, how to trade off between network density and network construction cost is an urgent and challenging problem for EIPs.

Focused on network connectivity in edge server network construction, this paper complements existing studies in designing sophisticated solutions to edge server network construction. It presents the first formal study of the *Edge Server Network Design* (ESND) problem, aiming to optimize the trade-off between network density and network construction cost. Its major contributions are as follows:

- We model the ESND problem as a constraint optimization problem formally and prove that it is $\mathcal{NP}$-hard.
- We propose an approach named ESND-O for finding optimal solutions to small-scale ESND problems based on integer programming, and an approximation approach named ESND-A for finding approximate solutions to large-scale ESND problems efficiently.
- We conduct comprehensive experiments on a real-world dataset to test the performance of ESND-O and ESND-A against two baseline approaches and two state-of-the-art approaches.

2 Motivating Example

Fig. 1 presents five edge servers in a specific area, e.g., Melbourne CBD, denoted as $\{s_1, s_2, \ldots, s_5\}$. The cost of constructing a link between two edge servers can be estimated by the EIP based on the surrounding environment. It is annotated as a weight of the edge between the two edge servers in the figure. To discuss and model the ESND problem generically, the cost of constructing a link between two edge servers is measured by the geographic distance between the edge servers. There are various ways to build links to construct the ESN in this area. From the EIP's perspective, the ESN must be adequately dense to ensure high performance for services deployed over the network. However, the network construction cost must be taken into account to achieve a cost-effective ESN design.

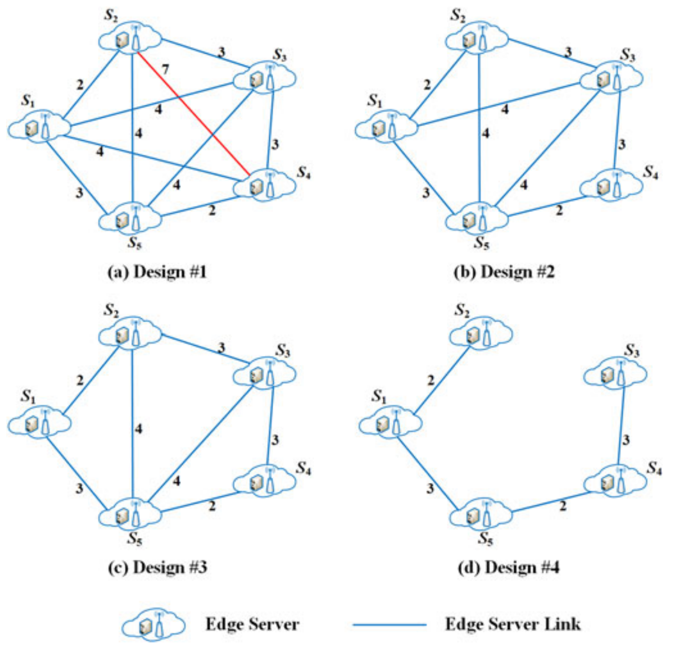


Fig. 1. Example of Edge server network design schemes.

Fig. 1a presents a straightforward ESN design, i.e., to build a link between every pair of edge servers. This design achieves the highest network density of 4 - each edge server is connected to an average of 4 other edge servers over the network. However, the network construction cost incurred by this strategy is high, reaching 36. This may not be the most cost-effective ESN design. For example, the long link between s_2 and s_4 incurs a high link construction cost, i.e., the cost of constructing a link. Fig. 1d depicts a low-cost ESN design that incurs a network construction cost of only 10, 72% less than that incurred by Design #1 presented in Fig. 1a. However, the network density produced by Design #4 is 60% lower than that produced by Design #1. This design significantly sacrifices network density, which translates into service performance directly, for a low cost. It is usually not the most cost-effective solution either from the EIP's perspective.

Figs. 1b and 1c present two ESN designs as trade-off solutions between Design #1 and Design #4, incurring network construction costs 31% and 42% lower than Design #1, respectively. Their network densities are only 20% and 30% lower than Design #1. The priorities for network density and network construction cost can be adjusted by the EIP to suit its need. For example, if the user density is high in the area, the EIP is likely to prioritize Design #2 over Design #3. Given a number of edge servers, there are many possible ESN designs. In real-world ESND scenarios, the number of

TABLE 1
Notations Summary

Notation	Description
$\mathcal{B}$	cost budget
$C(p)$	total cost of edge server links with strategy p
$CD(p)$	CD-CP of ESND with strategy p
$D(p)$	total density of edge server network with strategy p
d_i	degree of edge server s_i
$N(s_i)$	s_i's neighbor edge servers
n	finite number of edge servers
P	ESN design represented by a matrix
$p_{i,j}$	accessibility between s_i and s_j
S	finite set of edge servers
s_i	edge server i
W	distance matrix of edge servers
$w_{i,j}$	distance between s_i and s_j

edge servers to connect is usually much larger and different ESN designs offer various trade-offs between network density and network construction costs. From the EIP's perspective, it is important and challenging to be able to evaluate these designs and find one that offers the optimal trade-off.

3 Problem and Model Formulation

In this section, we first formulate the ESND problem and then prove its hardness theoretically. Table 1 summarizes the main notations and their definitions.

3.1 Problem Formulation

Given n edge servers in a specific area, denoted as s_1, ..., s_n, let graph $G(S, E)$ denote an ESN design that connects the edge servers, where each vertex represents an edge server $s_i \in S$ and each edge represents a link between edge servers.

As discussed in Section 1, the cost of building a link between two edge servers is mainly composed of hardware and labor costs. These are usually correlated with the physical distance between the edge servers. In general, it can be estimated by the EIP based on the surrounding environment and geographic distance between edge servers. Please note that the approaches proposed in this paper do not mandate that construction costs be measured by geographic distance. In practice, the costs of connecting edge servers can be estimated based on local cost factors, e.g., labor costs, geographic surroundings, hardware prices, etc. It can then be fed to our approaches as inputs. Let $w_{i,j}$ denote the geographic distance between edge servers s_i and s_j. An ESN design can be represented as a matrix $P = \{p_{i,j}, \forall s_i, s_j \in S\}$, where $p_{i,j} \in \{0, 1\}$ indicates whether edge servers s_i and s_j are connected in the design.

$$p_{i,j} = \begin{cases} 0 & \text{if } s_i \text{ is not connected to } s_j \\ 1 & \text{if } s_i \text{ is connected to } s_j \end{cases}. \tag{1}$$

Sometimes, it is too expensive or impossible to link two edge servers due to the sophistication in the surrounding environment, e.g., a river flowing between them. This is to be evaluated by the EIP area-specifically. If two edge servers can be connected, they are referred to as *neighbor edge servers*. Let $N(s_i)$ represent the set of adjacent edge servers of s_i. Let $|N(s_i)|$ denote the size of the set of s_i's neighbor edge servers. Thus, the maximum number of edge servers that edge server $s_i \in S$ can be connected to by any ESN design is $|N(s_i)|$:

$$\sum_{j=1}^{n} p_{i,j} \le |N(s_i)|, \ \forall s_i \in S. \tag{2}$$

The total network construction cost incurred by an ESN design P is calculated as follow:

$$C(P) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j} w_{i,j}, \ \forall s_i \in S, s_j \in N(s_i), \tag{3}$$

where $w_{i,j}$ denotes the distance between edge servers s_i and s_j. As discussed in Section 1, it indicates the cost of constructing the link between s_i and s_j.

Given the EIP's budget $\mathcal{B}$ for constructing the ESN, the total network construction cost must not exceed $\mathcal{B}$:

$$C(P) \le \mathcal{B}. \tag{4}$$

Let $D(P)$ denote the network density produced by P. It is measured by the ratio of the total degrees in the ESN over number of edge servers:

$$D(P) = \frac{\sum_{i=1}^{n} d_i}{n}, \quad \forall s_i \in S, \tag{5}$$

where d_i denotes the degree of edge server s_i, expressed as $\sum_{j=1, j \ne i}^{n} p_{i,j}$.

To allow for the highest service performance, the network density produced by P must be maximized:

$$\text{maximize } D(P). \tag{6}$$

In the meantime, the EIP usually also needs to minimize the cost of constructing the ESN:

$$\text{minimize } C(P). \tag{7}$$

As illustrated in Section 2, network density often conflicts with network construction cost and there is a trade-off to manage. To optimize the trade-off between the density and construction cost of an ESN, we employ the widely-used compromise programming technique [16], [25] to evaluate an ESN design P, which can be defined as follow:

$$CD(P) = \sqrt[\Phi]{\tau_c \left(\frac{C(P^*) - C(P)}{C(P^*) - C(P^{**})} \right)^{\Phi} + \tau_d \left(\frac{D(P^*) - D(P)}{D(P^*) - D(P^{**})} \right)^{\Phi}} \tag{8}$$

where P^* is the *hypothetically-best* ESN design, i.e., the one that produces the maximum network density and the minimum network construction cost of all the possible ESN designs. Here, $C(P^*)$ and $D(P^*)$ represent the minimum network construction cost (without consideration of network density) and the maximum network density (without consideration of network construction cost), respectively. They cannot be achieved at the same time by any ESN design. P^{**} is the *hypothetically-worst* ESN design, which is the opposite of P^*. Let Φ denote the distance measurement parameter between each decision criterion in compromise programming. Parameter $\Phi \in (0, \infty)$ is the measurement of a solution's closeness to the optimal solution P^* to the

ESND problem. The Φ value dictates how different factors contribute to the optimization objective collectively. A large Φ value will amplify the contributions made by the main factor. Usually, if one factor is not expected to overshadow the other factors, a small Φ value, typically 2, is used [17], [25]. In the context of ESND, both network density and construction cost are considered. Thus, $\Phi = 2$ is employed. The impact of different Φ values will be experimentally evaluated in Section 5. Parameter τ_c and τ_d are the standardized forms of the two optimization objectives' weights, i.e., the EIP's preference for network construction cost and network density where $\tau_c + \tau_d = 1.0$. For example, when $\tau_c = 1$ and $\tau_d = 0$, EIP pursues minimum network construction cost only. Such an ESN is suitable for latency-insensitive applications, e.g., distributing system update files over a large geographic area. When $\tau_c = 0$ and $\tau_d = 1$, network density maximization is EIP's only concern. Such an ESN is suitable for latency-sensitive applications like VR and online gaming in a small geographic area.

Let us suppose that $\tau_c = \tau_d = 0.5$ (as shown in Fig. 2), ESN design P_3 produces a higher network density than P_4 at the same network construction cost. In addition, ESN design P_2 incurs a lower network construction cost than P_3 while producing the same network density. Among the four ESN designs, P_2 is the most cost-effective.

To achieve the optimal trade-off, the optimization objective in the ESND problem is to minimize the compromise programming value, expressed as follow:

$$\text{minimize } CD(P). \tag{9}$$

3.2 Problem Hardness

Now, we introduce a classic $\mathcal{NP}$-hard problem named the *Travelling Salesman Problem* (TSP) to prove the $\mathcal{NP}$-hardness of the ESND problem by reducing it to the TSP problem. Given an undirected weighted graph $G = (V, E)$ and an $n \times n$ distance matrix $\Gamma = \{\gamma_{i,j}, \forall v_i, v_j \in V\}$, where $\gamma_{i,j}$ denotes the distance between vertexes v_i and v_j. Let $K = \{k_{i,j}, \forall v_i, v_j \in V\}$ denote the accessibility matrix, the TSP problem can be formulated as follows:

$$\min \sum_{j=1}^{n} \sum_{i \neq j, i=1}^{n} k_{i,j}\gamma_{i,j} \tag{10}$$

$$\text{s.t.: } k_{i,j} \in \{0,1\},\ i,j = 1,\dots,n \tag{11}$$

$$\sum_{i=1, i\neq j}^{n} k_{i,j} = 1,\ j = 1,\dots,n \tag{12}$$

$$\sum_{j=1, j\neq i}^{n} k_{i,j} = 1,\ i = 1,\dots,n. \tag{13}$$

The reduction from the TSP problem to the ESND problem can be conducted as follows: 1) Let the distance between any two edge servers s_i and s_j be a deterministic value; 2) Let the budget $\mathcal{B} = (n-1)w_{max}$ where n is the number of edge servers to be connected by the ESN and w_{max} denotes the maximum distance between edge servers over this ESN. Given an instance of the TSP problem $TSP(V, E, \Gamma, K)$, we can construct an instance of the ESND problem $ESND(V^*, E^*, W, P)$ with the reduction above where $|V| = |V^*|$ and $|\Gamma| = |W|$ in polynomial time. It is

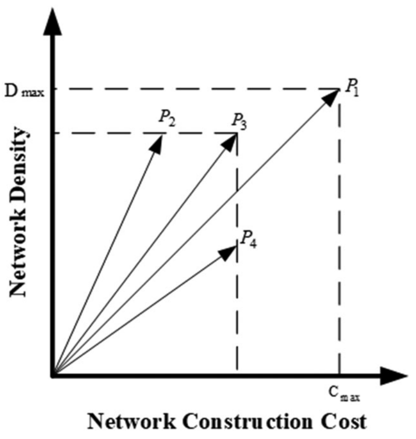


Fig. 2. Compromise programming strategies.

obvious that constraint (11) is equal to constraint (1). In the TSP problem, constraint (12) indicates that one vertex can only connect to another vertex once at maximum, while constraint (13) indicates that any vertex can connect to up to 2 other vertices. Therefore, constraints (12) and (13) can be converted to $\sum_{i=1}^{n}\sum_{j=1}^{n} k_{i,j} \leq 2$.

From the above reduction, the budget $\mathcal{B}$, i.e., the value of $(n-1)w_{max}$, must be greater than 2 when there are more than two edge servers. Thus, constraint (4) can be converted to $\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} \leq 2 \leq \mathcal{B}$, fulling constraint (12) and constraint (13). The optimization objective of the TSP problem is to minimize the total travel cost, i.e., to connect all the vertexes with the minimum number of links. In the ESND problem, $CD(P)$ is the compromise mean of $C(P)$ and $D(P)$. Eq. (8) can be reduced to $CD(P) = \frac{C(P*) - C(p)}{C(P*) - C(p**)}$ when the network construction cost is the only optimization objective considered. Thus, optimization objective (9) can be converted to minimizing $C(P)$, i.e., $\min \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}w_{i,j}$, so that it is equivalent to achieving objective (10).

In conclusion, any solution K to the TSP problem is a subset of the solution P to the corresponding ESND problem. Thus, the ESND problem can be reduced to the $\mathcal{NP}$-hard TSP problem and is thus also $\mathcal{NP}$-hard.

4 Approach Design

In this section, we first present ESND-O which finds the optimal solution to the ESND problem based on integer programming. Then, we present ESND-A for solving approximate solutions to large-scale ESND problems efficiently with a theoretical performance guarantee.

4.1 Optimal Approach

The ESND problem can be modeled as a *Constraint Optimization Problem* (COP) which consists of a finite set of variables $Y = \{y_1, \dots, y_n\}$ and a set of constraints $\mathcal{T}$ over Y. Given a set of edge servers $S = \{s_1, \dots, s_n\}$, a distance matrix Γ and an ESND budget $\mathcal{B}$ given by the EIP. Let a matrix $P = \{p_{i,j}, \forall s_i, s_j \in S\}$ denote the set of decisions for connecting edge servers, where $p_{i,j} \in \{0,1\}$ indicates whether edge servers s_i and s_j are connected. Let a $n-1$-dimensional vector $P[i]$ denote the connectivity between edge server s_i and other edge servers in ESN, where $P[i] = \{p_{i,1}, \dots, p_{i,i-1}, p_{i,i+1}, \dots, p_{i,n}\}$. Let a $n-1$-dimensional vector $\Gamma[i]$ denote the distance between edge server s_i and other edge servers in ESN. Thus, the network construction cost produced by an ESN design P can be calculated as follow:

$$C(p) = \sum_{i=1}^{n} P[i] \cdot \Gamma[i]^T. \tag{14}$$

The network density can be expressed as follow:

$$D(p) = \frac{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} p_{i,j}}{n}. \tag{15}$$

The value of Φ in Eq. (8) depends on the method for measuring ESN designs in the solution space. In compromise programming, $\Phi = 1, 2, \infty$ are commonly used. In the context of ESND, for $\Phi = 1$, weighted deviations of network construction cost and network density are assumed to compensate each other perfectly. This is usually not the case because network density and construction cost are not linearly correlated. If we take the limit as Φ goes to infinity, i.e., $\Phi = \infty$, Eq. (8) becomes Eq. (16):

$$\text{minmax}\left\{ \tau_c \left(\frac{C(P^*) - C(P)}{C(P^*) - C(P^{**})} \right), \tau_d \left(\frac{D(P^*) - D(P)}{D(P^*) - D(P^{**})} \right) \right\}, \tag{16}$$

where the main contributor will solely dictate the optimization objective and the problem degenerates into a typical minmax problem.

Thus, in Eq. (8), $\Phi = 2$ is chosen because the closeness of the the solution obtained based on our model to the optimal solution is measured by the euclidean distance. The COP model can be transformed to Eq. (17) based on compromise programming:

$$\min \sqrt{\tau_c \left(\frac{C(p^*) - C(p)}{C(p^*) - C(p^{**})} \right)^2 + \tau_d \left(\frac{D(p^*) - D(p)}{D(p^*) - D(p^{**})} \right)^2} \tag{17}$$

$$\text{s.t.: } p_{i,j} \in \{0, 1\} \tag{18}$$

$$c(P) \leq \mathcal{B} \tag{19}$$

Constraint (18) indicates that the connection decision between edge servers s_i and $s_j \forall s_i, s_j \in S$. Constraint (19) ensures that the network construction cost does not exceed the EIP's budget.

The above COP can be solved by integer programming solvers such as IBM CPLEX.1 There are a total of $\frac{n(n-1)}{2}$ possible network designs that connect n edge servers. To ensure that the edge server network is a connected graph, there are at least $n-1$ links in the network. Thus, a total of $C_{n(n-1)/2}^{n-1}$ possible solutions should be considered. Thus, the size of the solution space is $\frac{[n(n-1)/2]!}{(n-1)![(n-2)(n-1)/2]!}$. It confirms the $\mathcal{NP}$-hardness of the ESND problem such that optimal solutions to large-scale ESND scenarios cannot be found in polynomial time.

4.2 Approximation Approach

As proven in Section 3.2, the ESND problem is $\mathcal{NP}$-hard. It is intractable to find the optimal solution to a $\mathcal{NP}$-hard problem in large-scale scenarios. Although the ESDN problem can be solved offline, the memory of a typical computer usually would not suffice to solve a large-scale $\mathcal{NP}$-hard ESND problem. Thus, in practice, an efficient sub-optimal approach is needed to accommodate large-scale ESDN scenarios. In this section, we present ESDN-A, an approach specifically designed for finding sub-optimal solutions to large-scale ESDN problems rapidly. Together with ESDN-O, it offers a package of two approaches to the ESDN problem at different scales. The pseudo code of ESND-A is presented in Algorithm 1.

Algorithm 1. ESND-A

```
   Input: $G(S, E)$, $\mathcal{B}$, $W$
   Output: an ESND strategy $P = \{p_{i,j}, \forall s_i, s_j \in S\}$
1  initialization:
2  the decisions: $p_{i,j} \leftarrow 0 \; \forall s_i, s_j \in S$;
3  the CD-CP of strategy $P : CD(P) \leftarrow 0$;
4  the selected edge server set: $S^* \leftarrow \emptyset$;
5  the connection candidate set: $P^* \leftarrow \emptyset$;
6  end of initialization
7  for all $s_i \in S$ do
8    obtain $s_i$'s neighbor edge servers $N(s_i)$;
9    sort the edge servers in $N(s_i)$ by their CD-CP;
10   for $s_j \in N(s_i)$ do
11     if $|S^*| \neq |S| - |N(s_i)| \wedge \sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j} w_{i,j} < \mathcal{B}$ then
12       find the edge server $s_k$ with the lowest CD-CP;
13       $S^* \leftarrow S^* \cup \{s_k\}$;
14       update $P^*$ with $p_{i,k} = 1$;
15     end
16   end
17   if $CD(P^*) < CD(P)$ then
18     $CD(P) \leftarrow CD(P^*)$;
19     $P \leftarrow P^*$;
20   end
21 end
22 return $P$, $CD(P)$
```

The algorithm first initializes $p_{i,j}$, $CP(P)$, S^*, and P^* in Lines 1-6. Then, for each edge server $s_i \in S$, its neighbor edge servers are identified (Line 8). Next, the algorithm sorts the neighbor edge servers of s_i by their CD-CP $CD(P)$ in Line 9. Then, for each edge server in $N(s_i)$, this algorithm iterates for $|N(s_i)|$ times in Lines 10-16 to find the solution for edge server s_i. In each iteration, the algorithm selects $s_k \in N(s_i)$ with the lowest CD-CP, adds s_k to the set of candidate edge servers S^* and updates the decision matrix with $p_{i,k} = 1$, until every edge server in $N(s_i)$ is selected within the budget limit $\mathcal{B}$ (Lines 11-14). Finally, the algorithm compares the current CD-CP $CD(P^*)$ incurred by the candidate ESN design P^* with all the candidate ESND designs found in previous iterations. If it is lower than the current lowest one, the current ESN design is replaced by $P*$ (Lines 17-20).

4.3 Theoretical Analysis

The approximation ratio and time complexity of the ESND-A are analyzed in this section theoretically.

4.3.1 Approximation Ratio

Given an edge server set $S = \{s_1, s_2, \ldots, s_n\}$ in a specific area. Let λ_{OPT} denote the optimal solution to the ESND problem, and λ denote the approximation solution obtained

1. https://www.ibm.com/analytics/cplex-optimizer

by ESND-A. Let w^i_{min} denote the shortest distance between edge server s_i and its neighbor edge servers $N(s_i)$.

Now, we prove the approximation ratio of $CD(P)$ by measuring the gap between the solution found by ESND-A and the optimal solution found by ESND-O. As shown in Algorithm 1, ESND-A iterates for each neighbor edge server of each edge server $s_i \in S$ to find the decision that produces the lowest CD-CP. Thus, there is:

$$w_{i,j} \geq min\{w^i_{min}, w^j_{min}\}. \tag{20}$$

The total cost $C(\lambda_{OPT})$ incurred by the optimal solution λ_{OPT} is $\sum_{i=1}^n \sum_{j=1}^n w_{i,j}$. From (20), we can infer:

$$C(\lambda_{OPT}) \quad \geq \sum_{i=1}^{n}\sum_{j=1}^{n} min\{w^i_{min}, w^j_{min}\} = \sum_{i=1}^{n} \alpha_i w^i_{min}, \tag{21}$$

where α_i is the number of links between s_i and $N(s_i)$ selected by the optimal solution, i.e., $1 \leq \alpha_i \leq n-1$. Let us assume that k is the number of edge servers recently added into the set of selected edge servers S^*. Thus, there is $\sum_{i=1}^n \alpha_i = min\{2k, n\}$.

From (21), we can infer that the number of cases where $\alpha_i = 1$ is at most k, i.e, $\{w^1_{min}, w^2_{min}, \ldots, w^k_{min}\}$. The number of cases with $\alpha_i = 2$ is at least $\min\{2k, n\} - k$, i.e., $\{w^{k+1}_{min}, w^{k+2}_{min}, \ldots, w^{min\{2k,n\}}_{min}\}$.

$$C(\lambda_{OPT}) = \sum_{i=1}^{n} \alpha_i w^i_{min} \geq \sum_{i=k+1}^{min\{2k,n\}} 2w^i_{min} \tag{22}$$

Now we make $k = 2^0, 2^1, 2^2, \ldots, 2^{\log_2 n-1}$ and then sum the network construction cost incurred by each value of k. Based on (22), we can easily prove (23) by the mathematical induction proof.

$$\log_2 n \cdot C(\lambda_{OPT}) \geq \sum_{k=2^0}^{2^{\log_2 n-1}} \sum_{i=k+1}^{min\{2k,n\}} 2w^i_{min} \geq \sum_{i=2}^{n} 2w^i_{min} \tag{23}$$

Based on (22) and (23), we can obtain:

$$(log_2 n + 1)C(\lambda_{OPT}) \geq \sum_{i=1}^{n} 2w^i_{min} = C(\lambda). \tag{24}$$

The network density $D(\lambda_{OPT})$ produced by the optimal solution λ_{OPT} can be expressed as $D(\lambda_{OPT}) \leq n-1$. Let Δ^*_C denote the difference in network construction cost between $C(\lambda)$ and the optimal solution, and Δ^*_D denote the their difference in network density. According to (17) and (24), the following equation stands:

$$\begin{aligned} CD(\lambda_{OPT}) &\geq \sqrt{\tau_c(\Delta^*_C)^2 + \tau_d(\Delta^*_D)^2}\ CD(\lambda) \\ &\geq \sqrt{\frac{(\log_2 n + 1)^2 + (\frac{1}{n-1})^2}{2}}\ CD(\lambda) \\ &\geq \frac{\sqrt{2}(\log_2 n + 1)}{2}\ CD(\lambda). \end{aligned} \tag{25}$$

Therefore, the approximation ratio of ESND-A algorithm is $\frac{\sqrt{2}}{2}(\log_2 n + 1)$. There is only a logarithmic gap between the solution found by ESDN-A and the optimal solution.

4.3.2 Time Complexity

Let us suppose an ESND problem with n edge servers $S = \{s_1, s_2, \ldots, s_n\}$. For each edge server $s_i \in S$, x_i is the number of s_i's neighbor edge servers. The worst-case ESND scenario is that each edge server has $n-1$ neighbor edge servers, i.e., $x_i = n-1$. Thus, the inner iterations of Algorithm 1 (Lines 10-16) will run $n-1$ times until every edge server $s_j \in N(s_i)$ has been involved into the candidate edge server set S^*. The time complexity of sorting these edge servers in Lines 9 is at most $O(\log(n-1))$. Given an infinite budget $\mathcal{B} = \infty$, the time complexity of the overall iteration is no more than $O(n(n-1))$. Thus, the time complexity of ESND-A is $O(n(n-1)) + O(n\log(n-1))$, i.e., $O(n^2)$.

5 Evaluation

To evaluate the effectiveness and efficiency of ESND-O and ESND-A in different ESND scenarios, we conduct extensive and comprehensive experiments in this section.

5.1 Experimental Settings

5.1.1 Dataset

The experiments are conducted on the widely-used EUA data set [6], [26] which contains 1,464 real-world edge servers with their geographic coordinates in Metropolitan Melbourne, Australia.

5.1.2 Competing Approaches

This paper is the first attempt to solve the new ESDN problem. ESND-O and ESND-A are the first approaches designed specifically for solving the ESND problem. To properly evaluate the performance of ESND-O and ESND-A, we include ESND-R, ESND-C as well as Homa as representative competing approaches. However, these approaches are expected to suffer from low effectiveness because they do not consider network density and construction cost jointly. To bridge this gap, we include an approach designed based on the popular NSGA-II algorithm as another competing approach, which are widely employed to solve multi-objective optimization problems. In fact, NSGA-II is adopted as a competing approach in the many studies of MEC problems such as [19], [27]. The details of the four competing approaches can be found below.

- *ESND-R*: This approach randomly connects neighbour edge servers to construct the ESN under the budget constraint.
- *ESND-C*: This approach heuristically connects neighbour edge servers with the shortest distance without considering network density under the budget constraint.
- *Homa* [28]: This heuristic approach aims to minimize the network construction cost while ensuring the ESN is a connected network. At the start, Homa randomly selects an edge server from all the edge servers. After that, Homa always connects two edge servers that are closest to already-connected edge servers under the cost budget constraint in each iteration.

TABLE 2
Parameter Settings

	n	τ_d	Φ
Set #1.1	10, 15, ..., 35	0.5	2
Set #1.2	20	0, 0.25, ..., 1.0	2
Set #2.1	50, 100, ...,250	0.5	2
Set #2.2	150	0, 0.25, ..., 1.0	2
Set #2.3	150	0.5	1, 2, ..., 5

- *NSGA-II* [19], [29]: Based on the widely-used NSGA-II algorithm, this evolutionary approach first encodes the ESND problem according to the connection of edge servers, and then uses a partially mapped crossover operator, a uniform mutation operator, a proportional selection operator, and a fitness function to control the evolution toward the optimal solution during the entire iteration process. To evaluate the performance of the NSGA-II fairly and effectively, we conduct extensive experiments on NSGA-II to purse the best results and set the population size, mutation probability, and crossover probability to 100, 0.02, and 0.85, respectively. In each iteration, we select the optimal individual from the present populations based on the fitness function and proportional selection operator.

5.1.3 Parameter Settings

To evaluate ESND-O and ESND-A comprehensively, three parameters are considered in this experiments:

- *Number of edge servers* (n): This parameter decides the size of ESN G. It varies from 10 to 35 in Set #1.1, and from 50 to 250 in Set #2.1.
- *Distance measurement parameter* (Φ): As discussed in Section 4.1, this parameter dictates how different factors collectively contribute to optimization objective (17). It varies from 1 to 5 in Set #2.3.
- *Priority for network density* (τ_d): This parameter is used in Eq. (8) and decides the weight of network density in the optimization objective. It varies from 0 to 1.0 in steps of 0.25 in Set #1.2 and Set #2.2. Correspondingly, the weight of network construction cost τ_c varies from 1.0 to 0.

5.1.4 Performance Metrics

We employ four metrics for performance evaluation:

- *Overall CD-CP* ($CD - CP$): This metric indicates the compromise mean of the network density and network construction cost in the ESND problem. It is calculated by Eq. (17), the lower the better.
- *Network construction cost* ($cost$): This metric is calculated by Eq. (3) and indicates the network construction cost incurred by the ESN design, the lower the better.
- *Network density* ($density$): This metric indicates the network density produced by the ESN design. It is calculated by Eq. (5), the higher the better.
- *Computational overhead* ($time$): This metric is measured by the computational time taken by an approach to find the solution, the lower the better.

The parameter settings in the experiments are summarized in Table 2. When the value of each parameter changes, we repeat the experiments for 100 times and report the averaged value. As discussed in Section 2, the possibility of building a link between two edge servers is evaluated by the EIP based on the surrounding environment. In the experiments, we employ a distance threshold of 300m to decide whether two edge servers can be connected. Edge servers distanced more than 300m cannot be connected. This setting enables the reproduction of our experimental results based on the EUA dataset.

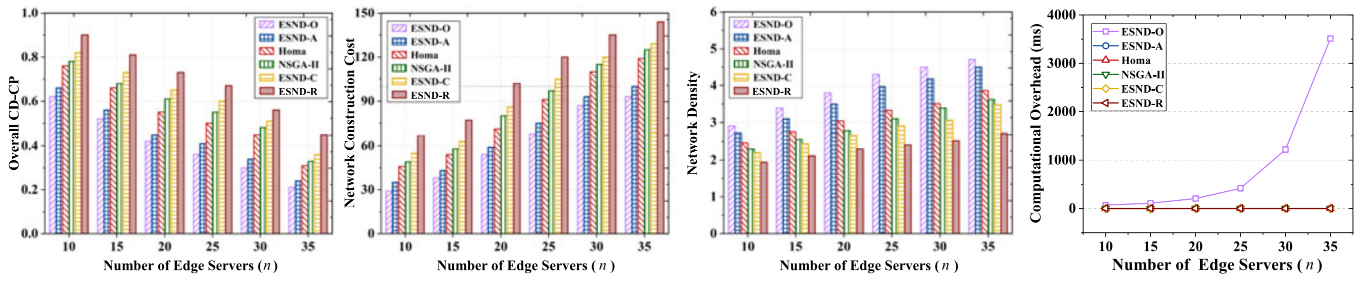
5.2 Experimental Results

The experimental results of Set #1 and Set #2 are presented and analyzed respectively in this section.

5.2.1 Experiment Set #1

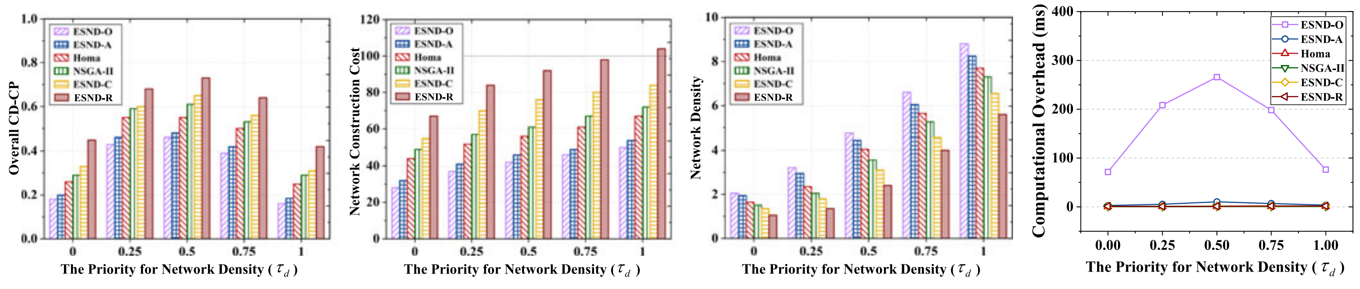
Effectiveness. Figs. 3 and 4 illustrate that ESND-O achieves the lowest CD-CP, the lowest network construction cost, and the highest network density in Set #1. On average across all the effectiveness metrics, ESND-O outperforms ESND-A by 6.88%, Homa by 25.39%, NSGA-II by 32.94%, ESND-C by 41.23%, and ESND-R by 59.47%.

Fig. 3a demonstrates the significant impact of n on the overall CD-CP in Set #1.1. The overall CD-CPs obtained by all the approaches decrease as n increases from 10 to 35. When n increases, the scale of the ESND problem increases. This immediately increases the number of links between edge servers that need to be constructed. Among all the six approaches, ESND-O always achieves the lowest overall CD-CP, 8.31% lower than ESND-A, 26.14% lower than Homa, 31.59% lower than NSGA-II, 35.23% lower than ESND-C, and 43.56% lower than ESND-R.



(a) $CD - CP$ vs. n (b) $cost$ vs. n (c) $density$ vs. n (d) $time$ vs. n

Fig. 3. Performance versus. n in set #1.1.



(a) $CD-CP$ vs. τ (b) $cost$ vs. τ (c) $density$ vs. τ (d) $time$ vs .τ

Fig. 4. Performance versus. τ_d in Set #1.2.

Fig. 3b shows the network construction costs achieved by the six approaches in Set #1.1. The network construction costs incurred by the approaches increase when n increases from 10 to 35. With the increase in the number of edge servers n, the number of needed links connect the edge servers increases. The number of each edge server's neighbor edge servers is more likely to increase when n increases. This makes it easier to improve network density at a low extra construction cost. Overall, more links must be built and the network construction cost increases.

Specifically, ESND-O achieves the lowest network construction cost, outperforming ESND-A, Homa, NSGA-II, ESND-C, and ESND-R by 7.12%, 25.64%, 31.27%, 35.82%, and 46.48%, respectively.

Fig. 3c demonstrates the impact of the number of edge servers n on network density in Set #1.1. When the number of edge servers n increases, the network densities produced by the ESN design formulated by the approaches increase. Interestingly, the increased rates of ESND-O and ESND-A are lower. When n increases from 10 to 35, the distance constraint discussed in Section 3 comes into play and leads to a higher number of links between each edge server and its neighbor edge servers. However, the number of each server's neighbor edge servers does not increase significantly. Among all the six approaches, ESND-O always produces the highest network density, with significant advantages over the other five approaches - 6.56%, 25.38%, 33.82%, 41.57%, and 68.72% over ESND-A, Homa, NSGA-II, ESND-C, and ESND-R on average.

Fig. 4a illustrates the overall CD-CP values achieved by the six approaches when the priority for network density τ_d increases from 0 to 1 in steps of 0.25 in Set #1.2. ESND-O and ESND-A achieve the lowest and second-lowest overall CD-CP values, respectively. Specifically, ESND-O outperforms ESND-A by 8.27%, Homa by 25.39%, NSGA-II by 32.18%, ESND-C by 37.35%, and ESND-R by 47.94%. When $\tau_d = 0$ or $\tau_d = 1$, the trade-off between network construction cost and network density becomes a single optimization objective, i.e., cost-effective oriented or density oriented. Thus, the six approaches all achieve the lowest overall CD-CP with $\tau_d = 0$ or $\tau_d = 1$ against other values of τ_d, i.e., $\tau_d \in (0, 1)$.

Fig. 4b shows the network construction cost incurred by the six approaches in Set #1.2. When the priority of network density τ_d increases, network density becomes increasingly important. All the approaches will pursue higher network density by building more links to construct the ESN. This immediately increases the network construction cost. Among the competing approaches, ESND-O and ESND-A always achieve the lowest and second-lowest network construction costs, respectively. ESND-O saves the network construction cost by 8.89%, 28.03%, 34.20%, 44.79%, and 54.70% against ESND-A, Homa, NSGA-II, ESND-C, and ESND-R on average. ESND-A saves the network construction cost by 21.72%, 28.15%, 39.24%, and 52.11% against Homa, NSGA-II, ESND-C, and ESND-R on average.

Fig. 4c shows the network density produced by the six approaches in Set #1.2. When τ_d increases from 0 to 1 in steps of 0.25, the network densities produced by all the six approaches increase linearly. Overall, ESND-O always achieves the highest network density, 6.21% more than ESND-A, 21.76% more than Homa, 34.57% more than NSGA-II, 52.45% more than ESND-C, and 89.47% more than ESND-R. Interestingly, the network density increases slower when τ_d increases. When the priority for network density increases, the network density becomes more important than the network construction cost. Thus, we can see that the network density experiences a significant increase when τ_d increases from 0 to 0.25. However, when τ_d varies from 0.5 to 1, it is difficult to achieve a huge improvement in network density while pursuing a low cost.

Efficiency. Fig. 3d and 4d demonstrate the high computational overheads of ESND-O, even in small-scale ESND scenarios. This validates the $\mathcal{NP}$-hardness of the ESND problem proved in Section 3.2. It shows that ESND-O is indeed not suitable for solving large-scale ESND problems.

In Fig. 3d, it is clear that ESND-O takes the most computation time of all in Set #1.1. All the six approaches' computation overheads increase when n increases from 10 to 35. When $n = 35$ in Set #1.1, ESND-O takes an average of 3,502.30 ms to find the optimal solution. The increase in n increases the scales of the simulated ESND scenarios. In these scenarios, more candidate solutions must be inspected to find the final solution.

In Fig. 4d, ESND-O still incurs the highest computational overhead compared with the other approaches, similar to the results presented in Fig. 3d. Interestingly, the computational overhead of ESND-O does not increase linearly with the increase in τ_d. When $\tau_d = 0$, the network density is the only optimization goal of the ESND problem without consideration of network construction cost. ESND-O can find the optimal solutions by pursuing only a single objective. This also applies to $\tau_d = 1$. ESND-O takes 71.60 ms to find the optimal solution when $\tau_d = 0$, 76.4 ms when $\tau_d = 1$. When $\tau_d = 0.5$, ESND-O needs to achieve the optimal trade-off between network density and network construction cost,

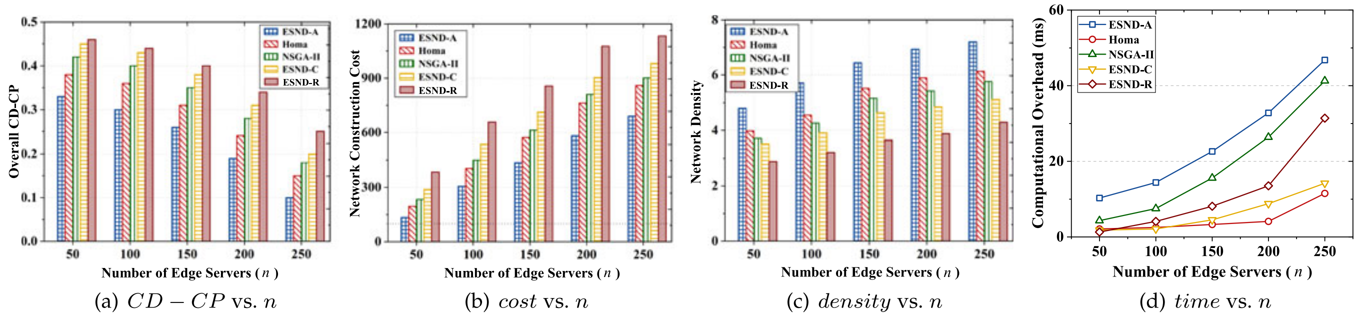


Fig. 5. Performance versus. n in Set #2.1.

i.e., achieving the highest network density and the lowest network construction cost with equal priorities. This complicates the ESND problem and takes the most time to find a solution, 265.8 ms on average.

Compared with ESND-O, ESND-A is much more efficient in solving large-scale ESND problems. For example, it takes only 2.06 ms on average to find a solution in Set #1, only 0.03% of what ESND-O takes. Unfortunately, Figs. 3 and 4 do not illustrate ESND-A's efficiency clearly. In the next section, we will clearly demonstrate the performance differences between ESND-A and Homa, NSGA-II, ESND-C, ESND-R in large-scale ESND scenarios in Set #2.

5.2.2 Experiment Set #2

Effectiveness. Figs. 5a, 5b, and 5c, and Figs. 6a, 6b, and 6c demonstrate the ESND-A's superior performance in maximizing network density and minimizing network construction cost in large-scale ESND scenarios. The network construction cost incurred by ESND-A is 21.60%, 27.85%, 34.77%, and 43.20% lower than Homa, NSGA-II, ESND-C, and ESND-R, respectively. The network density produced by ESND-A is 26.06%, 41.88%, 59.29%, and 103.26% higher than Homa, NSGA-II, ESND-C, and ESND-R, respectively.

Fig. 5a illustrates the significant advantages of ESND-A in achieving the best trade-off between network construction cost and network density, with an advantage of 20.81% over Homa, 29.75% over NSGA-II, 35.44% over ESND-C, and 39.84% over ESND-R. Fig. 5b shows the ability of ESND-A to minimize network construction cost. It outperforms Homa by 24.63%, NSGA-II by 31.03%, ESND-C by 40.18%, and ESND-R by 50.57%. Fig. 5c illustrates the significant advantages of ESND-A to maximize network density, i.e., 19.62% over Homa, 30.22% over NSGA-II, 41.01% over ESND-C, and 73.49% over ESND-R.

According to Fig. 6a, it shows the advantages of ESND-A, which are 21.43% over Homa, 29.03% over NSGA-II, 35.62% over ESND-C, and 47.47% over ESND-R. Fig. 6b shows that ESND-A achieves the lowest network construction cost of all, 18.57% lower than Homa, 24.67% lower than NSGA-II, 29.35% lower than ESND-C, and 35.83% lower than ESND-R. Fig. 6c shows the ability of ESND-A to maximize network density. It outperforms Homa by 32.50%, NSGA-II by 55.48%, ESND-C by 77.56%, and ESND-R by 132.04%.

Fig. 7 shows the impacts of parameter Φ on the overall CD-CP. As shown in Fig. 7a, when Φ increases from 1 to 5, the CD-CP obtained by ESND-A decreases first and then increases. When Φ exceeds 2, the increase in CD-CP slows down as Φ increases. This is because a large Φ value tends to turn the optimization problem into a minmax problem, as discussed before in Section 4.1. Interestingly, the CD-CP value obtained with $\Phi = 1$ is larger than that obtained with $\Phi = 2$. The reason is the changes in $C(p)$ and $D(p)$ are not linearly or negatively correlated. This increases the difficulty in balancing between construction cost and network density. Moreover, equal weights are set for network density and construction cost in the experiments, i.e., $\tau_c = \tau_d = 0.5$. An increase in Φ puts more emphasis on the impact of $C(p)$ in the optimization process. Therefore, the network density shown in Fig. 7c increases slower as Φ increases while the construction cost shown in Fig. 7b increases faster. This confirms the analysis in Section 4.1.

Efficiency. Figs. 5d and 6d demonstrate the time taken by the five approaches to find a solution in Set #2. ESND-O is excluded from Set #2 due to its unreasonable computational time taken to find solutions. ESND-A always spends more computational time than the other four competing approaches on finding a solution, i.e., 42.15%, 162.72%, 251.35%, and 288.40% more than NSGA-II, ESND-R, ESND-

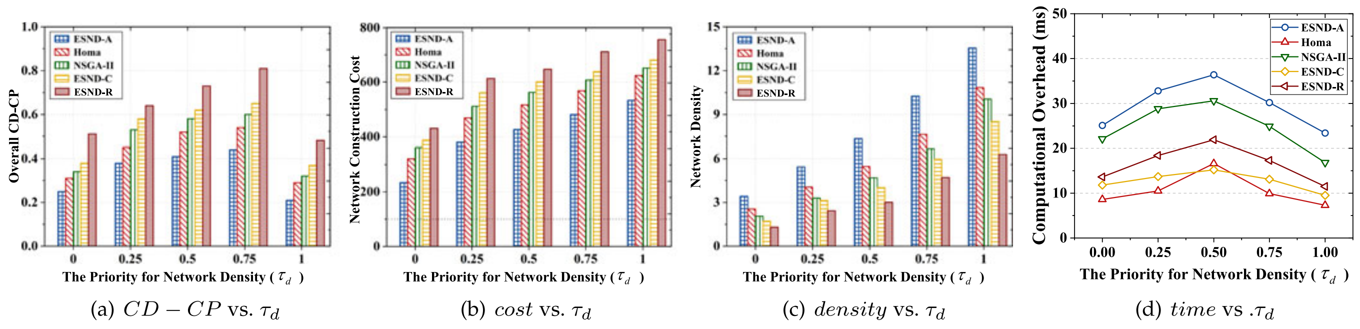
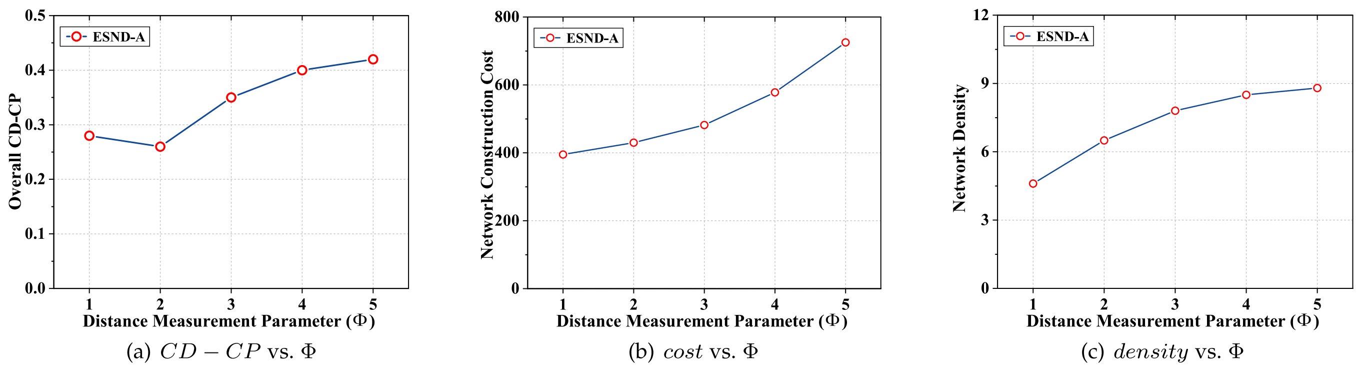


Fig. 6. Performance versus. τ_d in Set #2.2.



(a) $CD-CP$ vs. Φ (b) $cost$ vs. Φ (c) $density$ vs. Φ

Fig. 7. Performance versus. Φ in set #2.3.

C, and Homa, respectively. Overall, ESND-A scales with n and τ_d, taking no more than 50ms to find a solution in Set #2. Given its significant improvements in maximizing network density and minimizing network construction costs over Homa, NSGA-II, ESND-C, and ESND-R, its extra computational overhead is worthwhile in most, if not all, large-scale real-world ESND scenarios. The computational overhead of all the approaches increases when the number of edge servers n increases from 50 to 250, which is illustrated in Fig. 5d. With the increase of n, the space of possible ESN designs that can satisfy all the constraints increases. According to the process Lines 7-16 in Algorithm 1, it takes ESND-A more iterations to compute and sort the overall CD-CP values for each edge server. Thus, the computation time spent by ESND-A to find a solution increases. Fig. 6d demonstrates the computational overhead incurred by all the five approaches with different priorities for network density. The reason behind these phenomena is similar to what was discussed in Fig. 4d and thus are not discussed in detail here.

5.2.3 Result Summary

ESND-O and ESND-A are designed to solve ESND problems in small-scale scenarios and large-scale scenarios, respectively. Aiming to find the optimal solutions to the NP-hard ESND problem, ESND-O will always suffer from high computational overheads in large-scale scenarios, which is inevitable. Thus, an efficient approach for finding sub-optimal solutions to large-scale ESND problems is needed. However, this sub-optimal approach must be designed carefully to avoid significant effectiveness compromise. Designed based on straightforward heuristic or existing techniques, ESND-R, ESND-C, NSGA-II, and Homa unfortunately suffer from low effectiveness in terms of CD-CP, connection cost and network density. ESND-A outperforms these approaches by an average of 36.34% in CD-CP, 42.32% in connection cost, and 74.89% in network density, taking an average of only 13.14 more milliseconds to find a solution. Thus, compared with ESND-R, ESND-C, NSGA-II, and Homa, ESND-A offers a better trade-off between effectiveness and efficiency.

6 Related Work

The *mobile edge computing* (MEC) paradigm enables latency-sensitive and bandwidth-consuming applications to be deployed at the network edge by facilitating an *edge server network* (ESN) within users' close geographic proximity. Edge server network, as a significant part of the MEC infrastructure, is comprised of edge servers and network links between them. As experimentally demonstrated in a series of studies of MEC [6], [30], [31], these links allow edge servers to leverage their computing and storage resources collectively, which is critical to ensuring the performance of the services deployed in the MEC systems. In recent years, the importance of ESN is starting to attract researchers' attention.

Existing studies of ESN focus on the *edge server placement* (ESP) problem, trying to achieve various optimization objectives, e.g., minimizing edge server deployment cost [15], maximizing edge network robustness [17], minimizing energy consumption [32], [33], and maximizing user coverage [16], etc. To name a few, Zeng *et al.* [15] formulated a k-ESP model based on integer programming technology and proposed a greedy algorithm that aims to place as few edge servers as possible while guaranteeing user's *quality-of-service* (QoS) in wireless metropolitan area networks. Cui *et al.* [17] made the first attempt to consider the ESN's robustness from the EIP's perspective. They formulated a generic model to achieve maximum network robustness while minimizing the edge server deployment cost within the EIP's budget. Badri *et al.* [32] modeled the application placement problem as a multi-stage stochastic optimization problem and proposed a parallel sample average approximation algorithm to maximize system QoS under an energy budget. Cui *et al.* [16] formulated the ESP problem as a constraint optimization problem and solved it with heuristic algorithms to balance the trade-off between user coverage and network robustness.

Some researchers have employed genetic algorithms or machine learning to solve the problem of edge server placement [27], [29], [34], [35], [36]. Yuan *et al.* [34] proposed a deep learning-based approach for edge server placement, with the aim to minimize data migration cost incurred by user mobility. It employed a long short-term memory-based data prediction mechanism to predict users' task requirements and the resource prices. Then, it used a hierarchical clustering algorithm to place edge servers so that resources can be allocated properly to serve users. Lu *et al.* [35] modeled edge server placement over the internet of vehicles as a Markov decision process that aims to achieve a high task finish rate and low latency. Then, they proposed an approach to find the optimal solution to edge server placement based on a deep Q-network. Xu *et al.* [27] proposed an approach that finds solutions to edge server placement with the NSGA-II algorithm. Its key idea is to allow proper workload balance between edge servers. Zhao *et al.* [29] proposed a modified genetic algorithm to solve the k edge server placement problem where k is the number of edge servers to place. It jointly optimized the data retrieval latency and workload balance based on an elite policy enabled NSGA-II algorithm. To tackle the edge server placement problem over the dynamic internet of vehicles, Shen *et al.* [36] considered the traffic dynamics over the Internet-of-Vehicles and designed an approach for edge server placement that tries to balance between network performance and reconstruction cost incurred by edge server adjustment.

However, these studies have neglected the importance of network density. The density of an ESN greatly affects the performance of applications deployed on the ESN in various MEC scenarios, e.g., edge user allocation [18], edge data caching [19], [20], edge data delivery [6], edge data synchronization [22], etc. To name a few, Xia *et al.* [23] studied the *edge data caching* (EDC) problem for caching popular data on ESN. They aimed to minimize the number of cached data replica while ensuring that all the users can be served. The experimental results presented in [23] show that a high ESN density allows edge servers to connect to each other with links. This increases the user's chances of retrieving data from nearby edge servers within fewer hops. As a result, the data retrieval latency and caching cost decrease significantly. The authors of [6] studied the *edge data distribution* (EDD) problem with the aim to minimize the cost caused by distributing data from the cloud to edge servers while satisfying the latency limit. Their experimental results showed that the density of an ESN significantly impacts the overall data distribution cost. A higher network density usually makes edge servers have a higher chance to retrieve data within the latency limitation. This also significantly reduces the overhead incurred by distributing data from the remote cloud to edge servers via the links between them.

From the experimental results of these existing studies, we can clearly see the importance of ESN density. This drives us to study the novel ESND problem, and design ESND-O and ESND-A for finding cost-effective ESN designs that achieve the optimal trade-off between network density and network construction cost for EIPs.

7 Conclusion and Future Work

To summarise this paper, we made the first attempt to introduce, motivate, formulate, and solve the *edge server network design* (ESND) problem, aiming to achieve the optimal trade-off between the network density and network construction cost. We proved its $\mathcal{NP}$-hardness and proposed ESND-O and ESND-A to find the optimal and approximate solutions to small-scale and large-scale ESND problems, respectively. Experimental results conducted on a widely-used dataset demonstrated the effectiveness and efficiency of ESND-O and ESND-A. This paper complements existing studies in designing sophisticated solutions to edge server network construction. In the future, we will study the ESND problem further by taking the network robustness into consideration and considering the trade-off between network service performance and network construction cost.

References

[1] C. V. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update," *2017–2022 CISCO White Paper*, Feb. 2019.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[3] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021.

[4] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2833–2849, Dec. 2020.

[5] H. Jin, R. Luo, Q. He, S. Wu, Z. Zeng, and X. Xia, "Cost-effective data placement in edge storage systems with erasure code," *IEEE Trans. Serv. Comput.*, early access, Feb. 24, 2022, doi: 10.1109/TSC.2022.3152849.

[6] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 31–44, Jan. 2021.

[7] N. Tziritas *et al.*, "Data replication and virtual machine migrations to mitigate network overhead in edge computing systems," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 4, pp. 320–332, Oct.–Dec. 2017.

[8] Q. He, Z. Dong, F. Chen, S. Deng, W. Liang, and Y. Yang, "Pyramid: Enabling hierarchical neural networks with edge computing," in *Proc. Web Conf.*, 2022, pp. 1860–1870.

[9] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 2004–2017, Sep. 2018.

[10] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, early access, Mar. 12, 2019, doi: 10.1109/TSUSC.2019.2904680.

[11] L. Hu, W. Li, J. Yang, G. Fortino, and M. Chen, "A sustainable multi-modal multi-layer emotion-aware service at the edge," *IEEE Trans. Sustain. Comput.*, early access, Jul. 15, 2019, doi: 10.1109/TSUSC.2019.2928316.

[12] C. Ding, A. Zhou, Y. Liu, R. Chang, C.-H. Hsu, and S. Wang, "A cloud-edge collaboration framework for cognitive service," *IEEE Trans. Cloud Comput.*, May 25, 2020, doi: 10.1109/TCC.2020.2997008.

[13] F. Zhang *et al.*, "Online learning offloading framework for heterogeneous mobile edge computing system," *J. Parallel Distrib. Comput.*, vol. 128, pp. 167–183, 2019.

[14] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020.

[15] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, no. 1, 2019, Art. no. 32.

[16] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, "Trading off between user coverage and network robustness for edge server placement," *IEEE Trans. Cloud Comput.*, early access, Jul. 10, 2020, doi: 10.1109/TCC.2020.3008440.

[17] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, "Robustness-oriented k edge server placement," in *Proc. IEEE/ACM Int. Symp. Cluster Cloud Internet Comput.*, 2020, pp. 81–90.

[18] Q. Peng *et al.*, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Web Serv.*, 2019, pp. 91–98.

[19] Y. Liu, Q. He, D. Zheng, X. Xia, F. Chen, and B. Zhang, "Data caching optimization in the edge computing environment," *IEEE Trans. Serv. Comput.*, early access, Oct. 21, 2020, doi: 10.1109/TSC.2020.3032724.

[20] X. Xia, F. Chen, J. Grundy, M. Abdelrazek, H. Jin, and Q. He, "Constrained app data caching over edge server graphs in edge computing environment," *IEEE Trans. Serv. Comput.*, early access, Feb. 24, 2021, doi: 10.1109/TSC.2021.3062017.

[21] R. Luo, H. Jin, Q. He, S. Wu, Z. Zeng, and X. Xia, "Graph-based data deduplication in mobile edge computing environment," in *Proc. Int. Conf. Service-Oriented Comput.*, 2021, pp. 499–515.

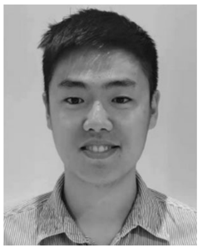
[22] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-based computing and storage offloading for data synchronization in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4272–4282, Jun. 2019.
[23] X. Xia *et al.*, "Graph-based optimal data caching in edge computing," in *Proc. Int. Conf. Service-Oriented Comput.*, 2019, pp. 477–493.
[24] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wirel. Commun.*, vol. 23, no. 1, pp. 72–79, Feb. 2016.
[25] E. S. Lee and R. J. Li, "Fuzzy multiple objective programming and compromise programming with pareto optimum," *Fuzzy Sets Syst.*, vol. 53, no. 3, pp. 275–288, 1993.
[26] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.
[27] X. Xu *et al.*, "Load-aware edge server placement for mobile edge computing in 5G networks," in *Proc. Int. Conf. Service-Oriented Comput.*, 2019, pp. 494–507.
[28] D. Z. Tootaghaj, F. Ahmed, P. Sharma, and M. Yannakakis, "Homa: An efficient topology and route management approach in SD-WAN overlays," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2351–2360.
[29] X. Zhao, Y. Zeng, H. Ding, B. Li, and Z. Yang, "Optimize the placement of edge server between workload balancing and system delay in smart city," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 6, pp. 3778–3792, 2021.
[30] L. Yuan *et al.*, "CoopEdge: A decentralized blockchain-based platform for cooperative edge computing," in *Proc. Web Conf.*, 2021, pp. 2245–2257.
[31] F. Chen, J. Zhou, X. Xia, H. Jin, and Q. He, "Optimal application deployment in mobile edge computing environment," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2020, pp. 184–192.
[32] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2022.
[33] P. Vitello, A. Capponi, C. Fiandrino, G. Cantelmo, and D. Kliazovich, "Mobility-driven and energy-efficient deployment of edge data centers in urban environments," *IEEE Trans. Sustain. Comput.*, early access, Feb. 03, 2021, doi: 10.1109/TSUSC.2021.3056621.
[34] X. Yuan, M. Sun, and W. Lou, "A dynamic deep-learning-based virtual edge node placement scheme for edge cloud systems in mobile environment," *IEEE Trans. Cloud Comput.*, early access, Feb. 18, 2020, doi: 10.1109/TCC.2020.2974948.
[35] J. Lu, J. Jiang, V. Balasubramanian, M. R. Khosravi, and X. Xu, "Deep reinforcement learning-based multi-objective edge server placement in internet of vehicles," *Comput. Commun.*, vol. 187, pp. 172–180, 2022.
[36] B. Shen, X. Xu, L. Qi, X. Zhang, and G. Srivastava, "Dynamic server placement in edge computing toward internet of vehicles," *Comput. Commun.*, vol. 178, pp. 114–123, 2021.

Ruikun Luo received the bachelor's degree from Dalian Maritime University, China, in 2018. He is currently working toward the PhD degree with the Huazhong University of Science and Technology in China. His research interests include edge computing, parallel and distributed computing, and network storage.

Hai Jin (Fellow, IEEE) received the PhD degree in computer engineering from the Huazhong University of Science and Technology, in 1994. He is a chair professor of computer science and engineering with the Huazhong University of Science and Technology (HUST) in China. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked with The University of Hong Kong between 1998 and 2000, and as a visiting scholar with the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. He is a fellow of the CCF, and a life member of the ACM. He has co-authored more than 20 books and published more than 900 research papers. His research interests include computer architecture, parallel and distributed computing, Big Data processing, data storage, and system security.

Qiang He (Senior Member, IEEE) received the first PhD degree from the Swinburne University of Technology, Australia, in 2009, and the second PhD degree in computer science and engineering from the Huazhong University of Science and Technology, China, in 2010. He is currently an associate professor with Swinburne. His research interests include service computing, software engineering, cloud computing and edge computing. For more details, please visit https://sites.google.com/site/heqiang/.

Song Wu (Member, IEEE) received the PhD degree from the Huazhong University of Science and Technology (HUST), in 2003. He is a professor of computer science with the HUST, China. He currently serves as the vice dean of the School of Computer Science and Technology and the vice head of Service Computing Technology and System Lab (SCTS) and the Cluster and Grid Computing Lab (CGCL), HUST. His current research interests include cloud resource scheduling and system virtualization.



Xiaoyu Xia received the master's degree from the University of Melbourne, Australia, in 2015. He is currently working toward the PhD degree with Deakin University. His research interests include edge computing, parallel and distributed computing, service computing, and cloud computing.