A Heuristic Algorithm for Optimal Facility Placement in Mobile Edge Networks

Jiping Jiao, Lingyu Chen*, Xuemin Hong and Jianghong Shi
Key Laboratory of Underwater Acoustic Communication and Marine Information Technology, Ministry of Education, Xiamen University, Xiamen 361005, Fujian, China
[e-mail: {jjp, chenly, xuemin.hong, shijh}@xmu.edu.cn]
* Corresponding author: Lingyu Chen

Received January 21, 2017; revised March 7, 2017; accepted March 11, 2017; published July 31, 2017

Abstract

Installing caching and computing facilities in mobile edge networks is a promising solution to cope with the challenging capacity and delay requirements imposed on future mobile communication systems. The problem of optimal facility placement in mobile edge networks has not been fully studied in the literature. This is a non-trivial problem because the mobile edge network has a unidirectional topology, making existing solutions inapplicable. This paper considers the problem of optimal placement of a fixed number of facilities in a mobile edge network with an arbitrary tree topology and an arbitrary demand distribution. A low-complexity sequential algorithm is proposed and proved to be convergent and optimal in some cases. The complexity of the algorithm is shown to be $O\left(H^2\gamma\right)$, where H is the height of the tree and γ is the number of facilities. Simulation results confirm that the proposed algorithm is effective in producing near-optimal solutions.

Keywords: Facility placement, mobile edge network, heuristic algorithm

https://doi.org/10.3837/tiis.2017.07.002 **ISSN : 1976-7277**

1. Introduction

The increasingly widespread and diverse usage of mobile data services imposes multiple challenges on future mobile communication networks. A primary challenge is exploding capacity demand, which requires the overall system capacity to increase in an order of hundreds of times [1] [2]. Another significant challenge is stringent delay requirement, which is essential to a multitude of Internet of Things (IoT) applications with time-critical control missions [3] [4]. Further challenges include dynamic spectrum management [5], high energy and spectrum efficiency [6] [7], and support for high mobility [8]. To address these challenges, it is envisioned that caching and computing facilities, which are traditionally centralized and located close to the core network (e.g., cloud computing [9]), should also be distributed and installed at the mobile edge network in the vicinity of users (e.g., fog computing [10]). Because a large portion of mobile traffic comes from content delivery services such as streaming of popular videos [1], *mobile edge caching* can effectively reduce the backhaul traffic and ease the backhaul capacity bottleneck. Similarly, *mobile edge computing* can effectively reduce communication hops between an end device and its server, thereby enabling ultra-low delay end-to-end communication services.

The mobile edge network is a hierarchical network consisting of multiple entities including base stations (BSs), switches, and/or routers. Typically, the network has a tree topology, with BSs being the bottom leaves of the tree. A computing/caching facility can be installed at different levels of the tree on either a BS, switch, or router. Placing a facility closer to the leaves of the tree will yield better performance in terms of delay, but at the cost of reduced service coverage. In practice, given a finite budget and a fixed number of caching/computing facilities, it is important to decide where to place these facilities in the network to achieve optimized overall performance. In addition, tentative placement of edge computing facilities is important for performance enhancement and recovery of overloaded mobile edge computing networks [11]. This motivates our study in this paper to propose an optimal facility placement algorithm, within the particular context of mobile edge computing and caching.

Content caching at the mobile edge network has recently become an active research field and has yielded a wealth of literature. For example, the problem of proactive caching were studied in [12] [13], the problem of collaborative caching strategy together with network coding were investigated in [14] [15], and the asymptotic analysis for content caching and delivery were provided in [16] [17]. These papers, however, typically focus on the problem of "content placement", which decides what subset of contents (depending on their local popularity) should be cached in a particular location. Our paper deals with a different problem of "facility placement", which decides where to place a given number of facilities in the network. The general problem of optimal facility placement is a classic and well-investigated topic. Mathematically, it is known as the *p*-median problem and was first formulated and studied by Haklmi in 1964 [18]. For a general graph, Haklmi *et al.* showed that the *p*-median problem is NP-hard [19]. In the context of communication networking, the facility placement problem were studied in [20] and the references therein, mostly within the context of placing data center in core networks. A brief literature review is provided in the next section. To our best knowledge, the problem of optimal facility placement in the particular context of mobile edge caching and computing has not been thoroughly investigated.

This paper studies the optimal facility placement problem in the mobile edge network with

tree topology. Our problem differs from the classic *p*-median problem in several aspects. First, the communication links in our scenario are assumed to be unidirectional, while the links in a *p*-median problem are bidirectional. Second, in our scenario, there is a pre-existing facility/server connected to the root of the tree, while the *p*-median problem does not have any pre-existing facility. These two differences essentially distinguish our problem from the *p*-median problem. The main contributions of our paper are summarized as follows.

1) Given a fixed number of facility γ, the network topology, and the distribution of user demand, we propose a heuristic algorithm to give optimized facility placement policy.
2) The proposed algorithm is proved to be convergent in an arbitrary tree topology and optimal in certain occasions. The time complexity of the algorithm is shown to be $O\left(H^2\gamma\right)$, where H is the height of the tree.
3) The performance of the proposed algorithm is validated via extensive simulations by comparing with the upper and lower bounds.

The remainder of this paper is organized as follows. Section 2 provides a brief literature review on the related work. Section 3 describes our system model and formulates an optimization problem. Section 4 presents the proposed algorithm. Section 5 analyses the properties of the algorithm. Simulation results and discussions are provided in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

Optimal facility placement is a well-defined mathematical problem and has attracted research interests for decades. Hakimi *et al.* were the first to define two closely-related problems, the *p*-median problem and the *k*-center problem [18]-[21]. The difference is that while the *p*-median problem aims to optimize the average performance, the *k*-center problem [22] aims to optimize the worst-case performance. In this paper, we consider the average performance weighted by user demand; hence our problem is more similar to the *p*-median problem. Paper [19] proved that the *p*-median problem is NP-hard for a general network topology and gave an optimal algorithm with a time complexity of $O\left(n^2p^2\right)$ to solve the *original* *p*-median problem on tree graphs. In [23], an algorithm was proposed with time complexity $O\left(n^2p\right)$ for tree graphs. To solve the general *p*-median problem, paper [24] proposed an algorithm to exploit the network structure, paper [25] proposed a heuristic algorithm based on dynamic programming theory, and paper [26] offered a three-layer gamma heuristic algorithm based on the concept of Heuristic Concentration (HC).

There is also a wealth of literature on variations of the facility placement problem in the context of telecommunications. Some early work in the 1990s considered the placement of web proxies in the Internet [27]. Later studies such as [28] and [29] addressed some particular facility location problems, but the demand distribution of users was not taken into account. Paper [28] proposed several intuitive strategies for the *p*-median problem: greedy algorithm, random algorithm, hop-spot algorithm, and super-optimal algorithm. Paper [27] leveraged dynamic programming theory to design an optimal solution for a linear array network. Paper [30] considered the user demand distribution and proposed a dynamic distributed algorithm with a time complexity of $O\left(n^2\right)$. Paper [31] addressed the optimal placement problem without/with constraint on the number of replicas (i.e., facilities) over tree networks and proposed two algorithms, the AGGregate Access (AGGA) and Weighted POPularity (WPOP)

algorithm. Unlike our study, [29]-[31] investigated the placement problem under a condition that the number of facilities is unrestricted and can be treated as a decision variable. In contrast, our study restricts the number of facilities as a given condition.

Recently, a similar problem of "content placement" in mobile edge networking has attracted significant research interest. The content placement problem considers what pieces of content should be cached in a facility, taken into account multiple factors such as content popularity and caching space. Paper [32] considered the content placement problem over a heterogeneous cellular network with cache-enabled femto-base stations, where a greedy algorithm is proposed for coded caching. Paper [33] studied the problem for a symmetric tree network by relaxing the integral decision variables as continuous ones. Papers [34] and [35] investigated the problem for Data Grid Systems (DGS), where the costs of reading and writing content are jointly considered. However, unlike the problem addressed in this paper, the content placement problem is essentially a different mathematical problem known as the Knapsack problem [36] [37].

In summary, although there is a wealth of literature addressing different variations of the facility placement problem, to our best knowledge, the specific problem of facility placement in mobile edge networks has not be addressed. In mobile edge networks, data flows are either aggregated upwards (uplink) or disseminated downwards (downlink). As a result, a user request cannot take a detour in the tree to find a serving facility, but can only go upward in the tree hierarchy until a facility is found. This results in a uni-directional topology instead of a bi-directional one and makes our problem essentially different from the existing literature.

3. Problem Formulation

As shown in **Fig. 1**, the communication scenario studied in this paper includes a tree-like communication network and a pre-existing facility (i.e., root server) connected to the root of the tree. The vertexes and edges of the tree stand for switches/routers and communication links in the mobile edge network, respectively. The leaves of the tree represent BSs, each of which has a random user demand indicated by a positive number. Apart from the root server, more facilities can be installed to some vertexes of the tree. Because communication links are assumed to be unidirectional, each user demand should go upward in the tree hierarchy until it finds a serving facility. The cost of serving a user demand is measured by the number of hops between the user and the serving facility. Such a cost measure can be naturally interpreted as either a reduced delay or a reduced backhaul traffic. The optimal facility placement problem is formulated as: given an arbitrary tree topology, an arbitrary user demand, and a fixed number of facilities, how to place these facilities in the vertexes of the tree to minimize the total cost of serving the user demand?

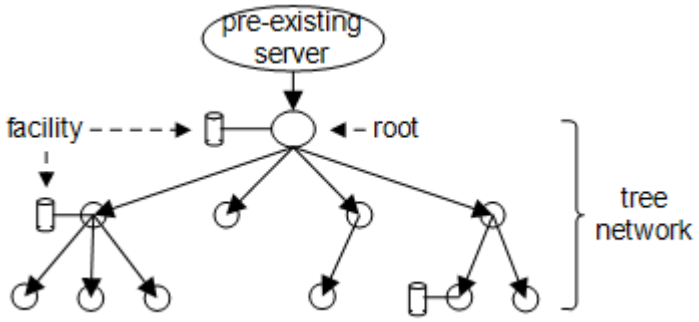

Fig. 1. Scenario of facility placement in a mobile edge network.

3.1 Symbol Specifications

Let $\boldsymbol{T}=(\boldsymbol{V},\boldsymbol{E})$ denote a tree, where $\boldsymbol{V}$ is the set of vertexes and $\boldsymbol{E}$ is the set of edges of the tree. Throughout the paper, we use v_i to denote a vertex in $\boldsymbol{V}$ and distinguish vertexes by the subscript i. Let

$$\boldsymbol{p}_{\to\text{root}}[v_i] := v_i \to,\cdots,\to v_{\text{root}}$$

be the directed path from v_i to *the root* v_{root}. Let l_i be the ***level*** of v_i, the number of the vertexes on $\boldsymbol{p}_{\to\text{root}}[v_i]$, and let

$$H := \max_{v_i\in\mathcal{T}} l_i$$

be the height of $\boldsymbol{T}$.

For two adjacent vertexes v_1 and v_2, we say that v_1 is the *parent* of v_2 or v_2 is a *child* of v_1 if v_1 is closer to v_{root} than v_2. Let $v_{i,j}$, $j=1,2,\cdots$, denote a child of v_i. Then

$$\boldsymbol{V}_{\text{child}}[v_i] := \{v_{i,1},\cdots,v_{i,j},\cdots\}$$

is the set of these children. We say that two vertexes v_1 and v_2 are directly correlated if there is a directed path to v_{root} that contains both vertexes. Moreover, v_1 is said to be a predecessor of v_2, and v_2 to be a descendant of v_1, if v_1 is closer to v_{root} than v_2. Let $\boldsymbol{T}[v_i]$ be the sub-tree rooted at v_i, a sub-tree that only contains v_i and all of its descendants. Let x_i be the facility state of v_i. We have $x_i=1$ if v_i is installed with a serving facility and $x_i=0$ otherwise. The facility placement policy $\mathbf{x}$ is a vector consisting of all x_i

$$\mathbf{x}=\left[x_1,x_2,\cdots,x_{N_{\text{vertex}}\{\boldsymbol{T}\}}\right]$$

where $N_{\text{vertex}}\{\boldsymbol{T}\}$ is the number of the vertexes in $\boldsymbol{T}$.

Let d_i be the demand of v_i, the total demand that can be served by v_i once v_i becomes a serving vertex. If v_i is a leaf, d_i is the demand that v_i possesses; otherwise d_i is the accumulated un-served demand from its children nodes, i.e.,

$$d_i = \sum_{v_{ij}\in\boldsymbol{V}_{\text{child}}[v_i]} \mathbb{I}\{x_{ij}=0\}\, d_{ij} \tag{3.1}$$

where $\mathbb{I}\{\ \}$ is a 0-1 indicator function.

Take **Fig. 2** for example. Here we have $d_{10}=d_1+d_2+d_3$, but $d_{11}=d_5+d_6$ because the demand that v_4 possesses will be served by v_4. Similarly, we can get $d_{12}=d_8+d_9$, $d_{13}=d_{10}+d_{11}$, and $d_{13}=d_1+d_2+d_3+d_5+d_6$.

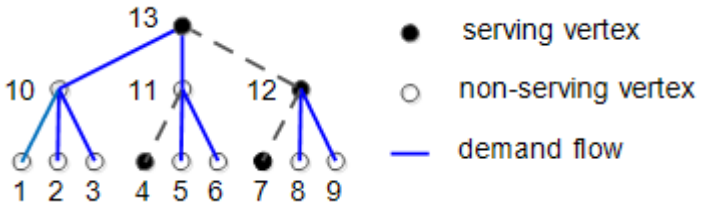

Fig. 2. Example of how to calculate the demand of a vertex.

3.2. Facility Placement Problem

Once a new facility is installed on a vertex, it will result in a non-negative reduction on the cost (i.e., hops). The number of reduced hops for installing a facility on v_i is called the *facility gain* and denoted by g_i. We have

$$g_i = l_i d_i \tag{3.2}$$

where l_i is the level of v_i and d_i is the demand of v_i. Let γ be the number of facilities to be installed in the network. Given a placement policy, the facility gain is a function of γ given by

$$g_\Gamma(\gamma) := \sum_{v_i \in \boldsymbol{T}} \mathbb{I}\{x_{ij} = 1\} g_i = \sum_{v_i \in \boldsymbol{T}} x_i g_i . \tag{3.3}$$

It follows that the cache placement problem can be formulated as a 0-1 integer program as

$$\max_{\mathbf{x}} g_\Gamma(\gamma)$$
$$s.t. \begin{cases} \sum_{v_i \in \boldsymbol{T}} x_i = \gamma . \\ x_i = \{0,1\} \end{cases} \tag{3.4}$$

It is easy to see that the number of useful facilities is not greater than $N_{\text{leaf}}\{\boldsymbol{T}\}$, the number of the leaves in $\boldsymbol{T}$. This is because if all the leaves are installed with a facility, all users can access service from the leaf vertex, so that all other vertexes have a facility gain of 0. Consequently, the value space of γ is from 0 to $N_{\text{leaf}}\{\boldsymbol{T}\}$, i.e.,

$$\gamma = 0,1,\cdots,N_{\text{leaf}}\{\boldsymbol{T}\}.$$

4. A Sequential Facility Placement Algorithm

4.1 Concept and definition of "balanced demand profile"

Let $\boldsymbol{D}[v_i]$ be the *demand profile* of v_i, which is a vector that consists of the demands of its children nodes. The demand profile of a leaf is a null vector, and the demand profile of an internal vertex v_i is given by the product items $\mathbb{I}\{x_{ij} = 0\} d_{ij}$ on the right-hand side of Eqn.

(3.1). A useful indicator of measuring whether a vertex is worth deploying a facility is the *balance degree* of the vertex's demand profile. A demand profile is called balanced when the demands are spread on multiple children nodes rather than concentrating on a single child node. Because if the demands are concentrated, it is more desirable to deploy a facility directly on the child node to save more hops. In what follows, we will present a heuristic algorithm based on such a concept of balanced demand profile.

Let ϕ_i be a binary indicator of whether $\boldsymbol{D}[v_i]$ is balanced. We set $\phi_i = 1$ if $\boldsymbol{D}[v_i]$ is considered balanced and $\phi_i = 0$ if $\boldsymbol{D}[v_i]$ is not considered balanced. To be as general as possible, here we do not give a specific formula for ϕ_i, but consider ϕ_i to be a general function (with a binary output) that fulfills the following three conditions:

1) $\phi_i = 0$ if $d_i = 0$;

2) $\phi_i = 1$ if v_i is a leaf;

3) $\phi_i = 0$ if $\boldsymbol{D}[v_i]$ has only one positive element.

The first condition implies that it is not worthwhile to place a facility on a vertex if the vertex does not serve any demand. The second condition implies that no other vertex is superior to a leaf node when we consider only the demand from the leaf. The third condition implies that if a father node has a single child node with a positive demand, then it is better to place a facility at the child node instead of the father vertex.

4.2 Procedure of a local iteration

The facility placement algorithm we proposed in this paper is a sequential one. Given an arbitrary tree and demand profile, we consider adding one facility into the tree at one time until we reach the pre-defined facility number. Adding a new facility will result in a chain of iteration to search for the optimal local placement policy, which will ultimately result in a non-negative increment on the facility gain. Let ρ_i be a lower bound on the increment of the facility gain when a new facility is added into a tree $\boldsymbol{T}[v_i]$ rooted at v_i. If v_i is a leaf, we have

$$\rho_i = \begin{cases} g_i & x_i = 0 \\ 0 & x_i = 1 \end{cases} \tag{4.1}$$

otherwise we have

$$\rho_i = \begin{cases} \tilde{\rho}_i & x_i = 0 \text{ and } \phi_i = 0 \\ \max\{\tilde{\rho}_i, g_i\} & x_i = 0 \text{ and } \phi_i = 1 \\ \tilde{\rho}_i - d_{\sim i} l_i & x_i = 1 \text{ and } g_{\sim i} = \tilde{\rho}_i \\ \tilde{\rho}_i & x_i = 1 \text{ and } g_{\sim i} \neq \tilde{\rho}_i \end{cases} \tag{4.2}$$

where $\tilde{\rho}_i$ is the lower bound of the incremental facility gain when the new facility is not allowed to be installed on vertex v_i. The difference between ρ_i and $\tilde{\rho}_i$ is that ρ_i includes the case where a new facility is installed on vertex v_i. Mathematically, we have

$$\tilde{\rho}_i = \max_{v_{i,j} \in \boldsymbol{V}_{\text{child}}[v_i]} \rho_{i,j} . \tag{4.3}$$

We further define $v_{\sim i} \coloneqq \arg \max_{v_{i,j} \in \boldsymbol{V}_{\text{child}}[v_i]} \rho_{i,j}$ to indicate the node/vertex on which the new facility should be installed to achieve the lower bound $\tilde{\rho}_i$.

Fig. 3 gives an simple example on how to calculate $\tilde{\rho}_i$. Let us first consider the case of $x_{13} = 0$, i.e., the root vertex v_{13} of the tree is not installed with a facility. By Eqn. (4.3) we have

$$\tilde{\rho}_{13} = \max\{\rho_{10}, \rho_{11}, \rho_{12}\}.$$

- If $\phi_{13} = 0$, which means that the new facility will not be installed on v_{13}, we have $\rho_{13} = \tilde{\rho}_{13}$ according to the definition in (4.2).
- If $\phi_{13} = 1$, which means that compared with the case of $\phi_{13} = 0$, there is a new option to install the new facility on v_{13}, hence the facility gain is maximum value of the two cases, i.e., $\rho_{13} = \max\{\tilde{\rho}_{13}, g_{13}\}$.

Now we move on to consider the case of $x_{13} = 1$, i.e., the root vertex v_{13} is already installed with a facility.

- Let us consider the case where the new facility is installed on v_2, to which v_{13} is a serving vertex. Installing a facility on v_2 will reduce the facility gain lower bound of $\boldsymbol{T}[v_{13}]$ by $d_2 l_{13}$.
- Let us consider another case where the new facility is installed on v_9, to which v_{13} is not a serving vertex. Because the demand of v_9 is hidden from v_{13} by v_{12}, installing a facility on v_9 will not caused any change on the facility gain lower bound of $\boldsymbol{T}[v_{13}]$.

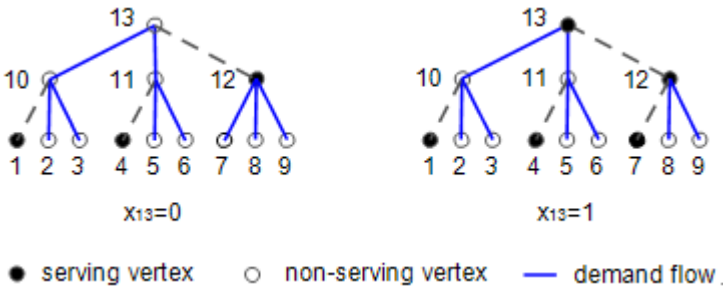

Fig. 3. Example of how to calculate ρ_i according to Eqn. (4.2).

4.3 The proposed sequential algorithm

We define a state vector $\mathbf{s}_i \coloneqq [d_i, g_i, \rho_i, \phi_i]$ for each vertex v_i. The state vector is recorded for each vertex and changes in each iteration. The pseudo-code of the proposed sequential algorithm is given below.

Algorithm I: Heuristic Algorithm for Optimal Facility Placement in Mobile Edge Networks

1 //--------*Initialization*--------
2 $\gamma \leftarrow 0$
3 **For all** $v_i \in \boldsymbol{T}$ **do**
4 **Update**($\mathbf{s}_i$)
5 $x_i \leftarrow 0$
6 //--------*Body*--------
7 **while** $\rho_{\text{root}} > 0$ **do**
8 Stk_1.**push**(v_{root})
9 **while** Stk_1.**IsNonEmpty**() **do**
10 $v_i \leftarrow \text{Stk}_1$.**pop**()
11 $v_j \leftarrow \rho_i^{-1}$
12 $x_j \leftarrow 1$
13 Stk_2.**push**($\boldsymbol{p}_{\rightarrow \text{root}}\left[v_{j,\text{ parent}}\right]$)
14 while Stk_2.**IsNonEmpty**() do
15 $v_k \leftarrow \text{Stk}_2$.**pop**()
16 **Update**($\mathbf{s}_i$)
17 **if** $x_k = 1 \wedge \rho_k > g_k$ **then**
18 $x_k \leftarrow 0$
19 Stk_1.**push**(x_k)
20 **break**
21 $\gamma \leftarrow \gamma + 1$
22 **Output**: $\left(\gamma, g_\Gamma(\gamma)\right)$

Explanation notes on Algorithm I:

1) The abbreviation "Stk" stands for "stack", a LIFO (last in, first out) data buffer. The data entries of both Stk_1 and Stk_2 are vertexes.
2) Line 13: The statement Stk_2.**push**($\boldsymbol{p}_{\rightarrow \text{root}}\left[v_j\right]$) means pushing vertexes on $\boldsymbol{p}_{\rightarrow \text{root}}\left[v_j\right]$ to Stk_2 following a top-down order from v_{root} to v_j.
3) Lines 8 and 19: The vertex in Stk_1 is used as an entrance of adding a facility, i.e., the added facility lies in the sub-tree rooted at the entrance.
4) Line 11: The symbol ρ_i^{-1} represents the vertex on which a facility will be installed to obtain at least an incremental gain of ρ_i.
5) Lines 17-20: These lines are used to decide whether or not an installed facility should continue to be located in the tested vertex. If not, the vertex is called an *unsuitable vertex*

for placement and the facility will be removed from the unsuitable vertex (Line 18). At the same time, the removed facility will be reinstalled in a certain vertex of the sub-tree rooted at the unsuitable vertex for compensation (Line 19), and the vertex is called a *moderate vertex for placement*. Such a process is called a *remove-and-reinstall iteration*.

5. Properties of the Proposed Algorithm

In this section, we present some proofs on the properties of the proposed algorithm.

5.1 Major Properties

Proposition I. *If the network is finite, Algorithm I is convergent.*

Proof: We prove this proposition by induction. If $\gamma = 1$, Lines 18-20 will not be executed, so there is no remove-and-reinstall iteration. Let us now suppose that the proposition holds when $\gamma = k$ and consider γ increases from k to $k+1$. Let

$$s_{k+1} := \sum_{v_i \in \boldsymbol{T}} x_i l_i$$

be the sum of the levels of the $k+1$ serving vertexes. There may be several remove-and-reinstall iterations. However, since each reinstalled facility is located in the sub-tree rooted at the previous unsuitable vertex for placement, a remove-and-reinstall iteration will lead to an increase in s_{k+1}. On the other hand, s_{k+1} must be finite for a finite network. Therefore, the number of remove-and-reinstall iterations is also finite and the algorithm is convergent.

■

Proposition II. *By Algorithm I, $g_\Gamma(\gamma)$ is monotonically increasing with respect to γ.*

Proof: We need to show that $g_\Gamma(k) \le g_\Gamma(k+1)$ for a non-negative integer k. Recall that the variation of γ from k to $k+1$ involves two kinds of processes, the *direct installation of a new facility* (Line 8) and a series of remove-and-reinstall iterations (Lines 18-20). The new facility can ensure that the facility gain has a positive change and Line 17 can also guarantee that an iteration will increase the facility gain.

■

Lemma I. *For all $\boldsymbol{T}[v_i]$, $\gamma\{\boldsymbol{T}[v_i]\} \le N_{\text{leaf}}\{\boldsymbol{T}[v_i]\}$, where $\gamma\{\boldsymbol{T}[v_i]\}$ is the number of the facilities in $\boldsymbol{T}[v_i]$ and $N_{\text{leaf}}\{\boldsymbol{T}[v_i]\}$ is the number of the leaves in $\boldsymbol{T}[v_i]$.*

Proof: We prove the lemma through three steps.

First, let A be the set of the facilities in $\boldsymbol{T}[v_i]$ that is closest to a certain leaf, i.e.,

$$A = \left\{ v_j \middle| \begin{array}{l} (1)\, x_j = 1. (2) \exists v_k \in \boldsymbol{V}_{\text{leaf}}\{\boldsymbol{T}[v_i]\} \text{ such that} \\ v_j \text{ is closer to } v_k \text{ than any other serving vertex.} \end{array} \right\}$$

where $\boldsymbol{V}_{\text{leaf}}\{\boldsymbol{T}[v_i]\}$ is the set of the leaves in $\boldsymbol{T}[v_i]$. Let $\gamma_A := |A|$. By the set A the leaves

can be partitioned into γ_A+1 disjoint subsets, i.e.,

$$B_j=\left\{ v_k \left| \begin{array}{l}(1)\, v_k \in \boldsymbol{V}_{\text{leaf}}\left\{\boldsymbol{T}\left[v_i\right]\right\}.(2)\,\tilde{v}_j \text{ is the j-th element of } A.\\ (3)\,\tilde{v}_j \text{ is } v_k \text{ or a predecessor of } v_k.\end{array} \right.\right\}$$

$$j=1,\cdots,\gamma_A,$$

and

$$B_{\gamma_A+1}=\boldsymbol{V}_{\text{leaf}}\left\{\boldsymbol{T}\left[v_i\right]\right\}-\bigcup_{j=1}^{\gamma_A}B_j\ .$$

Second, if $\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}>\gamma_A$, there are extra facilities that are not contained in A. At the same time, they have a facility gain 0 considering B_j. Such a case is impossible according to Lines 11 and 17. Therefore, $\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}=\gamma_A$.

Third, B_j must be non-empty and disjoint by definition, so γ_A is not greater than $N_{\text{leaf}}\left\{\boldsymbol{T}\left[v_i\right]\right\}$. Taking into account all the above three steps, we have

$$\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}=\gamma_A\le N_{\text{leaf}}\left\{\boldsymbol{T}\left[v_i\right]\right\}.$$

■

Proposition III. *When* $\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}=N_{\text{leaf}}\left\{\boldsymbol{T}\left[v_i\right]\right\}$, *all the leaves in* $\boldsymbol{T}\left[v_i\right]$ *are installed with a facility and the facility placement policy for* $\boldsymbol{T}\left[v_i\right]$ *is optimal. If* $\gamma=N_{\text{leaf}}\left\{\boldsymbol{T}\right\}$, *the facility placement policy for* $\boldsymbol{T}$ *is optimal.*

Proof:

If $\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}=\gamma_A=N_{\text{leaf}}\left\{\boldsymbol{T}\left[v_i\right]\right\}$, each B_j can only contain one vertex due to the non-empty and disjoint property. According to Line 11, all facilities will be installed on the leaves to yield an optimal placement policy.

■

5.2 Time Complexity

When γ increases from k to $k+1$, we define the *usage frequency* of a stack, both for Stk_1 and Stk_2, as the number of the vertexes pushed into the stack during the process. Once the usage frequency of Stk_1 is determined, the usage frequency of Stk_2 is never greater than H multiplied by the usage frequency of Stk_1. The complexity of the proposed algorithm is bounded by the usage frequencies of Stk_1 and Stk_2. In what follows, we will first show that the usage frequencies of Stk_1 and Stk_2 are $O(H)$ and $O(H^2)$, respectively.

Lemma II. When $\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}$ increases by one, $\gamma\left\{\boldsymbol{T}\left[v_j\right]\right\}$ will never decrease for any $v_j\in\boldsymbol{T}\left[v_i\right]$.

Proof: This Lemma shows that the number of pre-existing facilities in any sub-tree will not reduce if the total number of facilities increases. Because a direct adding will never reduce the number of facilities in any sub-tree, our focus is on the remove-and-reinstall iteration. Considering the reinstalled cache vertex as a new facility, it will not lead to a reduction of facilities at least for the sub-tree rooted at the vertexes except for the unsuitable vertex used for placement and its predecessors. On the other hand, since the reinstalled facility must be located in the sub-tree rooted at the unsuitable vertex, the total number of facilities in the sub-tree rooted at the unsuitable vertex or one of its predecessors will also remain unchanged.

■

Lemma III. When $\gamma\left\{\boldsymbol{T}\left[v_i\right]\right\}$ increases by one, ρ_j will never increase for any $v_j \in \boldsymbol{T}\left[v_i\right]$.

Proof: By definition, the process of installing or reinstalling a facility will not lead to an increase in ρ_j for any $v_j \in \boldsymbol{T}\left[v_i\right]$, hence we focus on the removing process in a remove-and-reinstall iteration. First, the following symbols are introduced:

v_1: vertex whose facility is removed

v_2: vertex where the reinstallation is implemented

$v_{1,k}$: child of v_1

$v_{1,1}$: specific child vertex that is just v_2 or direct correlated to v_2.

Before the re-adding, we have $\rho_1^{-1} = v_2$ and $\rho_1 = g_2$. Moreover, we have $\rho_{1,k} \le g_2, \forall k = 2,3,\cdots$, and $g_1 \le g_2$. Once v_2 becomes a serving vertex, we have

$$\rho_1 \le \max\left\{g_1, \rho_{1,k}, j = 1,2,\cdots\right\}$$

and $\rho_{1,1} \le g_2$. Therefore,

$$\rho_1 \le \max\left\{g_1, \rho_{1,k}, j = 1,2,\cdots\right\} \le g_2.$$

■

Proposition IV. *The time complexity of Algorithm I is* $O\left(H^2\gamma\right)$.

Proof: First, let us consider the case that γ varies from k to $k+1$. We use $v^{(1)}$ to denote the vertex with a new facility, use $\tilde{v}^{(p)}$, $p = 1,2,\cdots$, to denote the vertex with the *p*-th removed facility, and use $v^{(p+1)}$ to denote the vertex with the *p*-th reinstalled facility. When $v^{(p)}$ is installed or reinstalled, only the facility on $\boldsymbol{p}_{\to\text{root}}\left[v_{\text{parent}}^{(p)}\right]$ that is closest to $v^{(p)}$, denoted by $\hat{v}^{(p)}$, is likely to be removed, where $v_{\text{parent}}^{(p)}$ is the parent of $v^{(p)}$. In other words, an installation or reinstallation causes at most one iteration. A detailed reasoning is given below..

1) If the facility in $\hat{v}^{(p)}$ is not removed, the other facilities on $\boldsymbol{p}_{\to\text{root}}\left[v_{\text{parent}}^{(p)}\right]$ have an unchanged demand and a non-increasing ρ by Lemma IV. As a result, the condition on Line 17 will not be satisfied and these facilities will not be removed.

2) If the facility in $\hat{v}^{(p)}$ is removed, the other facilities on $\boldsymbol{p}_{\to\text{root}}\left[v_{\text{parent}}^{(p)}\right]$ have a larger or at least an unchanged demand and a non-increasing ρ. According to a similar analysis in case 1), they will not be removed either.

If the facility in $\hat{v}^{(p)}$ is removed, $\hat{v}^{(p)}$ will become $\tilde{v}^{(p)}$ and $v^{(p+1)}$ will be located in the sub-tree rooted at $\tilde{v}_{\text{child}}^{(p)}$, where $\tilde{v}_{\text{child}}^{(p)}$ stands for a certain child of $\tilde{v}^{(p)}$. At the same time, $\boldsymbol{p}_{\to\text{root}}\left[v_{\text{parent}}^{(p+1)}\right]$ can be partitioned into two parts:

$$v_{\text{parent}}^{(p+1)} \to \cdots \to \tilde{v}_{\text{child}}^{(p)}$$

and

$$\boldsymbol{p}_{\to\text{root}}\left[\tilde{v}^{(p)}\right].$$

Following the above analysis, $\boldsymbol{p}_{\to\text{root}}\left[\tilde{v}^{(p)}\right]$ cannot contain the facilities that can be removed and we thus call it a false tested path. On the other hand, $v_{\text{parent}}^{(p+1)} \to \cdots \to \tilde{v}_{\text{child}}^{(p)}$ contains at most one facility that can be removed. If there is such a facility, the false tested path will be expanded into $\boldsymbol{p}_{\to\text{root}}\left[\tilde{v}^{(p+1)}\right]$ where $\tilde{v}^{(p+1)}$ is the (p+1)-th removed facility that is located on $v_{\text{parent}}^{(p+1)} \to \cdots \to \tilde{v}_{\text{child}}^{(p)}$. That is, each iteration will give rise to a longer false tested path. When the length of the false tested path is $H-1$, there will be no facility that can be removed and the algorithm will terminate. Therefore, the number of iterations is at most $H-1$. Considering the direct adding of the new facility, the usage frequency of Stk_1 is upper bounded by H. Moreover, considering the number of facilities γ, we can obtain that the time complexity of Algorithm I is $O\left(H^2\gamma\right)$.

■

For comparison purpose, we note that the computing complexity of a brutal-force enumeration is an exponential function of γ given by $O\left(N_{\text{vertex}}^{\gamma}\left\{\boldsymbol{T}\right\}\right)$ when $\gamma \le \frac{1}{2}N_{\text{vertex}}\left\{\boldsymbol{T}\right\}$, where $N_{\text{vertex}}\left\{\boldsymbol{T}\right\}$ is the total number of vertexes in the tree. To our best knowledge, the most efficient heuristic algorithms with polynomial complexity proposed for a similar problem is the one in [30], which has a time complexity of $O\left(N^2\right)$, where N is the number of nodes. Our algorithm has a complexity of $O\left(H^2\gamma\right)$, where H is the height of the tree. Obviously, we have $H < N$, hence our algorithm yields a better performance in terms of time complexity, especially when we consider fat trees. For example, considering a *M*-tree (i.e., a tree where each father node has *M* children), H is approximately proportional to $\log_M\left(N\right)$.

6. Simulation Results and Discussions

This section evaluates the performance of the proposed algorithm by simulations. First, we should give a specific formula for ϕ_i, which measures whether a demand profile is balanced or not. Let $\max d_{i,1}$ and $\max d_{i,2}$ be the two largest elements in the demand profile $\boldsymbol{D}[v_i]$, i.e.,

$$\begin{cases} \max d_{i,1} := \max\limits_{d_{i,j} \in \boldsymbol{D}[v_i]} d_{i,j} \\ \max d_{i,2} := \max\limits_{d_{i,j} \in \boldsymbol{D}[v_i] - \{\max d_{i,1}\}} d_{i,j} \end{cases}.$$

In the special case where $\boldsymbol{D}[v_i]$ contains only one element, we have $\max d_{i,2} = 0$. If v_i is a leaf, both $\max d_{i,1}$ and $\max d_{i,2}$ are set to be 0. Let us further define $\overline{d}_i$ to be the average value taken over elements in $\boldsymbol{D}[v_i]$. If v_i is a leaf, we have $\overline{d}_i = d_i$; otherwise it is given by

$$\overline{d}_i = \frac{1}{N_{\text{child}}[v_i]} \sum_{d_{i,j} \in \boldsymbol{D}[v_i]} d_{i,j}.$$

Let $\Delta d_i := \max d_{i,1} - \max d_{i,2}$, we can give a practical formula for ϕ_i as

$$\phi_i = \begin{cases} 0, & \Delta d_i > \alpha \overline{d}_i \\ 1, & \Delta d_i \le \alpha \overline{d} \end{cases}, \tag{5.1}$$

where α is an extra parameter that takes non-negative real values. Essentially speaking, a smaller value of α means that a demand profile is less likely to be considered as balanced. When $\alpha \to 0$, for a non-leaf vertex, only the demand profiles that have at least two *identical* positive elements are considered balanced. When $\alpha \to \infty$, all demand profiles that have at least two positive elements are considered balanced. For convenience, we use $g_\Gamma(\gamma;\alpha)$ to denote the facility gain from Eqn. (6.1).

To systematically evaluate the performance of the proposed algorithm, we set up four different simulation scenarios, the parameters of which are summarized in **Table 1**. For comparison purpose, we also consider a special case where all facilities are installed only on the leaf nodes. This special case yields a performance *lower bound*.

Table 1. Parameter settings in different simulation scenarios.

	Tree height H	Node branch M	Probability of having child nodes	Demand distribution on tree leaves	Vertex availability
Scenario I	3	2	A deterministic tree in Fig. 4		All available
Scenario II	4	5	0.835/0.625	Uniform(1~50)	All available
Scenario III	6	7	0.835/0.625	Uniform(1~50)	All available
Scenario IV	4	5	0.835	Uniform(1~50)	Selective

The first scenario represents a small and deterministic tree topology, in which the *optimal facility gain* can be obtained by brutal-force enumeration. With varying values of parameter α, the facility gains achieved by the proposed algorithm is compared with the optimal gain in

Table 2. We can see that in most cases, the proposed algorithm is able to achieve the exact or approximate results compared with the optimal gain. Moreover, the performance of the algorithm is not very sensitive to the value of α. For comparison, we also adopt and apply the algorithm in [30] to our model and evaluate its performance by continuously changing the storage cost. It is observed that both heuristic algorithms can obtain near-optimal results in this simple scenario, while our algorithm slightly outperforms the algorithm in [30]. We further note that our algorithm is more advantageous in large-scale tree networks because it has a less complexity.

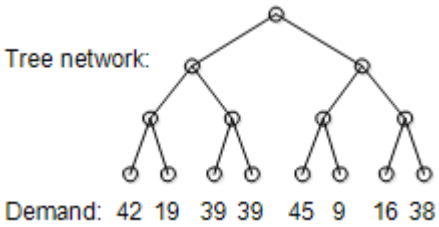

Fig. 4. A deterministic tree network and demand distribution in Scenario I.

Table 2. Facility gains determined by different methods in Scenario I.

γ	LOWER BOUND	$\alpha=1$	$\alpha=20$	OPTIMAL GAIN	[30]
1	180	278	278	278	247
2	348	494	494	494	494
3	504	620	620	633	633
4	660	759	759	759	741
5	812	820	820	837	819
6	888	898	898	898	880
7	952	952	952	952	938
8	988	988	988	988	988

To further validate the performance of our algorithm, we will subsequently generate random tree topology with random demands. A random tree is generated by a classic random branching process, where the probability for a new vertex not being a leaf (a leaf has no further branching) is denotes as $p_{non-leaf}$. The maximum tree height H and maximum number of branch M are pre-defined parameters. The demand on each leaf is a random variable following a uniform distribution from 1 to 50.

Scenario II considers a medium-size tree. First, setting $p_{non-leaf}$ =0.835, we generate a random tree snapshot with 93 leaves and a total user demand of 2423. **Fig. 5** plots the facility gain as a function of γ (number of facilities) with varying α. As a testament to Proposition 2, we observe that $g_\Gamma(\gamma;\alpha)$ is a strictly increasing function of γ, indicating the effectiveness of the proposed algorithm. The facility gain rises quickly when γ is small, suggesting that increasing the number of facilities is effective initially, but has diminishing returns. As **Fig. 5** only shows the performance in a snapshot, it is desirable to see the statistically averaged performance of our algorithm. To this end, **Fig. 6** sets $p_{non-leaf}$ =0.635 and generate 100 snapshots of random tree topology. For each snapshot, the x-axis is normalized to the maximum number of facilities (i.e., number of leaves in the random tree) and the y-axis is

normalized to the maximum facility gain. Such a normalization procedure allows us to evaluate the average performance in different tree topologies. It can be observed that our algorithm consistently outperforms the lower bound. If 10 percent of the vertexes can be installed with facilities, we can achieve about 50 percent of the maximum facility gain. Moreover, the performance is insensitive to the values of α when the value of α is not near zero. This indicates that when we have a relative relaxed regulation on whether a vertex is balanced, the proposed algorithm will yield close-to-optimal results.

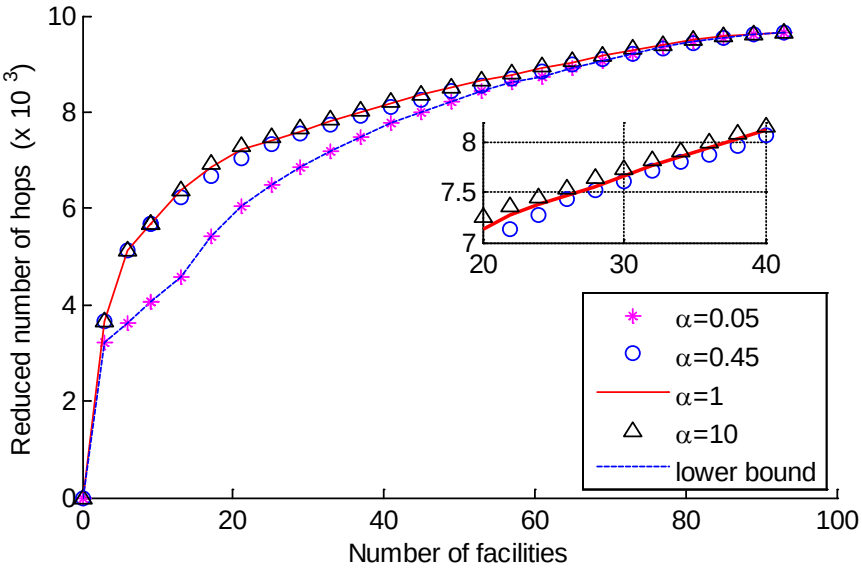

Fig. 5. Facility gain as a function of the number of facilities in Scenario II ($H = 4$, $M = 5$, $p_{\text{non-leaf}} = 0.835$).

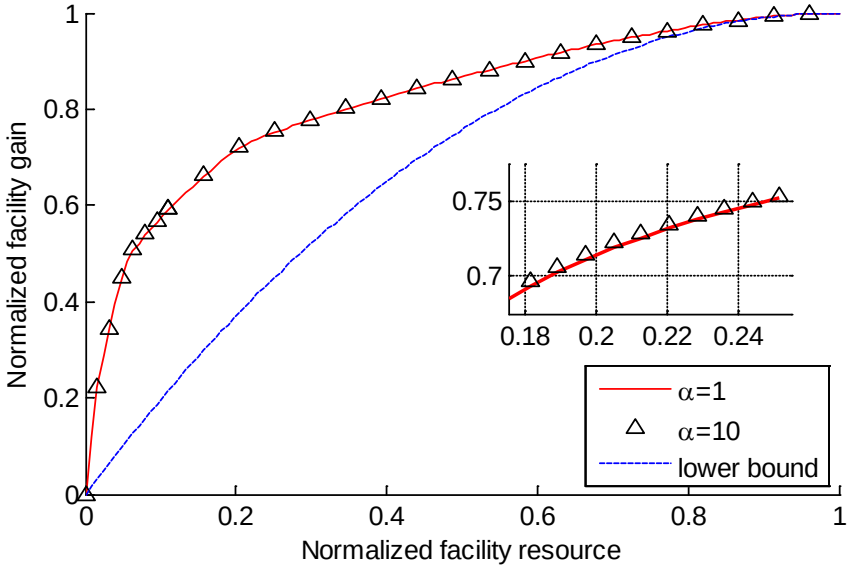

Fig. 6. Normalized facility gain as a function of normalized facility resource in Scenario II ($H = 4$,

$M = 5,\ p_{\text{non-leaf}} = 0.625$).

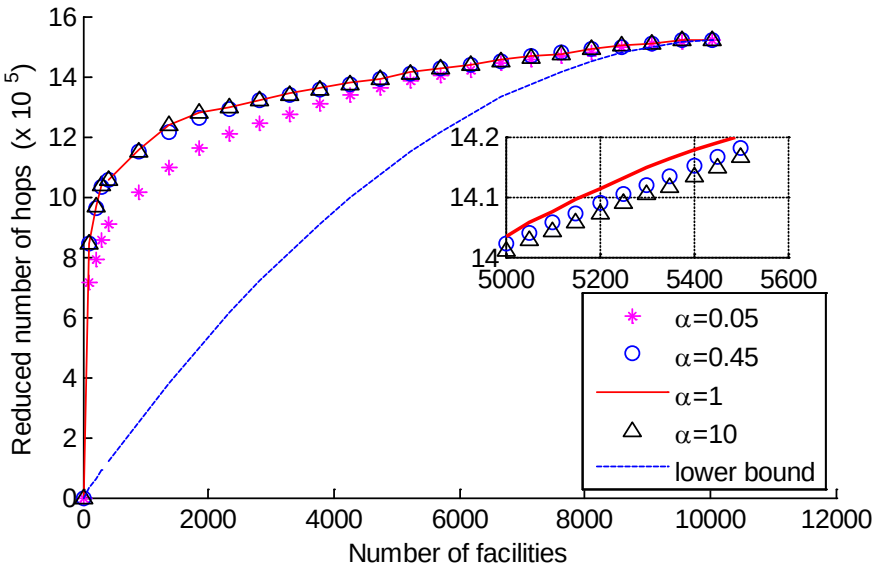

Fig. 7. Facility gain as a function of the number of facilities in Scenario III ($H = 6$, $M = 7$, $p_{\text{non-leaf}} = 0.835$).

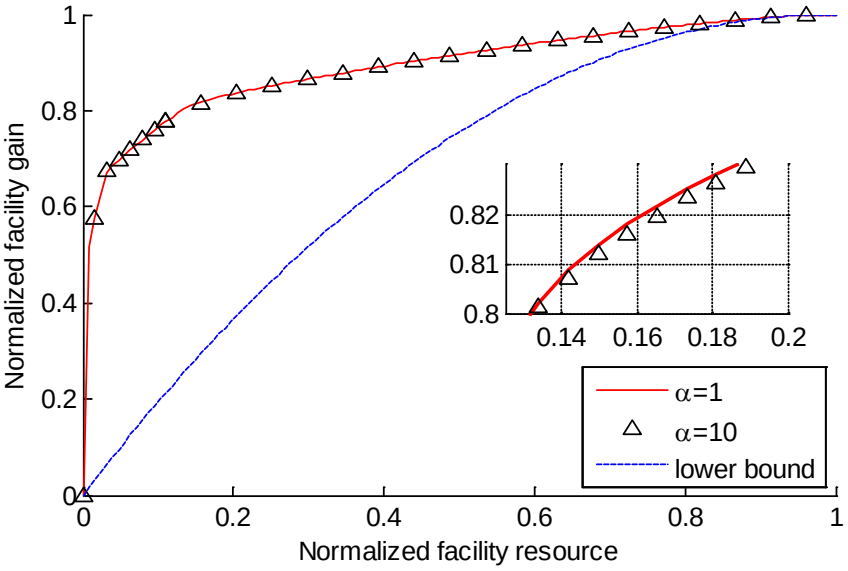

Fig. 8. Normalized facility gain as a function of normalized facility resource in Scenario III ($H = 6$, $M = 7$, $p_{\text{non-leaf}} = 0.625$).

Scenario III simulates a large-size tree topology. First, we generate a snapshot with 10,507 leaves and a total demand of 255,857. The corresponding facility gain for the snapshot

is shown in **Fig. 7.** We see that the overall tendencies in **Fig. 7** are similar to **Fig. 5** and **Fig. 6**. However, there are some new differences. First, the facility gain takes a steeper rising slope when γ is small. Second, our algorithm shows a more significant advantage over the lower bound, even for the case of $\alpha = 0.05$. Third, we find that the results for $\alpha = 1$ is slightly better than the results for $\alpha = 0.45$ and 10, revealing a delicate fact that $g_\Gamma(\gamma;\alpha)$ is not always monotonically increasing with α for all γ. Nevertheless, the differences are insignificant. In **Fig. 8**, the normalized performance is shown after taking average over 100 snapshots. We see that if 10 percent of the vertexes can be installed with facilities, we can achieve about 75 percent of the maximum facility gain.

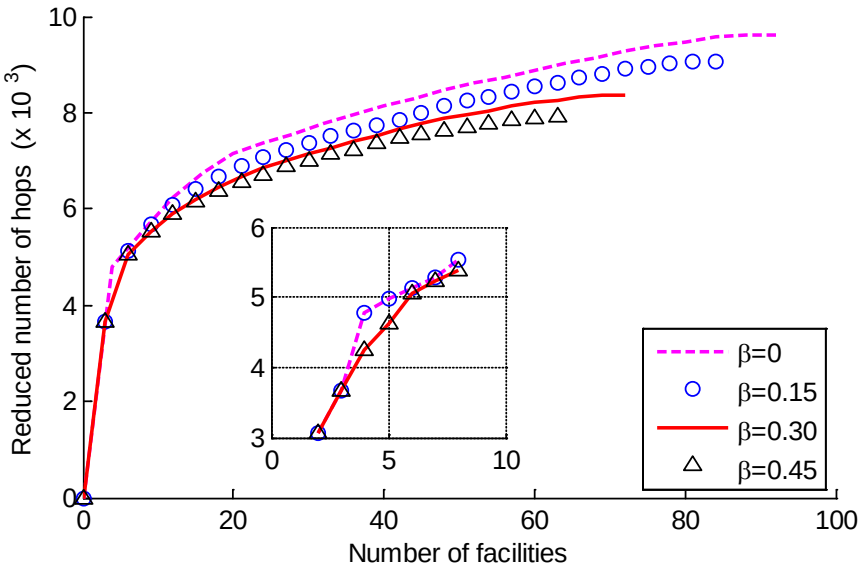

Fig. 9. Facility gain as a function of the number of facilities in Scenario IV ($H = 4$, $M = 5$, $p_{\text{non-leaf}} = 0.835$).

In all the above scenarios, we assume that all vertexes are available for facility placement. In reality, it is likely that some vertexes are not available for practical reasons. It is hence interesting to see whether our algorithm can adapt well to such conditions. In Scenario IV, we simulate a medium-size tree topology and randomly make some vertexes unavailable. It follows that

$$g_i = \begin{cases} l_i d_i, & \text{if } v_i \text{ is cache-enabled} \\ 0 & \text{if } v_i \text{ is cache-disabled} \end{cases}. \tag{5.2}$$

In such case, ϕ_i will always be equal to 0 if v_i is a unavailable vertex. We use β to denote the fraction of the unavailable vertexes in all vertexes. The performance of our algorithm is shown in **Fig 7**. As β grows from 0 to 0.45, it is observed that the performance degrades only slightly. In particular, when the number of facilities is small, the performance is almost identical. This indicates that our algorithm is able to cope well with the additional unavailability constraint.

Finally, we briefly discuss the applicability of our algorithm to networks with general topology. The problem of facility placement in a general network is known to be NP hard [19]. Our algorithm adapts a heuristic that relies on the tree structure, therefore cannot be directly applied to a general network. However, one may still adapt a two-step, heuristic approach to apply our algorithm by first establishing a multi-cast tree in the general network base on certain criteria. This is an interesting research direction to be pursued in our future work.

7. CONCLUSIONS

In this paper, we have studied the problem of optimal facility placement in mobile edge networks for achieving maximum gains in reducing the number of hops in the backhaul network. A literature review has been given to explain how our problem differs from the classic facility placement problem. A heuristic algorithm has been proposed to find sub-optimal solutions for placing a fixed number of facilities in an arbitrary tree topology with an arbitrary demand distribution. Proofs have been given regarding some important properties of the proposed algorithm. The complexity of the algorithm has been shown to scale linearly with the number of facilities and scale in a square order with the height of the tree. The effectiveness of the proposed algorithm has been validated by extensive simulations.

Acknowledgement

The authors acknowledge the support from the Natural Science Foundation of China (Grant NO. 61571378 and 61601388) and the Key Laboratory of Wireless Sensor Network & Communication，Shanghai Institute of Microsystem and Information Technology，Chinese Academy of Sciences.

References

[1] "Cisco visual networking index: global mobile data traffic forecast update, 2015-2020," *CISCO White paper*, February 2016. Article (CrossRef Link)

[2] X. Ge, S. Tu, G. Mao, C. X. Wang and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72-79, February 2016. Article (CrossRef Link)

[3] J. Plachy, Z. Becvar and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. of IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* , pp. 1-6, May 2016. Article (CrossRef Link)

[4] M. Condoluci, G. Araniti, T. Mahmoodi and M. Dohler, "Enabling the IoT machine age with 5G: machine-type multicast services for innovative real-time applications," *IEEE Access*, vol. 4, no.2 , pp. 5555-5569, February 2016. Article (CrossRef Link)

[5] X. Cheng, L. Yang and X. Shen, "D2D for Intelligent Transportation Systems: A Feasibility Study," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1784-1793, Aug. 2015. Article (CrossRef Link)

[6] X. Ge, S. Tu, T. Han, Q. Li and G. Mao, "Energy efficiency of small cell backhaul networks based on Gauss–Markov mobile models," *IET Networks*, vol. 4, no. 2, pp. 158-167, March 2015. Article (CrossRef Link)

[7] X. Cheng *et al.*, "Communicating in the real world: 3D MIMO," in *Proc. of IEEE Wireless Communications*, vol. 21, no. 4, pp. 136-144, August 2014. Article (CrossRef Link)

[8] X. Ge, J. Ye, Y. Yang and Q. Li, "User mobility evaluation for 5G small cell networks based on individual mobility model," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 528-541, March 2016. Article (CrossRef Link)

[9] B. P. Rimal, D. P. Van and M. Maier, "Mobile-edge computing vs. centralized cloud computing in fiber-wireless access networks," in *Proc. of IEEE INFOCOM*, pp. 991-996, April 2016. Article (CrossRef Link)
[10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, Fourth quarter 2015. Article (CrossRef Link)
[11] D. Satria , D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Elsevier Future Generation Computing Systems*, vol. 70, pp.138–147, May 2017. Article (CrossRef Link)
[12] J. Tadrous, A. Eryilmaz and H. El-Gamal, "Joint smart pricing and proactive content caching for mobile services," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2357-2371, Augest 2016. Article (CrossRef Link)
[13] J. Tadrous and A. Eryilmaz, "On optimal proactive caching for mobile networks with demand uncertainties," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2715-2727, October 2016. Article (CrossRef Link)
[14] Y. Zhang, X. Xu, X. Wang, Z. Fang and J. Tang, "NC-COCA: network coding-based cooperative caching scheme," in *Proc. of IEEE International Conference on Computational Science and Engineering*, pp. 976-980, May 2014. Article (CrossRef Link)
[15] A. Khreishah and J. Chakareski, "Collaborative caching for multicell-coordinated systems," in *Proc. of IEEE InfoCom Workshop*, pp. 257-262, Augest 2015. Article (CrossRef Link)
[16] S. Gitzenis, G. S. Paschos and L. Tassiulas, "Asymptotic laws for joint content replication and delivery in wireless networks," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2760-2776, May 2013. Article (CrossRef Link)
[17] N. Golrezaei, A. G. Dimakis and A. F. Molisch, "Scaling behavior for device-to-device communications with distributed caching," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 4286-4298, July 2014. Article (CrossRef Link)
[18] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations research* , vol. 12, no. 3, pp. 450-459, March 1964. Article (CrossRef Link)
[19] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. II: The p-medians," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 539-560, March 1979. Article (CrossRef Link)
[20] J. Sahoo, M. Salahuddin, R. Glitho, H. Elbiaze and W. Ajib, "A Survey on Replica Server Placement Algorithms for Content Delivery Networks," *IEEE Communications Surveys & Tutorials* , vol.PP, no.99, pp.1-1. Article (CrossRef Link)
[21] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. I: The p-centers," *SIAM Journal on Applied Mathematics*, vol.37, no. 3, pp. 513-538, March 1979. Article (CrossRef Link)
[22] R. Z. Farahani, M. Steadieseifi, and N. Asgari, "Multiple criteria facility location problems: A survey," *Applied Mathematical Modelling*, vol. 34, no. 7, pp. 1689-1709, July 2010. Article (CrossRef Link)
[23] A. Tamir. "An O (pn 2) algorithm for the p-median and related problems on tree graphs," *Operations Research Letters*, vol. 19, no. 2, pp. 59-64, February 1996. Article (CrossRef Link)
[24] B. C. Tansel, R. L. Francis, and T. J. Lowe, "State of the art—location on networks: a survey. Part I: the p-center and p-median problems," *Management Science*, vol. 29, no. 4, pp. 482-497, April 1983. Article (CrossRef Link)
[25] M. Hribar and M. S. Daskin, "A dynamic programming heuristic for the p-median problem," *European Journal of Operational Research*, vol. 101, no. 3, pp. 499-508, March 1997. Article (CrossRef Link)
[26] K. E. Rosing, C. S. Revelle, and D. A. Schilling, "A gamma heuristic for the p-median problem," *European Journal of Operational Research*, vol. 117, no. 3, pp. 522-532, March 1999. Article (CrossRef Link)

[27] B. Li, M. J. Golin, G. F. Italiano, X. Deng and K. Sohraby, "On the optimal placement of web proxies in the Internet," in *Proc. of IEEE INFOCOM '99*, pp. 1282-1290, 1999. Article (CrossRef Link)
[28] L. Qiu, V. N. Padmanabhan and G. M. Voelker, "On the placement of web server replicas," in *Proc. of IEEE INFOCOM*, pp. 1587-1596, April 2001. Article (CrossRef Link)
[29] P. Nuggehalli, V. Srinivasan, C. F. Chiasserini and R. R. Rao, "Efficient cache placement in multi-hop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 1045-1055, October 2006. Article (CrossRef Link)
[30] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," *Computer Networks*, vol. 40, no. 2, pp. 205-218, Feburary, 2002. Article (CrossRef Link)
[31] J. Xu, B. Li and D. L. Lee, "Placement problems for transparent data replication proxy services," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1383-1398, September 2002. Article (CrossRef Link)
[32] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch and G. Caire, "Femto caching: wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402-8413, December 2013. Article (CrossRef Link)
[33] S. Borst, V. Gupta and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. of IEEE INFOCOM*, pp. 1-9, 2010. Article (CrossRef Link)
[34] Y. Mansouri, S. T. Azad, and A. Chamkori, "Minimizing cost of k-replica in hierarchical data grid environment," in *Proc. of IEEE International Conference on Advanced Information Networking and Applications*, pp. 1073-1080, 2014. Article (CrossRef Link)
[35] U. Tos, R Mokadem, A. Hameurlain, T. Ayav, and S. Bora, "Dynamic replication strategies in data grid systems: a survey," *The Journal of Supercomputing*, vol. 71, no. 11, pp. 4116-4140, November 2015. Article (CrossRef Link)
[36] J. Jiang, S. Zhang, B. Li and B. Li, "Maximized cellular traffic offloading via device-to-device content sharing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 1, pp. 82-91, January 2016. Article (CrossRef Link)
[37] N. Tziritas, T. Loukopoulos, S. U. Khan and C. Z. Xu, "Distributed algorithms for the operator placement problem," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 182-196, December 2015. Article (CrossRef Link)

Jiping Jiao received his B.S. degree in Electronic Information Engineering from Hainan University, China, in 2008. He is currently a PhD candidate in the School of Information Science and Engineering, Xiamen University, China. His research interests include cognitive radio networks and wireless ad-hoc networking.

Lingyu Chen received a PhD degree from Xiamen University, China, in 2011. Since 2012, he has been an assistant professor at Xiamen University, China. Dr Chen's research interests include signal detection and processing in wireless communications, wireless ad-hoc networks, and software define wireless networks.

Xuemin Hong (S'05–M'12) received a PhD degree from Heriot-Watt University, UK, in 2008. He is currently a professor in the School of Information Science and Technology, Xiamen University. His research interests include wireless channel modeling, cognitive radio networks, and information-centric networks. He has published more than 40 technical papers in major international journals and conferences and 1 book chapter in the area of wireless communications.

Jianghong Shi received his PhD from Xiamen University, China, in 2002. He is currently a professor in the School of Information Science and Technology, Xiamen University. He is also the director of the West Straits Communications Engineering Center, Fujian Province, China. His research interests include wireless communication networks and satellite navigation.