This paper was presented as part of the main technical program at IEEE INFOCOM 2011

ESPN: Efficient Server Placement in Probabilistic Networks with Budget Constraint

Dejun Yang, Xi Fang and Guoliang Xue

***Abstract*—The notion of *probabilistic network* has been used to characterize the unpredictable environment in wireless communication networks or other unstable networks. In this paper, we are interested in the problem of placing servers in probabilistic networks subject to budget constraint, so as to maximize the expected number of servable clients that can successfully connect to a server. We study this problem in both the single-hop model and the multi-hop model. We discuss the computational complexity of this problem and show that it is NP-hard under both models. We then develop efficient approximation algorithms, which produce solutions provably close to optimal. If the costs of candidate locations are uniform, when extra budget is available in the future, the progressive feature of our algorithms allows for placing additional servers instead of relocating all the servers, while retaining the guaranteed performance. Results of extensive experiments on different topologies confirm the performance of our algorithms compared to the optimal algorithm and other heuristic algorithms.**

I. Introduction

Most studies on wireless networks have been assuming that the wireless communication between two nodes is deterministic, in the sense that two nodes can communicate with each other as long as they are within the transmission range of each other. However, due to unpredictable environmental effects, which make wireless channels lossy and unreliable, the success of a transmission is inherently *probabilistic* [3]. Motivated by this fact, *probabilistic network*, a well-known notion from social science, bioscience, artificial intelligence and machine learning communities, is introduced into the wireless communication community by Berchmans *et al.* [2].

In this paper, we consider the following scenario. A service provider decides to offer wireless connection service in an area. Through marketing investigations, it obtains the information (e.g. locations and access technologies) regarding the future clients within this area. In addition, it preselects a number of candidate locations to place servers, taking into account the policy of network service provider, physical constraints, and rental and maintaining cost. For each possible pair of server&client or client&client, the quality/reliability of the wireless channel in-between is estimated by experimental measurements [12, 23, 24]. To improve the quality of service, a client is allowed to have multiple connections to different servers. Based on the above information, the service provider needs to consider several issues before placing servers. First, there is a cost for service providers to operate and maintain these servers. Moreover, the cost of interconnecting these servers for data and state synchronization increases as the number of deployed servers grows. Given a fixed budget, in order to serve more clients, a limited number of servers must be placed strategically at certain locations in the network. Our study addresses this placement problem, which can be briefly stated as: *Given a set of candidate locations for servers and a fixed budget, how to place a number of servers to maximize the expected number of servable clients subject to budget constraint*. We investigate algorithms to intelligently perform placement which achieve guaranteed performance in comparison with the optimal solution. This enables service providers the insight into the relation of network cost and the achievable service quality, based on which they can further adjust their marketing strategies to reflect their revenue streams. When the costs of candidate locations are the same, our algorithm has a nice progressive feature: when extra budget becomes available in the future, service providers can just place additional servers instead of relocating all the servers, while retaining the guaranteed performance.

Yang, Fang and Xue are all with the CSE Dept. at Arizona State University, Tempe, AZ 85287. Email: {dejun.yang, xi.fang, xue}@asu.edu. This research was supported in part by NSF grants 0901451, 0830739 and 0905603, and ARO grant W911NF-09-1-0467. The information reported here does not reflect the position or the policy of the federal government.

The contributions of this paper are:

- We study the budgeted server placement problem in probabilistic networks, denote as BSP. We first consider the uniform cost case where all candidate locations have the same cost and then extend our algorithms to the non-uniform case. Our models include both the single-hop model and the multi-hop model.
- In each model, we analyze the computational complexity of the BSP problem and prove it to be NP-hard. For the uniform cost case, we design a 0.63-approximation algorithm for the single-hop model and an approximation algorithm returning a solution with value at least $0.63 \cdot OPT - \varepsilon$ and probability at least $(1-\delta)$ for the multi-hop model, where OPT is the value of the optimum, $\varepsilon > 0$ and $\delta \in (0,1)$ are two given arbitrarily small numbers.
- For the non-uniform cost case, we design a 0.31-approximation algorithm for the single-hop model and an approximation algorithm returning a solution with value at least $0.31 \cdot OPT - \varepsilon$ and probability at least $(1-\delta)$ for the multi-hop model. By using partial enumeration technique, we can improve the results to 0.63 and $0.63 \cdot OPT - \varepsilon$, respectively.
- We evaluate the performance of our algorithms on different topologies. Extensive experiment results confirm the advantage of our algorithms in both cases, especially significant for the non-uniform cost case.

The rest of the paper is organized as follows. In Section II, we briefly survey related work. We describe the network models studied in this paper and formally define our problem in Section III. Then in Section IV, we present simple algorithms with guaranteed performance to solve the problem for the uniform cost case. In Section V, we study the non-uniform cost case and design corresponding approximation algorithms. Before concluding this paper in Section VII, we evaluate our algorithms using comprehensive experiments in Section VI.

II. Related Work

Two well-known graph theoretic approaches have been applied to solve placement problems in networks: *Facility Location Problem* and *Minimum K-median Problem*, both of which are NP-hard [4]. The Facility Location Problem is to minimize the total cost of server installation, connecting each client to its designated server. The Minimum K-Median Problem is to select K servers so as to minimize the total cost of connecting the clients and servers. If the cost metric satisfies the triangle inequality, a number of algorithms with constant approximation ratios have been proposed [22]. However, none of these algorithms can be directly applied to solve the problem studied in this paper, since the metric in our model does not meet the requirement.

There has also been considerable work on server placement to achieve different objectives [7–9, 15, 17, 18, 20, 21]. In [15], Li *et al.* approached the server placement problem with the objective of minimizing the latency and used dynamic programming to obtain the optimal solution. However, their algorithm works under the assumption that the underlying network topologies are trees, which is not the case in our paper. Inspired by them, Jia *et al.* [9] also studied the server placement problem with the objective of minimizing the longest delay. Same as [15], their algorithm only works for tree topology. In [8], Jamin *et al.* showed that there is a rapid diminishing return to place more servers in terms of both client latency and server load-balancing. In [17], Qiu *et al.* developed several greedy algorithms to solve the server placement problem with a similar objective to the ones aforementioned. Note that one of the greedy algorithms is similar to our algorithm. However, they only showed that it is within a factor of 1.1-1.5 of optimal by simulating the algorithm over several synthetic and real network graphs. There is no theoretical proof given in their paper. In [18], Radoslavov *et al.* also proposed a max degree strategy by placing servers in decreasing order of node degrees. With a different objective, Shi *et al.* [20] studied the problem of placing a minimum number of servers in a network to cover all clients subject to distance constraints. It is noted that none of these works studied the server placement problem in probabilistic networks, which is more challenging. Although some of the heuristic algorithms proposed in previous works can be modified to apply to our problem, there is no performance guarantee.

Berchmans *et al.* [2] considered the problem of assigning links in probabilistic networks such that the number of satisfied clients (whose overall service success probability is met) is maximized subject to server capacity constraints. In their models, servers are given and no cost issues are considered, which is different from our problem.

Another closely related work is the budgeted maximum coverage problem studied by Khuller *et al.* [11]. It can be considered a special case of our problem, where the link is deterministic and the model is the single-hop model. Our problem is more general and hence more challenging.

III. Network Model and Problem Formulation

In this paper, we study the budgeted server placement problem in both single-hop networks and multi-hop networks.

A. Single-hop Probabilistic Network Model

A single-hop probabilistic network can be modeled by a bipartite graph $G_S = (\mathcal{S}, \mathcal{C}, E_S)$, where the vertex set $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$ represents n candidate locations for placing servers, the vertex set $\mathcal{C} = \{c_1, c_2, \cdots, c_m\}$ represents m clients requesting wireless service from servers, and the link set E represents potential connections between $\mathcal{S}$ and $\mathcal{C}$. Each candidate location s_i is associated with a weight $w(s_i) > 0$ denoting the cost of placing a server at s_i. Let $w(\mathcal{S}')$ denote the total cost of placing servers at candidate locations $\mathcal{S}'$, *i.e.*, $w(\mathcal{S}') = \sum_{s_i \in \mathcal{S}'} w(s_i)$. Abusing the notation a little bit, we will also use s_i to denote the server placed at candidate location s_i if no confusion can be caused. For each link $e = (s_i, c_j) \in E_S$, let $p(s_i, c_j)$ denote the probability of a successful connection between server s_i and client c_j. Obviously, we have $p(s_i, c_j) \in (0, 1]$. Links with $p(s_i, c_j) = 0$ can be removed from the graph without affecting the results. We assume that the link probabilities are independent. Let $\mathcal{S}(c_j)$ denote the set of neighbors of client c_j in G_S, *i.e.*, $\mathcal{S}(c_j) = \{s_i \in \mathcal{S} | (s_i, c_j) \in E_S\}$. Let $\mathcal{C}(s_i)$ denote the set of neighbors of server s_i in G_S, *i.e.*, $\mathcal{C}(s_i) = \{c_j \in \mathcal{C} | (s_i, c_j) \in E_S\}$. Similarly, let $\mathcal{C}(\mathcal{S}')$ denote the set of neighbors of $\mathcal{S}'$, *i.e.*, $\mathcal{C}(\mathcal{S}') = \cup_{s_i \in \mathcal{S}'} \mathcal{C}(s_i)$. A *server placement* in this paper is defined as a subset of candidate locations $\mathcal{S}'$ chosen to place servers. Thus, given a server placement $\mathcal{S}'$, the probability of a client $c_j \in \mathcal{C}(\mathcal{S}')$ successfully receiving service from $\mathcal{S}'$ is

$$P_{\mathcal{S}'}(c_j) = 1 - \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j)) . \qquad (1)$$

Here, we assume that the load balancing issue on servers has been considered while deciding the candidate server locations and blueprinting the network topology as the given input. In addition, we focus on the infrastructure construction in this paper. How to schedule communications to avoid interference is also out of the scope of this paper.

B. Multi-hop Probabilistic Network Model

We model the multi-hop probabilistic network as a graph $G_M = (V, E_M)$, where $V = \mathcal{S} \cup \mathcal{C}$ and $E_M \subseteq (\mathcal{S} \cup \mathcal{C}) \times \mathcal{C}$. Different from the single-hop model, we allow clients to receive service from servers directly or via other client(s) as intermediate relay(s) in this model. Therefore, multi-hop technology can increase the coverage area of the service. In this model, each candidate location $s_i \in \mathcal{S}$ is associated with weight $w(s_i)$, denoting the cost of placing a server at s_i, and each link $(u, v) \in E$ is associated with $p(u, v)$, representing the probability of a successful connection between u and v. Unlike the single-hop model, we cannot give a closed-form formulation to calculate $P_{\mathcal{S}'}(c_j)$ for a client c_j and a given server placement $\mathcal{S}'$ in general. Instead, we will show how to approximate this value arbitrarily close in Section IV.

C. Problem Definition

We first define our objective function.

Definition 3.1: (Quality of Server Placement): Given a server placement $\mathcal{S}' \subseteq \mathcal{S}$, the *Quality of Server Placement* of $\mathcal{S}'$, denoted as $Q(\mathcal{S}')$, is defined as the expected number of clients that can successfully obtain service from the deployed servers, *i.e.*, $Q(\mathcal{S}') = \sum_{c_j \in \mathcal{C}(\mathcal{S}')} P_{\mathcal{S}'}(c_j)$. □

Now, we formally define the problem studied in this paper.

Definition 3.2: (Budgeted Server Placement (BSP)): Given a probabilistic network and a fixed budget B, the *Budgeted Server Placement* problem is to find a server placement $\mathcal{S}'$ such that $Q(\mathcal{S}')$ is maximized, subject to the budget constraint, *i.e.*, $w(\mathcal{S}') \leq B$. □

In order to have a better understanding of the BSP problem, we first consider a simple case where the costs of all the candidate locations are uniform, denoted as ω. We will study the case where different locations have different costs in Section V. If the cost is the same for all candidate locations, the BSP problem can be stated in an alternative form. Given a probabilistic network, find a server placement $\mathcal{S}'$ of size k such that $Q(\mathcal{S}')$ is maximized. Here k is a function of B and ω, and can be calculated by $k = \lfloor \frac{B}{\omega} \rfloor$. As we will show later, even with uniform cost, it is NP-hard to determine the optimal server placement under both models.

IV. BSP Problem with Uniform Cost

In this section, we first present a general algorithmic framework to solve the BSP problem in both single-hop and multi-hop models. Then, we analyze the computational complexity of the problem and give the detailed algorithm for each model.

A. Overall Approach

Although the BSP problem will be proved to be NP-hard in both single-hop and multi-hop models in the following sections respectively, some tractable properties of the objective function enable us to design a simple greedy algorithm with performance provably close to optimal. These properties include nonnegativity, monotonicity and submodularity.

Definition 4.1: (Monotonicity): Let U be a non-empty finite set. Let f be a mapping from the power set of $\mathcal{U}$ to non-negative real numbers. We say f is *monotone* if $f(\mathcal{A}) \leq f(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$. □

Definition 4.2: (Submodularity): Let U be a non-empty finite set. Let f be a mapping from the power set of $\mathcal{U}$ to non-negative real numbers. We say f is *submodular* if $f(\mathcal{A} \cup \{v\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{v\}) - f(\mathcal{B})$ for all $v \in \mathcal{U} \setminus \mathcal{B}$ and all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$. □

A result of Nemhauser *et al.* [16] proves that, for a problem whose objective function has the above properties, the following greedy algorithm guarantees a solution with value at least $(1 - 1/e) > 0.63$ of the optimum, where e is the base of natural logarithm. *Starting with an empty set, an element that maximizes the newly added value is added in each iteration.* This result is formally stated in the following theorem.

Theorem 4.1: [16] Let f be a non-negative, monotone and submodular function. Let $\mathcal{A}$ be a set of k elements iteratively chosen as follows. Initially, $\mathcal{A}$ is empty. In each iteration, select an element v that maximizes $f(\mathcal{A} \cup \{v\}) - f(\mathcal{A})$ and add it into $\mathcal{A}$. Let $\mathcal{A}^*$ be a set that maximizes the value of f over all possible sets of size k. Then $f(\mathcal{A}) \geq (1 - 1/e) f(\mathcal{A}^*)$. □

Based on Theorem 4.1, we design a general algorithmic framework in Algorithm 1.

Algorithm 1: Approximation algorithmic framework for the BSP problem with uniform cost

```
Input : A probabilistic network and a positive integer $k$
Output: A server placement $\mathcal{S}_k$
1 $\mathcal{S}_0 \leftarrow \emptyset$;
2 for $i = 1$ to $k$ do
3     $s \leftarrow \arg\max_{s \in \mathcal{S} \setminus \mathcal{S}_{i-1}} (Q(\mathcal{S}_{i-1} \cup \{s\}) - Q(\mathcal{S}_{i-1}))$;
4     $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \{s\}$;
5 end
6 return $\mathcal{S}_k$.
```

Now the problem reduces to that of evaluating Q and proving the aforementioned properties, especially submodularity, of Q. We will discuss these issues in the following sections.

B. BSP in the Single-hop Model

We first show that the BSP problem is NP-hard in this model. We then give an exact formula to evaluate Q. Finally, we prove that Q is non-negative, monotone and submodular.

Theorem 4.2: The BSP problem is NP-hard in the single-hop model. □

Proof. We prove this theorem by a reduction from a well-known NP-hard problem, *Set Cover* problem (SC), to the BSP problem. The decision version of the SC problem is: Given a universe $\mathcal{U} = \{u_1, u_2, \cdots, u_q\}$, a family $\mathcal{S} = \{S_1, S_2, \cdots, S_p\}$ of subsets of $\mathcal{U}$ and an integer k, does there exist a subset $\mathcal{S}' \subseteq \mathcal{S}$ of size k, such that every element in $\mathcal{U}$ belongs to at least one member in $\mathcal{S}'$?

The decision version of the BSP problem in the single-hop model is: Given a bipartite graph $G_S = (\mathcal{S}, \mathcal{C}, E_S)$ with weight on each link, an integer k and a real number r, does there exist a subset $\mathcal{S}' \subseteq \mathcal{S}$ of size k, such that $Q(\mathcal{S}') \geq r$?

Given an instance of the SC problem, we construct an instance of the BSP problem as follows. For each $u_j \in \mathcal{U}$, we construct a node u_j as a client. For each $S_i \in \mathcal{S}$, we construct a node S_i as a server. If u_j belongs to S_i, we have a link between S_i and u_j. We set the weight on each link to 1 and set the parameter r to q. One reduction example is illustrated in Fig. 1.

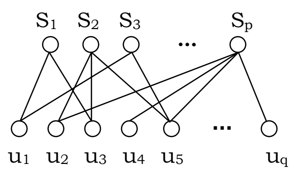

Fig. 1. Reduction from Set Cover to BSP

Obviously, the construction can be finished in polynomial time. Now we prove that there exists a solution for the SC problem if and only if there exists one for the BSP problem.

$\Longleftarrow$: Assume $\mathcal{S}' \subseteq \mathcal{S}$ is a solution of the BSP problem. Then each client u_j must have a link with at least one server

S_i in $\mathcal{S}'$. Mapping back to the instance of the SC problem, it means the corresponding element u_j belongs to S_i. Since we set the value of r to q, it means all the elements in $\mathcal{U}$ satisfy this condition. We have a solution of the SC problem.

$\Longrightarrow$: Suppose there exists a solution $\mathcal{S}' \subseteq \mathcal{S}$ of size k. Then each element u_j in $\mathcal{U}$ belongs to at least one member S_i in $\mathcal{S}'$. Mapping to the instance of the BSP problem, we know that the corresponding client u_j can successfully receive service from server S_i since $p(S_i, u_j)$ is 1. Furthermore, we have the number of such clients is q.

We proved that the BSP problem is NP-hard. ■

One important component in our algorithmic framework is to evaluate Q taking a server placement $\mathcal{S}'$ as an input. It will affect not only the quality of the solution but also the time complexity of the resulting algorithm. Based on (1) and Definition 3.1, we can compute $Q(\mathcal{S}')$ as follows, for any given $\mathcal{S}' \subseteq \mathcal{S}$.

$$Q(\mathcal{S}') = \sum_{c_j \in \mathcal{C}(\mathcal{S}')} \left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j))\right). \quad (2)$$

To be able to apply the algorithmic framework in Section IV-A to achieve an approximation algorithm, it suffices to prove Q to be non-negative, monotone and submodular. Clearly, Q is non-negative by Definition 3.1. As to monotonicity, for all $\mathcal{S}' \subseteq \mathcal{S}'' \subseteq \mathcal{S}$, we have $\mathcal{C}(\mathcal{S}') \subseteq \mathcal{C}(\mathcal{S}'')$ and thus

$$\begin{aligned} Q(\mathcal{S}'') &= \sum_{c_j \in \mathcal{C}(\mathcal{S}'')} \left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}''} (1 - p(s_i, c_j))\right) \\ &\geq \sum_{c_j \in \mathcal{C}(\mathcal{S}')} \left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j))\right) \\ &= Q(\mathcal{S}'). \end{aligned}$$

The submodular property is confirmed in the following lemma.

Lemma 4.1: Q is submodular in the single-hop model. □

Proof. First, we need to analyze the property of expression $Q(\mathcal{S}' \cup \{s\}) - Q(\mathcal{S}')$ for any $\mathcal{S}' \subseteq \mathcal{S}$ and any server s. We define $\Phi(\mathcal{S}') = Q(\mathcal{S}' \cup \{s\}) - Q(\mathcal{S}')$. Then, we have

$$\begin{aligned} \Phi(\mathcal{S}') &= \sum_{c_j \in \mathcal{C}(\mathcal{S}' \cup \{s\})} \left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap (\mathcal{S}' \cup \{s\})} (1 - p(s_i, c_j))\right) \\ &\quad - \sum_{c_j \in \mathcal{C}(\mathcal{S}')} \left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j))\right) \\ &= \sum_{c_j \in \mathcal{C}(s)} \left(\left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap (\mathcal{S}' \cup \{s\})} (1 - p(s_i, c_j))\right)\right. \\ &\quad \left. - \left(1 - \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j))\right)\right) \quad (3) \\ &= \sum_{c_j \in \mathcal{C}(s)} \left(\prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j))\right) p(s, c_j), \end{aligned}$$

where (3) follows from the fact that the marginal gain only comes from server s. It is clear that $\prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}'} (1 - p(s_i, c_j)) \geq \prod_{s_i \in \mathcal{S}(c_j) \cap \mathcal{S}''} (1 - p(s_i, c_j))$ for all $\mathcal{S}' \subseteq \mathcal{S}'' \subseteq \mathcal{S}$. Thus we have $\Phi(\mathcal{S}') \geq \Phi(\mathcal{S}'')$, which proves that Q is submodular according to Definition 4.2. ■

Having proved the required properties, it follows from Theorem 4.1 that with (2) plugged in for the evaluation of Q, Algorithm 1 is an approximation algorithm for the BSP problem in single-hop model. Specifically, we have the following theorem.

Theorem 4.3: Algorithm 1 computes a server placement that is a $(1 - 1/e)$-approximation of the optimum in time bounded by $O(n^3 m)$. □

Proof. The approximation ratio of our algorithm is guaranteed by Theorem 4.1 and Lemma 4.2. We focus on the time complexity. There are exactly $k \leq n$ iterations in total. In each iteration, we check all n candidate locations to select the one that gives the largest marginal gain. It requires $O(nm)$ time to compute the value of $Q(\mathcal{S}')$. Therefore, the time complexity of all executions is bounded by $O(n^3 m)$. ■

C. BSP in the Multi-hop Model

In this section, following the same analysis pattern for the single-hop model, we first analyze the computational complexity of the BSP problem in the multi-hop model. We then discuss how to evaluate the objective function Q approximately, for which the reason will be explained later. At the end of this section, we show that Q is non-negative, monotone and submodular in this model.

Theorem 4.4: The BSP problem is NP-hard in the multi-hop model. □

Proof. Since the BSP problem is proved to be NP-hard in the single-hop model in Theorem 4.2 and the single-hop model is a special case of the multi-hop model, it is clear that the BSP problem is also NP-hard in the multi-hop model. ■

Compared to the single-hop model, the multi-hop essence makes it unlikely to have a closed-form formulation to exactly compute the value of Q. However, we can consider the client connection process a random process and the number of clients successfully connecting to servers a random variable $\mathcal{Q}$. The *Law of Large Numbers* [6] guarantees that by simulating the random process enough number of times, we can approximate arbitrarily close to the exact value of $Q = E[\mathcal{Q}]$, with high probability.

Law of Large Numbers [6]: Let $X_1, X_2, \cdots, X_N$ be an independent trials process, with finite expected value μ and finite variance σ^2. Then for any $\alpha > 0$, $P\left(\left|\sum_{i=1}^{N} X_i/N - \mu\right| < \alpha\right) \to 1$ as $N \to \infty$.

Next we describe how to simulate the random process efficiently. Let us first examine the way successful service propagates. Suppose we have placed servers at candidate locations $\mathcal{S}'$. During the process, each pair of nodes u and v try to set up a connection with certain probability. Then, starting from $\mathcal{S}'$, the service will propagate through successful connections. Note that the above procedure is equivalent to that of first determining the success of each link and removing all the failed links, then checking if a client can connect to at least one server. In other words, a client can successfully obtain the

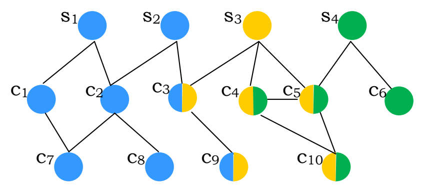

Fig. 2. Example showing the submodularity of Q_I, where $\mathcal{S}' = \{s_1, s_2\}$, $\mathcal{S}'' = \{s_1, s_2, s_3\}$, $\mathcal{C}(\mathcal{S}' \cup \{s_4\}) \setminus \mathcal{C}(\mathcal{S}') = \{c_4, c_5, c_6, c_{10}\}$ and $\mathcal{C}(\mathcal{S}'' \cup \{s_4\}) \setminus \mathcal{C}(\mathcal{S}'') = \{c_6\}$.

service if and only if there exists at least one successful path (consisting entirely of successful links) connecting it and at least one server. For each random process $I \in \mathcal{I}$, where $\mathcal{I}$ is the whole possibility space, the Quality of Server Placement $Q_I(\mathcal{S}')$ is equal to the number of clients that can be reached from $\mathcal{S}'$ in the resulting graph.

We give the detailed description in Algorithm 2.

Algorithm 2: Approximation algorithm for the BSP problem in the multi-hop model

```
Input  : A probabilistic network, a positive integer $k$ and
         two given arbitrarily small numubers $\varepsilon$ and $\delta$
Output: A server placement $\mathcal{S}_k$
1  $\mathcal{S}_0 \leftarrow \emptyset$;
2  $N = \frac{16n^2m^2}{\varepsilon^2\delta}$;
3  for $i = 1$ to $k$ do
4      Set $Q(\{s\}) = 0$ for all $s \in \mathcal{S} \setminus \mathcal{S}_{i-1}$;
5      for $j = 1$ to $N$ do
6          Construct $G' = (V, E')$ by removing each link
           $(u, v)$ from $E$ with probability $1 - p(u, v)$;
7          Compute $Q_j(\mathcal{S}_{i-1} \cup \{s\}) - Q_j(\mathcal{S}_{i-1})$, for all
           $s \in \mathcal{S} \setminus \mathcal{S}_{i-1}$;
8          for $s \in \mathcal{S} \setminus \mathcal{S}_{i-1}$ do
9              $Q(\{s\}) += \big(Q_j(\mathcal{S}_{i-1} \cup \{s\}) - Q_j(\mathcal{S}_{i-1})\big)$;
10         end
11     end
12     $Q(\{s\}) = Q(\{s\})/N$ for all $s \in \mathcal{S} \setminus \mathcal{S}_{i-1}$;
13     $s \leftarrow \arg\max_{s \in \mathcal{S} \setminus \mathcal{S}_{i-1}} Q(\{s\})$;
14     $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \{s\}$;
15 end
16 return $\mathcal{S}_k$.
```

In order to guarantee that Theorem 4.1 can be applied to this model, we are required to prove that Q is non-negative, monotone, and submodular. Similar with the function Q in the single-hop model, it is clear that Q is non-negative in this model as well. To prove the monotonicity, we only need to consider the fact that, if a client can connect to a server in $\mathcal{S}'$ then it must be able to connect to a server in $\mathcal{S}''$, for all $\mathcal{S}' \subseteq \mathcal{S}'' \subseteq \mathcal{S}$. Now we prove the submodularity of Q in the following lemma.

Lemma 4.2: Q is submodular in the multi-hop model. □

Proof. Our proof is similar to the analysis in [10]. First, we prove that for each possible outcome of random process $I \in \mathcal{I}$, Q_I is submodular. Let $\mathcal{S}'$ and $\mathcal{S}''$ be two subsets of $\mathcal{S}$ such that $\mathcal{S}' \subseteq \mathcal{S}''$. Let $\mathcal{C}(\mathcal{A})$ denote the set of clients that connect to at least one server in $\mathcal{A}$, for any $\mathcal{A} \subseteq \mathcal{S}$. We then have $Q_I(\mathcal{S}' \cup \{s\}) - Q_I(\mathcal{S}') = |\mathcal{C}(\mathcal{S}' \cup \{s\}) \setminus \mathcal{C}(\mathcal{S}')| = |\mathcal{C}(\mathcal{S}' \cup \{s\})| - |\mathcal{C}(\mathcal{S}')|$. For any $c \in \mathcal{C}(\mathcal{S}'' \cup \{s\}) \setminus \mathcal{C}(\mathcal{S}'')$, we know $c \notin \mathcal{C}(\mathcal{S}'')$ and $c \in \mathcal{C}(\{s\})$. Since $\mathcal{C}(\mathcal{S}'') \supseteq \mathcal{C}(\mathcal{S}')$, we have $c \notin \mathcal{C}(\mathcal{S}')$ and thus $c \in \mathcal{C}(\mathcal{S}' \cup \{s\}) \setminus \mathcal{C}(\mathcal{S}')$. This proves that $\mathcal{C}(\mathcal{S}'' \cup \{s\}) \setminus \mathcal{C}(\mathcal{S}'') \subseteq \mathcal{C}(\mathcal{S}' \cup \{s\}) \setminus \mathcal{C}(\mathcal{S}')$. An example is illustrated in Fig. 2. It follows that $Q_I(\mathcal{S}' \cup \{s\}) - Q_I(\mathcal{S}') \geq Q_I(\mathcal{S}'' \cup \{s\}) - Q_I(\mathcal{S}'')$.

We have proved that Q_I is submodular for each possible outcome of the link success test $I \in \mathcal{I}$. Since

$$Q(\mathcal{A}) = \sum_{I \in \mathcal{I}} P(I) \cdot Q_I(\mathcal{A}),$$

where $P(I)$ is the probability that outcome I happens, $Q(\mathcal{A})$ is a positive linear combination of submodular functions. Therefore Q is also submodular. ■

The following theorem states the performance guarantee and the time complexity of Algorithm 2. Note that the time complexity derived here is fairly conservative. We will show in Section VI that the simulating processes converge very fast. Therefore in practice, N can take a value much smaller than the one state in Line 2 of Algorithm 2, without significantly degrading the performance.

Theorem 4.5: For any given arbitrarily small numbers $\varepsilon > 0$ and $\delta \in (0, 1)$, Algorithm 2 computes a server placement such that $Q(\mathcal{S}_k) \geq (1 - 1/e)Q(\mathcal{S}^*) - \varepsilon$ with probability at least $(1 - \delta)$ in time bounded by $O(|E_M| \cdot \frac{n^3m^2}{\varepsilon^2\delta})$, where $\mathcal{S}^*$ is the optimal solution of the BSP problem and $|E_M|$ is the number of links in graph G_M. □

Proof. We prove the approximation ratio and the time complexity separately.

Performance Analysis: We first prove that by setting $N = \frac{16n^2m^2}{\varepsilon^2\delta}$, we can achieve an estimated value $\bar{Q} = \frac{\sum_{i=1}^{N} Q_i}{N}$ of Q, such that $|\bar{Q}(\mathcal{S}') - Q(\mathcal{S}')| < \frac{\varepsilon}{4n}$ with probability at least $(1-\delta)$ for any $\mathcal{S}' \subseteq \mathcal{S}$, where Q_i is the result of ith simulation.

Let σ^2 be the variance of Q. We then have

$$\sigma^2 = E[(\mathcal{Q} - Q)^2] \leq (max_{I \in \mathcal{I}} Q_I - Q)^2 \leq m^2,$$

since $0 \leq \mathcal{Q} \leq m$ and $0 \leq Q \leq m$. By Chebyshev Inequality [6], we know that $P(|\bar{Q} - Q| \geq \delta) \leq \frac{\sigma^2}{N\delta^2}$. Equivalently, $P(|\bar{Q} - Q| < \delta) \geq 1 - \frac{\sigma^2}{N\delta^2}$. Plugging in $\delta = \frac{\varepsilon}{4n}$, we have

$$P\left(|\bar{Q} - Q| < \frac{\varepsilon}{4n}\right) \geq 1 - \frac{16n^2\sigma^2}{N\varepsilon^2} \geq 1 - \frac{16n^2m^2}{N\varepsilon^2} = 1 - \delta.$$

Now we are ready to prove the performance. Since we proved $|\bar{Q}(\mathcal{S}') - Q(\mathcal{S}')| < \delta = \frac{\varepsilon}{4n}$, we have $-2\delta \leq (\bar{Q}(\mathcal{S}' \cup \{s\}) - \bar{Q}(\mathcal{S}')) - (Q(\mathcal{S}' \cup \{s\}) - Q(\mathcal{S}')) \leq 2\delta$. Therefore

$$\begin{aligned} & Q(\mathcal{S}_i) - Q(\mathcal{S}_{i-1}) + 2\delta \geq \bar{Q}(\mathcal{S}_i) - \bar{Q}(\mathcal{S}_{i-1}) \\ = & \max_{s \in \mathcal{S}} (\bar{Q}(\mathcal{S}_{i-1} \cup \{s\}) - \bar{Q}(\mathcal{S}_{i-1})) \\ \geq & \max_{s \in \mathcal{S}} (Q(\mathcal{S}_{i-1} \cup \{s\}) - Q(\mathcal{S}_{i-1}) - 2\delta). \end{aligned} \tag{4}$$

Next we prove that $Q(\mathcal{S}_i) - Q(\mathcal{S}_{i-1}) \geq \frac{1}{k}(Q(\mathcal{S}^*) - Q(\mathcal{S}_{i-1})) - 4\delta$. Using monotonicity of Q, we have

$$\begin{aligned} Q(\mathcal{S}^*) - Q(\mathcal{S}_{i-1}) &\leq Q(\mathcal{S}^* \cup \mathcal{S}_{i-1}) - Q(\mathcal{S}_{i-1}) \\ &= Q(\mathcal{S}^* \setminus \mathcal{S}_{i-1} \cup \mathcal{S}_{i-1}) - Q(\mathcal{S}_{i-1}). \end{aligned}$$

Assume $\mathcal{S}^* \setminus \mathcal{S}_{i-1} = \{s_1, s_2, \cdots, s_t\}$ and $Z_j = Q(\mathcal{S}_{i-1} \cup \{s_1, s_2, \cdots, s_j\}) - Q(\mathcal{S}_{i-1} \cup \{s_1, s_2, \cdots, s_{j-1}\})$.

Then $Q(\mathcal{S}^*) - Q(\mathcal{S}_{i-1}) \le \sum_{j=1}^{t} Z_j$. Now, we have

$Z_j - 2\delta \le Q(\mathcal{S}_{i-1} \cup \{s_j\}) - Q(\mathcal{S}_{i-1}) - 2\delta \le Q(\mathcal{S}_i) - Q(\mathcal{S}_{i-1}) + 2\delta$,

where the first inequality follows from the submodularity and the second follows from (4). Thus, it follows that

$$Q(\mathcal{S}^*) - Q(\mathcal{S}_{i-1}) \le \sum_{j=1}^{t} Z_j \le t(Q(\mathcal{S}_i) - Q(\mathcal{S}_{i-1}) + 4\delta).$$

Manipulating the inequality and noting that $t \le k$, we have

$$Q(\mathcal{S}_i) - Q(\mathcal{S}_{i-1}) \ge \frac{1}{k}(Q(\mathcal{S}^*) - Q(\mathcal{S}_{i-1})) - 4\delta. \quad (5)$$

Next, we prove $Q(\mathcal{S}_i) \ge (1 - (1 - \frac{1}{k})^i)Q(\mathcal{S}^*) - 4k\delta$ using induction. First, we have $Q(\mathcal{S}_1) \ge \frac{1}{k}Q(\mathcal{S}^*) - 4\delta$ following from (5). Let us prove it holds when $i > 1$. We have

$$\begin{aligned}
Q(\mathcal{S}_i) &= Q(\mathcal{S}_{i-1}) + (Q(\mathcal{S}_i) - Q(\mathcal{S}_{i-1})) \\
&\ge Q(\mathcal{S}_{i-1}) + \frac{1}{k}(Q(\mathcal{S}^*) - Q(\mathcal{S}_{i-1})) - 4\delta \quad (6) \\
&= \left(1 - \frac{1}{k}\right) Q(\mathcal{S}_{i-1}) + \frac{1}{k}Q(\mathcal{S}^*) - 4\delta \\
&\ge \left(1-\frac{1}{k}\right)\left(\left(1-\left(1-\frac{1}{k}\right)^{i-1}\right)Q(\mathcal{S}^*)-4k\delta\right)+\frac{1}{k}Q(\mathcal{S}^*)-4\delta \quad (7) \\
&= \left(1 - \left(1 - \frac{1}{k}\right)^i\right) Q(\mathcal{S}^*) - 4k\left(1 - \frac{1}{k}\right)\delta - 4\delta \\
&= \left(1 - \left(1 - \frac{1}{k}\right)^i\right) Q(\mathcal{S}^*) - 4k\delta,
\end{aligned}$$

where (6) follows from (5), and (7) follows from the induction.

Thus plugging in $\delta = \frac{\varepsilon}{4n}$ and knowing $k \le n$, we have

$$\begin{aligned}
Q(\mathcal{S}_k) &\ge \left(1 - \left(1 - \frac{1}{k}\right)^k\right) Q(\mathcal{S}^*) - 4k\delta \\
&\ge (1 - 1/e)Q(\mathcal{S}^*) - \varepsilon. \quad (8)
\end{aligned}$$

Time Complexity: There are exactly $k \le n$ iterations in total. In each iteration, we need to simulate the random process for N times. For each random process, Line 7 takes $O(|E_M|)$ if we use BSF search. Therefore, the time complexity of the whole algorithm can be bounded by $O(nN|E_M|)$. ■

V. BSP Problem with Non-uniform Cost

We have been assuming that the cost associated with each candidate server location is the same. However, sometimes this may not be the case in reality. For instance, one may need to pay more for some candidate location to rent the space for placing a server. It costs more to maintain or operate the server at some locations. Hence we have to deal with the BSP problem where the costs of the candidate locations are different. It is clear that the BSP problem is NP-hard in both the single-hop and the multi-hop models for this case as well.

Obviously, the criteria in Section IV for selecting a candidate location will not work in this case. An intuitive idea would be to take into account both marginal gain and cost in the form:

$$\underset{s \in \mathcal{S} \setminus \mathcal{S}_{i-1}, w(\mathcal{S}_{i-1} \cup \{s\}) \le B}{\arg\max} \frac{Q(\mathcal{S}_{i-1} \cup \{s\}) - Q(\mathcal{S}_{i-1})}{w(s)}. \quad (9)$$

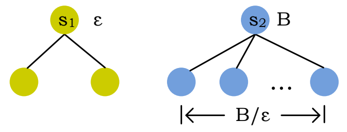

Fig. 3. Example showing simple greedy criteria fails badly, where the labels beside the candidate locations denote the costs.

However, a simple example in Fig. 3 indicates that the above criteria could return an arbitrarily bad solution. In this example, we have $w(s_1) = \varepsilon$ and $w(s_2) = B$. The way we set the weights allows us to select only one of two candidate locations. Assume the success probability is 1 on each link. Also, let s_1 connect to 2 clients and s_2 connect to B/ε clients. Thus the ratios for s_1 and s_2 are $\frac{2}{\varepsilon}$ and $\frac{B/\varepsilon}{B} = \frac{1}{\varepsilon}$, respectively. Based on the selecting criteria, s_1 would be picked. Whereas, if picking s_2 instead of s_1, we can have the Quality of Server Placement equal to B/ε instead of 2 otherwise. When ε is approaching 0, the greedy algorithm performs arbitrarily bad.

However, a provably good approximation can be obtained as follows. Instead of returning a server placement solely based on criteria (9), we return the better one of two server placements selected according to Equation (9) and the criteria regardless of costs. We illustrate this algorithmic framework for the BSP problem with non-uniform costs in Algorithm 3.

Algorithm 3: Approximation algorithmic framework for the BSP problem with non-uniform costs

```
Input  : A probabilistic network and a positive budget $B$
Output: A server placement $\mathcal{S}'$
1  $\mathcal{S}_1 \leftarrow \emptyset$, $\mathcal{S}_2 \leftarrow \emptyset$;
2  while $\exists s \in \mathcal{S} \setminus \mathcal{S}_1$ such that $w(\mathcal{S}_1 \cup \{s\}) \le B$ do
3      $s \leftarrow \arg\max_{s \in \mathcal{S} \setminus \mathcal{S}_1, w(\mathcal{S}_1 \cup \{s\}) \le B} (Q(\mathcal{S}_1 \cup \{s\}) - Q(\mathcal{S}_1))$;
4      $\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{s\}$;
5  end
6  while $\exists s \in \mathcal{S} \setminus \mathcal{S}_2$ such that $w(\mathcal{S}_2 \cup \{s\}) \le B$ do
7      $s \leftarrow \arg\max_{s \in \mathcal{S} \setminus \mathcal{S}_2, w(\mathcal{S}_2 \cup \{s\}) \le B} \frac{Q(\mathcal{S}_2 \cup \{s\}) - Q(\mathcal{S}_2)}{w(s)}$;
8      $\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{s\}$;
9  end
10 return $\arg\max_{\mathcal{S}' \in \{\mathcal{S}_1, \mathcal{S}_2\}} Q(\mathcal{S}')$.
```

Theorem 5.1: For the single-hop model, Algorithm 3 computes a server placement $\mathcal{S}'$ that is a $\frac{1}{2}(1-1/e)$-approximation of the optimum in time bounded by $O(n^3 m)$. For the multi-hop model, given any two arbitrarily small numbers $\varepsilon > 0$ and $\delta \in (0, 1)$, Algorithm 3 computes a server placement $\mathcal{S}'$ such that $Q(\mathcal{S}') \ge \frac{1}{2}(1 - 1/e)Q(\mathcal{S}^*) - \varepsilon$ with probability at least $(1 - \delta)$ in time bounded by $O(nN|E_M|)$, where $\mathcal{S}^*$ is the optimal solution of the BSP problem, $|E_M|$ is the number of links in graph G_M and $N = \frac{16B^2m^2}{\varepsilon^2\delta \min\{w(s_i)\}}$. □

Proof. For the single-hop model, the performance ratio is guaranteed by Theorem 3 in [14], which proves the ratio for any submodular function. The time complexity analysis is similar to the proof of Theorem 4.3. The difference is that there are at most n iterations. For the multi-hop model, the proof is similar as in Theorem 4.5 and [13], and hence omitted due to space limitation. The time complexity analysis is similar to the proof of Theorem 4.5 except the difference that there are at most n iterations. ■

By using partial enumeration technique [11], we can achieve an even better theoretical bound with a higher time complexity. We illustrate the improved approximation framework in Algorithm 4.

Algorithm 4: Improved approximation algorithmic framework for the BSP problem with non-uniform costs

```
Input  : A probabilistic network and a positive budget $B$
Output: A server placement $\mathcal{S}'$
1 $\mathcal{S}_1 \leftarrow \arg\max_{\mathcal{S}'\subseteq\mathcal{S},|\mathcal{S}'|<3,w(\mathcal{S}')\leq B} Q(\mathcal{S}')$, $\mathcal{S}_2 \leftarrow \emptyset$;
2 for $\mathcal{S}' \subseteq \mathcal{S}$, $|\mathcal{S}'| = 3$, $w(\mathcal{S}') \leq B$ do
3     while $\exists s \in \mathcal{S} \setminus \mathcal{S}'$ such that $w(\mathcal{S}' \cup \{s\}) \leq B$ do
4         $s \leftarrow \arg\max_{s\in\mathcal{S}\setminus\mathcal{S}', w(\mathcal{S}'\cup\{s\})\leq B} \frac{Q(\mathcal{S}'\cup\{s\})-Q(\mathcal{S}')}{w(s)}$;
5         $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{s\}$;
6     end
7     $\mathcal{S}_2 \leftarrow \arg\max_{\mathcal{S}''\in\{\mathcal{S}_2,\mathcal{S}'\}} Q(\mathcal{S}'')$;
8 end
9 return $\arg\max_{\mathcal{S}'\in\{\mathcal{S}_1,\mathcal{S}_2\}} Q(\mathcal{S}')$.
```

Theorem 5.2: For the single-hop model, Algorithm 4 computes a server placement $\mathcal{S}'$ that is a $(1-1/e)$-approximation of the optimum in time bounded by $O(n^6 m)$. For the multi-hop model, given any two arbitrarily small numbers $\varepsilon > 0$ and $\delta \in (0, 1)$, Algorithm 4 computes a server placement $\mathcal{S}'$ such that $Q(\mathcal{S}') \geq (1 - 1/e)Q(\mathcal{S}^*) - \varepsilon$ with probability at least $(1 - \delta)$ in time bounded by $O(n^4 N|E_M|)$, where $\mathcal{S}^*$ is the optimal solution of the BSP problem, $|E_M|$ is the number of links in graph G_M and $N = \frac{64B^2m^2}{\varepsilon^2\delta\min\{w(s_i)\}}$. □

Proof. The proofs of the performance guarantees are similar as in Theorem 4.5 and [13]. We focus on the time complexity. Obviously, the most time consuming part is the for-loop, which has at most n^3 iterations. For each iteration, it takes $O(n^3 m)$ and $O(nN|E_M|)$ time for the single-hop model and the multi-hop model, respectively. Therefore, the total time complexity is $O(n^6 m)$ and $O(n^4 N|E_M|)$, respectively. ■

VI. Experimental Evaluations

In this section, we evaluate the performance of our approximation algorithms through extensive experiments. Since we are the first to study the BSP problem in probabilistic networks, we compare our algorithms with brute force optimal algorithm and various heuristic placement algorithms. Although these algorithms were intentionally designed to solve different problems [17–19], simple modification will adapt them to the BSP problem.

A. Various Algorithms

Random Deployment Algorithm (RDA): The random algorithm arbitrarily selects a number of candidate server locations with uniform probability among all locations subject to budget constraint. We consider it as a "lower bound" of the server placement algorithms, since an efficient algorithm should perform at least as well as the random placement.

Degree-based Algorithm (DBA): For the uniform cost case, DBA sorts all the candidate locations in non-increasing order of their degrees, and then selects the top k locations. For the non-uniform cost case, DBA sorts all the candidate locations in non-increasing order of the ratio of the degree to the cost, and then keeps selecting the location sequentially until none of the remain candidate locations is applicable without violating the budget constraint. DBA is motivated by the intuition that a candidate location with a higher degree has potential to serve more clients.

Optimal Algorithm (OPT): We simply use brute force to find the optimal solution. Therefore, it is only practical for small instances in the uniform cost case, because the size of the server placement is unknown for the non-uniform cost case.

Throughout this section, we denote our algorithm as SGA (*Submodularity-based Greedy Algorithm*). Note that we implemented Algorithm 3 for the BSP problem with non-uniform cost due to its low time complexity.

B. Experiment Setup

In order to see the impact of network topology on the performance of server placement algorithms, we use two graph models to generate different topologies.

Random Model (RAM): In this model, we randomly generate a number of candidate server locations and clients in a square of size 1000×1000. There is a link between a client and a server, or two clients for the multi-hop model only, if the distance between them is below a threshold. We use RAM-S and RAM-M to denote the single-hop RAM and the multi-hop RAM, respectively.

Preferential Attachment Model (PAM): It is a well-known model proposed by Barabasi and Albert [1] to generate a power-law graph [5]. The model generates a graph node by node. For each new node u, it creates γ links, where γ is a constant parameter same for all nodes. It creates a link to another node v with probability proportional to v's degree $d(v)$. In the experiments, we set $\gamma = |\mathcal{S}|/10$ in the single-hop model and $\gamma = (|\mathcal{S}| + |\mathcal{C}|)/50$ in the multi-hop model. We use PAM-S and PAM-M to denote the single-hop PAM and the multi-hop PAM, respectively. To make the topology satisfy our network model, we remove all the links between two server candidate locations and between two clients for the single-hop model, and remove all the links between two candidate server locations for the multi-hop model.

For both types of topologies, we assigned a value drawn from a uniform distribution in $(0, 1]$ to each link as its success probability. For cost assignment, we assigned unit cost to each candidate location for the uniform cost case, and cost uniformly chosen from $[0.6, 3]$ for the non-uniform cost case.

Besides impact of the network topology, we are also interested in the impacts of budget and the number of clients on the performance of various algorithms. For the former, we ran experiments with B varying from 1 to 20, while keeping $|\mathcal{S}| = 50$ and $|\mathcal{C}| = 50$. For the latter, we ran experiments with $|\mathcal{C}|$ varying from 10 to 100, while keeping $|\mathcal{S}| = 50$ and $B = 10$. We ran each different setup 100 times and averaged the results.

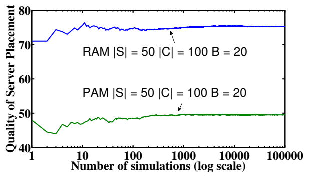

Fig. 4. Convergence of simulating processes

For the multi-hop model, we set $\varepsilon = 0.1$ and $\delta = 0.05$. When we decided the value of N, we noticed that the quality of estimation of Q after 10000 simulating processes is comparable to that after 100000 or more simulating processes even for the instance of the largest size as shown in Fig. 4. Therefore, throughput all experiments, we set $N = 10000$.

C. Experiment Results

In this section, we discuss the impacts of various factors on the performance of different server placement algorithms.

1) Uniform Cost Case: Fig. 5 and Fig. 6 show the results of different experiment setups for the uniform cost case. In general, SGA (red solid line with circle markers) outperforms the other two algorithms and RDA (blue solid line with star markers) has the worst performance among them. More specifically, Fig. 5 depicts the performance of various algorithms when the budget is varying. As we can see, when the budget increases, the Quality of Server Placement increases as expected. However, the increasing speed tends to slow down gradually. These are also evidences of the submodularity of our objective function Q.

For RAM, the Quality of Server Placement is higher in the multi-hop model than in the single-hop model for the same budget. Because in the multi-hop model, a client can connect to a server through other clients. This confirms our statement in Section III that the multi-hop model can improve the coverage. *The same improvement cannot be observed for PAM, because we have different parameters γ for the single-hop and the multi-hop models, which make these two models not comparable.*

As for the impact of network topology, we note that DBA (magenta dashed line with square markers) performs almost as well as SGA in PAM. This is because, in PAM, the node degree distribution follows power law, which allows a small number of servers to have dramatically higher degrees than others. DBA can take advantage of this property to have a better performance than in other network topologies.

To check the impact of the number of clients on the performance, we plot the experiment results in Fig. 6. We have the same observations as in Fig. 5 about the impact of network topologies. However, we note that the multi-hop model has a different performance curve pattern from the single-hop model. In the single-hop model, all the performance curves are approximately lines. Whereas in the multi-hop model, all the performance curves have accelerating returns within the given range. The reason is that the number of clients connected to servers grows exponentially in the multi-hop model.

Although it is impractical to run experiments using OPT in general, we generated some small instances ($|\mathcal{S}| = 20$ and $|\mathcal{C}| = 50$) to compare the performance of SGA to OPT, as shown in Fig. 7. As we can observe, the performance of SGA is almost optimal.

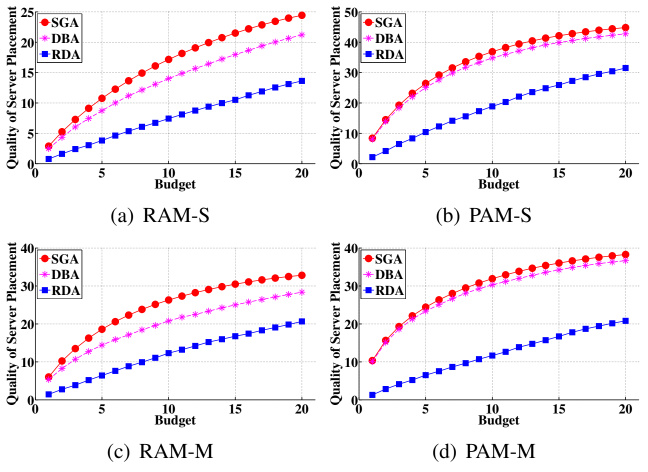

Fig. 5. Varying budget for uniform cost case.

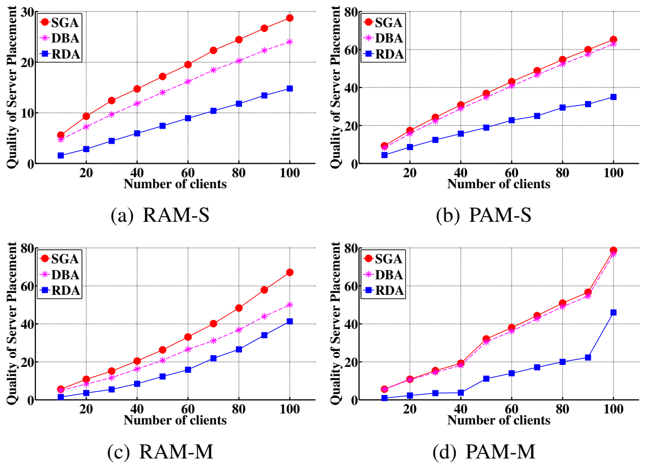

Fig. 6. Varying the number of clients for the uniform cost case.

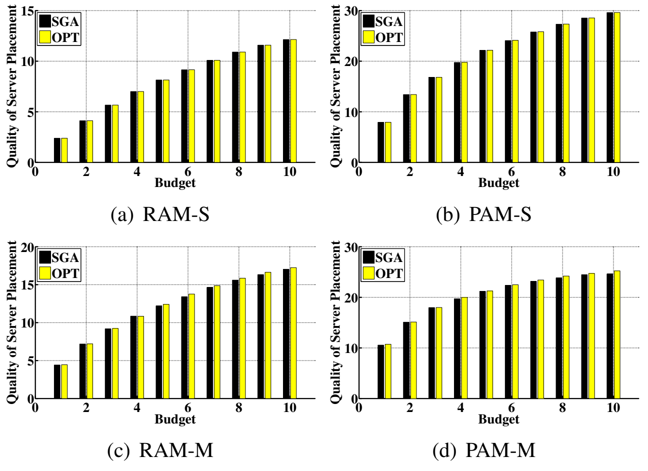

Fig. 7. Comparison between SGA and OPT.

2) Non-uniform Cost Case: Fig. 8 and Fig. 9 show the experiment results for the non-uniform cost case.

Most of the observations from the uniform cost case still hold here. SGA has the best performance while RDA has the

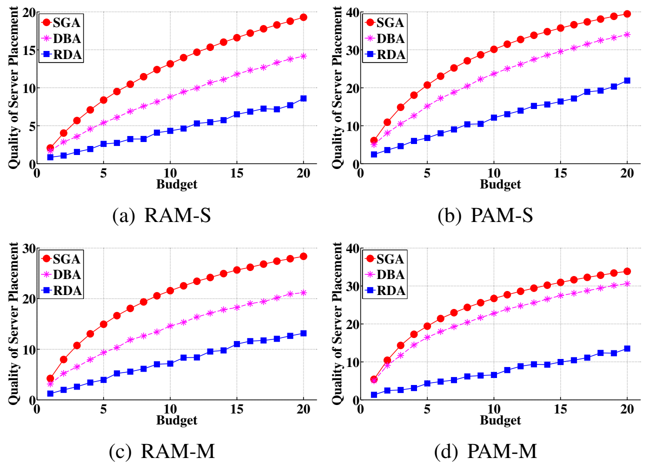
Fig. 8. Varying budget for the non-uniform cost case.

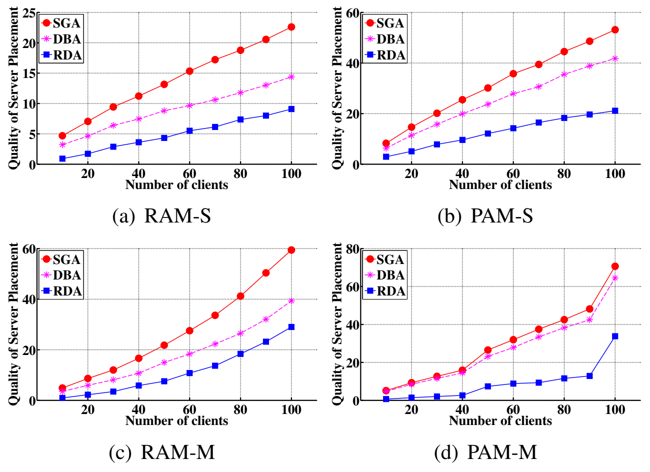
Fig. 9. Varying the number of clients for the non-uniform cost case.

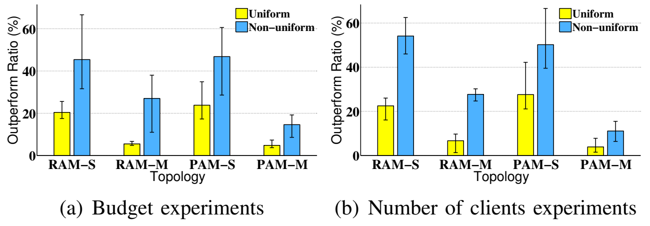
Fig. 10. Outperform Ratio of SGA to DBA.

worst. However, it is noted that the performance gap between SGA and DBA is widen, compared to the uniform cost case. In order to capture this difference, we define the performance improvement from Algorithm A to Algorithm B as

$$OutperformRatio = \frac{Q(\mathcal{S}_A) - Q(\mathcal{S}_B)}{Q(\mathcal{S}_B)}.$$

We plot the maximum, minimum and average outperform ratio of SGA to DBA across all instances in Fig. 10. Fig. 10(a) shows the results when the budget is varying and Fig. 10(b) shows the results when the number of clients is varying. In both scenarios, we can see that SGA outperforms DBA more in the non-uniform cost case than in the uniform cost case.

VII. Conclusions

In this paper, we have studied the budgeted server placement problem in probabilistic networks, denoted as BSP. We first considered the case where the cost associated with all candidate locations is the same. We then extended our results to the non-uniform cost case. For each case, we studied our problem in both the single-hop model and the multi-hop model. We proved that the BSP problem is NP-hard in both models and presented efficient approximation algorithms. If the costs of candidate locations are uniform, with extra budget, the progressive feature of our algorithms allows for placing additional servers instead of relocating all the servers, while retaining the guaranteed performance. Extensive experiment results confirmed the performance of our algorithms compared to the optimal algorithm and other heuristic algorithms.

References

[1] A.L. Barabasi and R. Albert; Emergence of scaling in random networks; *Science*, Vol.286(1999), pp. 509–512.

[2] F.J. Berchmans, W-K. Hon, A.C.Y. Huang, C-S. Liu, E. Lo and D.K.Y. Yau; Optimizing link assignment to enhance service in probabilistic network; *SECON'10*, pp. 235–243.

[3] A. Cerpa, J.L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statiscal model of lossy links in wireless sensor networks; *IPSN'05*, pp. 81–88.

[4] P. Crescenzi and V. Kann; A compendium of NP optimization problems; *http://www.csc.kth.se/∼viggo/wwwcompendium/*, 1998.

[5] M. Faloutsos, P. Faloutsos and C. Faloutsos; On power-law relationships of the Internet topology; *SIGCOMM'99*, pp. 251–262.

[6] C.M. Grinstead and J.L. Snell; *Introduction to probability*; American Mathematical Society, 1997.

[7] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt and L. Zhang; On the placement of Internet instrumentation; *INFOCOM'00*, pp. 295–304.

[8] S. Jamin, C. Jin, A.R. Kurc, D. Raz and Y. Shavitt; Constrained mirror placement on the Internet; *INFOCOM'01*, pp. 31–40.

[9] X.H. Jia, D.Y. Li, X.D. Hu, H.J. Huang and D.Z. Du; Optimal placement of proxies of replicated Web servers in the Internet; *WISE'00*, pp. 55–59.

[10] D. Kempe, J. Kleinberg and E. Tardos; Maximizing the spread of influence through a social network; *SIGKDD'03*, pp. 137–146.

[11] S. Khuller, A. Moss and J. Naor; The budgeted maximum coverage problem; *Information Processing Letters*, vol.70 (1999), pp. 39–45.

[12] K. Kim and K.G. Shin; On accurate measurement of link quality in multi-hop wireless mesh networks; *MOBICOM'06*, pp. 38–49.

[13] A. Krause and C. Guestrin; A note on the budgeted maximization of submodular functions; *Technical Report. CMU-CALD-05-103*, 2005.

[14] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriensen and N. Glance; Cost-effective outbreak detection in networks; *SIGKDD'07*, pp. 420–429.

[15] B. Li, M.J. Golin, G.F. Italiano and X. Deng; On the optimal placement of web proxies in the Internet; *INFOCOM'99*, pp. 1282–1290.

[16] G. Nemhauser, L. Wolsey and M. Fisher; An analysis of the approximations for maximizing submodular set functions; *Mathematical Programming*, 1978.

[17] L. Qiu, V.N. Padmanabhan and G.M. Voelker; On the placement of Web server replicas; *INFOCOM'01*, pp. 1587–1596.

[18] P. Radoslavov, R. Govindan and D. Estrin; Topology-informed Internet replica placement; *Proceedings of the Sixth International Workshop on Web Caching and Content Distribution (WCW'01).*

[19] S. Roy, H. Pucha, Z. Zhang, Y.C. Hu and L. Qiu; On the placement of infrastructure overlay nodes; *IEEE/ACM Trans. on Networking*, Vol.14(2009), pp. 1298–1311.

[20] S. Shi and J. Turner; Placing servers in overlay networks; *SPECTS'02*.

[21] X. Tang and J. Xu; On replica placement for QoS-aware content distribution; *INFOCOM'04*, pp. 806–815.

[22] V. Vazirani; *Approximation algorithms*; Springer, 2001.

[23] A. Woo, T. Tong and D. Culler; Taming the underlying challenges of reliable multihop routing in sensor networks; *SENSYS'03*, pp. 14–27.

[24] H. Zhang, A, Arora and P. Sinha; Learn on the fly: data-driven link estimation and routing in sensor network backbones; *INFOCOM'06*, pp. 1–12.