An Approximation Algorithm for Data Storage Placement in Sensor Networks

Bo Sheng, Chiu C. Tan, Qun Li, and Weizhen Mao
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795, USA
Email:{shengbo, cct, liqun, wm}@cs.wm.edu

Abstract

Data storage has become an important issue in sensor networks as a large amount of collected data needs to be archived for future information retrieval. This paper proposes to introduce storage nodes that can store data collected from the sensors in their proximities. The storage nodes alleviate the heavy load of transmitting all the data to a central place for archiving and reduce the communication cost induced by the network query. This paper considers the storage node placement problem to minimize the total power consumption for data funneling to the storage nodes and data query. We formulate it as an integer linear programming problem and present an approximation algorithm based on a rounding technique. Our simulation shows that our approximation algorithm performs well in practice.

1 Introduction

We consider a data storage system in sensor networks so that data collected by sensors could be stored in the network rather than being sent to the base station. Specifically, we consider a two tier-structure composed of storage nodes and associated normal sensors, proposed in our previous work [20]. Storage nodes are special sensors with much larger permanent storage (e.g., flash memory) and more battery power. In such a hybrid sensor network, these storage nodes collect the data from normal sensors nearby. Upon receiving a query, the storage nodes will process the query and then reply back to the sink. If needed, the data accumulated on each storage node can be transported periodically to a data warehouse by robots or traversing vehicles using physical mobility as Data Mule [19]. The basic model is shown in Fig. 1 (with 3 storage nodes), where solid lines indicate raw data transfer and dashed lines denote query replies. Since the storage nodes only collect data from the sensors in their proximity and not all of the raw data are transmitted to the sink via a hop-by-hop relay of other sensor nodes, the concerns of limited storage, communication capacity, and battery power are ameliorated.

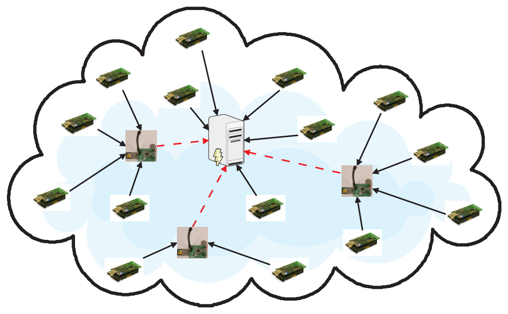

Figure 1. Access Model with Storage Nodes

Due to the higher cost of storage nodes compared to regular sensor nodes, there are usually only a limited number of storage nodes in the entire sensor network. Thus, the placement of such storage nodes is a crucial problem that affects data transmission in the whole network. Under this two-tier model,each sensor, apart from sensing data, is also involved in routing data for two network services: transmitting raw data to storage nodes, and diffusing/replying queries. In this paper, we use power consumption as the metric for evaluating our solution. Thus, we aim to minimize the total power consumption in data accumulation and data query by judicious placing of storage nodes.

In our prior work [20], we discussed the problem of placing storage nodes on a communication tree. In this paper, we consider a more general case without topology assumption. We formulate the problem as an integer programming problem and propose a 10-approximation algorithm to resolve it. Our simulation results show that our algorithm performs well even though the approximation factor is large.

The problem in this paper is quite similar to the k-median problem and the uncapacitated facility location problem (UFL), which have been well studied in literature [4,5,7–9,15] and [12–14,17,18,21]. Our approximation algorithm follows the ideas in [8], which give an approximation factor of $6\frac{2}{3}$ to the k-median problem. In our problem, however, the sink is a special facility as the final destination of all data. From another aspect, our problem is similar to the two-level facility location problem ([2, 3, 6, 23]) with the sink as the only one level-2 facility. However, in our problem, the cost triangle inequality does not always hold, which makes the problem more complicated, as a special case of the non-metric two-level facility location problem. The best known solution to the metric k-median problem has an approximation factor of $(3+\epsilon)$ ([5]). No prior work guarantees a constant approximation factor for the general UFL and 2-UFL problems. The best known solution has an approximation factor of $O(ln(C))$ ([23]), where C is the number of clients.

2 Problem Formulation

In this paper, we consider an application in which sensor networks provide real-time data services to users. A sensor network is given with one sensor identified as the sink (or base station) and each sensor generating (or collecting) data from its environment. Users specify the data they need by submitting queries to the sink and they are usually interested in the latest readings generated by the sensors[1]. There are two types of sensors (or nodes) in this hybrid network, defined as follows.

- *Storage nodes*: This type of nodes store all the data it has received from other nodes or generated by themselves. The sink only sends queries to storage nodes. According to the query description, storage nodes obtain the results needed from the raw data they are holding and then send these results back to the sink. The sink itself is considered as a storage node.

- *Forwarding nodes*: Each forwarding node is associated with a storage node. A forwarding node always forwards the data generated by itself to the associated storage node. Since forwarding nodes are not aware of queries, the forwarding operation is independent of queries and there is no data processing at these nodes.

Since storage nodes hold raw data sent from nearby forwarding nodes, it requires a large local disk space (flash memory), which makes storage nodes more expensive than normal forwarding nodes. Considering the total budget of a sensor network, we probably can afford only a limited number of storage nodes (a small fraction of all the deployed sensors). Thus, given an input parameter k, our goal is to strategically allocating at most k storage sensors in a sensor network to minimize the energy cost (power consumption) associated with raw data transfers, query diffusion, and query replies.

[1]Our algorithms also apply to the queries to the historic data. For the ease of presentation, we assume all queries are corresponding to the latest generated data.

In the deployment, we first deploy normal forwarding nodes. After collecting their location information, we select at most k of them to be storage nodes. We can attach large flash memory to these selected forwarding nodes or replace them by deploying more powerful storage nodes at the same locations. We also associate each forwarding node with a storage node which will hold the raw data from the forwarding node. We broadcast the association information to the network in the initial phase.

In this model (shown in Fig. 1), queries are only diffused to every storage node. Since we consider a very limited number of storage nodes in this paper and query message size is negligible compared to the data transmission, the query diffusion cost is ignored. Thus, in the following of this paper, energy cost includes transmission cost of the raw data and query reply cost but not query diffusion cost.

We make the following assumptions about the characteristics of data generation, query diffusion, and query reply. First, for data generation, assume that each node generates r_d readings per time unit and the data size of each reading is s_d. Second, for query rate, assume that r_q queries of the same type are submitted from users per time unit. Third, for query reply, assume that the size of data needed to reply a query is a fraction α of that of the raw data. Specifically, we define a data reduction function f for query reply. For input x, which is the size of the raw data generated by a set of nodes, function $f(x) = \alpha x$ for $\alpha \in (0, 1]$ returns the size of the processed data needed to reply the query. This characterizes many queries satisfied by a certain fraction of the all sensing data, e.g., a query may be "return all the nodes that sense a temperature higher than 100 degree". The characteristics of queries can be estimated from historic query records and analytical models.

In this paper, we consider multi-hop communication for relaying data. We assume the data routing between a pair of sensors, e.g., a normal sensor and a storage node, a storage node and the sink, follows the geographic routing algorithm [16], which looks for the shortest path connecting them. Thus, the energy cost model is simplified by the assumption that the transmission cost is proportional to the data size and the hop distance between the sender and the receiver. In a densely deployed sensor network, the hop distance between two sensors is proportional to the Euclidean distance ([10, 11, 22]). Therefore, in this paper, we use

$$\text{Euclidean distance} \times \text{Data size}$$

to measure the energy consumed to send data.

Therefore, the problem in this paper is to find the optimal placement of the storage nodes such that the energy cost associated with raw data transfer and query reply is minimized. This problem is a general case of the k-median problem[2]. Especially when there is no data transfer between storage nodes and the sink, i.e. $r_q = 0$, the problem becomes the classic k-median problem, which is NP-hard. In the following, we give an approximate algorithm for our optimal storage node placement problem.

More specifically, given L as a set of locations of sensor nodes including the sink, the problem is to select at most k sensors to be storage nodes such that the total energy cost is minimized. Assume different nodes are placed at distinct locations, L can be also regarded as the set of sensor nodes. All nodes/locations are labeled from 0 to n and node 0 is the sink. We define y_i as the type flag of node i,

$$\forall i \in L, y_i = \begin{cases} 1 & \text{if } i \text{ is a storage node;} \\ 0 & \text{if } i \text{ is a forwarding node.} \end{cases}$$

Let c_{ij} be the Euclidean distance[3] between node i and j and l_i be the Euclidean distance between node i and the sink, i.e. $l_i = c_{i0}$. We use x_{ij} as an indicator denoting if the raw data generated by node j are sent to storage node i and stored there,

$$x_{ij} = \begin{cases} 1 & \text{if } y_i = 1 \text{ and node } j \text{ forwards its raw data to } i; \\ 0 & \text{otherwise.} \end{cases}$$

Thus, our problem can be formulated as an integer program,

$$\begin{aligned} \text{IP:} \quad \min \quad & \sum_{i,j \in L} x_{ij}(c_1 c_{ij} + c_2 l_i) & \\ s.t. \quad & \forall j \in L, \sum_{i \in L} x_{ij} = 1, & (1) \\ & \sum_{i \in L} y_i \le k, & (2) \\ & \forall i, j \in L, y_i \ge x_{ij} \ge 0, y_0 = 1. & (3) \end{aligned}$$

where $c_1 = r_d s_d$ and $c_2 = r_q \alpha s_d$. In the objective, the cost incurred by a node j includes two parts. The first part ($c_1 c_{ij}$) is the cost for raw data transfer from node j to the associated storage node i. The second part ($c_2 l_i$) is the cost of sending the query reply, which is derived from the raw data generated by j, from the storage node i to the sink. The first constraint requires every sensor to send its data through a storage node. Since we treat the sink as a storage node, it includes the case that sensors send data directly to the sink. The second constraint is for the number of storage nodes, where k is given as a parameter of this problem. In the third constraint, if node j forwards data to node i, node i must be a storage node. It shows the connection between variables x and y.

Since c_1 and c_2 are constant, the objective function is equivalent to

$$\min \sum_{i,j \in L} p_{ij} x_{ij},$$

where $p_{ij} = c_{ij} + \beta l_i$ with $\beta = \frac{c_2}{c_1} = \frac{r_q \alpha}{r_d}$. We are going to use the above objective function for the IP problem from now on. Its LP-relaxation is

$$\begin{aligned} \text{LP-relaxation:} \quad \min \quad & \sum_{i,j \in L} p_{ij} x_{ij} \\ s.t. \quad & \forall j \in L, \sum_{i \in L} x_{ij} = 1, \\ & \sum_{i \in L} y_i \le k, \\ & \forall i, j \in L, y_i \ge x_{ij} \ge 0, y_0 = 1. \end{aligned}$$

Note that the difference between this LP-relaxation and the k-median problem is that p_{ij} is neither symmetric nor proportional to the Euclidean distance between i and j.

Theorem 1 *If $\beta \ge 1$, there is no need to place storage nodes.*

Proof: Assume node i is a storage node, and a node j (j may be equal to i) sends data via node i. Recall that the cost incurred by node j is $p_{ij} = c_{ij} + \beta l_i$. If j sends data directly to the sink, the cost will be l_j. According to the triangle inequality

$$l_j \le c_{ij} + l_i \le c_{ij} + \beta l_i = p_{ij}.$$

It shows that when $\beta \ge 1$, there is no benefit from transmitting data through a storage node. Thus, there is no need to deploy storage nodes. ■

In the following, we only consider the scenario with $\beta < 1$.

3 Approximation Algorithm

In this section, we describe a rounding algorithm to resolve the problem. We first modify the LP-relaxation problem to an equivalent problem by introducing a demand d_j to every node. Intuitively, d_j can be regarded as the size of the raw data generated by node j. Parameter d_j is set to 1 for each node and we keep the same constraints of the LP-relaxation problem. But the objective function of this problem becomes

$$\text{Original: } \min \sum_{i,j \in L} d_j p_{ij} x_{ij}.$$

We call this problem the original problem. Obviously, a feasible solution to the LP-relaxation is feasible to the original problem and an integer solution to the original problem is feasible to the IP problem.

[2]Definition of k-median problem ([8]): Given n points, we must select k of them to be cluster centers, and then assign each point j to the selected center that is closest to it. The goal is to minimize the sum of the distance between each node and its associated center.

[3]We use the Euclidean distance to approximate the minimal number of communication hops between two nodes, which translates to the total optimal power consumption of the nodes on the communication path between those two nodes. This approximation is valid when a large number of nodes are deployed ([10, 11, 22]).

3.1 Outline of the Algorithm

Initially, we obtain a feasible solution $(\bar{x}, \bar{y})$ to the LP-relaxation problem, which is also feasible to the original problem. Let $\bar{C}_{LP}$ be the value of the objective of the original problem. For any node $j \in L$, we use $\bar{C}_j$ to represent the cost of raw data transfer and query reply incurred by a unit data from node j in solution $(\bar{x}, \bar{y})$:

$$\bar{C}_j = \sum_{i \in L} p_{ij} \bar{x}_{ij}. \tag{4}$$

And the total cost of $(\bar{x}, \bar{y})$ in the original problem is

$$\bar{C}_{LP}(\bar{x}, \bar{y}) = \sum_{j \in L} d_j \bar{C}_j. \tag{5}$$

We use the following three steps to obtain an integer solution to the original problem.

Step 1: We modify the demand of every node by moving some nodes' demands to the others. We call this process *consolidating demands*. After this step, only some nodes hold demands while the other nodes' demands become 0. Since we keep the same constraint, $(\bar{x}, \bar{y})$ is also feasible to the modified problem. Additionally, our modification follows some rules such that an integer solution to the modified problem can be converted to an integer solution to the original problem with no more than $4\bar{C}_{LP}(\bar{x}, \bar{y})$ extra cost.

Step 2: In solution $(\bar{x}, \bar{y})$, the values of the variables are not integers. We call node i a *fractional storage node* if $\bar{y}_i \in (0, 1)$. In this step, we simplify the problem by *consolidating fractional storage nodes*, i.e. moving $\bar{y}_i$ of fractional storage nodes to other nodes. We modify $(\bar{x}, \bar{y})$ to another solution (x', y'), such that $x', y' \in [\frac{1}{2}, 1]$ and the cost of (x', y') is at most three times of the cost of $(\bar{x}, \bar{y})$. We will further modify (x', y') to another $\{\frac{1}{2}, 1\}$-integral solution (x'', y'') to the modified problem without increasing the cost.

Step 3: Finally, we apply a rounding algorithm to convert (x'', y'') to a $\{0, 1\}$-integral solution to the modified problem with at most twice the cost of (x'', y''). As we mentioned in Step 1, this integer solution can be further converted to an integer solution to the original problem.

3.2 Consolidating Demands

Originally, every node has demand of 1. In this step, we try to reallocate demands from all nodes to fewer number of nodes such that for any pair of nodes i and j with positive demands, $c_{ij} > 4\max(\bar{C}_i, \bar{C}_j)$. The following procedure is applied to modify the demands.

1. We re-index the nodes in an increasing order of $\bar{C}_j$, i.e., $\bar{C}_1 \le \bar{C}_2 \le \ldots \le \bar{C}_n$.
2. We modify the demands of nodes in the new order. Let d'_j be the new demands. Initially, $d'_j = d_j$. For a node j, we check if there is another node i satisfying $i < j$, $d_i > 0$ and $c_{ij} \le 4\max\{\bar{C}_i, \bar{C}_j\} = 4\bar{C}_j$. If there exists such a node i, we move the demand of j to node i by:

$$d'_i \leftarrow d'_i + d'_j; \qquad d'_j \leftarrow 0.$$

After this process, we get a new problem with the modified demands. This problem has the same constraints as the original problem, but the objective becomes:

$$\text{Modified: min} \sum_{i,j \in L} d'_j p_{ij} x_{ij}.$$

A node with positive demand is called a *demand node*.

In the process above, we only modify the demands, but nothing on the constraints. Thus, the feasible solution $(\bar{x}, \bar{y})$ to the original problem is also feasible to the modified problem. Let $\hat{C}_{LP}$ be the cost in the modified problem,

$$\hat{C}_{LP}(\bar{x}, \bar{y}) = \sum_{j \in L} d'_j \bar{C}_j.$$

Theorem 2 *After modifying the demands, the cost of $(\bar{x}, \bar{y})$ in the modified problem is less than that in the original problem, i.e., $\hat{C}_{LP}(\bar{x}, \bar{y}) < \bar{C}_{LP}(\bar{x}, \bar{y})$.*

Proof: To see this, assume that during the modification, we move demands from j to i with $\bar{C}_j > \bar{C}_i$. Thus, the change of the total costs is:

$$\begin{aligned} & \hat{C}_{LP}(\bar{x}, \bar{y}) - \bar{C}_{LP}(\bar{x}, \bar{y}) \\ = & (d'_i \bar{C}_i + d'_j \bar{C}_j) - (d_i \bar{C}_i + d_j \bar{C}_j) \\ = & ((d_i + d_j)\bar{C}_i + 0 \cdot \bar{C}_j) - (d_i \bar{C}_i + d_j \bar{C}_j) \\ = & d_j(\bar{C}_i - \bar{C}_j) < 0. \end{aligned}$$

■

Theorem 3 *For any feasible integer solution (x, y) to the modified problem, there is a feasible integer solution to the original problem with cost at most $4\bar{C}_{LP}(\bar{x}, \bar{y})$ more than the cost of (x, y) in the modified problem.*

Proof: Let $(x1, y1)$ be an integer solution to the modified problem. We will convert it to an integer solution $(x2, y2)$ to the original problem. First, we set $y2 = y1$. Secondly, assume node j moves its demand d_j to another node j' during the consolidating process and j' is assigned to a storage node i in the modified problem according to the integer solution $(x1, y1)$. We also assign j to the storage node i in the original problem, i.e., $x2_{ij} = x1_{ij'} = 1$, as illustrated as Fig. 2.

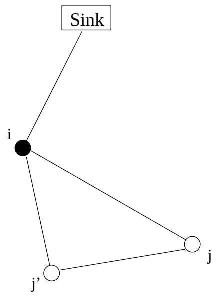

Figure 2. Black node i is a storage node and white nodes j and j' are forwarding nodes.

In the original problem, the cost of sending demand $d_j = 1$ to the sink via i is

$$p_{ij} = c_{ij} + \beta l_i.$$

Similarly, in the modified problem, the cost of sending $d_j = 1$, which is actually a part of $d'_{j'}$, is

$$p_{ij'} = c_{ij'} + \beta l_i.$$

The difference is

$$p_{ij} - p_{ij'} = c_{ij} - c_{ij'} < c_{jj'} \leq 4\bar{C}_j.$$

The first inequality ($<$) follows from the triangle inequality and the second inequality ($\leq$) follows from the rule of demand modification we mentioned earlier.

Therefore, summing up for all $j \in L$,

$$\begin{aligned} &\sum_{j \in L}(p_{ij}x2_{ij} - p_{ij'}x1_{ij'}) = \sum_{j \in L}(p_{ij} - p_{ij'}) \\ \leq\ &\sum_{j \in L} 4\bar{C}_j = 4\sum_{j \in L} d_j\bar{C}_j = 4\bar{C}_{LP}(\bar{x},\bar{y}). \end{aligned}$$

We can claim that any feasible integer solution to the modified problem can be converted to a feasible integer solution to the original problem with at most $4\bar{C}_{LP}(\bar{x},\bar{y})$ more cost. ■

3.3 Consolidating Storage Nodes

The goal of this step is to modify the values of $\bar{y}$ and obtain a new solution (x', y'), such that

$$\begin{aligned} y'_i = 0, &\quad \text{if } d'_i = 0; \\ y'_i \geq \tfrac{1}{2}, &\quad \text{if } d'_i > 0. \end{aligned}$$

For each node j, recall $\bar{C}_j = \sum_{i \in L} p_{ij}\bar{x}_{ij}$. we have

$$\bar{C}_j \geq \sum_{p_{ij} > 2\bar{C}_j} p_{ij}\bar{x}_{ij} > \sum_{p_{ij} > 2\bar{C}_j} 2\bar{C}_j\bar{x}_{ij} \Leftrightarrow \sum_{p_{ij} > 2\bar{C}_j} \bar{x}_{ij} < \frac{1}{2}.$$

Since $\sum_i \bar{x}_{ij} = 1$ and $\bar{x}_{ij} \leq \bar{y}_i$,

$$\sum_{p_{ij} \leq 2\bar{C}_j} \bar{y}_i \geq \sum_{p_{ij} \leq 2\bar{C}_j} \bar{x}_{ij} = 1 - \sum_{p_{ij} > 2\bar{C}_j} \bar{x}_{ij} > \frac{1}{2}.$$

Additionally, because $p_{ij} > c_{ij}$, we have

$$\sum_{c_{ij} \leq 2\bar{C}_j} \bar{y}_i > \sum_{p_{ij} \leq 2\bar{C}_j} \bar{y}_i > \frac{1}{2}.$$

Starting with $x' = \bar{x}$ and $y' = \bar{y}$, we modify $(\bar{x}, \bar{y})$ to (x', y') as follows: For each fractional storage node i, i.e. $1 > y'_i > 0$, if $d'_i = 0$,

1. We will move the value of y'_i to the closest demand node j,

$$y'_j \leftarrow \min(1, y'_j + y'_i); \qquad y'_i \leftarrow 0.$$

2. Also, we need move the forwarding nodes assignments, for each $j' \in L$

$$x'_{jj'} \leftarrow x'_{jj'} + x'_{ij'}; \qquad x'_{ij'} \leftarrow 0.$$

After these changes, we obtain a new solution (x', y') to the modified problem and we can prove the following lemmas.

Theorem 4 $\hat{C}_{LP}(x', y') \leq 3\hat{C}_{LP}(\bar{x}, \bar{y})$.

Proof: Consider that a fractional storage node i has moved its $\bar{y}_i$ to node j during the modification, as shown in Fig. 3. For any demand node j', the previous association $\bar{x}_{ij'}$ is also transferred to j. Since j is the closest demand node to i, $c_{ij} \leq c_{ij'}$. Recall $p_{jj'} = c_{jj'} + \beta l_j$, from the triangle inequality,

$$c_{jj'} < c_{ij} + c_{ij'} \leq 2c_{ij'}.$$

For the second term of $p_{jj'}$,

$$\beta l_j < \beta l_i + \beta c_{ij} < \beta l_i + c_{ij} \leq \beta l_i + c_{ij'} = p_{ij'}.$$

Therefore,

$$p_{jj'} < 2c_{ij'} + p_{ij'} \leq 2p_{ij'} + p_{ij'} = 3p_{ij'}.$$

Considering all the modified fractional storage nodes, e.g., $\bar{y}_{i_1}, \bar{y}_{i_2}, \ldots$ are moved to y'_j,

$$\begin{aligned} \hat{C}_{LP}(x', y') &= \sum_{j,j' \in L} d'_{j'} p_{jj'} x'_{jj'} \\ &= \sum_{j,j' \in L} d_{j'} p_{jj'}(\bar{x}_{jj'} + \bar{x}_{i_1 j'} + \bar{x}_{i_2 j'} + \cdots) \\ &< \sum_{j,j' \in L} d_{j'}(p_{jj'}\bar{x}_{jj'} + 3p_{i_1 j'}\bar{x}_{i_1 j'} + 3p_{i_2 j'}\bar{x}_{i_2 j'} + \cdots) \\ &< 3\sum_{j,j' \in L} d_{j'} p_{jj'}\bar{x}_{jj'} = 3\hat{C}_{LP}(\bar{x}, \bar{y}). \end{aligned}$$

Therefore, the cost $\hat{C}_{LP}(x', y')$ is at most triple of $\hat{C}_{LP}(\bar{x}, \bar{y})$. ■

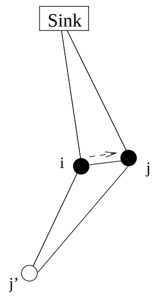

Figure 3. Fractional storage node i moves it $\bar{y}_i$ to y'_j and white node j' is a forwarding node.

Lemma 1 *For a demand node j, any node i satisfying $c_{ij} \le 2\bar{C}_j$ will move its value of $\bar{y}_i$ to y'_j.*

Proof: First, $\forall i$, if $c_{ij} \le 2\bar{C}_j$, the demand of i is 0. Recall the first step, we guarantee for any two remaining demand nodes, $c_{ij} > 4\max(\bar{C}_j, \bar{C}_i)$.

Next, we prove that all these nodes will move their fractions to node j. Assume there exists one node i with $c_{ij} \le 2\bar{C}_j$ moves its $\bar{y}_i$ to another demand node j', which implies $c_{ij'} < c_{ij}$. According to the triangle inequality,

$$c_{jj'} < c_{ij'} + c_{ij} < 2c_{ij} \le 4\bar{C}_j.$$

It is a contradiction with the requirements of demand nodes. Thus, after modifying $(\bar{x}, \bar{y})$ to (x', y'), all the nodes within distance of $2\bar{C}_j$ to node j will move their values of $\bar{y}$ to y'_j. ■

As we mentioned earlier in this section,

$$\sum_{c_{ij} \le 2\bar{C}_j} \bar{y}_i \ge \frac{1}{2}.$$

Hence, after the modification, we get

$$y'_i \ge \frac{1}{2}, \text{ if } d'_i > 0.$$

Next, we will modify (x', y') to another feasible solution (x'', y'') subject to $x'', y'' \in \{\frac{1}{2}, 1\}$, and the cost of (x'', y'') is no more than the cost of (x', y'). The condition that $y', y'' \ge \frac{1}{2}$ implies that there are at most $2k$ nodes with positive demands in both solutions (x', y') and (x'', y''). Initially, we assign $x'' = x'$ and $y'' = y'$. For each i with positive demand, the best choice is to send data through itself. To get the minimum cost, we should assign

$$x''_{ii} = y''_i, \text{ if } d'_i > 0.$$

The remaining $(1 - y''_i)$ fraction should be assigned to another demand node i', where $p_{i'i}$ is the minimum among all demand nodes. Let $s(i)$ denote such node i'. The minimum cost is

$$\begin{aligned} & \sum_{d'_i > 0} d'_i (p_{ii} y''_i + p_{s(i)i}(1 - y''_i)) \\ = & \sum_{d'_i > 0} d'_i (\beta l_i y''_i + p_{s(i)i} - p_{s(i)i} y''_i) \\ = & \sum_{d'_i > 0} d'_i p_{s(i)i} - \sum_{d'_i > 0} d'_i y''_i (p_{s(i)i} - \beta l_i), \end{aligned} \qquad (6)$$

where

$$p_{s(i)i} = c_{s(i)i} + \beta l_{s_i} > \beta(c_{s(i)i} + l_{s(i)}) > \beta l_i.$$

So far, we only modify x'', but y'' is still equal to y'. Since formula (6) only depends on y'', we can use $f(y'')$ to represent it.

Next, we will show that under the constraint $\frac{1}{2} \le y''_i \le 1$, we can obtain a $\{\frac{1}{2}, 1\}$-integral solution y'' such that $f(y'')$ is the minimum. The first term of Eq. (6) is a constant independent of y''. To minimize the cost, we should maximize y''_i for the nodes with largest values of $d'_i(p_{s(i)i} - \beta l_i)$. Let n' be the number of demand nodes, as we know, $n' < 2k$. We reorder demand nodes according to $d'_i(p_{s(i)i} - c_2 l_i)$ decreasingly. We set $y''_i = 1$ for the first $2k - n'$ nodes and $y''_i = \frac{1}{2}$ for the remaining $2(n' - k)$. It is actually a greedy algorithm to maximum the second term of Eq. (6). Thus,

$$f(y'') \le f(y') \le \hat{C}_{LP}(x', y').$$

Accordingly, x'' is also a $\{\frac{1}{2}, 1\}$-integral solution. For a demand node i, if $y''_i = 1$,

$$x''_{ji} = \begin{cases} y''_i = 1 & \text{if } j = i; \\ 0 & \text{otherwise.} \end{cases}$$

Otherwise, if $y''_i = \frac{1}{2}$,

$$x''_{ji} = \begin{cases} y''_i = \frac{1}{2} & \text{if } j = i; \\ 1 - y''_i = \frac{1}{2} & \text{if } j = s(i); \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 5 $\hat{C}_{LP}(x'', y'') \le \hat{C}_{LP}(x', y')$.

Proof: It is obvious because (x'', y'') yields the minimum value of the cost function f. ■

3.4 Rounding

Finally, we apply a rounding algorithm to get a $\{0, 1\}$ integer solution. First, we place a storage node at node j if $y''_j = 1$. For the remaining nodes with $y''_i = \frac{1}{2}$, half data is sent via $s(i)$. Consider a directed graph G consisting of the remaining demand nodes, where each edge is from i to $s(i)$.

Lemma 2 *There is no loop of length more than 2 in G.*

Proof: Assume there is a loop in G involving nodes $n_1, n_2, \cdots, n_m$, where $m > 2$ and $\forall t \leq m$ there is a directed edge from n_t to $n_{(t \bmod m)+1}$. For each node n_t, $s(n_t) = n_{(t \bmod m)+1}$. According to the definition of $s(n_t)$ that $p_{n_t s(n_t)}$ is the minimum, we have

$$\begin{aligned} p_{n_2 n_1} &< p_{n_m n_1} \\ p_{n_3 n_2} &< p_{n_1 n_2} \\ &\cdots \\ p_{n_1 n_m} &< p_{n_{m-1} n_m}. \end{aligned}$$

Recall $p_{ij} = c_{ij} + \beta l_i$, the conditions above become

$$\begin{aligned} c_{n_2 n_1} + \beta l_{n_2} &< c_{n_m n_1} + \beta l_{n_m} \\ c_{n_3 n_2} + \beta l_{n_3} &< c_{n_1 n_2} + \beta l_{n_1} \\ &\cdots \\ c_{n_1 n_m} + \beta l_{n_1} &< c_{n_{m-1} n_m} + \beta l_{n_{m-1}}. \end{aligned}$$

Thus, the summation of the left side should be less than the summation of the right side. We find, however, that the summation of both sides are equal. This contradiction means that the series of conditions can not be held at a same time. ■

Furthermore, if there are two edges between two nodes, i.e $s(i) = j$ and $s(j) = i$, we arbitrarily choose one of them as a root and eliminate the directed edge from the root to the other node. Finally, G becomes a forest graph, which consists of multiple rooted trees. Additionally, we assign every node a level value, which is the distance to the root of the tree that it belongs to. We can divide these nodes into two sets based on odd and even level values and select the smaller set of nodes to be storage nodes. Totally, $\{i|y''_i = \frac{1}{2}\}$ has $2(n' - k)$ nodes. Thus, we place at most $n' - k$ storage nodes at this step. Plus the storage nodes set earlier in $\{i|y''_i = 1\}$, which has $2k - n'$ nodes, the total number of storage nodes is at most $\sum y''_i \leq k$. In addition, each unselected node i in the tree will associate itself with $s(i)$, which must be a storage node, i.e., $(x''_{s(i)i} = 1)$. Finally, we get an integer solution of the modified problem from a feasible solution $(\bar{x}, \bar{y})$.

Theorem 6 *After rounding, the cost of the integer solution is no more than double the cost of (x'', y'').*

Proof: In the routing process above, for j with $y''_j = \frac{1}{2}$, the previous cost is $\frac{1}{2}\beta l_j + \frac{1}{2} p_{s(j)j}$ and after rounding, it becomes $p_{s(j)j}$ or βl_j. Thus, the cost is at most doubled. ■

Let $\hat{C}_{INT}$ be the cost of this integer solution in the modified problem. Based on the previous theorems,

$$\begin{aligned} \hat{C}_{INT} &\leq 2\hat{C}_{LP}(x'', y'')\text{(Theorem 6)} \\ &\leq 2\hat{C}_{LP}(x', y')\text{(Theorem 5)} \\ &\leq 6\hat{C}_{LP}(\bar{x}, \bar{y})\text{(Theorem 4)}. \end{aligned}$$

As we mentioned in Theorem 3, we can derive an integer solution to the original problem from an integer solution to the modified problem. Let $\bar{C}_{INT}$ denote the cost of this integer solution in the original problem,

$$\begin{aligned} \bar{C}_{INT} &\leq \hat{C}_{INT} + 4\bar{C}_{LP}(\bar{x}, \bar{y}) \\ &\leq 6\hat{C}_{LP}(\bar{x}, \bar{y}) + 4\bar{C}_{LP}(\bar{x}, \bar{y}) \\ &\leq 6\bar{C}_{LP}(\bar{x}, \bar{y}) + 4\bar{C}_{LP}(\bar{x}, \bar{y})\text{(Theorem 2)} \\ &= 10\bar{C}_{LP}(\bar{x}, \bar{y}). \end{aligned}$$

Since $(\bar{x}, \bar{y})$ is the optimal fractional solution, the cost of $(\bar{x}, \bar{y})$ must be no more than the cost of the optimal integer solution. Therefore, combining three steps together, we get a 10-approximation($3 \times 1 \times 2 + 4$) algorithm for this problem.

4 Performance Evaluation

We have implemented the approximation algorithm and compared the performance of the algorithm with the optimal solution. We consider a network composed of 100 sensor nodes randomly deployed in a 100×100 square field, where the sink is in the center. We vary the number of storage nodes k (including the sink) from 2 to 15 with β taking $0.1, 0.15$, and 0.2 respectively. In our approximation algorithm implementation, we use GLPK package (GNU Linear Programming Kit [1]) to get the fractional solution in the first step of our algorithm. The optimal solution is done by using integer linear programming, which is provided by MIP (mixed integer program).

Fig. 4 shows the simulation results when the parameter β is set to 0.1, 0.15 and 0.2. We first calculate a maximum cost C_{max}, which is the energy cost when there is no storage node and every sensor sends data directly to the sink. The performance shown in the figures is the ratio over C_{max}. From the figures, we observe that our approximation algorithm achieves the optimal performance when the number of storage nodes is small, which is a valid assumption since a storage node is expected to be in charge of tens of regular sensor nodes. When the number of storage nodes becomes larger, the disparity between the optimal solution and our approximation algorithm gets bigger. Even though the approximation algorithm proposed in the paper has a high approximation factor, our simulation shows that in practice, the algorithm performs well when the number of storage nodes is small.

5 Conclusion

This paper considers the storage node placement problem in a sensor network. Introducing storage nodes into the sensor network alleviates the communication burden of sending all the raw data to a central place for data archiving and facilitates the data collection by transporting data from

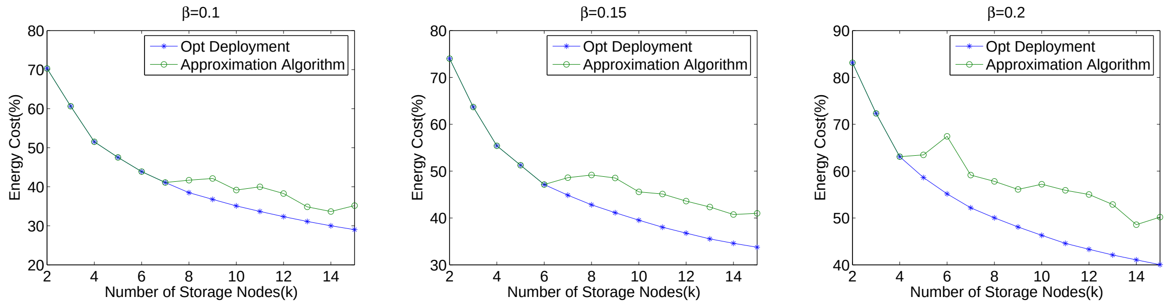

Figure 4. Select k storage nodes from 100 randomly deployed sensors and $\beta = 0.1, 0.15, 0.2$.

limited number of storage nodes. In this paper, we examine how to place storage nodes to save energy for data collection and data query. We formulate the problem as an integer linear programming problem and propose a 10-approximation rounding algorithm. We also implement the algorithm and conduct simulation on different network parameters. Our simulation shows that the performance of our approximation algorithm is very close to optimal when the number of storage nodes is small. Our future work includes how to optimize query reply in a sensor network and how to solve the storage node placement problem in terms of other performance metrics.

Acknowledgment

This project was supported in part by US National Science Foundation award CCF-0514985.

References

[1] GLPK (GNU Linear Programming Kit), available [online] http://www.gnu.org/software/glpk/glpk.html.
[2] K. Aardal, F. A. Chudak, and D. B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Inf. Process. Lett.*, 72(5-6):161–167, 1999.
[3] A. A. Ageev. Improved approximation algorithms for multilevel facility location problems. In *APPROX '02*.
[4] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC '98*.
[5] V. Arya, N. Garg, R. Khandekar, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *STOC '01*.
[6] A. Bumb and W. Kern. A simple dual ascent algorithm for the multilevel facility location problem. In *APPROX '01/RANDOM '01*.
[7] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *FOCS '99*.
[8] M. Charikar, S. Guha, Éva Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *STOC '99*.
[9] J. Chuzhoy and Y. Rabani. Approximating k-median with non-uniform capacities. In *SODA '05*.
[10] S. De. On hop count and euclidean distance in greedy forwarding in wireless ad hoc networks. *IEEE Communication Letters*, 9, 2005.
[11] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *INFOCOM '05*.
[12] A. D. Flaxman, A. M. Frieze, and J. C. Vera. On the average case performance of some greedy approximation algorithms for the uncapacitated facility location problem. In *STOC '05*.
[13] S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. In *SODA '98*.
[14] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *STOC '02*.
[15] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.
[16] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00*.
[17] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *SODA '98*.
[18] D. Krivitski, A. Schuster, and R. Wolff. A local facility location algorithm for sensor networks. In *DCOSS '05*.
[19] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *First IEEE International Workshop on Sensor Network Protocols and Applications (SPNA)*.
[20] B. Sheng, Q. Li, and W. Mao. Data storage placement in sensor networks. In *MobiHoc '06*.
[21] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *STOC '97*.
[22] S. Vural and E. Ekici. Analysis of hop-distance relationship in spatially random sensor networks. In *MobiHoc '05*.
[23] J. Zhang. Approximating the two-level facility location problem via a quasi-greedy approach. In *SODA '04*.