Optimal placement of storage nodes in a wireless sensor network

Gianlorenzo D'Angelo[1], Daniele Diodati[2], Alfredo Navarra[2], and Cristina M. Pinotti[2]

[1] Gran Sasso Science Institute (GSSI), L'Aquila, Italy.
gianlorenzo.dangelo@gssi.infn.it
[2] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy.
daniele.diodati@dmi.unipg.it; alfredo.navarra@unipg.it; cristina.pinotti@unipg.it

Networks of sensor nodes are usually employed to monitor large areas, collecting data with regular frequency. This large volume of data has to be stored somewhere for answering to external user queries [3]. There are usually two main ways to store data. Source nodes, which are responsible for collecting data, can either locally store the data or transmit them to the *sink*, a powerful node connected to the external world. Both solutions present some disadvantages. If data are locally stored, several problems may arise: (i) data cannot be accumulated for long periods because nodes are equipped with only limited memory space; (ii) stored data are lost once the energy of a source node – battery operated – is depleted; and (iii) searching data for serving query demand results in network-wide communications. Alternatively, source nodes can forward the collected data to the sink. However, communicating data from the source nodes up to the sink makes the network congested, especially if data are transmitted *raw*, that is, uncompressed. Limitations to the number of packets a sensor can transmit to the sink per time unit must be also considered [2].

Recently [9, 10], a hybrid solution has been proposed which makes use of a limited number of "special" sensors, more powerful than standard ones in terms of storage, energy, and computational capabilities. Under this model, source nodes may forward their raw data to such special nodes, referred to as *storage nodes*. Here, raw data are stored and *compressed*, i.e., reduced in size, to be transmitted to the sink at the time a query demand from external users is submitted. With this two-tier model, if the number of storage nodes is kept limited, the network becomes less congested at the price of a moderate increase of the sensor cost of the network. Indeed, the integration of storage nodes in the tiered architecture for sensor networks is made possible by the new storage-enriched hardware [6, 11] considered to be very practical [5]. The introduction of the storage nodes helps to alleviate the transmission bandwidth problem by distributing the local data transmission to the storage nodes. This hierarchical structure has been instantiated by the popular stargate device [11] and the memory-enhanced sensor nodes by UC Riverside [6]. Those special powerful nodes take advantage of their high transmission, storage and even computational capabilities to alleviate the bandwidth limitation, and also provide auxiliary support for surrounding vulnerable sensors for data back-up. In [9, 10] the problem of selecting a subset

of storage nodes so as the overall communication cost is minimized is called *optimal storage placement* problem. When the number of storage nodes is limited by an integer k, we talk about the *minimum k-storage problem*, which is formally stated in the next paragraph.

Problem statement. Let $G = (V, E)$ be a connected directed graph of n nodes representing a sensor network. Each node $v \in V$ generates raw data of size $s(v)$. Arcs of the network have different weights. The energy cost propagation of a message over the arc $(u, v) \in E$ is denoted by $w(u, v)$. When a communication link is bidirectional, $w(u, v) = w(v, u)$. Let $d(u, v)$ be the minimum energy cost for propagating a message from u to v which is given by the shortest path distance from u to v in G. Each $v \in V$ can be set to serve as *storage* node. A solution is a set $S \subseteq V$ of storage nodes such that $|S| \leq k$, for some $k \in \mathbb{N}$. External users retrieve data from a special storage node $r \in S$, named *sink*. Each node v in V is associated to a storage node in S, denoted as $\sigma(v, S)$. Clearly, if $v \in S$, then $\sigma(v, S) = v$. For replying to a query, a storage node compresses and sends to r the last data generated from its associated nodes. The compressed size of the data produced by a node v becomes $\alpha s(v)$, with $\alpha \in [0, 1]$. The compressed data cannot be further compressed if they reach another storage node. A node $v \in V$ is associated to the storage node $s \in S$ that minimizes $s(v)d(v, s) + \alpha s(v)d(s, r)$, ties are arbitrarily broken. The total cost for a set S of storage nodes is given by: $cost(S) = \sum_{v \in V} s(v) \left(d(v, \sigma(v, S)) + \alpha d(\sigma(v, S), r)\right)$. The *minimum k-storage* problem (briefly, *MSP*) consists in finding a subset $S \subseteq V$, with $|S| \leq k$ that minimizes $cost(S)$.

Related Work. There has been a lot of prior research on data collection in sensor networks. Initially, no in-network storage was considered: the request for data was routed from the sink to every sensor by flooding messages. The data were sent to the sink by following the same path but in the reverse direction [4]. Recently, a two-tier model has been proposed [10] to ameliorate the problem of communication congestion. The authors formulate the problem as an integer programming problem and propose a 10-approximation rounding algorithm. Differently from us, they assume that (i) raw data have size independent from the source nodes; and (ii) the energy spent for transmitting one unit of data between any pair of sensors is proportional to their Euclidean distance. For us, instead, different source nodes may generate data of different size, since sensors can monitor different environment aspects. Moreover, we assume that communications follow an underlying network represented by a graph. Each edge of the graph has its own weight that measures the energy required to traverse it. In [9], the problem is solved assuming that the communication network topology is a directed tree T, rooted at the sink. The arcs are directed towards the sink to collect the data, and they are directed away from the sink to broadcast the query. When a sensor s sends one unit data upwards to the sink the energy cost is fixed, while when a sensor s sends one unit data downwards, the energy cost can be high because it is proportional to the number of children of s in T. In this

paper and in [10], instead, storage nodes simply send query replies in a proactive manner with a predefined query frequency and hence the query cost is null.

Our results. In the following we summarize our results. In detail we first focus on the case of directed graphs and then in that of undirected ones.

Directed graphs. First, we focus on the approximation properties of the problem and we show that indeed the problem is not in APX, that is, we cannot devise a polynomial time algorithm with a constant factor approximation guarantee. This is stated in the next theorem.

Theorem 1. *Unless $P = NP$, MSP in directed graphs does not belong to APX.*

The above theorem implies that it is not possible to find any practical approximation guarantee for the general case. Therefore, we focus on a restricted case where the topology is a tree rooted at the sink and all the arcs are directed towards the sink. In this case, we show that *MSP* can be optimally solved by a dynamic programming algorithm in $O(\min\{kn^2, k^2P\})$ time, where P is the *path length* of the tree [8].[3] We observe that for a balanced binary tree $P = \Theta(n \log n)$, for random general trees $P = \Theta(n\sqrt{n})$, and in the worst case $P = O(n^2)$.

Undirected graphs. The proof of Theorem 1 is strictly based on the fact that the links of the network are unidirectional and does not hold if all the links of the network are bidirectional, that is the underlying graph is undirected. This is a realistic assumption as links of a sensor network are usually bidirectional.

As the minimum k-storage problem for undirected graphs is similar to the well-known metric k-median problem [1], then it is easy to show that also in this case the problem is *NP*-complete.[4] However, we are able to show that for graphs with bounded treewidth [7], the problem is optimally solvable in polynomial time. We note that this result also holds for the metric k-median problem which is interesting by itself.

Theorem 2. *Given a tree-decomposition of size w, there exists an algorithm that optimally solves MSP in $O(w \cdot k \cdot n^{w+3})$ time.*

Notice that the above theorem does not prove that *MSP* is fixed parameter tractable and we leave such proof (or disproof) as an open problem.

We then characterize the minimum k-storage problem on undirected graphs from the approximation viewpoint. To this aim, we first prove that it is *NP*-hard to approximate the metric k-median problem within a factor of $1 + \frac{1}{e}$, and then we extend such a bound to the minimum k-storage problem by means of a polynomial time reduction that preserves approximation.

[3] The path length of a tree is the sum of the lengths over all nodes of the paths from the root to each node.

[4] The metric k-median problem is defined as follows: Given a complete graph $G = (V, E)$, a metric distance function $dist : V \times V \to \mathbb{N}$, and an integer k, find a set $V' \subseteq V$ such that $|V'| \leq k$ and $\sum_{u \in V} \min_{v \in V'} dist(u, v)$ is minimized.

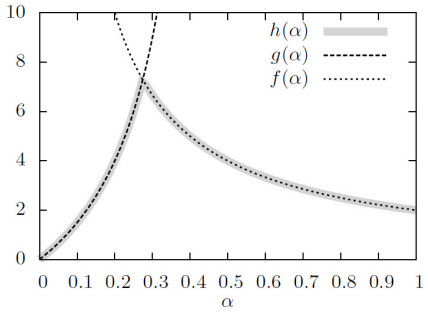

Fig. 1a. The two upper bound functions to the locality gap.

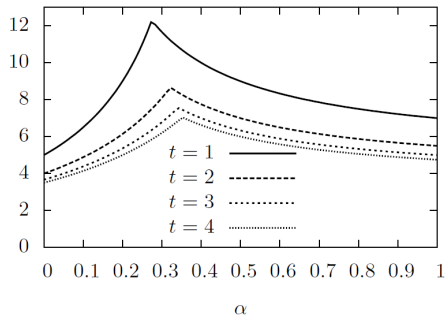

Fig. 1b. Function h'.

Theorem 3. *It is NP-hard to approximate the metric k-median problem within a factor* $\gamma < 1 + \frac{1}{e}$.

Corollary 1. *It is NP-hard to approximate MSP within a factor* $\gamma < 1 + \frac{1}{e}$.

According to Theorem 3, we propose a local search algorithm which guarantees a constant approximation ratio greater than $1+1/e$. In detail, the algorithm is denoted by $\mathcal{L}$ and defined as follows. Each solution is specified by a subset $S \subseteq V$ of exactly k nodes. To move from one feasible solution S to a neighboring one S', we define a *swap* operation between two nodes $s \in S$ and $s' \in V \setminus S$ which consists in adding s' and removing s, that is $S' = S \cup \{s'\} \setminus \{s\}$. In $\mathcal{L}$, we repeatedly check whether any swap move yields a solution of lower cost. In the affirmative case, we apply to the current solution any swap move that improves the solution cost and the resulting solution is set to be the new current solution. This is repeated until, from the current solution, no swap operation decreases the cost, that is, the current solution represents a local optimum. To give a bound on the locality gap, let us define the following three functions: $f : (0,1] \to \mathbb{R}$, $f(\alpha) = 2/\alpha$; $g : [0, \frac{1}{2}) \to \mathbb{R}$, $g(\alpha) = \frac{12\alpha}{1-2\alpha}$; $h : [0,1] \to \mathbb{R}$,

$$h(\alpha) = \begin{cases} g(\alpha) & \text{if } \alpha = 0 \\ \min\{f(\alpha), g(\alpha)\} & \text{if } \alpha \in (0, \frac{1}{2}) \\ f(\alpha) & \text{if } \alpha \in [\frac{1}{2}, 1]. \end{cases}$$

Theorem 4. *The local search algorithm* $\mathcal{L}$ *for MSP with compression ratio* $\alpha \in [0,1]$ *exhibits a locality gap of at most* $5 + h(\alpha)$.

Theorem 4 provides two upper bounds to the locality gap given by $5 + f(\alpha)$ and $5+g(\alpha)$. Functions f, g, and h are plotted in Fig. 1a. Function f is monotonic decreasing, while g is monotonic increasing, in their intervals of definition. We have that $f(\alpha) = g(\alpha)$ for $\alpha = \frac{1}{6}(\sqrt{7} - 1) \approx 0.274$ where $f(\alpha) = g(\alpha) < 7.3$. For all the other values of α, one of the two functions is always below such a threshold, that is the approximation ratio is always below 12.3.

Actually, algorithm $\mathcal{L}$ is not yet an approximation algorithm, as the number of iterations needed to find a local optimum solution might be superpolynomial. To fix this problem, as in [1], we can change the stopping condition of $\mathcal{L}$ so it finishes as soon as it finds an approximate local optimum solution, i.e., when the solution S is such that every neighboring solution S' of S has $cost(S') > (1-\epsilon)cost(S)$, for some $\epsilon \in (0,1)$. This leads to the next corollary.

Corollary 2. *There exists an* $\frac{1}{1-\epsilon}(5+h(\alpha))$*-approximation algorithm for MSP for any* $\epsilon \in (0,1)$.

Finally, by following the arguments in [1], the algorithm can be improved by allowing t simultaneous swaps. This leads to a locality gap of $h'(\alpha)$, where

$$h' : [0,1] \to \mathbb{R},\ h'(\alpha) = \begin{cases} g'(\alpha) & \text{if } \alpha = 0 \\ \min\{f'(\alpha), g'(\alpha)\} & \text{if } \alpha \in (0, \frac{t}{t+1}) \\ f'(\alpha) & \text{if } \alpha \in [\frac{t}{t+1}, 1] \end{cases};$$

$f' : (0,1] \to \mathbb{R},\ f'(\alpha) = 1 + \frac{t+1}{t}\frac{1+2\alpha}{\alpha}$; $g' : [0, \frac{t}{t+1}) \to \mathbb{R},\ g' = \frac{(3+\alpha)t+2+\alpha}{(1-\alpha)t-\alpha}$.

Function h' is plotted in Fig. 1b for $t = 1, 2, 3, 4$. To give an idea on the improvement provided by this method, we computed the maximum value of the upper bounds on the approximation ratio for $t = 2, 3, 4$, which is less than 8.67, 7.78 and 7.05, respectively. It follows that for $t \geq 2$ and any value of α, our algorithm improves over the 10-approximation algorithm provided in [10].

References

1. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
2. E. Duarte-Melo and M. Liu. Data-gathering wireless sensor networks: Organization and capacity. *Computer Networks (COMNET)*, 43(4):519–537, November 2003.
3. J. Gehrke and S. Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 3(1):46–55, 2004.
4. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of ACM SIGMOD/PODS Conference*, pages 491–502, 2003.
5. J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler. The tenet architecture for tiered sensor networks. *ACM Transactions on Sensor Networks*, 6(4):34:1–34:44, 2010.
6. Rise project. http://www.cs.ucr.edu/~rise, 2014.
7. N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.
8. R. Sedgewick and P. Flajolet. *An introduction to the analysis of algorithms.* Addison-Wesley, 1996.
9. B. Sheng, Q. Li, and W. Mao. Optimize storage placement in sensor networks. *IEEE Transactions on Mobile Computing*, 9(10):1437–1450, 2010.
10. B. Sheng, C. C. Tan, Q. Li, and W. Mao. An approximation algorithm for data storage placement in sensor networks. In *Proceedings of the 2nd International Conference on Wireless Algorithms, Systems and Applications (WASA)*, pages 71–78. IEEE, 2007.
11. Stargate gateway (spb400). http://www.xbow.com, 2014.