Data Collection with Multiple Sinks in Wireless Sensor Networks

Sixia Chen, Matthew Coolbeth, Hieu Dinh, Yoo-Ah Kim, and Bing Wang*

Computer Science & Engineering Department, University of Connecticut

Abstract. In this paper, we consider Multiple-Sink Data Collection Problem in wireless sensor networks, where a large amount of data from sensor nodes need to be transmitted to one of multiple sinks. We design an approximation algorithm to minimize the latency of data collection schedule and show that it gives a constant-factor performance guarantee. We also present a heuristic algorithm based on breadth first search for this problem. Using simulation, we evaluate the performance of these two algorithms, and show that the approximation algorithm outperforms the heuristic up to 60%.

1 Introduction

A wireless sensor network, which consists of an ad hoc group of small sensor nodes communicating with one another, have a variety of applications such as traffic monitoring, emergency medical care, battlefield surveillance, and underwater surveillance. In such applications, it is often necessary to collect the data accumulated by each sensor node for processing. We call the operation of transmitting accumulated data from sensor nodes to the sinks *data collection*. In this paper, we consider the problem to find a schedule to quickly collect a large amount of data from sensor nodes to sinks. Data need to be collected without merging. If the data can be merged, the operation is called data aggregation.

One of the challenges in data collection in wireless networks is radio interferences that may prevent nearby sensor nodes from transmitting packets simultaneously. Scheduling data transmissions without carefully considering such inferences can result in significant delay in data collection.

When there is only one gateway (or sink) to collect data and the amount of data at all sensor nodes is similar, a greedy algorithm may work well. For example, there is a 3-approximation algorithm for general graphs and optimal algorithms for simpler graphs in this case [5]. When there are multiple gateways and the distribution of the data is not uniform, the problem becomes more challenging as the amount of data to be sent to each gateway need to be balanced to avoid congestion.

In this paper, we consider the problem of minimizing the latency of data collection in settings where (i) data cannot be merged, (ii) there are multiple sinks, and (iii) the amount of data accumulated at sensor nodes as well as the capacity of links may not be uniform. We present an approximation algorithm to find a schedule with a constant-factor guarantee. We evaluate the performance

* Y. Kim and B. Wang were partially supported by UConn Faculty Large Grant and NSF CAREER award 0746841, respectively.

of this algorithm as well as a greedy heuristic algorithm using simulation in various settings. Our simulation results show that the approximation algorithm outperforms the naive heuristic up to 60%.

The rest of the paper is organized as follows: We present related work and problem formulation in Sections 2 and 3, respectively. In Section 4, we introduce an LP-based approximation algorithm and show that it gives a constant-factor performance guarantee. We then present a greedy breadth-first-search (BFS) based heuristic algorithm in Section 5, and the experimental results in Section 6. Conclusion is presented in Section 7.

2 Related Work

Data broadcast and collection are among the most fundamental operations in many communication networks. Algorithms for broadcasting in wireless networks, in which the objective is to transmit data from one source to all the other nodes, have been extensively studied in the literature [1, 3, 7, 13, 12, 8]. Data collection and aggregation problem in wireless networks, on the other hand, are relatively new but have recently received much attention [14, 15, 11, 18, 5].

Florens and McEliece [5] studied data distribution and collection problems with a single source/sink. They present data distribution/collection schedule for several types of graphs, and show that it provides 3-approximation for general graphs and optimal solution for simpler graphs such as trees. Their algorithm (and the analysis) cannot be easily extended to the case of multiple sinks. In addition, they assume that all the links have uniform capacity, i.e., one packet can be transmitted over a link at each time unit.

Another problem closely related data collection is the data aggregation problem (also called convergecast), in which gathered data can be merged by taking the maximum or minimum for example. Huang et al. [14] designed a nearly constant approximation algorithm for minimum latency data aggregation problem.

Cheng et al. [4] considered large-scale data collection problem in wired networks, in which a large amount of data need to be collected from multiple hosts to a single destination. They present coordinated data collection algorithm using a time-expanded graph to minimizes the latency. Unlike our problem, radio interference is not an issue in their problem as they consider data collection in wired networks.

3 Problem Formulation

3.1 Network Model

A wireless sensor network can be modeled as a graph $G = (V, E)$ in which each node in V represents a sensor node and each link, $e \in E$, represents a wireless communication link between two nodes. The capacity of a link, e, specifies the amount of data that can be transmitted over e during a unit time. For any two network nodes, $u, v \in V$, (u, v) and $C(u, v)$ respectively denote the link from u to v and its capacity. Finally, every link $(u, v) \in E$ has an *interference set*, denoted as $I(u, v)$, which represents the set of links in E that may not be used concurrently with (u, v) due to interference. Interference has been modeled in a variety of ways in the literature, including *protocol model* [9], *transmitter model (Tx-model)* [19], *transmitter-receiver model (Tx-Rx model)* [2, 16], and so on.

Our algorithm is applicable to any of those models by appropriately defining interference sets $I(u, v)$ according to the model.

3.2 Data Collection Problem

We consider the Multiple-Sink Data Collection Problem (MSDC), in which there are multiple sinks and a large amount of data from sensor nodes need to be transmitted to one of the sinks. Formally, we are given a wireless directed network graph, $G = (V, E)$. Each link (v_i, v_j) in the graph from node $v_i \in V$ to $v_j \in V$ is associated with a capacity, $C(v_i, v_j)$, and an interference set, $I(v_i, v_j) = \{(v_{i'}, v_{j'}) \in E \mid \text{link } (v_{i'}, v_{j'}) \text{ interferes with link } (v_i, v_j)\}$. A link cannot be used when another link in its interference set is activated. We have a set of sinks $D \subset V$. For node $v_i \in V \setminus D$, let s_i denote the amount of accumulated data at this node, and the data need to be sent to one of the nodes in D. Note that the data can be sent to *any* of the sinks. We want to find a data-collection schedule to transmit all the data to the sinks. The schedule must specify in each time slot which links are used for data transfer. In order for the schedule to be valid, it must not allow two interfering links to be used for data transfer at the same time. Our objective is to find a data-collection schedule of the minimum latency.

4 A constant approximation algorithm

In this section, we present an approximation algorithm with a constant-factor guarantee. The algorithm consists of three components. The first component constructs a *time-expanded graph* [10, 6], which represents the progression of the state of the network throughout a given time period T, during which data collection occurs. The second component uses a *linear programming* (LP) formulation to generate *feasible flows* to send data over edges in the time-expanded graph. The first and second components need to be repeatedly run until we can find a feasible flow with latency close enough to the optimal solution. The details of the iterations are given in the end of Section 4.2 and Algorithm 1. Once a feasible flow solution is obtained, the third component finds an *exact data collection schedule* for each time slot. We refer to this algorithm as *LP-based algorithm*. In Section 4.4 we show that it gives constant factor guarantees, with the constant depending on the interference model (e.g., it gives 5-approximation under unit disk model). We next describe the three components in the algorithm in detail.

4.1 Generating a time-expanded graph

One core component of our algorithm is expanding the network graph, G, to a time-expanded graph, $G_T = (V_T, E_T)$, which represents the network over a period of T time units, and in which we can compute a network flow that corresponds to a valid data collection schedule. Figure 1 depicts an example network graph and its corresponding time-expanded flow graph.

Given a graph G and time T, the expanded time graph, $G_T = (V_T, E_T)$, can be constructed as follows. For each network node $v_i \in V$, and for every $t \in \{0...T\}$, we add a vertex $v_{i,t}$ to V_T. These vertices represent v_i at $T + 1$ points of time, where $v_{i,t+1}$ is one time unit later than $v_{i,t}$. To represent the storage of data over time at a node in V, we assign an unlimited flow capacity, $C_T(v_{i,t}, v_{i,t+1}) = \infty$, to any $v_i \in V$ and $t \in \{0...T - 1\}$. Furthermore, we assign

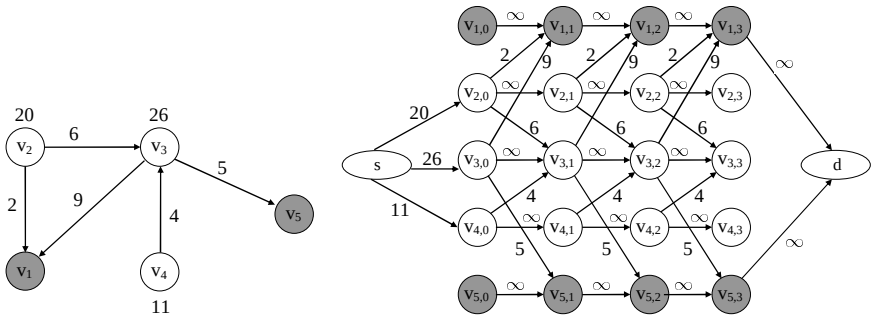


Fig. 1. A network graph, G, and its corresponding time-expanded graph, G_T, where $T = 3$, and the shaded nodes represent the sinks.

a flow capacity $C_T(v_{i,t}, v_{j,t+1}) = C(v_i, v_j)$ to each edge $(v_{i,t}, v_{j,t+1}) \in E_T$, which represents the maximum amount of flow allowed on this edge in a time unit. In addition, we associate a flow interference set, $I_T(v_{i,t}, v_{j,t+1})$, to $(v_{i,t}, v_{j,t+1}) \in E_T$, which can be inferred from the interference set of the corresponding link in E. Specifically, for all $v_i, v_j, v_{i'}, v_{j'} \in V$ with $(v_{i'}, v_{j'}) \in I(v_i, v_j)$, and for all $t \in \{1...T-1\}$, $(v_{i',t}, v_{j',t+1}) \in I_T(v_{i,t}, v_{j,t+1})$. Last, we add a super source s and a super sink d into the time expanded graph. We also create a link from s to $v_{i,0}$ with capacity s_i for any node $v_i \in V \setminus D$. Meanwhile, for any node $v_j \in D$ in the original graph, we have a link from $v_{j,T}$ to d with unlimited capacity.

4.2 Computing a data flow

The constructed time-expanded graph G_T, along with the associated supplies, edge capacities, and interference constraints, form an interference-bound network flow problem. In this problem, if the maximum flow from s to d is $\sum_{v_i \in V \setminus D} s_i$, then in the original graph we can collect all the data to the sink in time T. We solve the network flow problem using a linear programming (LP) formulation as follows. Denote $f_{u,v}$ as the flow along the edge from vertex u to v for any $u, v \in V_T$ and $(u, v) \in E_T$. In particular, $f_{s,u}$ represents the flow along the link from the super source s to node $u \in V_T$, and $f_{v,d}$ represents the flow along the link from node $v \in V_T$ to the super sink d. Let $Out(u)$ denote the set of the endpoints of the outgoing links of u, $u \in V_T$, and let $In(v)$ denote the set of the endpoints of the incoming links of v, $v \in V_T$. Then the LP formulation is:

$$\textbf{Maximize}: \quad \sum_{u \in Out(s)} f_{s,u}$$

$$\textbf{Subject to}: \quad \sum_{u \in Out(s)} f_{s,u} - \sum_{v \in In(d)} f_{v,d} = 0 \tag{1}$$

$$\sum_{v \in In(u)} f_{v,u} - \sum_{v \in Out(u)} f_{u,v} = 0, \ \forall\ u \neq s, d \tag{2}$$

$$\frac{f_{u,v}}{C_T(u,v)} + \sum_{(u',v') \in I(u,v)} \frac{f_{u',v'}}{C_T(u',v')} \leq 1, \ \forall\ u, v \in V_T \tag{3}$$

$$f_{u,v} \leq C_T(u,v), \ \forall\ u, v \in V_T \tag{4}$$

$$f_{u,v} \geq 0, \ \forall \ u, v \in V_T \tag{5}$$

The above LP formulation represents a network flow problem with additional constraints to ensure interference-free property. More specifically, Constraints (1) and (2) are for flow conservation, and Constraint (4) is for link capacity. Constraint (3) provides a sufficient condition for interference-free schedule. In other words, the constraints ensure that, throughout any time slot, only one network link in each interference set can be active.

For a given time T, if the resulting maximum flow from s to d after solving the LP problem is $\sum_{v_i \in V \setminus D} s_i$, then we can infer a complete schedule for all data in the network to be collected in time at most T. By performing a standard doubling/binary search, we can find the minimum latency T_{min} necessary to have a valid network flow solution in logarithmic time. Once T_{min} is found, the solution of network flow in the time-expanded graph $G_{T_{min}}$ can be obtained, and will be used in the third step (see Section 4.3) to generate an exact data collection schedule. Algorithm 1 presents a high-level description of the above procedure.

Algorithm 1 Compute feasible data flow

1. Guess T_{min} in logarithmic time, using doubling and binary search:
   (a) Generate a time-expanded graph G_T of length T from the given network G.
   (b) Check whether there exists a valid flow in G_T using the LP formulation.
2. Compute the solution of network flow in time-expanded graph $G_{T_{min}}$.

4.3 Data collection schedule

We now present a scheduling algorithm for each time interval. We assume that the amount of data to be collected is sufficiently large, and so we can split the data into pieces as small as necessary. Without loss of generality, we assume the time interval to be one second.

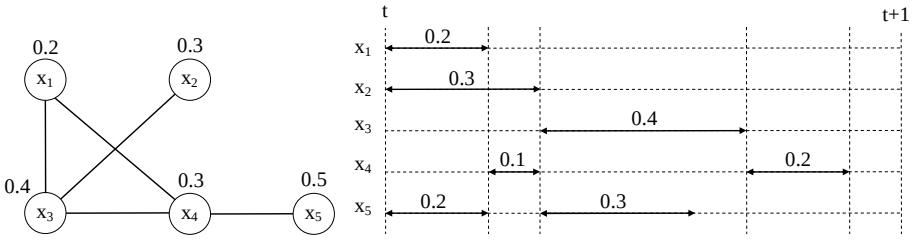
Consider time interval $[t, t+1]$. Let $G_t = (V_t, E_t)$ denote a subgraph of $G_T = (V_T, E_T)$, where $V_t = \{v_{i,t}, v_{i,t+1} \mid v_i \in V\}$ and $E_t = \{(v_{i,t}, v_{j,t+1}) \mid (v_i, v_j) \in E\}$. We derive an link interference graph $G_{I_t} = (V_{I_t}, E_{I_t})$ from $G_t = (V_t, E_t)$, where node $v^I_{i,t} \in V_{I_t}$ corresponds to an edge in E_t, and an edge in E_{I_t} connects two vertices, $v^I_{i,t}$ and $v^I_{j,t}$, if and only if $v^I_{i,t}$ and $v^I_{j,t}$ interfere with each other in G_t. Let f^t_i denote the amount of flow that is transmitted through $v^I_{i,t}$. Let $N(v^I_{i,t})$ be the set of nodes in V_{I_t} that are connected to $v^I_{i,t}$. That is, $N(v^I_{i,t})$ contains all the edges that interfere with $v^I_{i,t}$ in E_t. Let $x^t_i = f^t_i / C_T(v^I_{i,t})$, where $C_T(v^I_{i,t})$ represents the capacity along the edge $v^I_{i,t} \in E_t$. Then x^t_i represents the amount of time that $v^I_{i,t}$ needs to occupy in interval $[t, t+1]$.

Algorithm 2 describes how to construct a data collection schedule for interval $[t, t+1]$. Basically, for each node $v^I_{i,t} \in V_{I_t}$, it schedules a duration of x^t_i as early as possible while not causing any interference. An example illustrating the algorithm is shown in Figure 2. In the figure, (a) shows an example of link

Algorithm 2 Construct data collection schedule for interval $[t, t+1]$

for $i = 1$ to $|V_{I_t}|$ **do**

1. In interval $[t, t+1]$, find the earliest slot, $[t', t' + x_i^t]$, for $v_{i,t}^I$ while not causing any interference (i.e., this slot has no overlap with the slots already allocated to nodes in $N(v_{i,t}^I)$).
2. If no consecutive slot of duration x_i^t is found, split x_i^t into shorter durations and schedule each duration separately.



(a) $G_{I_t} = (V_{I_t}, E_{I_t})$ (b) Data collection schedule

Fig. 2. An example illustrates Algorithm 2

interference graph G_{I_t} where the number beside each node shows the amount of flow to be sent over the link. A valid data collection schedule for the link interference graph can be constructed as shown in (b). That is, the flows for x_1 and x_2 can be scheduled starting from time t. Transmission for x_3 can be scheduled only after $t + 0.3$ due to the interference with x_2 which is already scheduled. For x_4, 0.1 is scheduled after x_1 is finished and before x_3 starts sending, and the remaining is scheduled after $t + 0.7$. x_5 can be scheduled in a similar way.

In Section 4.4, we prove that this algorithm produces a schedule of length at most 1 for time interval $[t, t+1]$. Therefore, we can concatenate the T unit schedules to produce a complete schedule of length at most T. In practice, the length of the schedule for each interval may be substantially less than one, leading to a complete schedule substantially less than T. In our experiments, we found that a computed complete schedule is often below T (it is often less than $0.7T$).

4.4 Analysis

Theorem 1. *The data collection schedule ultimately produced by our algorithm is valid and satisfies the interference constraints.*

Proof. We only sketch the proof due to space limit. First, it is easy to see that, in Algorithm 2, any two nodes, $v_{i,t}^I, v_{j,t}^I \in V_{I_t}$, that interfere with each other will not be scheduled during the same time slot in $[t, t+1]$. Secondly, since Constraint (3) in the LP formulation is equivalent to $x_i^t + \sum_{v_{k,t}^I \in N(v_{i,t}^I)} x_k^t \leq 1$, in each unit time interval, an amount of x_i^t for v_i^t will be completely scheduled.

Last, Constraints (1) in the LP formulation guarantee that all the data from the sources will be delivered to the sinks at the end of the schedule. Therefore, the above claim holds. □

We next establish that, under several network models, our algorithm approximates an optimal schedule within a constant factor. In particular, under a unit disk model, it approximates the optimal schedule within a factor of five. To prove the approximation guarantee, we use the following lemma from [17].

Lemma 2. *[17] For every edge $(u, v) \in E$, there are at most c edges in $I(u, v)$ that can transfer data at the same time, where c is a constant depending on the model used to determine the network's interference constraints. Under a unit disk model, $c = 5$.*

Theorem 3. *The LP-based algorithm gives c-approximation for Multi-sink Data Collection problem.*

Proof. Let LP_k refer to a linear programming formulation, which differs from that in Section 4.2 in that the interference constraint (3) is replaced by

$$\frac{f_{u,v}}{C_T(u,v)} + \sum_{(u',v') \in I(u,v)} \frac{f_{u',v'}}{C_T(u',v')} \leq k, \ \forall \ u, v \in V_T. \tag{6}$$

That is, LP_k allows using k edges simultaneously in each interference set (our original LP formulation is LP_1).

By Lemma 2, an optimal data collection schedule may use no more than c edges in a given interference group simultaneously. It has been shown that Constraints (6) provide a necessary condition to obtain a valid flow without interferences when k is set to c (c depends on the interference model) [17]. Therefore, the optimal solution is lower bounded by the solution to LP_c. We can also see that, at any point in time, the schedule produced by LP_c allows c edges in each interference group to be active. If the amount of time available is multiplied by c, then each of these potentially interfering transmissions can be placed in its own time period. The resulting schedule satisfies the constraints imposed in LP_1. Let OPT refer to an optimal algorithm, and let $\mathcal{T}(X)$ denote the latency of the schedule produced by an algorithm, X. Therefore, $\mathcal{T}(LP_1) \leq c \cdot \lceil \mathcal{T}(LP_c) \rceil \leq c \cdot \lceil \mathcal{T}(OPT) \rceil$. □

5 A Greedy BFS-based Algorithm

In this section, we design a heuristic algorithm for the special case where all the links have the same capacity. This heuristic algorithm is greedy in nature and uses breadth first search. Therefore, we call it *Greedy BFS-based Algorithm*.

The main idea of the algorithm is to divide nodes into layers according to their distance to the set of sinks, and let nodes far away from the sinks transmit data to nodes close to the sinks, which in turn forward the data to the sinks. More specifically, let $Dist(u, v)$ denote the distance between u and v in graph G, which is the length (in hop counts) of the shortest path connecting u and v in G.

Let $Dist(v, U)$ denote the distance from u to a set of nodes U, which is defined to be $\min_{u \in U} Dist(u, v)$. We divide nodes into layers according to their distance to the sink set, D. More specifically, let L_i denote the set of nodes whose distance to D is i. Let l_{max} denote the index of the layer that is farthest away from D. We can obtain L_i using breadth first search, $i = 1, \ldots, l_{max}$. Let $Data(u, t)$ denote the amount of data at node u at time t. Consider a heuristic function, $F(t) = \sum_{u \in V} Dist(u, D) Data(u, t)$. At the end of the schedule (i.e., after all data have been transmitted to the sinks), this function equals zero. Our heuristic takes a greedy approach to reduce $F(t)$ as fast as possible by maximizing the number of nodes that transmit data at every time t, while ensuring that (i) the transmissions do not create any interference, and (ii) nodes in L_{i+1} transmit to nodes in L_i (so that $F(t)$ decreases over time). We repeat the above procedure until $F(t)$ goes to zero. The pseudo-code of the algorithm is given in Algorithm 3.

Algorithm 3 Greedy BFS-based Algorithm

```
Classify nodes into layers using breadth first search
t ← 0
while F(t) > 0 do
    X ← ∅ /* the set of transmitting links at time t */
    for i = l_max to 1 do
        sort Data(u,t) in decreasing order, u ∈ L_i
        for each u ∈ L_i such that Data(u,t) > 0 do
            if u has neighbor v ∈ L_{i-1} such that the link (u,v) does not interfere with
            any link in X then
                X ← X ∪ {(u,v)}
            end if
        end for
    end for
    Transmit data on every link in X
    Update Data(u,t) for all u ∈ V
    t ← t + 1
end while
```

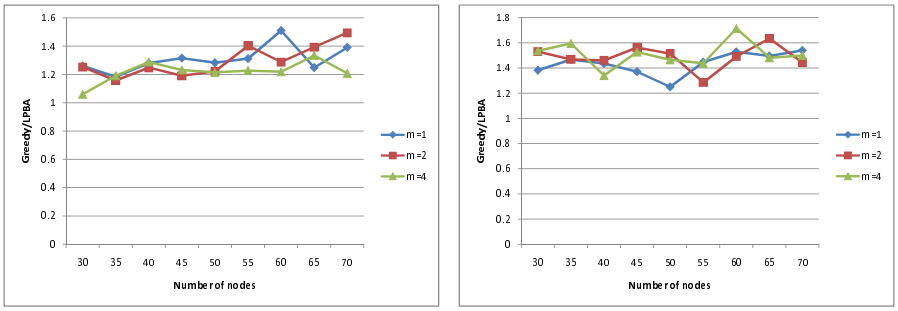
6 Experimental Results

Network setting: We generate a unit disk graph by placing n nodes uniformly at random over a square region of size $\sqrt{2n} \times \sqrt{2n}$, where n is the number of nodes. The area of the network, as a result, increases in proportion to the number of nodes, maintaining an approximately constant network density. We vary the number of nodes n in the random network from 30 to 70 with increments of five. There is an edge between two nodes if their geometric distance is no more than one. The capacities of all the links are the same. For convenience, we set all capacities to be 1. A random subset of the nodes is chosen as sinks. The number of sinks is set to be $1, 2, 4$ and 8. We consider two distributions for the initial

amount of data at non-sink nodes: uniform distribution, where the initial data at each non-sink node is set to be 1, and skewed distribution, where the initial data s_i at node i is $1/i$.

Interference model: Our algorithm is applicable to any interference model. In our experiments, we use two-hop interference model, in which two edges can interfere with each other if they are within two-hop distance [16].

Performance comparison. For each setting, we generate ten independent inputs and compare the average latencies of the two algorithms over the ten inputs. Figure 3(a) and (b) depict the ratio between average latencies of Greedy BFS-based Algorithm and LP-based Algorithm for uniform and skewed distribution, respectively (the results using 8 sinks are similar and omitted for clarity). The confidence intervals are tight and hence omitted. We observe that the latency of LP-based Algorithm is around 20% to 70% shorter than that of Greedy BFS-based Algorithm. For uniform distribution, the gain of LP-based Algorithm increases from 1.1 to 1.5 as the number of nodes increases. For skewed distribution, LP-based algorithm outperforms Greedy Algorithm by 40%-60%, even for small networks. Last, we observe similar results for different number of sinks.



(a) Uniform distribution (b) Skewed distribution

Fig. 3. Ratio between average latencies of Greedy BFS-based Algorithm and LP-based Algorithm. m is the number of sinks

7 Conclusion

We proposed a constant approximation algorithm and a heuristic for Multiple-Sink Data Collection Problem in wireless sensor networks. The constant approximation algorithm first constructs a time-expanded graph, then generates feasible flows in the expanded graph, and finally finds an exact data collection schedule. The heuristic algorithm is based on greedy breadth first search. Our experimental results show that the approximation algorithm can significantly outperform the simple heuristic (up to 60%).

References

1. N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290C298, 1991.

2. H. Balakrishnan, C. Barrett, V. S. Anil Kumar, M. Marathe, and S. Thite. The distance 2-matching problem and its relationship to the mac layer capacity of adhoc wireless networks. *IEEE J. Selected Areas in Connunications*, 22(6), August 2004.
3. R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104C126, 1992.
4. William C. Cheng, Cheng-Fu Chou, Leana Golubchik, Samir Khuller, and Yung-Chun (Justin) Wan. Large-scale data collection: a coordinated approach. In *IEEE Infocom*, 2003.
5. Cedric Florens and Robert McEliece. Packets distribution algorithms for sensor networks. In *IEEE Infocom*, 2003.
6. L.R. Ford and D.R. Fulkerson. Flows in networks. In *Princeton University Press, New Jersey*, 1962.
7. R. Gandhi, S. Parthasarathy, and A. Mishra. Minimizing broadcast latency and redundancy in ad hoc networks. In *ACM MobiHoc03*, pages 222–232, 2003.
8. Rajiv Gandhi, Yoo-Ah Kim, Seungjoon Lee, Jiho Ryu, and Peng-Jun Wan. Approximation algorithm for data broadcast and collection in wireless networks. In *IEEE Infocom (Mini conference)*, 2009.
9. P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.
10. B. Hopper and E. Tardors. Polynomial time algorithms for some evacuation problems. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 512 – 521, 1994.
11. Qingfeng Huang and Ying Zhang. Radial coordination for convergecast in wireless sensor networks. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 542–549, Washington, DC, USA, 2004. IEEE Computer Society.
12. Scott C.-H. Huang, Peng-Jun Wan, Jing Deng, and Yunghsiang S Han. Broadcast scheduling in interference environment. *IEEE Trans. on Mobile Computing*, 7(11):1338 – 1348, November 2008.
13. Scott C.-H. Huang, Peng-Jun Wan, Xiaohua Jia, Homngwei Du, and Weiping Shang. Minimum-latency broadcast scheduling in wireless ad hoc networks. In *Proceedings of 26th IEEE Infocom*, pages 733 – 739, 2007.
14. Scott C.-H. Huang, Peng-Jun Wan, Chinh T. Vu, Yingshu Li, and Frances. Yao. Nearly constant approximation for data aggregation scheduling in wireless sensor networks. In *Proceedings of 26th IEEE Infocom*, pages 366 – 372, 2007.
15. Alex Kesselman and Dariusz R. Kowalski. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. *J. Parallel Distrib. Comput.*, 66(4):578–585, 2006.
16. V.S. Anil Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet scheduling in ad hoc networks. In *ACM-SIAM symposium on Discrete Algorithms, SODA*, pages 1014–1023, 2004.
17. V.S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. Algorithmic aspects of capacity in wireless networks. In *SIGMETRICS*, Banff, Alberta, Canada, June 2005.
18. S. Upadhyayula, V. Annamalai, and S. K. S. Gupta. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *IEEE Global Telecommunications Conference*, pages 3525–3530, 2003.
19. S. Yi, Y. Pei, and S. Kalyanaraman. On the capacity improvement of ad hoc wireless networks using deirectional antennas. In *Proceedings of 4th ACM international symposium on Mobile ad hoc networking and computing*, pages 108–116, 2003.