QoS Constrained Optimal Sink and Relay Placement in Planned Wireless Sensor Networks

Abhijit Bhattacharya, Akhila Rao, Naveen K. P., Nishanth P. P.,
S.V.R. Anand, and Anurag Kumar
Dept. of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore 560012, India.
Email: {abhijit, naveenkp, anand, anurag}@ece.iisc.ernet.in, {akhila.suresh.rao, nishanth.pp93}@gmail.com

***Abstract*— We are given a set of sensors at given locations, a set of potential locations for placing base stations (BSs, or sinks), and another set of potential locations for placing wireless relay nodes. There is a cost for placing a BS and a cost for placing a relay. The problem we consider is to select a set of BS locations, a set of relay locations, and an association of sensor nodes with the selected BS locations, so that the number of hops in the path from each sensor to its BS is bounded by $h_{\max}$, and among all such feasible networks, the cost of the selected network is the minimum. The hop count bound suffices to ensure a certain probability of the data being delivered to the BS within a given maximum delay under a light traffic model. We observe that the problem is NP-Hard, and is hard to even approximate within a constant factor. For this problem, we propose a polynomial time approximation algorithm (SmartSelect) based on a relay placement algorithm proposed in our earlier work, along with a modification of the greedy algorithm for weighted set cover. We have analyzed the worst case approximation guarantee for this algorithm. We have also proposed a polynomial time heuristic to improve upon the solution provided by SmartSelect. Our numerical results demonstrate that the algorithms provide good quality solutions using very little computation time in various randomly generated network scenarios.**

***Index Terms*—Wireless sensor network design; Multiple sink and relay placement; QoS-aware network design**

I. Introduction[1]

Recently there has been increasing interest in replacing wireline industrial sensor networks with wireless packet networks ([1], [2], [3]). Owing to the small communication range of the sensing nodes (typically a few tens of meters), usually multi-hopping is needed to communicate to the control center. The practical problem that we consider in this paper is the following: there are already deployed, static sensors (also referred to as *sources*) from which measurements, encapsulated into packets, need to be collected. Additional relays and base stations (BS) need to be placed in the region in order to provide multi-hop paths from each of the sources to at least one BS. The sources can also act as relays for the packets from other sources. The network so obtained needs to provide certain quality-of-service (QoS) to the packets flowing over it, in terms of, e.g., delivery probability, or packet delay.

In most practical applications, due to the presence of obstacles or taboo regions, we cannot place relays and sinks anywhere in the region, but only at certain designated locations. This leads to the problem of *constrained node placement* in which the nodes are constrained to be placed at certain *potential locations*. Further, only certain links are permitted[2]. See Figure 1 for a depiction of the problem.

We assume that there is a cost associated with each sink, and each relay. The objective of the design is to *place a minimum cost selection of sinks and relays* (at the potential locations) while achieving a network that meets the following QoS objectives:

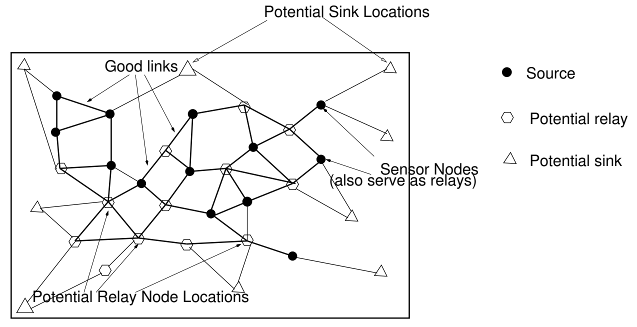

Fig. 1. The constrained sink and relay placement problem; the edges denote the useful links between the nodes.

1) There is a path from each source node to at least one BS.
2) The maximum delay on any path is bounded by a given value $d_{\max}$, and the packet delivery probability (the probability of delivering a packet within the delay bound) on any path is $\geq p_{\text{del}}$.

In wireless networks, the actual link qualities are unknown *a priori*, and can only be ascertained by field measurements; designs based only on *approximate* stochastic RF propagation models cannot be guaranteed to work when deployed on field. This motivates iterative algorithms that explore the field by making partial deployments and link quality measurements, and build the network iteratively (see, for example, [4]). As explained in [4], such iterative algorithms require to invoke, at each iteration, a design module which extracts from a network graph with *given* link qualities, a subgraph that meets the QoS requirements assuming the given link qualities to be true. Since this module has to be invoked at each of potentially many iterations, it should have *low time complexity*. We therefore, seek *fast* algorithms that provide optimal or close to optimal solutions. In our earlier work [5], [6], [4], we addressed this problem for the special case where there is a single BS at a designated location. In this paper, we study the more general problem where there is the possibility of deploying multiple sinks. This general problem is of interest for scalability of the network design, i.e., when we want to deploy a network over an area so large that a single sink based solution fails to meet the desired QoS requirements.[3]

II. The Network Design Problem

The Lone Packet Model: In this paper, we assume that the traffic from the source nodes is *very light*. Formally, we define "light traffic" as follows: at any point of time, there is at most one packet flowing in the network. We call this the "lone packet traffic model," which is realistic for many applications,

[1]This work was supported by the Department of Science and Technology (DST) under the Indo-Brazil project, and under a J. C. Bose fellowship

[2]This could be because some links could be too long, leading to high bit error rate and hence large packet delay, or due to an obstacle, e.g., a firewall

[3]For example, since in low power wireless networks the usable link lengths are limited, with just one sink the number of hops from some of the sources to the sink can become so large as to make the packet delivery probability unacceptably small

including the so called *condition monitoring/industrial telemetry* applications ([7]), where the time between successive measurements being taken is sufficiently long so that the measurements can be staggered so as not to occupy the medium at the same time. The main motivation behind the lone-packet model comes from the following important result (formally proved in [6]): for a design (network) to satisfy the QoS objectives for a given positive traffic arrival rate, it is *necessary* that the network satisfies the QoS objectives under the lone packet model.

This light traffic assumption facilitates the conversion of QoS objectives into simple graph constraints. The designs based on lone-packet model can be used as a starting point for network design with more general arrival processes. Packet level simulation results reported in [6] suggest that such lone-packet model based designs suffice up to some small (but useful) positive arrival rates. For a more detailed discussion on the applicability and justification of the lone-packet model, see [6].

The Network Design Setting: Given a set of source nodes or required vertices Q, a set of potential relay locations R, each with cost c_r, and a set of potential sink locations B, each with cost c_s, we consider a graph $G = (V, E)$ on $V = Q \cup R \cup B$ with E consisting of all *feasible* edges. Throughout this work, we assume that all nodes operate at the same fixed power level. We can then define the set of *feasible* edges E, either by imposing a bound on the packet error rate (PER) of each link, or alternately, by constraining the maximum allowed link length (which, in turn, affects the link PER). Having thus characterized the link quality of each feasible link in the graph G, it can be shown by an elementary analysis that the QoS objectives ($d_{\max}$ and p_{del}) can be met by imposing a hop count bound of $h_{\max}$ between each source node and the sink. Details of this analysis are provided in [6], where we have considered the practical situation of slowly fading links, and packet losses due to random channel errors. Thus, there is a random delay at each hop due to packet retransmissions, and packets could be dropped if a retransmission limit is reached. Note that as a consequence of the lone packet assumption, the delay along a path is additive, i.e., it is simply the sum of the delays on each hop along the path.

Problem Formulation: Given the graph $G = (V, E)$ on $V = Q \cup R \cup B$ with E consisting of all *feasible* edges (as explained earlier), costs c_s and c_r of each sink and relay respectively, and a hop constraint $h_{\max}$, *the problem is to extract from this graph, a minimum cost subgraph spanning Q, such that each source has a path to at least one sink with hop count* $\leq h_{max}$. We call this the *MultiSink Steiner Network-Minimum Cost-Hop Constraint* (MSSN-MC-HC) problem.[4]

Complexity of the Problem

Proposition 1: 1) **Complexity:** The MSSN-MC-HC problem is NP-Hard. 2) **Inapproximability:** The MSSN-MC-HC problem is not constant factor approximable. In particular, it cannot be approximated to within a factor better than $O(\log(m))$, where m is the number of sensors.

Proof: Proof uses the restriction argument [8, p. 63, Section 3.2.1] along with a reduction from set cover to the subclass of problems with $c_r = 0$. For details, see [9]. ■

Proposition 2: If $\frac{c_s}{c_r} \geq \overline{m}(\overline{m}+1)(h_{\max}-1)$, for some $\overline{m} \in \mathbb{N}$, $m > \overline{m} \geq 1$, then the worst case approximation guarantee of any algorithm for the MSSN-MC-HC problem is upper bounded by $m\left(1+\frac{1}{\overline{m}(\overline{m}+1)}\right)$, where $|Q| = m$.

Proof: The proof follows by observing that the cost of the optimal solution to the MSSN-MC-HC problem is lower bounded by c_s (since at least one sink is required), and the cost of the outcome of any algorithm is upper bounded by $mc_s + m(h_{\max}-1)c_r$. See [9] for details. ■

Related Work: [10], [11] proposed approximation algorithms for variations of the *NP-Hard* problem of *unconstrained relay placement for connectivity*, where a minimum number of relays have to be placed (they can be placed anywhere; no *potential* locations are given) to obtain a tree spanning a given set of sources and BS. No QoS constraint was imposed in their formulations. Bredin et al. [12] proposed an $O(1)$ approximation algorithm for the *NP-hard* problem of optimal *(unconstrained)* relay placement for $k-$connectivity, but without any QoS constraint. [13] and references therein proposed approximation algorithms for variations of the NP-Hard problem of *constrained relay placement* for connectivity and survivability, but *without any constraint on the end-to-end QoS.* In [5], [6], [4], we have studied the *NP-Hard* problem of *constrained* relay placement *with end-to-end QoS objective*, but with a single Base Station at a given location. In our current work, we aim at extending this formulation to incorporate the possibility of deploying multiple sinks. Recently, Sitanayah et al. [14] have proposed a local search heuristic (GRASP-MSRP) of *exponential time-complexity* for the MSSN-MC-HC problem. However, no theoretical study of either the problem, or the proposed algorithm was provided. *The time-complexity of the algorithm prohibits its use in an iterative network design process* such as SmartConnect [4]. Hence, we seek, instead, *fast* heuristics that perform reasonably close to optimal.

III. MSSN-MC-HC: A Heuristic and its Analysis

In this section, we present a polynomial time approximation algorithm for the MSSN-MC-HC problem. The algorithm proceeds by reducing the problem to a modified version of the weighted set cover problem, and the greedy algorithm for weighted set cover is used to obtain a solution. Since the greedy algorithm for weighted set cover is polynomial time [15], the proposed algorithm is polynomial time.

A. SmartSelect: A Greedy Algorithm for Sink and Relay Selection

1) **The single sink, zero relay case:** Consider the restriction of the graph G to only the sources, Q, and the potential sinks, B. For each sink $b \in B$, find the shortest path tree rooted at b spanning the sources in Q. If there exists a sink b_0 such that the SPT rooted at b_0 satisfies the hop constraint for each source in Q, then we are done; the optimal solution requires a single sink, and no relays. Otherwise, go to the next step.
2) **Checking feasibility:** On graph G (i.e., now including all sources and relays), obtain a shortest path tree rooted at each potential sink location (i.e., we have as many shortest path trees as there are potential sinks). If there exists a source such that its shortest paths to all the sinks have lengths exceeding $h_{\max}$, declare the problem infeasible. Else, go to the next step.
3) For each sink $b_i \in B$, $i = 1, \ldots, |B|$, identify the set of sources, say Q_i, whose shortest paths to b_i have lengths $\leq h_{\max}$. The set $Q_i \subseteq Q$, is said to be *covered* by b_i. Note that, having ensured feasibility in Step 2, $\cup_{i=1}^{|B|} Q_i = Q$. Also identify the set of relays, $R_i \subseteq R$, whose shortest paths to b_i have lengths $\leq h_{\max} - 1$ (this helps to reduce the complexity of Step 4; indeed R_i is the set of relays that may ever be used to connect the sources in Q_i to b_i).

[4]In this formulation, we have not taken into account the energy expenditure at a node due to transmission and reception since, in a light traffic setting, the fraction of time a node is transmitting or receiving a packet is small compared to the idle time of the node.

4) Set $j \leftarrow 0$. The iterations will be indexed by j. Set $Q_i^{(0)} = Q_i$, $B^{(0)} = B$. $B^{(j)}$ denotes the set of sinks not yet picked at the start of iteration j, $j \geq 0$, and $Q_i^{(j)}$ denotes the set of *uncovered* sources that are associated with a BS $b_i \in B^{(j)}$ at the start of iteration j.

The greedy iterative algorithm:

5) For each i such that $b_i \in B^{(j)}$, let $G_i^{(j)}$ be the restriction of G to $Q_i^{(j)} \cup R_i \cup \{b_i\}$. Run an algorithm (e.g., the SPTiRP algorithm in [6]) on $G_i^{(j)}$ to obtain a *near-optimal* subset of relays, $\hat{R}_i^{(j)} \subseteq R_i$, that connect the sources in $Q_i^{(j)}$ to b_i with $\leq h_{\max}$ hops.
6) For each i such that $b_i \in B^{(j)}$, consider the restriction of $G_i^{(j)}$ to $Q_i^{(j)} \cup \hat{R}_i^{(j)} \cup b_i$. Denote this graph by $\tilde{G}_i^{(j)}$.
7) For each i such that $b_i \in B^{(j)}$, define the cost of $\tilde{G}_i^{(j)}$ as $C_i^{(j)} = \frac{c_s + c_r \times |\hat{R}_i^{(j)}|}{|Q_i^{(j)}|}$, i.e., the cost of a subgraph is computed as the total cost per source.
8) **The greedy selection:** Pick the subgraph with the least cost among the subgraphs not yet picked, i.e., pick $\tilde{G}^{(j)} = \arg\min_{\tilde{G}_i^{(j)}} C_i^{(j)}$. Break ties by picking the subgraph that covers more sources. Let $\tilde{b}^{(j)}$ be the sink associated with $\tilde{G}^{(j)}$.
9) Let $Q^{(j)} = Q \cap \tilde{G}^{(j)}$, $R^{(j)} = R \cap \tilde{G}^{(j)}$. If $\cup_{k=1}^{j} Q^{(k)} = Q$, **STOP**, i.e., stop when all the sources have been *covered*. Else, go to next step.
10) **Update step:** Set $B^{(j+1)} = B^{(j)} \backslash \tilde{b}^{(j)}$. For each i such that $b_i \in B^{(j+1)}$, set $Q_i^{(j+1)} = Q_i^{(j)} \backslash (\tilde{G}^{(j)} \cap Q_i^{(j)})$. Moreover, set the cost of each relay in $R^{(j)}$ to zero for all future iterations. This is done so that sources and relays that are shared by covers do not get counted more than once.
11) Set $j \leftarrow j + 1$, and go to Step 5.

B. Analysis of SmartSelect

Some observations:

1) If the optimal solution uses a single sink, and no relays, SmartSelect *achieves the optimal solution* (follows from Step 1).
2) For the sink placement problem (i.e., $c_r = 0$), the SmartSelect algorithm reduces exactly to the greedy algorithm for weighted set cover ([15]), and hence *achieves the best possible worst case approximation guarantee ($O(\log(m))$) for the sink placement problem*, where m is the number of sources.

Worst Case Approximation Guarantee: We already know that when an optimal solution uses a single sink, and *no relays*, the SmartSelect algorithm gives the optimal solution. We, therefore, focus on instances where any optimal solution uses at least one sink, and *at least one relay*. We start with the following lemma.

Lemma 1: Suppose $\frac{c_s}{c_r} \geq \overline{m}(\overline{m}+1)(h_{\max}-1)$, for some $\overline{m} \in \mathbb{N}, m > \overline{m} \geq 1$.[5] Then the following holds: In an iteration in which the number of sources remaining to be covered is $> \overline{m}$, if there is a (not yet selected) sink that covers all the remaining sources, the greedy algorithm will select a sink that covers at least $(\overline{m}+1)$ sources.

Proof: Suppose, up to the start of iteration j, $j \geq 0$, $m^{(j)}$ sources have been covered, and $m - m^{(j)} > \overline{m}$. Suppose there exists a sink (not yet picked) that covers all the remaining sources, and another sink (not yet picked) that covers $\overline{m}$ of the remaining sources. We index these two sinks by 1 and 2 respectively. Let $n_1^{(j)}$ and $n_2^{(j)}$ be the number of relays picked in Step 5 of the SmartSelect algorithm in iteration j for sinks 1 and 2 respectively. Then, the algorithm favors sink 2 over sink 1 iff

$$\frac{c_s + n_1^{(j)} c_r}{m - m^{(j)}} > \frac{c_s + n_2^{(j)} c_r}{\overline{m}}$$
$$\Leftrightarrow \frac{c_s}{c_r} < \frac{\overline{m} n_1^{(j)} - (m - m^{(j)}) n_2^{(j)}}{m - m^{(j)} - \overline{m}} \quad (1)$$

Note that we have made use of the tie-breaking mechanism introduced in Step 8 to obtain the *strict* inequality above.

Now, we make the following observations:

1) $n_1^{(j)} \leq (m - m^{(j)})(h_{\max} - 1)$, since each of the remaining sources connects to the sink 1 using at most $h_{\max}$ hops.
2) $\frac{\overline{m}(m-m^{(j)})}{m-m^{(j)}-\overline{m}} = \frac{\overline{m}}{1-\frac{\overline{m}}{m-m^{(j)}}} \leq \overline{m}(\overline{m}+1)$, where the last inequality follows since $m - m^{(j)} \geq \overline{m}+1$.

Then we have that

$$\begin{aligned}\frac{c_s}{c_r} &\geq \overline{m}(\overline{m}+1)(h_{\max}-1) \\ &\geq \frac{\overline{m}(m-m^{(j)})(h_{\max}-1)}{m-m^{(j)}-\overline{m}} \\ &\geq \frac{\overline{m} n_1^{(j)}}{m-m^{(j)}-\overline{m}} \\ &\geq \frac{\overline{m} n_1^{(j)} - (m-m^{(j)}) n_2^{(j)}}{m-m^{(j)}-\overline{m}}\end{aligned}$$

Thus, condition (1) cannot hold, and hence, the algorithm cannot favor sink 2 over sink 1.

Proceeding similarly, and using the monotonicity of $f(\overline{m}) \triangleq \overline{m}(\overline{m}+1)$ in $\overline{m}$, it can be shown that the algorithm cannot favor a sink covering less than $\overline{m}$ sources over sink 1. Then the lemma follows. ■

Equipped with the above lemma, we derive bounds on the worst case approximation factor of the SmartSelect algorithm as follows.

Theorem 1: Consider the subclass of MSSN-MC-HC problems where there exists a feasible solution that uses exactly one sink. Further assume that $\frac{c_s}{c_r} \geq \overline{m}(\overline{m}+1)(h_{\max}-1)$, for some $\overline{m} \in \mathbb{N}$, $m > \overline{m} \geq 1$. Then the following hold:
1) The number of sinks picked by the SmartSelect algorithm is at most $\lceil \frac{m}{\overline{m}+1} \rceil$.
2) The worst case approximation guarantee of the SmartSelect algorithm is upper bounded by $\epsilon + \frac{m}{\overline{m}}$, where $\epsilon \in [0, 1)$ satisfies $\lceil \frac{m}{\overline{m}+1} \rceil = \frac{m}{\overline{m}+1} + \epsilon$.

Proof: We provide an outline here. For details, see [9].

Since there exists a sink that can cover *all* the sources, as long as this sink is not picked during the course of the SmartSelect algorithm, the hypothesis in Lemma 1 continues to hold. Then Part 1 follows by using the monotonicity of $\overline{m}(\overline{m}+1)$, and repeatedly applying Lemma 1.

Part 2 follows by using Part 1, and observing that the optimum cost is at least $(c_s + c_r)$. ■

Corollary 1: Consider the subclass of problems introduced in Theorem 1. Further, suppose $\frac{c_s}{c_r} \geq \lceil \alpha m \rceil (\lceil \alpha m \rceil + 1)(h_{\max} - 1)$ for some $\alpha \in (0, 1]$. Then, the worst case approximation factor of the SmartSelect algorithm is upper bounded by $(1 +$

[5]This assumption is not too restrictive because in practice, the cost of a base station is much more than the cost of a relay node, since setting up a base station typically requires infrastructure such as uninterrupted power supply from mains, ethernet or WiFi connectivity to a backhaul network etc.

$\frac{1}{\alpha}$), i.e., for this subclass of problems, SmartSelect provides an $O(1)$ approximation guarantee.

Proof: Follows by putting $\overline{m} = \lceil \alpha m \rceil$, and proceeding as in the proof of Part 2 of Theorem 1. See [9] for details. ■

Theorem 2: Suppose $\frac{c_s}{c_r} \geq \overline{m}(\overline{m}+1)(h_{\max}-1)$, for some $\overline{m} \in \mathbb{N}$, $m > \overline{m} \geq 1$. Then, for the general class of MSSN-MC-HC problems, the worst case approximation guarantee of the SmartSelect algorithm is upper bounded by $\max\{\epsilon + \frac{m}{\overline{m}}, \frac{m}{2}\left(1 + \frac{1}{\overline{m}(\overline{m}+1)}\right)\}$, where $\epsilon \in [0,1)$ is as defined in Theorem 1.

Proof: We provide a sketch here. See [9] for details.

Theorem 1 gives the worst case approximation guarantee for a subclass of problems. For the remaining (complementary) class of problems, the worst case approximation guarantee is upper bounded by observing that the optimum cost $\geq 2c_s$, and trivially upper bounding the cost of the algorithm. The theorem follows by combining the worst case guarantees. ■

C. A Destroy and Repair Heuristic to Improve upon SmartSelect

We propose below a polynomial time heuristic to improve upon the solution provided by the SmartSelect algorithm.

Destroy and Repair Heuristic

1) Let $N^{(0)}$ be the outcome of the SmartSelect algorithm. $N^{(0)}$ is the restriction of the initial network graph G to the sources, selected sinks and selected relays. Set $k = 0$. Also set $N_{best} = N^{(0)}$, and $solution\_update = false$. Let K be the maximum number of iterations allowed.

2) For each sink b_j in $N^{(k)}$, do the following:

- Pretend to prune the sink b_j.
- Run the SmartSelect algorithm using *only the remaining sinks in* $N^{(k)}$, and *all* potential relays to obtain a solution N_1.
  If N_1 is feasible, and cost of N_1 is better than cost of N_{best}, set $N_{best} = N_1$, and $solution\_update = true$. Else go to the next step.
- Pretend to prune the sink b_j, and Run the SmartSelect algorithm using *all* the remaining sinks and relays in G to obtain a solution N_2.
  If N_2 is feasible, and cost of N_2 is better than cost of N_{best}, set $N_{best} = N_2$, and $solution\_update = true$.

3) After all the sinks in $N^{(k)}$ have been tried (for pruning), if $solution\_update = true$, set $k \leftarrow k+1$, $N^{(k)} = N_{best}$, and go to Step 2.

4) **Stop** when no further solution update is possible, or the maximum number of iterations have been exceeded.

Remark: Since each iteration of the Destroy and Repair heuristic uses the SmartSelect algorithm which is polynomial time, and since the number of iterations is upper bounded by a constant K, it follows that the heuristic is polynomial time.

IV. Numerical Results

Since computing an optimal solution is NP-Hard, we obtain a lower bound on the optimum cost of a problem instance by solving the LP relaxation of an ILP formulation for the MSSN-MC-HC problem. See [9] for details of the ILP. We compare the performance of our algorithms against this LP-based lower bound on the optimum cost, as well as the exponential search heuristic (GRASP-MSRP) proposed by [14] in three different experimental settings. In all the experiments, we chose $c_s = 10$, $c_r = 1$, and $h_{\max} = 5$; note that these choices satisfy the hypothesis on c_s/c_r made in Theorems 1 and 2 with $\overline{m} = 1$. For Setups 1 and 2, we selected the source locations and potential node locations according to a continuous uniform distribution over the area, whereas for Setup 3, we partitioned the area into square cells of side $10m$, and picked the sources and potential node locations randomly from among the corner points of the cells. The other details of the experimental settings are provided in Table I.

TABLE I
Details of the Experimental Settings

Setup	Instances generated	Sources	potential relays	potential sinks	Area (in m^2)	$r_{\max}$ (in meters)
1	60	20	30	10	100×100	20
2	60	40	50	15	140×140	20
3	60	30	50	15	140×140	30

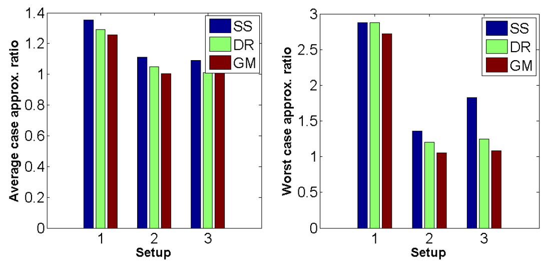

Fig. 2. Performance comparison of the algorithms against the LP-relaxation based lower bound

For ease of exposition, from now on, we use the abbreviations SS, DR, and GM respectively to denote the SmartSelect algorithm, the Destroy and Repair heuristic, and the GRASP-MSRP heuristic proposed in [14]. For each instance in each setting, we ran all the three algorithms on that instance, and also computed the LP-based lower bound on the optimum solution for that instance. The maximum number of iterations in the DR heuristic was chosen to be 25. The experiments were run using MATLAB R2011b on a Linux based desktop with 8 GB RAM. Figure 2 summarizes the performance of the algorithms as compared to the LP-based lower bound. The number of feasible instances for the three settings were 47, 32, and 60 respectively. For each of the feasible instances in each setting, we computed the empirical approximation ratio of each algorithm (with respect to the LP-based lower bound) as Approx. ratio $= \frac{\text{Cost of the algorithm outcome}}{\text{Cost of LP solution}}$. The maximum of these over the feasible instances in that setting was taken as the empirical worst case approximation ratio for that setting.

We also computed the empirical average case approximation ratio of the algorithms as follows: let $C_{avg}^{(algo)}$ be the average cost of the algorithm outcome over the feasible instances in a setting, and let $\overline{C}_{lp}$ be the average cost of the LP solution over those feasible instances. Then, the empirical average case approximation ratio, $\overline{\alpha}_{algo}$ was computed as $\overline{\alpha}_{algo} = \frac{C_{avg}^{(algo)}}{\overline{C}_{lp}}$. The theoretical upper bound on the worst case approximation ratio of SS algorithm was computed using Theorem 2.

In Table II, we compare the execution times of the algorithms and that of the LP-based lower bound computation.

From Figure 2 and Table II, we make the following observations:

1) In all the experimental settings considered, the *average* empirical performance of both the SS and DR algorithms in terms of cost are within a factor of about 1.4 of the LP based *lower bound* on the optimum cost. Notice that the actual performance would be even better since we are only comparing

TABLE II
EXECUTION TIMES OF THE ALGORITHMS, AND THE LP-BASED LOWER BOUND COMPUTATION

Experimental setup	Execution time in secs							
	SS		DR		GM		LP	
	mean	max	mean	max	mean	max	mean	max
1	1.148	2.72	3.4565	15.0878	258.645	673.415	37.36	138.14
2	4.5344	12.37	23.64	86.02	1885.8	6296.4	3819.7	41785
3	5.173	11.46	12.095	34.922	311.514	638.15	489.039	5395.3

TABLE III
PERFORMANCE COMPARISON OF THE DR AND SS ALGORITHMS

Experimental setup	Improvement in average cost by DR over SS (in %)	Max. improvement in cost by DR over SS (in %)
1	4.87	37.5
2	5.65	29.3
3	7.68	75

TABLE IV
PERFORMANCE COMPARISON OF THE DR AND GM ALGORITHMS

setup	Degradation in average cost of DR w.r.t GM (in %)	Max. degradation in cost of DR w.r.t GM (in %)	Max. improvement in cost of DR w.r.t GM (in %)
1	2.76	33.33	11.54
2	4.55	20.51	5.36
3	0.52	25	7.69

against a lower bound on the optimum cost. In the worst case, the algorithms are off from the lower bound by a factor of about 2.9, which is still much better than the theoretically predicted performance bound for the SS algorithm in the corresponding setting.
2) Both the SS and the DR algorithms achieve *orders of magnitude improvement in running time* compared to the LP (and hence, obviously with respect to the exact ILP).
3) In all the settings considered, both the SS and DR algorithms are *order of magnitude faster compared to the GM algorithm.*

While this extremely fast execution time of DR (and SS) is desirable, it comes with a degradation in cost. We now proceed to further quantify the degradation in cost when DR (or SS) algorithm is used instead of the GM algorithm. We first quantify the improvement in cost achieved by the DR heuristic over SS algorithm. Our findings are summarized in Table III.

Since we observe from Table III that the DR algorithm achieves an improvement in average cost of about 5% to 8% (and a maximum improvement of 75% over all the instances) over the SS algorithm in all the settings considered, and has the same order of running time as the SS algorithm (as observed from Table II), we next compare the performance of the DR algorithm against that of the GM algorithm ([14]). Our findings are summarized in Table IV.

From Table IV, we make a couple of observations:
1) In all the settings considered, the *average* cost of the DR heuristic is within at most 4.6% of that of the GM algorithm, and in the worst case (over all the tested scenarios in all the settings), the cost of the DR algorithm is off by 33.33% from that of the GM algorithm. On the other hand, as can be observed from Table II, *the improvement in average run time of the DR algorithm compared to that of the GM algorithm is up to a factor of about 80.*
2) Surprisingly, in all the three settings considered, there were instances where the DR algorithm in fact did better than the (more complex) GM algorithm even in terms of cost, as indicated by the last column in Table IV, and the improvement was up to 11.54%.

In summary, we conclude that the DR heuristic (with the SS algorithm as its starting point) achieves significant improvement in running time compared to both the GM algorithm ([14]), and ILP based solutions, while incurring only a small penalty in terms of cost. This extremely fast running time, and insignificant penalty in cost make the Destroy and Repair heuristic (with SmartSelect as starting point) an excellent choice for use in an iterative network design procedure such as SmartConnect [4].

V. CONCLUSION

In this paper, we have studied the problem of determining an optimal relay and sink node placement strategy such that certain performance objective(s) (in this case, hop constraint, which, under a lone-packet model, ensures data delivery to the BS within a certain maximum delay) is (are) met. We found that the problem is NP-Hard, and is even hard to approximate within a factor of $O(\ln m)$, where m is the number of sources. We have proposed a polynomial time approximation algorithm for the problem. The algorithm is simple, intuitive, and as can be concluded from numerical experiments presented in Section IV, gives solutions of very good quality in extremely reasonable computation time. We have also provided worst case bound on the performance of the algorithm. Further, we are working on combining our algorithm with that proposed by Sitanayah et al. [14] to further improve the cost efficiency of our algorithm while retaining the benefits of fast running time.

REFERENCES

[1] Honeywell, "www.honeywell.com/ps/wireless."
[2] ISA100, "www.isa.org/isa100."
[3] GE, "http://www.ge.com/stories/industrial-internet."
[4] A. Bhattacharya, S. M. Ladwa, R. Srivastava, A. Mallya, A. Rao, D. G. R. Sahib, S. Anand, and A. Kumar, "Smartconnect: A system for the design and deployment of wireless sensor networks," in *5th International Conference on Communication Systems and Networks (COMSNETS)*, 2013.
[5] A. Bhattacharya and A. Kumar, "Delay Constrained Optimal Relay Placement for Planned Wireless Sensor Networks," in *18th IEEE International Workshop on Quality of Service (IWQoS)*, 2010.
[6] A. Bhattacharya and A. Kumar, "QoS Aware and Survivable Network Design for Planned Wireless Sensor Networks," tech. rep., available at arxiv.org/pdf/1110.4746, 2013.
[7] B. Aghaei, "Using wireless sensor network in water, electricity and gas industry," in *3rd IEEE International Conference on Electronics Computer Technology*, pp. 14–17, April 2011.
[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Bell Telephone Laboratories, Inc., 1979.
[9] A. Bhattacharya, A. Rao, K. P. Naveen, P. P. Nishanth, S. Anand, and A. Kumar, "Qos constrained optimal sink and relay placement in planned wireless sensor networks," tech. rep., available at www.ece.iisc.ernet.in/~abhijit/temp/multi-sink-draft.pdf, 2014.
[10] G.-H. Lin and G. Xue, "Steiner tree problem with minimum number of Steiner points and bounded edge length," *Information Processing Letters*, vol. 69, pp. 53–57, 1999.
[11] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks," *IEEE Transactions on Computers*, vol. 56, January 2007.
[12] J. L. Bredin, E. D. Demaine, M. T. Hajiaghayi, and D. Rus, "Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance," in *MobiHoc'05*, ACM, 2005.
[13] D. Yang, S. Misra, X. Fang, G. Xue, and J. Zhang, "Two-Tiered Constrained Relay Node Placement in Wireless Sensor Networks: Computational Complexity and Efficient Approximations," *IEEE Transactions on Mobile Computing*, 2011. accepted for publication.
[14] L. Sitanayah, K. N. Brown, and C. J. Sreenan, "Multiple sink and relay placement in wireless sensor networks," in *European Conference on Wireless Sensor Networks*, 2013.
[15] V. V. Vazirani, *Approximation Algorithms*. Springer.