Optimal Cluster-head Deployment in Wireless Sensor Networks with Redundant Link Requirements

Xu Ning and Christos G. Cassandras *
Department of Manufacturing Engineering and
Center for Information and Systems Engineering
Boston University
Brookline, MA 02446, USA
nx@bu.edu, cgc@bu.edu

ABSTRACT

In this paper, a Wireless Sensor Network (WSN) deployment problem is posed: in a two-level heterogeneous WSN, we need to optimally determine the location of cluster-heads in order to minimize communication power. We require that each sensor node connects to at least p cluster-heads for reliability, and each cluster-head can accept at most q connections. The optimization problem in formulated as a Mixed Integer Nonlinear Programming (MINLP) problem. To overcome the fact that a MINLP solver fails to solve large-scale cases or obtain a global optimum, we propose an iterative decomposition algorithm and use a randomized multi-start technique for global optimization. We also propose an incremental deployment approach and use it to solve the original problem as if the WSN is built incrementally. Numerical results show that the decomposition algorithm is very efficient. While the incremental deployment method is slower in each run, it produces a better solution distribution compared to the pure multi-start approach. Both, however, are capable of solving large-scale problems.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Nonlinear programming, Integer programming*; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless communication, Network topology*

General Terms

Algorithms

*The authors' work is supported in part by the National Science Foundation under grant DMI-0330171, by AFOSR under grants FA9550-04-1-0133 and FA9550-04-1-0208, and by DOE under grant DE-FG52-06NA27490.

Valuetools '07, October 23-25, 2007, Nantes, France

Keywords

Wireless Sensor Network, Clustering, Facility Location

1. INTRODUCTION

A Wireless Sensor Network (WSN) consists of low-cost nodes which are mainly battery powered and have sensing and wireless communication capabilities [13]. Usually, the nodes in such a network share a common objective, such as environmental monitoring or event detection. In designing and deploying a WSN, there are two considerations. First, due to limited on-board power, power saving is crucial in WSNs, since it directly impacts their lifetime in the likely absence of human intervention for most applications of interest. Energy in WSN nodes is consumed by the CPU, by sensors, and by radio, with the latter consuming the most [19]. Second, reliability is another key issue. Due to the volatile nature of wireless channels and the multi-hop nature of transmissions in WSNs, packet delivery has no guarantee.

Various approaches have been proposed to increase the network life time. While some focus on wireless transmission and use specialized MAC/sleeping control mechanisms such as [15], some other approaches use the concept of clustering or hierarchical deployment. Unlike a flat topology WSN, e.g., [7], in a hierarchical deployment, some nodes process different tasks according to their hierarchy level. The reason for clustering in WSNs is that data are usually regionally correlated. By processing data locally instead of transmitting them all the way back to the base, energy can be saved. There are two cases in hierarchical deployment. The first case is that all nodes are homogeneous in hardware but some nodes serve as "cluster-heads" or "fusion centers". These nodes collect information from other sensors, process them and relay the data back to the base station, e. g. [17]. The other case is that the WSN consists of heterogeneous devices, such as [9]. Both cases raise optimization problems regarding how to elect or deploy the cluster-heads to minimize power consumption.

On the other hand, WSN reliability is less addressed comparing to life time maximization. Some approaches are proposed, such as [18] and [6]. The main idea in both approaches are to introduce redundancy. For example, [6] proposes a braided-path routing method, where data is routed simultaneous via multiple paths to increase reliability. In this paper, we consider a static two-layer WSN, as shown in Figure 1, which is typical in the state of the art practice [14]. We pose a cluster-head deployment problem where reliabil-

ity and power consumption are jointly taken into account.

As shown in Figure 1, a two-layer network consists of a lower layer of sensor nodes and an upper layer of cluster-heads. Sensor nodes are fixed, operate at low power, have low computation capability, perform sensing tasks and transmit data to cluster-heads. In this type of network, it is typical that the sensor nodes do not possess routing capability, i.e., data must be routed by a cluster-head. Cluster-heads have greater capability in terms of computation and transmission. In this type of WSN, generally, cluster-heads are powered back-bone nodes which accept data transmitted from sensor nodes. They may process the data on the spot or further upload the data to a base station. In this design problem, we do not consider any possible scheduling scheme among transmissions, as well as possible wireless interference. Instead, we carry out a "worst case" analysis where once a connection is made between a sensor node and a cluster-head, the sensor node transmits at a low constant data rate. For redundancy purposes, we require that each sensor node must connect to at least p cluster-heads. As a capacity constraint, each cluster-head cannot connect to more than q sensor nodes. Our goal is to find: (i) the optimal cluster-head locations and (ii) the optimal network connectivity such that communication power consumption is minimized.

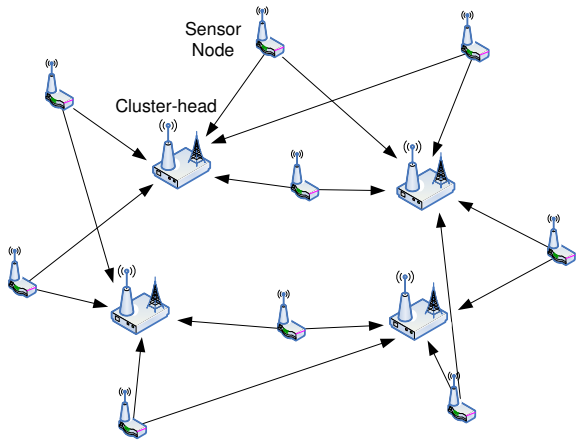

Figure 1: An example of a two-layer sensor network with redundant links.

The cluster-head deployment problem belongs to the category of location-allocation (LA) problems [2][4], which is difficult because it is neither convex nor concave and possesses multiple local minima. Although many approaches are proposed [5], they cannot be applied directly since our problem has its own complicating factors: (i) the allocation for each sensor node is binary while the location of cluster-heads is continuous; (ii) unlike p-center or p-median problems, each sensor node connects to more than one cluster-head; (iii) cluster-heads have capacity constraints due to wireless interference, computing power limits, etc., so the maximum number of sensor nodes under one cluster-head's coverage is limited.

In this paper, we first formulate this deployment problem as a Mixed Integer Nonlinear Programming (MINLP) problem. Although the MINLP problem can be solved using general solvers, only local minima can be obtained and scalability is a serious issue due to the huge memory consumption in the branch-and-bound process. Therefore, we exploit the problem structure and propose a decomposition algorithm based on the sequential location-allocation (SLA) decomposition scheme [3]. The decomposition algorithm is not only much more efficient and scalable, but also compared to the solution obtained by a MINLP solver, the quality is minimally degraded, although it still finds local minima only. To find the global optimal solution, we incorporate a randomized "multi-start" scheme. Since in practice it is common to encounter cases where the existing deployment needs to expand, we propose an incremental deployment approach. This approach utilizes the "incremental-friendliness", i. e., a previous solution can be used to accelerate solving the new expanded problem. Finally, we propose to use the incremental deployment approach to solve the original deployment problem, as if the deployment is grown from a very simple topology till the specification is reached. This new approach, although more computationally expensive, produces a better solution distribution than a pure multi-start approach.

The paper is organized as follows: the MINLP problem is formulated in Section 2. In Section 3, the decomposition algorithm is proposed. In Section 4, we introduce the incremental deployment, including the approach of using such a scheme to solve the original problem. Section 5 shows some numerical results and the conclusions are in Section 6.

2. PROBLEM FORMULATION

We first introduce our notation:

S_i Known fixed location of sensor node i. $i \in I$ where I is an index set. The total number of sensor nodes is $|I|$. We assume that all sensor nodes and cluster-heads are in a plane, so $S_i \in \mathbb{R}^2$.

R_j Controllable location of cluster-head j. $j \in J$ where J is an index set as well. The total number of cluster-heads is $|J|$, and $|J| \geq p$. We assume that J is given and fixed. We also assume that all cluster-heads are in the same plane as the sensor nodes, so $R_j \in \mathbb{R}^2$.

c_{ij} Controllable 0-1 binary variable indicating whether sensor node i is connected to cluster-head j.

T_{ij} Transmission power of sensor node i to cluster-head j.

In typical WSN devices, transmission power level is adjustable [10]. When a sensor node i transmits data to cluster-head j, energy is consumed at rate T_{ij} at the sensor node. To determine T_{ij}, we incorporate a general wireless transmission model. Define $P(T_{ij}, S_i, R_j)$ to be the signal strength received at cluster-head R_j when S_i is transmitting at power level T_{ij}. Function $P(T_{ij}, S_i, R_j)$ is always non-negative, monotonically increasing with respect to T_{ij} and monotonically decreasing with respect to $\|S_i - R_j\|$ which is the Euclidean distance between points S_i and R_j. In this paper, we assume the model in [12]:

$$P(T_{ij}, S_i, R_j) = \frac{\alpha T_{ij}}{\|S_i - R_j\|^d}$$

where α is a constant scaling factor for all i, j and d is the exponent characterizing the signal attenuation with respect to

transmission distance. Usually $2 \leq d \leq 3.5$. During transmission, channel and electronic noise are added. In order to recover the data from a noisy wireless signal, the signal to noise ratio (SNR) $P(T_{ij}, S_i, R_j)/N_0$ must be greater than γ, a threshold determined by the electronic characteristics of the receiver. We assume N_0 and γ are both known constants.

The relationship between c_{ij} and T_{ij} is:

$$c_{ij} = \mathbf{1}\left\{\frac{\alpha T_{ij}}{N_0 \|S_i - R_j\|^d} \geq \gamma\right\}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. If $c_{ij} = 0$, we have $\alpha T_{ij}/N_0 \|S_i - R_j\|^d < \gamma$, and since we want to minimize T_{ij}, we can set T_{ij} to zero. On the other hand, if $c_{ij} = 1$, then $\alpha T_{ij}/N_0 \|S_i - R_j\|^d \geq \gamma$. Since there is no incentive for T_{ij} to be larger than needed, we can conclude that:

$$T_{ij} = \frac{N_0 \gamma \|S_i - R_j\|^d}{\alpha}$$

Therefore, combining both cases, we can see that T_{ij} is a function of c_{ij} and R_j:

$$T_{ij}(c_{ij}, R_j) = c_{ij}\frac{N_0 \gamma \|S_i - R_j\|^d}{\alpha}, \quad i \in I, j \in J \qquad (1)$$

2.1 Optimization problem formulation

As stated, our goal is to minimize total end-point power:

$$\min_{R_j, c_{ij}} \sum_{i \in I}\sum_{j \in J} T_{ij}$$

Using the observation in (1), the main optimization problem is formulated as Problem 1:

PROBLEM 1 (MAIN PROBLEM).

$$\min_{R_j, c_{ij}} \sum_{i \in I}\sum_{j \in J} c_{ij}\frac{N_0 \gamma \|S_i - R_j\|^d}{\alpha} \qquad (2)$$

$$s.t. \sum_{j \in J} c_{ij} \geq p, \quad \forall i \qquad (3)$$

$$\sum_{i \in I} c_{ij} \leq q, \quad \forall j \qquad (4)$$

$$c_{ij} \in \{0, 1\}$$

In this problem formulation, control variables are the connections $c_{ij}, i \in I, j \in J$ and cluster-head locations $R_j, j \in J$ where c_{ij} are binary. The constraint (3) specifies the requirement for at least p connections for each sensor node. The constraint (4) specifies the capacity of q connections for each cluster-head. We assume no explicit limit on T_{ij}; however, the cost for a sensor node to reach a faraway cluster-head will be prohibitive and $T_{ij} \to \infty$ as $\|S_i - R_j\| \to \infty$. Problem 1 is a mixed integer non-linear programming (MINLP) problem which can be solved directly using a MINLP solver, such as MINLPBB [11] which uses branch-and-bound methods. To utilize a solver, we need to provide an initial solution. In Problem 1, as there is no constraint on cluster-head locations and end-point transmission range, we can pick the initial locations of the cluster-heads $\{R_j^0 : j \in J\}$ arbitrarily, and find $\{c_{ij}^0 \in \{0,1\} : i \in I, j \in J\}$ such that the constraints (3) and (4) are satisfied. One way is to choose R_j^0's by uniformly sampling in the bounding box of all S_i's, $i \in I$. Let $S_i = (x_i, y_i)^T$. The bounding box is a rectangle defined by $(\min_I x_i, \min_I y_i)$ and $(\max_I x_i, \max_I y_i)$. We exclude the trivial case where the bounding box is a point. One can also limit the sampling to the convex hull of S_i's, but the bounding box has a simple topology making it easier to sample uniformly within it. To find feasible c_{ij}^0's we can solve a simple auxiliary optimization problem:

PROBLEM 2 (AUXILLIARY).

$$\min_{c_{ij}} \sum_{i \in I}\sum_{j \in J} c_{ij}$$

$$s.t. \sum_{j \in J} c_{ij} \geq p, \quad \forall i$$

$$\sum_{i \in I} c_{ij} \leq q, \quad \forall j$$

$$c_{ij} = \{0, 1\}$$

As a matter of fact, this 0-1 integer programming problem can be cast into a min-cost transportation problem, which is easily solved using network simplex methods. Then, with R_j^0 and c_{ij}^0's, we can invoke the MINLP solver to solve the problem and obtain $R_j^*, j \in J$ and $c_{ij}^*, i \in I, j \in J$ which is the optimal solution to Problem 1. The transmission power of each node can then be obtained using (1). Figures 2(a) and 2(b) show a very small scale example of the problem. In this example, we have 5 sensor nodes and 3 cluster-heads whose locations are to be determined. We require that each sensor connects to $p = 2$ cluster-heads, while each cluster-head can only accept $q = 5$ sensor nodes. We first randomly place the cluster-heads, solve Problem 2 and obtain a feasible solution, shown in Figure 2(a). Then, we solve Problem 1 and the result is shown in Figure 2(b).

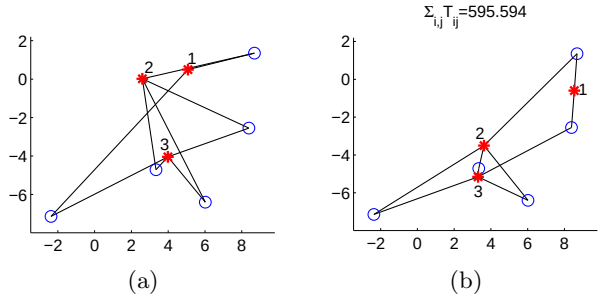

Figure 2: Figure 2(a) visualizes the solution of Problem 2, where circles represent sensor nodes and stars with numbers represent randomly placed cluster-heads. The solution, namely c_{ij}'s, are represented by lines connecting cluster-heads and sensor nodes. Figure 2(b) shows the optimal solution obtained from a MINLP solver (TOMLAB/MINLPBB), with initial feasible solution given in 2(a). The total transmission power is shown at the top.

Although we are able to obtain an optimal solution using a MINLP branch and bound solver, we encounter some problems. The first one is the scalability issue. Due to the massive number ($|I| \cdot |J|$) of integer variables c_{ij} in large-scale cases, branch and bound will introduce a large number of sub-problems which the solver has to keep track of, and this process requires an excessive amount of memory. Second,

the optimal solution depends on the initial feasible solution. Figure 3(a) shows a different initial solution, which results in the optimal solution shown in Figure 3(b).

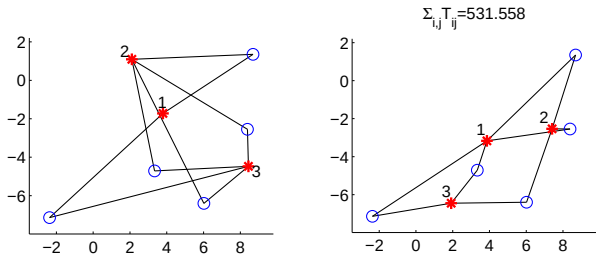

Figure 3: With a different initial feasible solution, shown in Figure 3(a), the solution obtained by MINLP solver, 3(b), differs from the previous optimal solution, shown in Figure 2(b). This implies that the optimal solution is local.

As we can see, Figure 3(b) has a smaller cost than Figure 2(b), which is a convincing fact that the optimal solution obtained from the MINLP solver is local. To find the global optimal solution, randomization in the optimization algorithm is needed. In the following sections, we first provide an algorithm which decomposes the problem into two sub-problems and solve them respectively and iteratively. By decomposition, we reduce the computational complexity, which enables solving larger instances. Then, we introduce randomization for global optimization.

3. SLA-BASED DECOMPOSITION ALGORITHM

Problem 1 is difficult partly because it contains both integer variables c_{ij} and continuous variables R_j. In [3], a general decomposition scheme of the location-allocation problem is suggested, which decomposes the problem into two sub-problem and alternates between finding the optimal location given the allocation, and finding the optimal allocation given the location. This approach is also known as a "sequential location and allocation" (SLA) method. Originally SLA is applied to LA problems, such as p-center or p-median, where the allocation of a demand node is assigned to the nearest supply center, and the capacity of supply centers is unlimited. However, in Problem 1, we have constraints (3) and (4) so the allocation problem is no longer trivial. Notice that the constraints apply on c_{ij} only, while R_j are unconstrained. Therefore, the feasible regions of c_{ij} and R_j are decoupled, in the sense that at one feasible solution, if we fix all c_{ij} and change R_j, the feasibility will be maintained. This implies that we can still apply the SLA idea. The iterative decomposition algorithm is outlined as follows:

Initialization: Assign initial locations to the cluster-heads, namely R_j^1.

At the kth iteration:

Step 1: Solve Problem 1 with R_j being substituted by fixed R_j^k. In effect, Problem 1 reduces to a combinatorial optimization problem determining which sensor node connects to which cluster-head, namely c_{ij}. This is a linear programming problem, and can be further transformed into a transportation problem [16]. By solving the LP, we obtain:

$$c_{ij}^k = \arg \min_{c_{ij}\in\{0,1\}} c_{ij} \frac{N_0\gamma \left\|S_i - R_j^k\right\|^d}{\alpha}$$
$$s.t. \sum_{j\in J} c_{ij} \geq p, \quad \forall i$$
$$\sum_{i\in I} c_{ij} \leq q, \quad \forall j$$

Step 2: Solve Problem 1 with c_{ij} being substituted by fixed c_{ij}^k obtained from Step 1. Problem 1 reduces to an unconstrained optimization of cluster-head locations R_j. This problem is a convex non-linear programming problem, which is easily solvable, e.g., using Newton's method. By solving the problem, we obtain:

$$R_j^{k+1} = \arg \min_{R_j, j\in J} \sum_{i\in I}\sum_{j\in J} c_{ij}^k \frac{N_0\gamma \left\|S_i - R_j\right\|^d}{\alpha}; \quad j \in J$$

Stopping Criterion: If step 2 does not move R_j's substantially, e.g., for all j,

$$\left\|R_j^{k+1} - R_j^k\right\| \leq \varepsilon$$

for a given small $\varepsilon > 0$, we conclude that the algorithm has converged and quit. Otherwise, go to step 1 and repeat the iteration.

In what follows we will describe each step of the iterative decomposition algorithm in detail.

3.1 Step 1: optimizing connections c_{ij}

While the cluster-head locations R_j are fixed, we define:

$$Q_{ij} = \frac{N_0\gamma \left\|S_i - R_j\right\|^d}{\alpha} \tag{5}$$

so we are treating Q_{ij} as the "cost" for establishing a connection from i to j. We formulate the following 0-1 integer programming (IP) problem:

PROBLEM 3.

$$\min_{c_{ij}\in\{0,1\}} \sum_{i\in I}\sum_{j\in J} Q_{ij}c_{ij}$$
$$s.t. \sum_{j\in J} c_{ij} \geq p, \quad \forall i$$
$$\sum_{i\in I} c_{ij} \leq q, \quad \forall j$$

This IP problem can be reformulated as a min-cost flow problem over a bipartite digraph. To see that Problem 3 has an exact LP relaxation, let the digraph be $G = (V, A)$. Let M be the node-arc incidence matrix where:

$$m_{st} = \begin{cases} 1 & \text{node } s \text{ is the tail of arc } t \\ -1 & \text{node } s \text{ is the head of arc } t \\ 0 & \text{otherwise} \end{cases}$$

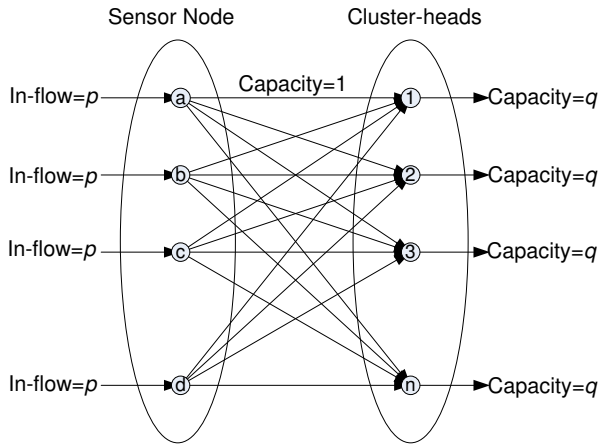

Figure 4: Connection graph between endpoints and cluster-heads. Each link between endpoint i and cluster-head j has capacity 1 and cost Q_{ij}. Each endpoint generates a flow of p, each cluster-head has a sink with capacity q.

and $s \in V, t \in A$. Then, Problem 3 can be reformulated as:

$$\min_{x_a \in \{0,1\}} \sum_{a \in A} Q_a x_a$$

$$\text{subject to } M\mathbf{x} \geq \begin{bmatrix} \mathbf{p} \\ -\mathbf{q} \end{bmatrix}$$

where

$$\mathbf{x} = \left[x_{a_1}, x_{a_2}, ..., x_{a_{|A|}} \right]'$$

$$\mathbf{p} = [p, p, ..., p]', \mathbf{p} \in \mathbf{R}^{|I|}$$

$$\mathbf{q} = [p, p, ..., p]', \mathbf{q} \in \mathbf{R}^{|J|}$$

By the total unimodularity property of matrix M, the integer constraints can be relaxed [16]:

$$0 \leq x_a \leq 1$$

so that it can be efficiently solved by a typical min-cost flow algorithm in polynomial time. After we solve Problem 3, we either encounter an infeasible result, which implies the infeasibility of Problem 1, or we obtain an optimal c_{ij}.

3.2 Step 2: optimizing locations R_j

After fixing the connections c_{ij} in Problem 3, Problem 1 is transformed into a convex optimization problem:

PROBLEM 4.

$$\min_{R_j} \sum_{j \in J} \sum_{i \in I} c_{ij} \frac{N_0 \gamma \left\| S_i - R_j \right\|^d}{\alpha}$$

Since Problem 4 is convex and continuously differentiable, it can be solved easily. In fact, in Problem 4, due to the additivity in the objective function, all R_j are decoupled, making the problem equivalent to a family of sub-problems that can be solved independently and parallely:

$$R_j^* = \arg\min_{R_j} \sum_{i \in I} c_{ij} \frac{N_0 \gamma \left\| S_i - R_j \right\|^d}{\alpha}, j \in J \qquad (6)$$

where $\left\{ R_j^*, j \in J \right\}$ is an optimal solution to Problem 4 if and only if R_j^* is the optimal solution to (6) for all j.

If $d = 2$, then we can immediately differentiate the objective function in (6) and obtain the solution:

$$R_j^* = \frac{\sum_{i \in I} c_{ij} S_i}{\sum_{i \in I} c_{ij}}, \forall j$$

which is obviously the center of mass of all sensor nodes connecting to j. For other $d \neq 2$, numerical methods such as Newton's method can be used.

A special case arises when for some $j_0 \in J$, $c_{ij_0} = 0, \forall i$ so the objective function in (6) equals 0. This indicates that cluster-head j_0 is not used, possibly because it is assigned to an unpopulated area in the initialization step. In this case, we perform a simple heuristic to relocate R_{j_0}:

$$j^* = \arg\max_j \sum_{i \in I} c_{ij} \frac{N_0 \gamma \left\| S_i - R_j \right\|^d}{\alpha}$$

$$R_{j_0} = \frac{\sum_{i \in I} c_{ij^*} S_i}{\sum_{i \in I} c_{ij^*}}$$

where essentially we drop R_{j_0} into the most heavily loaded cluster.

Since step 1 is a linear program, there is no need to use a separate way to find an initial feasible solution, as opposed to the case where a MINLP solver is used. Since both steps 1 and 2 are feasible convex programming problems where the global optimal solution can be found, and we have always maintained feasibility, in each iteration, the objective function value of Problem 1 is always decreasing. The algorithm stops when no further improvements can be made. However, the obtained solution will depend on the initial locations of the cluster-heads, which is similar to the case when an MINLP solver is used.

3.3 Global Optimization

As stated before, Problem 1 is extremely difficult because it is neither convex nor concave and possesses multiple local minima. In solving general LA problems, one can restart SLA algorithm with randomly assigned initial locations, also known as "multi-start", and select the best result from the obtained result set. In [20], it is mentioned that good results can be obtained and for small cases, usually the global optimal point can be obtained. In [1], a probabilistic analysis of the solution distribution is performed where the multi-start results are fitted with a Weibull distribution to estimate the global optimum. Although estimation is fairly accurate for large-scale problems, the estimation is for objective values only and can only serve as a guide. Metaheuristics such as simulated annealing and tabu search have also been proposed [5]. In this paper, however, we use the multi-start approach where initial R_j^1 is randomly assigned each time, because it is easy to implement and the decomposition algorithm is computationally inexpensive.

4. INCREMENTAL DEPLOYMENT

In practice, it is common to encounter the following situation: we need to add new sensor nodes or cluster-heads to an existing network. In the decomposition algorithm, because step 1 is essentially an LP, after adding a new sensor or cluster-head or both, we can start from the previous solution and continue the algorithm. Therefore, the decomposition

algorithm is "incremental-friendly". We will first discuss the case in which a new cluster-head is added.

4.1 Add a cluster-head

Denote by R_s the location of the new cluster-head s. There are two cases when a cluster-head is added: (i) moving existing cluster-heads is not allowed; (ii) moving existing cluster-heads is allowed. We first consider case (i). In this case, we can formulate an optimization problem to determine optimal R_s^*:

PROBLEM 5.

$$R_s^* = \arg\min_{R_s} \sum_{j\in J}\sum_{i\in I} c_{ij}Q_{ij} + \sum_{i\in I} c_{is}\frac{N_0\gamma \left\|S_i - R_s\right\|^d}{\alpha}$$

$$s.t. \sum_{j\in J\cup\{s\}} c_{ij} \geq p, \quad \forall i$$

$$\sum_{i\in I} c_{ij} \leq q, \quad \forall j \in J\cup\{s\}$$

$$c_{ij} \in \{0,1\}, \forall i, \forall j \in J\cup\{s\}$$

where Q_{ij} is from (5) and is fixed.

Problem 5 is also a LA problem which can be solved by the aforementioned decomposition algorithm, where step 2 involves R_s only. However, it can be also expected that the initial location of R_s will affect the final result R_s^*. In this paper, we assume that the initial R_s is randomly chosen within the bounding box of sensor nodes. In case (ii) where existing cluster-heads can also be moved again, the situation is much more complicated. In this case, after we have chosen the initial R_s and set $c_{is} = 0$ for all $i \in I$, we continue with the decomposition algorithm. Same as the previous case, the initial location of R_s will affect the final result R_s^*.

4.2 Add a sensor node

Adding a sensor node is relatively simple because the sensor node is given and fixed in Problem 1. However, adding a sensor node may cause Problem 1 to be infeasible, because the new sensor node's connection demand exceeds the cluster-heads' capacity. To check the feasibility, we can compute step 1. If Problem 3 is infeasible, we need to add a cluster-head and retry step 1, otherwise we can proceed with the decomposition algorithm.

4.3 Solving Problem 1 with incremental deployment

We can actually use incremental deployment to solve Problem 1, and the idea is straightforward. Assume we are to deploy $|I_0|$ sensor nodes, and $|J_0|$ cluster-heads, where I_0, J_0 are the index sets of sensor nodes and cluster-heads to be added, respectively, and $|I_0| \geq 2$, $|I_0| \geq p$. We begin with 2 sensor nodes and p cluster-heads, in which case the optimal deployment of cluster-heads is right at the middle point of the 2 sensor nodes. We apply the decomposition algorithm iteratively with one additional sensor node each time during which a cluster-head is added on a need-basis. Eventually, we will have $|I| = |I_0|$ sensor nodes and

$$|J| = \left\lceil \frac{I_0 p}{q} \right\rceil$$

cluster-heads. Then, we will add the remaining $|J_0| - |J|$ cluster-heads.

When sequentially adding sensor nodes, we can choose the adding order arbitrarily. Here we suggest three ways but these are by no means the only possible ways. The first way is to add the sensor node which is closest to the nodes already added:

$$i^* = \arg\min_{i\in I_0\backslash I} \left\{ \min_{i'\in I} \left\|S_i - S_{i'}\right\| \right\}$$

while the second way is to add the sensor node which is farthest away from the nodes already added:

$$i^* = \arg\max_{i\in I_0\backslash I} \left\{ \min_{i'\in I} \left\|S_i - S_{i'}\right\| \right\}$$

and the third way is to randomly pick a sensor node from $I_0\backslash I$ and add it. It is expected that the final result will depend on the adding-order of sensor nodes so the incremental deployment method is by no means ensuring global optimality. However, in practice, the incremental strategy actually produces good results, as seen in the next section. The reason is that when $|I|, |J|$ are small, we can guarantee global optimality, e. g. $|I| = 2$ case. When new sensor nodes are added one-by-one, in each iteration, the existing cluster-heads are already located at "good" places.

Although we cannot ensure the global optimality of Problem 1, we can still find good solutions with much less computational effort. Because the scale of Problem 1 in terms of number of variables is $O(|I||J|)$, if we solve it using a multi-start technique, the total effort will be at least $O(n|I||J|)$ where n is the number of multi-start trials and is typically large. However, if we solve it incrementally, the total computational effort will be much lower because each time we solve an instance smaller than $O(|I||J|)$ (much smaller at the beginning), and no more than $|I| + |J|$ times. In large-scale problems the advantage will be obvious. Meanwhile, we can also accelerate the incremental process by not solving Problem 1 at every sensor node increment, but every k increments, and hence the computation effort is reduced. In the following section, we will show numerical examples of both the decomposition algorithm and the incremental deployment method to solve Problem 1.

5. NUMERICAL EXAMPLES

5.1 Optimization Result of Decomposition Algorithm

In the implementation of the iterative decomposition algorithm, we used CPLEX[8] for step 1, and KNITRO[21] for step 2. Both solvers are from the TOMLAB suite. The computing platform is a Dell Precision 650, Dual 3.06GHz Xeon CPU, 3.0GBytes RAM, Windows XP SP2, MATLAB 2006b. As stated before, both algorithms find a local minimum of Problem 1. Therefore, we resort to a multi-start technique: in each run, we choose initial R_j's by uniformly sampling in the bounding box of all S_i's, $i \in I$. The results are given in Tables 1 and 2.

Table 1 shows the comparison of computation efforts between the MINLP solver (MINLPBB) and the decomposition algorithm (DECOMP) under various scenarios. The results are obtained by running the algorithm 100 times with random initial cluster-head deployments. Both algorithms use the same initial deployments. We can see that although DECOMP is iterative, its total effort is lower than

Case	$\|I\|$	$\|J\|$	# variables	100 runs time (secs)	
				MINLPBB	DECOMP
1	25	4	108	40	35
2	50	8	416	158	102
3	75	12	924	522	212
4	100	16	1632	N/A	381
5	400	64	25728	N/A	8131

Table 1: Comparison of Computation Effort.

Case	$\|I\|$	$\|J\|$	Best Solution in 100 runs	
			MINLPBB	DECOMP
1	25	4	12673	12673
2	50	8	70888	70888
3	75	12	243416	245120
4	100	16	N/A	441135
5	400	64	N/A	21348391

Table 2: Comparison of Solution Quality.

MINLPBB because both steps in DECOMP are easy convex problems. MINLPBB fails at large scale cases 4 and 5, outputting "out of memory" error. In case 3 where we have 924 variables, the memory consumption by MINLPBB is around 500MBytes, while a smaller case of 416 consumes about 300MBytes of memory. On the other hand, DECOMP can handle a much larger case, such as case 5, on the same computational platform.

Table 2 and Figures 5 to 7 compare the quality of solutions obtained by MINLPBB and DECOMP. Figures 5 to 7 show the quality distribution of solutions. The horizontal axis is the relative cost with respect to the best solution in the 100 runs. For example, in case 1, the best solution out of both cases has value 12673. In Figure 5, category "106%" includes solution values within the interval of $[12673, 12673 \times 106\%)$, and category "113%" includes solution values within $[12673 \times 106\%, 12673 \times 113\%)$. When the best solutions differ between MINLPBB and DECOMP, we choose the lower one as the basis. The vertical axis is the percentage of solutions that fall into a particular relative cost interval. While in the smaller cases 1 and 2, both algorithms found the same best solution, in the larger case 3, MINLPBB outperforms DECOMP by a small margin. This is to be expected since both step 1 and 2 are convex problems in DECOMP so the solving process is always descending, while MINLPBB uses a branch-and-bound method, which allows "restart" at branching. Distributionally, DECOMP has a larger tail, so MINLPBB tends to produce better quality solutions. However, the best solutions of both algorithms are very close in value.

5.2 Incremental Deployment

Table 3, Figures 8 and 9 show a comparison among the MINLP solver, the decomposition algorithm and the incremental deployment approach (nearest increment). We adjusted the number of runs in "MINLP", "DECOMP" and "INC-1" so that they consume roughly the same amount of time. Among "INC-1", "INC-5" and "INC-10", the incremental deployment approach is performed in 3 ways where the decomposition algorithm is invoked upon each, every 5, or every 10 sensor node insertions. Notice that in the latter two cases in Table 3, the elapsed time is not reduced by a factor

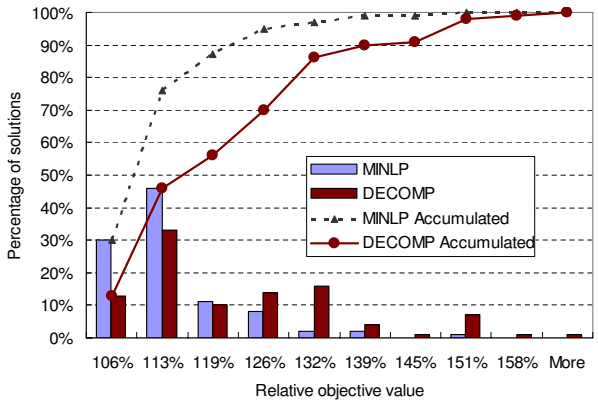

Figure 5: Case 1

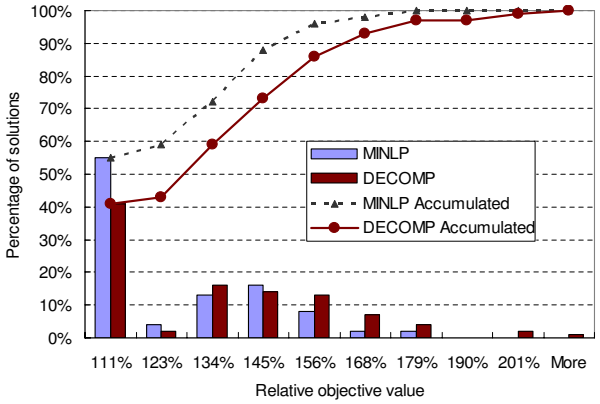

Figure 6: Case 2

of 5 or 10. This is partially due to setup time in each run, as well as the fact that in INC-1 case, since each invocation of the decomposition algorithm involves small increments only, the previous solution can be effectively utilized as a starting point, while in large increment cases (INC-5/10) this is less efficient.

Figure 9 shows that in terms of solution quality, the MINLP solver still leads, although it fails to solve largeriscale cases. The incremental approach produces better quality solutions compared to a pure mulit-start scheme. On the other hand, we can also see that the incremental deployment approach takes much more time than the pure multi-start in each run,

	Runs	Time (secs)	Best Solution
MINLP	150	802	243416
DECOMP	400	825	243472
INC-1	30	857	243576
INC-5	30	349	244226
INC-10	30	246	243472

Table 3: Using MINLP, decomposition algorithm and incremental deployment to solve Case 3.

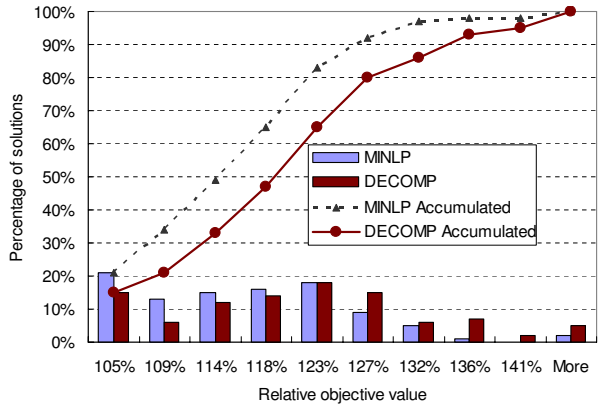

Figure 7: Case 3

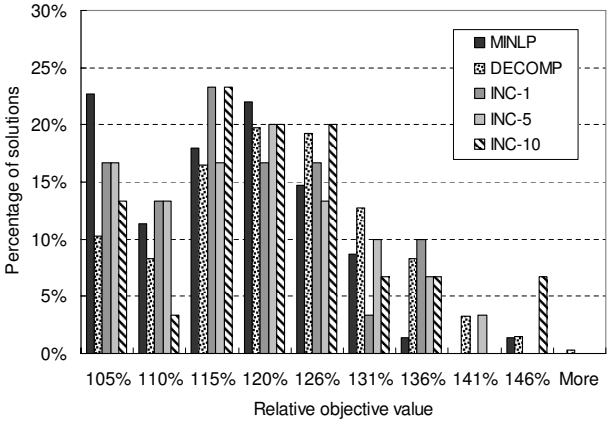

Figure 8: Distribution of solutions

but, overall, if given the same amount of CPU time, we can conclude that the incremental deployment approach is *more likely* to find a better solution than the pure multi-start. Finally, since incremental deployment invokes the decomposition algorithm, we can handle problems of very large scale.

6. CONCLUSIONS

In this paper, we considered a practical problem in WSNs: deploying cluster-heads in order to save energy and provide reliability. We formulated the problem as a MINLP location-allocation problem, which is extremely difficult and contains multiple local minima. In order to combat the scalability issue that arises in a MINLP solver, we exploited structural properties of the problem and developed an SLA-based iterative decomposition algorithm. Since the decomposition algorithm is "incremental-friendly", we proposed an incremental deployment scheme for the scenario where new sensor nodes or cluster-heads are added to an existing deployment. We further proposed to solve the original "whole" deployment problem by way of the incremental scheme. Numerical results show that the incremental deployment scheme produces better solutions in terms of distribution.

The fact that the MINLP branch-and-bound solver can-

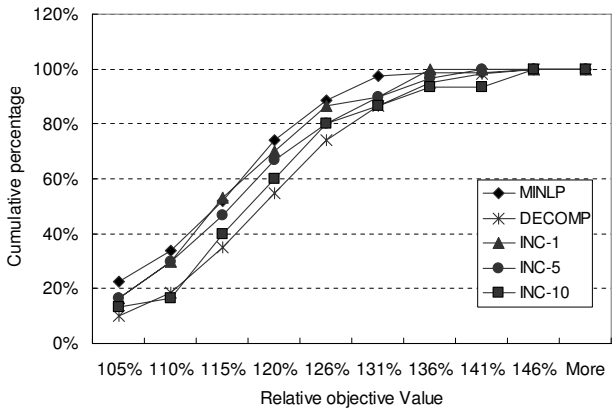

Figure 9: Cumulative distribution of solutions

not handle a large-scale problem is not surprising. There are simply too many branches in each node of the branch-and-bound tree. Since each branching needs to store the current problem plus many new constraints introduced during the branching process, the whole tree will consume a huge amount of memory. One interesting fact is the failure of the MINLP solver to find the global optimal solution even in very simple cases. This is not surprising either. Because even if we relax the integral constraints in Problem 1, it is still non-convex so an initial feasible solution will affect the final result.

For the incremental deployment approach, although it takes more time to compute in our numerical example, we believe it is promising because it has more potential: different ways of incremental deployment and new heuristics may evolve within the framework, leading to efficient solving of the original problem.

Future research directions include: (*i*) incorporating a more detailed model of wireless interference and considering the effect of different medium access control schemes; (*ii*) a probabilistic model regarding the connection, where the new reliability criterion will be probabilistic as well; (*iii*) scenarios where the search space for cluster-head locations is discrete and subject to routing constraints.

7. REFERENCES

[1] M. L. Brandeau and S. S. Chiu. Sequential locaiton and allocation: Worst case performance and statistical estimation. *Location Science*, 1(4):289–298, 1993.

[2] L. Cooper. Location-allocation problems. *Operations Research*, 11(3):331–343, 1963.

[3] L. Cooper. Heuristic methods for location-allocation problems. *SIAM Review*, 6(1):37–53, 1964.

[4] L. Cooper. Solutions of generalized locational equilibrium models. *Journal of Regional Science*, 7(1):1–18, 1967.

[5] Z. Drezner and H. W. Hamacher, editors. *Facility Location: Applications and Theory*. Springer-Verlag, Berlin, Germany, 2002.

[6] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput.*

Commun. Rev., 5(4):11–25, 2001.

[7] J. L. Hill and D. E. Culler. MICA: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov/Dec 2002.

[8] ILOG, Inc. and Tomlab Optimization. TOMLAB/CPLEX v10.0. http://tomopt.com/tomlab/products/cplex, 2007.

[9] A. Iranli, M. Maleki, and M. Pedram. Energy efficient strategies for deployment of a two-level wireless sensor network. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 233–238, New York, NY, USA, 2005. ACM Press.

[10] J. Jeong, D. E. Culler, and J.-H. Oh. Empirical analysis of transmission power control algorithms for wireless sensor networks. *University of California at Berkeley Technical Report*, (UCB/EECS-2005-16), November 21 2005. http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-16.html.

[11] S. Leyffer and R. Fletcher. Dundee solvers for MINLP/NLP/QP. http://www-unix.mcs.anl.gov/ leyffer/solvers.html.

[12] W. Li and C. G. Cassandras. A minimum-power wireless sensor network self-deployment scheme. In *Proceedings of 2005 IEEE Wireless Communications and Networking Conference*, pages 1897–1902, New Orleans, LA, USA, March 2005.

[13] S. Megerian and M. Potkonjak. *Wireless Sensor Networks*. Wiley Encyclopedia of Telecommunications. Wiley-Interscience, New York, NY, January 2003.

[14] Millenial Net Inc. Millenial Net: wireless sensor mesh networking system software and modules. http://www.millennial.net/technology/topologies.asp.

[15] X. Ning and C. G. Cassandras. Dynamic sleep time control in event-driven wireless sensor networks. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 2722–2727, San Diego, CA, USA, December, 2006.

[16] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc, Mineola, NY, 1998.

[17] L. Qing, Q. Zhu, and M. Wang. Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks. *Computer Communications*, 29(12):2230–2237, August 2006.

[18] Y. Sankarasubramaniam, Ö. B. Akan, and I. F. Akyildiz. ESRT: Event-to-sink reliable transport in wireless sensor networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188, New York, NY, USA, 2003. ACM Press.

[19] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200, New York, NY, USA, 2004. ACM Press.

[20] H. Späth. Computational experiences with the exchange method, applied to four commonly IUser partitioning cluster analysis criteria. *European Journal of Operational Research*, 1(1):23–31, Janurary 1977.

[21] Ziena Optimization Inc. and Tomlab Optimization. TOMLAB/KNITRO v5.1. http://tomopt.com/tomlab/products/knitro, 2007.