Efficient mobile sink-based data gathering in wireless sensor networks with guaranteed delay

Charalampos Konstantopoulos
Department of Informatics
University of Piraeus, Greece
konstant@unipi.gr

Grammati Pantziou
Department of Informatics
Technological Educational
Institution of Athens, Greece
pantziou@teiath.gr

Nikolaos Vathis
School of Electrical and
Computer Engineering
National Technical University
of Athens, Greece
nvathis@softlab.ntua.gr

Vasileios Nakos
School of Electrical and
Computer Engineering
National Technical University
of Athens, Greece
billynak@gmail.com

Damianos Gavalas
Department of Cultural
Technology and
Communication
University of the Aegean,
Greece
dgavalas@aegean.gr

ABSTRACT

In this paper, we present a rendezvous-based data gathering protocol for wireless sensor networks employing a mobile sink. For satisfying timely delivery of sensory data to the mobile sink, the mobile sink is forced to visit only an appropriate number of rendezvous nodes while the remaining nodes send their data through multi-hop communication toward the rendezvous nodes. The proposed technique achieves prolonged network lifetime by selecting energy rich paths for this multi-hop communication. Specifically, first the network is partitioned into a number of clusters and then the cluster heads of an appropriate number of energy rich clusters are selected as rendezvous nodes in such a way that (i) the rendezvous nodes are appropriately distributed across the network area to ensure energy efficient data gathering from the other sensor nodes to the rendezvous nodes and (ii) the length of the mobile sink trajectory is below a certain limit. Then, by exploiting the clustering structure, energy efficient routes are determined for all sensor nodes of the network toward the mobile sink trajectory. Experimental results confirm the effectiveness of our approach compared with other competitive approaches from the literature.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed Networks; Network Communications; Wireless Communication

General Terms

Algorithms; Experimentation; Performance

1. INTRODUCTION

Most typical applications of wireless sensor networks (WSNs) involve the collection of sensory data at a base station (sink) often under certain data gathering delay constraints. A fundamental challenge for these networks is to support timely data gathering with minimum energy consumption at the sensor nodes (SNs). The energy requirements due to wireless data transmission, can be reduced by employing a mobile sink (MS) that roams in the network and collects data from the SNs. A number of different approaches exploiting sink mobility for data collection in WSNs have been proposed in the literature. The MS may visit each SN and gather its data via single-hop communication [10, 9] or may visit only some locations of the WSN and SNs send their data to MS through multi-hop communication [2, 5]. In the first approach only single-hop communication is required, therefore energy consumption is minimized, however, at the expense of long data delivery delay. In the second approach, this delay is short but the energy consumption due to multi-hop communication is rather high. Also, SNs should constantly learn the MS's current location thereby creating considerable routing overhead. A solution in between is to have a number of nodes (rendezvous nodes) acting as caching points for data from other SNs. The data stored in the rendezvous nodes (RNs) are sent to the MS when it is within their transmission range [13, 14, 1, 8, 6].

In this article we study the problem of rendezvous-based data gathering defined as follows. We are given a WSN, a MS and a data gathering delay limit. The problem is to determine an appropriate number of RNs, organise the WSN into routing trees each rooted at a RN, and compute a trajectory for the MS to visit the RNs and gather the buffered data. The objective is to maximize the network lifetime defined as the time to the first SN failure due to energy exhaustion, provided that the data gathering delay is lower than the given limit. We formulate the problem as a Mixed Integer Linear Programming (MILP) problem, we prove that it is NP-complete and then we present a novel

heuristic approach for giving a solution to the problem.

The proposed protocol first builds a multi-hop clustering structure over the WSN. Then, a technique is employed to compute the MS trajectory by selecting a subset of the set of CHs as RNs. Since the RNs as well as their neighboring nodes are prone to fast energy depletion due to their data relay overhead, the RNs are selected to be CHs of energy rich clusters which at the same time are appropriately distributed across the network field to ensure energy efficient data gathering from the other SNs i.e. routing trees that minimize the distance from each SN to the corresponding RN. Thus, the MS trajectory is designed to ensure prolonged network lifetime and at the same time its length is kept under the imposed input constraint, guaranteeing timely data delivery. Note that although several rendezvous-based data gathering protocols have been proposed [13, 14, 1, 8, 6], they do not consider SNs residual energy when they build the MS trajectory. As a result, the MS may pass through low energy regions and due to high data traffic observed near the MS trajectory, the RNs or the nodes in their neighborhood may suffer rapid energy exhaustion. The rest of the paper is organised as follows. Section 2 presents the related work and Section 3 provides a MILP formulation of the problem. Section 4 presents the proposed protocol while Section 5 gives the experimental results.

2. RELATED WORK

As mentioned in the introduction, several works exploit sink mobility for data collection in WSNs. In this section we briefly review approaches most relevant to our proposal. In these approaches, the SNs send their data to the RNs which buffer the received data and send them to the MS when MS is within their transmission range [13, 14, 1, 8, 6]. Also, a constraint on the length of the MS trajectory is imposed and the objective is to maximize the network lifetime or minimize the total energy consumption.

In [13], rendezvous-based solutions are presented for fixed MS trajectories and for scenarios where the trajectory is controllable. In the second scenario, which is more relevant to our work, full aggregation is assumed, i.e. at each node all incoming messages are aggregated to a single outgoing message. Also, the objective is to minimize the total energy consumption in the network while respecting the trajectory length constraint. This trajectory is built based on a Steiner tree rooted at a fixed point which connects all source nodes of the network. Specifically, after a pre-order traversal of the tree, a tour on the traversed nodes is determined. This procedure is repeated, increasing the current length bound by a certain amount each time until the imposed length constraint is reached. Then, the subtrees having not been traversed are the trees along which data are routed toward rendezvous nodes. In [14], rendezvous-based solutions are proposed for two scenarios, depending on whether the sink trajectory is constrained or not. For unconstrained trajectory, a Steiner tree rooted at fixed point is built again and then a tour initially empty is incrementally formed. Specifically, at each step, a new node is inserted in the tour which considerably shortens the routing paths from nodes not in the tour to the nearest RNs and also increases the tour length after its insertion by the least amount. The objective in [14] is to minimize the total energy consumption while obeying the tour length constraint. Also, there is no aggregation and thus each node relays the receiving messages intact.

In [1], the objective is to select RNs which minimize the distance of network nodes from the RNs while obeying the length constraint again. Notice that this objective also minimizes the total energy consumption in the network when there is no data aggregation at nodes. The proposed solution repeatedly creates a specific number of clusters in the network. At each iteration, the shortest tour passing through all the clusters is determined and then, if the tour length is below the length limit, the number of clusters to be constructed in the next iteration is increased by a binary-search like method. After the longest sink tour obeying the length constraint has been determined, it is fine-tuned in order to get closer to the length limit. In [8], the proposed solution maximizes the network lifetime and follows a simple greedy approach for finding the MS trajectory and the routing trees rooted at the RNs. At each step, a tentative MS trajectory is determined and then shortest path trees for each RN is then found. The weight of each node is the number of descendants of this node in the routing tree it belongs to times the number of hops separating this node from its RN. Initially, the trajectory includes only the initial position of the MS and a shortest path tree rooted at MS is built all over the network. Then, in the following steps, the MS trajectory is extended by one node at time, namely by the node with maximum weight over all other nodes not in the trajectory. Before inserting the new node in the trajectory, the algorithm checks if the TSP route passing over the new node and all nodes of the existing trajectory exceeds the length constraint. In that case, the insertion is canceled and the algorithm considers the node with the next higher weight.

In [6], a minimum spanning tree is built on the network and then the median of the tree is found. This node has the lowest sum of distances over all tree nodes and is a good starting point for finding a length-constrained trajectory such that sensory data can be gathered at RNs with low energy consumption. Starting from an empty tour, at each step a tree traversal is directed toward the subtree with highest number of nodes which also has not been yet traversed. Then, a tour on the nodes already traversed is built and if its length is still below the length limit, the traversal starts over toward the heaviest non-traversed subtree.

A common feature of the above rendezvous-based solutions is that they do not consider node energies in their approaches. As a result, the MS may pass through low energy regions and therefore, the RNs or the nodes in their neighborhood may suffer rapid energy exhaustion. In contrast, in our work, node energies are considered and the whole data gathering structure including sink trajectory is determined so that only energy-rich nodes are used for relaying data traffic.

3. PROBLEM FORMULATION

The proposed protocol employs a MS which visits the rendezvous nodes of the WSN and gathers all buffered data previously sent by the other SNs. The nodes of the network will be partitioned into a number of groups each corresponding to a different RN. Then, the data from nodes of each group are gathered to the corresponding RN through a tree rooted at that RN which also includes only nodes of this

group. The primary objective in our work is the maximum network lifetime provided that the data gathering delay is lower than a certain limit. In what follows, we formulate the problem at hand as a Mixed Integer Linear Programming (MILP) problem. Let $G(V, E)$ be the directed graph where $\{v_1, v_2, \ldots, v_n\}$ are the nodes of the network and $e_{ij} \in E$ iff v_j is within the transmission range of node v_i. Let also E_{ij} be the energy consumption at node v_i for transmitting one data unit over the edge e_{ij}. Let E_{rcv} denote the energy consumption for receiving one data unit, and f the data units generated at each sampling period at each node. Let also E_i be the available energy at node v_i and d_{ij} the euclidean distance between two nodes v_i, v_j. We also consider a fictitious node v_{n+1} which corresponds to the MS, and the edges $e_{i,n+1}$ $(i = 1, \ldots, n)$. Finally, we assume that $E_{i,n+1}$, $i = 1, \ldots, n$, is the energy consumed for sending one data unit from node v_i to MS which has just arrived at v_i, $(i = 1, \ldots, n.)$. The problem of the maximum lifetime can now be formulated as follows:

$$\max T \quad (1)$$

$$\sum_{i,j=1,i\neq j}^{n} d_{ij} z_{ij} \leq L \quad (2)$$

$$x_k^k = y_k,\ k = 1, \ldots, n \quad (3)$$

$$x_i^k \leq y_k,\ i, k = 1, \ldots, n,\ i \neq k \quad (4)$$

$$\sum_{k=1}^{n} x_i^k = 1,\ i = 1, \ldots, n \quad (5)$$

$$x_{ij}^k \leq \frac{x_i^k + x_j^k}{2},\ i, j, k = 1, \ldots, n,\ i \neq j \quad (6)$$

$$x_{i,n+1}^k \leq \frac{x_i^k + y_i}{2} \leq \frac{1}{2} + \frac{1}{2} x_{i,n+1}^k,\ i, k = 1, \ldots, n \quad (7)$$

$$\sum_{k=1}^{n} \sum_{j=1,j\neq i}^{n+1} x_{ij}^k = 1,\ i, k = 1, \ldots, n \quad (8)$$

$$f_{ij}^k \leq x_{ij}^k M,\ i, k = 1, \ldots, n,\ j = 1, \ldots, n+1,\ i \neq j \quad \text{and } M \text{ is a very large positive constant} \quad (9)$$

$$\sum_{j=1,j\neq i}^{n} f_{ji}^k + f = \sum_{j=1,j\neq i}^{n+1} f_{ij}^k,\ i = 1, \ldots, n \text{ and } k = 1, \ldots, n \quad (10)$$

$$T(E_{rcv} \sum_{j=1,j\neq i}^{n} f_{ji}^k + \sum_{j=1,j\neq i}^{n+1} E_{ij} f_{ij}^k) \leq E_i,\ i = 1, \ldots, n \quad (11)$$

$$\sum_{j=1,j\neq i}^{n} z_{ij} = y_i,\ i = 1, \ldots, n \quad (12)$$

$$\sum_{j=1,j\neq i}^{n} z_{ji} = y_i,\ i = 1, \ldots, n \quad (13)$$

$$2(1 - \delta_i') \leq \sum_{j=1}^{i} y_j \leq \delta_i' + n(1 - \delta_i'),\ i = 1, \ldots, n \quad (14)$$

$$\frac{y_i + \delta_i'}{2} - \frac{1}{2} \leq \delta_i' \leq \frac{y_i + \delta_i'}{2} + \frac{1}{4},\ i = 1, \ldots, n \quad (15)$$

$$u_i - u_j + n(z_{ij} - \delta_i) \leq n - 1,\ i, j = 1, \ldots, n, i \neq j \quad (16)$$

$$x_i^k, x_{ij}^k, y_i, \delta_i', \delta_i \in \{0, 1\}\ i, k = 1, \ldots, n, j = 1, \ldots, n+1, j \neq i \quad (17)$$

$$z_{ij} \in \{0, 1\}, u_i \in R, i, j = 1, \ldots, n, T \in N \quad (18)$$

$$f_{ij}^k \geq 0, i, k = 1, \ldots, n, j = 1, \ldots, n+1, j \neq i \quad (19)$$

In the formulation above, the variables y_i indicate the RNs, the variables x_i^k indicate the fact that the node v_i sends its data to the RN v_k and the variables x_{ij}^k are equal to 1 iff the edge e_{ij} exists, and also this edge belongs to the tree whose root is the node v_k. Finally, the variables z_{ij} indicate that the nodes v_i, v_j are consecutive along the sink trajectory. The objective is to maximize the network lifetime T while the data gathering delay is kept under a certain limit. Notice that the length of the TSP tour over the RNs of the network is a measure of data gathering delay (assuming a constant MS speed) and thus in (2), the length of this tour cannot be larger than the input parameter L.

The constraints (3)-(11) describe feasible rendezvous-based solutions, whereas the constraints (12)-(16) determine a single tour which includes all the elected RNs. Specifically, (3) ensures that each RN v_k should belong to the tree rooted at the same node v_k; (4) ensures that the SN v_i cannot belong to a tree of non existent RN; (5) guarantees that each SN should belong to exactly one tree rooted at a RN; (6) ensures that if $x_{ij}^k = 1$ then $x_i^k = 1$ and $x_j^k = 1$, that is, an edge e_{ij} cannot be a part of a tree if the corresponding vertices v_i and v_j do not belong to this tree; (7) combined with (3) guarantees that $x_{i,n+1}^k = 1$ iff $i = k$ and v_k is a RN, that is, only RNs can communicate with the MS; (8) ensures that each SN has only one parent in the tree of a RN while RNs can only send their gathered data to the MS; (9) ensures that each SN can send data only along valid edges defined in the previous constraints; (10) is the flow conservation constraint at each node while (11) states that the total energy consumption for receiving and sending data at each SN for T rounds should be lower than the initial residual energy of the SN. Note also that (10) with (8) and (9) ensures that the routing structure rooted at each RN is actually a tree and that no routing cycles exist over the tree nodes.

Regarding the TSP problem, (12) and (13) ensure that there exists exactly one outgoing and ingoing edge at each RN along the tour. The constraints (14),(15),(16) eliminate any possible subtours and guarantee that only a single tour including all RNs is built. Specifically, we follow the well-known MTZ technique [7] for subtour elimination. However, unlike in our setting, the above technique assumes that the goal is to find the TSP tour over all the nodes of the input graph. Then, a vertex is singled out, usually the lowest-indexed (e.g. v_1 in our input graph), and a constraint similar to (16) ensures that every tour should contain that vertex. This combined with constraints similar to (12) and (13) guarantees that only a single tour can be constructed including all the nodes of the graph. In our problem, the TSP tour passes only through a subset of nodes and thus the above special vertex cannot be selected in advance. Actually, the constraints (14),(15) help in selecting a vertex from the set of RNs. Specifically, if $\{v_{i_1}, v_{i_2}, \ldots, v_{i_p}\}$ is the set of elected RNs with $i_k < i_{k+1}$ $(k = 1, \ldots, p-1)$, RN v_{i_1} is selected as the special node of the subtour elimination technique. For this node, it holds that $\sum_{j=1}^{i_1} y_j \leq 1$ and $y_{i_1} = 1$ and thus constraints (14),(15) ensure that $\delta_i = 1$ iff $i = i_1$. Then, using the variables δ_i, node v_{i_1} is specially treated in the subtour elimination constraint (16), following similar reasoning as in the original MTZ technique. Note also for any feasible tour $v_{j_1} \rightarrow v_{j_2} \rightarrow \ldots v_{j_p} \rightarrow v_{j_1}$ where j_1 is the lowest index among j_k $(k = 1, \ldots, p)$ we can always

find a valid assignment for u_i $(i = 1, \ldots, n)$, for instance, $u_{j_1} = p$, $u_{jk} = k-1$ $(k = 2, \ldots, p)$ and $u_i = 1$ for any $i \neq j_k$ $(k = 1, \ldots, p)$. In the following lemma, we prove that MILP problem (1)-(19) is an NP-complete problem by a reduction from the TSP problem.

LEMMA 1. *The mixed integer linear programming problem (1)-(19) is a NP-complete problem.*

PROOF. We prove the Lemma through a reduction from the TSP problem. First, we assume that energies E_{ij} and E_{rcv} are all infinite while energies $E_{i,n+1}$ are all of the same finite value. Thus, the lifetime T is non-zero iff the MS visits each node with the length of the MS route not exceeding the bound L. However, this is exactly the decision version of the TSP problem. □

4. THE PROTOCOL

In this section we describe our MS-based data gathering protocol that aims at maximizing the network lifetime defined as the time to the first SN failure due to energy exhaustion, provided that the data gathering delay is below a certain limit. The protocol (Algorithm 7) consists of two stages: a network organization or setup stage (Algorithms 1-5) and an operational data gathering stage (Algorithm 6). In the setup stage, the WSN is appropriately organized i.e. an appropriate number of SNs are chosen as RNs, a routing structure for data collection from all nodes to RNs is built, and a trajectory is computed along which the MS visits the elected RNs. Then the data gathering stage follows where the MS periodically visits the RNs and gathers all data buffered at these nodes. To ensure a fair load balance among SNs, to prevent fast energy depletion of the RNs and their neighbors and hence prolong the network lifetime, the network organization stage is periodically invoked to compute a new set of RNs and rebuild the data gathering structure.

The network organization stage comprises three phases. First, a multi-hop clustering structure is built over the WSN. Then, the CHs of an appropriate number of energy rich clusters are selected as RNs in such a way that (i) the selected RNs are appropriately distributed across the network area to ensure energy efficient data gathering from the other SNs to the RNs and (ii) the length of the trajectory followed by the sink for visiting the RNs and gathering the buffered data is below a certain limit. Finally, in the third phase, data gathering trees rooted at the RNs are determined with the objective of maximizing the network lifetime. In what follows, we describe the three phases in more detail.

Phase 1: Clustering. In this phase, a clustering structure is built over the WSN employing the multi-hop distributed clustering algorithm in [4]. Note also that the energy rich regions around each CH are ideal stops for a MS to gather data coming from the members of the corresponding cluster. For the shake of completeness of the presentation of our protocol we give the basic steps of the *Clustering* algorithm (Algorithm 1); for the detailed description the reader is referred to [4]. Initially, each node u broadcasts at a fixed power level a message announcing its residual energy E_u. Then each node u waits until it receives all the messages sent by its neighboring nodes and builds the sets GE and EQ of neighbors that have more or equal residual energy to u, respectively. Next, u is attached to an appropriate neighbor v and considers v as its parent in the clustering structure, as follows. If the set GE of u is nonempty then u is attached to its neighbour v that maximizes the ratio $\frac{E_v - E_u}{d_{uv}}$. Thus, it is ensured that the sensory data of u will be forwarded toward its CH through an energy rich neighbor and at the same time the one-hop communication with the neighbor is relatively cheap. If the set GE and EQ of u is empty and non-empty respectively, then u is attached to the node v in EQ that minimizes d_{uv}. In the case that both sets GE and EQ are empty, u has the largest residual energy from all its neighbors and becomes a CH. Therefore, it broadcasts a CH announcement message (u, u) to its neighbors announcing its decision to become a CH. When a node u receives a CH announcement message (p, h)) from the node p, if p is its parent node then it sets h as its CH and forwards the CH announcing message (u, h) to its neighbors. Note that a node becomes a CH if it has the highest residual energy in its neighborhood (a local energy-maximum, resulting to empty GE and EQ sets). The CH announcing messages travel along decreasing energy paths and the nodes of each path are associated with the CH at the head of the path. Therefore, the CH has the highest residual energy among all nodes in its cluster.

Algorithm 1: Clustering()

```
begin                                              /* runs at SNs */
    foreach node u do
        u broadcasts E_u to neighbours;
        GE ⟵ ∅;
        EQ ⟵ ∅;
        while a predefined timer has not elapsed do
        /* wait for its neighbors */
            on u receiving (v, E_v) do
                if E_v > E_u then
                    GE ⟵ GE ∪ {v}
                else
                    if E_v = E_u and v > u then
                        EQ ⟵ EQ ∪ {v}
        if GE = ∅ and EQ = ∅ then
            u broadcasts CH announcement (u, u);
        else
            if GE ≠ ∅ then
                parent ⟵ argmax_{v∈GE} (E_v − E_u)/d_uv
            else
                parent ⟵ argmin_{v∈EQ} d_uv
            on u receiving CH announcement (p, h) do
                if parent = p then
                    CH_u ⟵ h;
                    broadcast CH announcement (u, h) ;
```

After the clustering, the sink should learn the IDs and the positions of the CHs, as well as the *neighboring* clusters. Two clusters C and C' are considered as *neighbors* if there are at least two nodes $v \in C$ and $v' \in C'$ that are within the transmission range of one another. The above information

will prove useful in the following Phases 2 and 3. For gathering this information, the MS makes an initial walk across the whole network deployment area following a predefined route. Depending on the speed of the MS, the coverage of the WSN area may take relatively long time, however, it takes place only once i.e. after the first completion of the clustering phase.

Phase 2: MS Trajectory Estimation. Based on the known CHs locations, a trajectory can be computed by the MS for visiting the energy rich regions around each CH and collecting data coming from the members of the corresponding cluster. Unfortunately, such an approach is not always working because the trajectory length may exceed the limit imposed by the need for a guaranteed data gathering delay. Therefore, a technique is needed to restrict the number of stops of the MS by selecting a subset of the set of CHs as rendezvous CHs (RCHs). Since the selected RCHs as well as their neighboring nodes are prone to fast energy depletion due to their data relay overhead, they must be the CHs of energy rich clusters which at the same time are appropriately distributed across the network field to ensure energy efficient data gathering from the other SNs i.e. routing trees that minimize the distance from each SN to the corresponding RCH. Thus, if the sink trajectory is designed to pass through energy rich clusters appropriately distributed across the network area and at the same time the total trajectory length is kept under the imposed input constraint, we can guarantee timely data delivery while also ensuring prolonged network lifetime.

The problem at hand resembles the orienteering problem (OP) [11], an NP-hard problem formulated as follows: Given an edge-weighted graph with profits on its nodes, a depot node s and a time limit B, the goal is to find a tour starting and ending at s with length at most B such that the total profit of the visited nodes is maximized. We may formulate our problem as a variant of the OP by considering the CHs as the nodes of the graph, the depot node as the base station location where the MS starts and ends its trajectory, and the time budget equal to the given data gathering delay D. The profit $Profit_u$ of each CH u is defined as a function of the average energy of the corresponding cluster and the location of the CH in the network field. In the sequel we present the *MS Trajectory Estimation* algorithm (Algorithm 3) for computing the MS trajectory, inspired by the Iterated Local Search (ILS) heuristic which provides near optimal solution almost in real time to the Team Orienteering Problem with Time Windows [12], the extension of OP to multiple tours where each node is also associated with a service time window. The ILS defines an "insertion" and a "shake" step. The insertion step adds, one by one, new nodes in a tour, ensuring that all nodes after the insertion point remain feasible to visit and the time budget is not violated. The shake step is used to escape from local optima. During this step, one or more nodes are removed in each tour looking for non-included nodes that may either decrease the tour time length or increase the overall collected profit.

The modeling of the MS *Trajectory Estimation* algorithm involves two variables for each CH u: (a) $Profit_u$ defined as the average energy of the cluster whose CH is u, normalized to have a maximum value of 100, and (b) $DistC_u$ defined as the distance between the CH u and the circle with center the center of the WSN field (the point where the diagonals meet) and radius equal to 1/4 of the diagonal of the field. Actually, this distance is equal to the difference of the distance of u from the circle center and the radius of the circle. Let $R = v_1, v_2, ..., v_l$, $v_1 = v_l$, be the tour of the current solution and u a CH not included in R. Let $Length_j^u(R)$ be the total length of the solution after inserting the CH u after node v_j in R i.e.,

$$Length_j^u(R) = \sum_{i=1}^{j-1} d_{v_i v_{i+1}} + d_{v_j u} + d_{u v_{j+1}} + \sum_{i=j+1}^{l-1} d_{v_i v_{i+1}}$$

Let also $DistSum_u(R) = \sum_{i=1}^{l} d_{uv_i}$, i.e. $DistSum_u(R)$ is equal to the sum of the distances of the candidate node u from each node in the current solution R. The insertion step (Algorithm 2) proceeds as follows: For each CH u not included in the current solution R (candidate node), u's best possible insert position is determined by computing the least total length $Length_u(R) = min_{j \in \{1,...,l-1\}} Length_j^u(R)$ over all possible insert positions in R. If $Length_u(R) \leq L$ where L the trajectory length constraint imposed by the data gathering delay D, the algorithm calculates the ratio

$$ratio_u = \frac{Energy_u * DistSum_u(R)}{Length_u(R) * DistC_u}$$

which represents a measure of how profitable in terms of energy is to insert u and how this insertion contributes towards an appropriate distribution of the nodes in the network area versus the total length of the tour after the insertion, and the distance of u from the circle with center the center of the field and radius equal to 1/4 of the diagonal of the field. Among all candidate nodes, the heuristic selects for insertion the one with the highest ratio.

Algorithm 2: Insert Step(R)

```
begin
    Candidates ⟵ the set of CHs not included in the
    current solution R;
    maxRatio ⟵ −∞;
    foreach u ∈ Candidates do
        pos_u ⟵ position of u in R that gives the
        Length_u(R);
        if Length_u(R) < L then
            ratio_u ⟵ Energy_u*DistSum_u(R) / (Length_u(R)*DistC_u);
            if ratio_u > maxRatio then
                bestCandidate ⟵ u;
                maxRatio ⟵ ratio_u;
                Pos ⟵ pos_u;
    if maxRatio > −∞ then
        Insert bestCandidate at Pos in R;
        return the new route R;
    else
        local optimum ⟵ true ;
```

The MS *Trajectory Estimation* algorithm defines the value of a solution $R = v_1, v_2, ..., v_l$, $v_1 = v_l$ as the ratio

$$\frac{TotalEnergy(R) * TotalDist(R)}{TotalDistC(R) * Size(R)}$$

where $TotalEnergy(R) = \sum_{i=1}^{l-1} Energy_{v_i}$ is the total energy of all nodes in R, $TotalDist(R) = \sum_{i=1}^{l-1} \sum_{j=1}^{l-1} d_{v_i v_j}/2$ is the sum over all i of the distances from node v_i in R to all other nodes in R, $TotalDistC(R) = \sum_{i=1}^{l-1} DistC_{v_i}$ is the sum of the distances from all nodes in R to the circle with center the center of the field and radius equal to 1/4 of the diagonal of the field, and *Size(R)* is the number of the nodes in R. Note that the algorithm favors solutions consisting of high energy CHs that are far away from each other to span the whole network, and are locating close to the circle with center the center of the field and radius equal to 1/4 of the diagonal of the field to avoid solutions whose nodes are close to the borders of the network field. The locations of the elected RCHs and their distribution in the network field facilitate Phase 3 of the protocol to construct routing trees with low distance paths from each SN to its corresponding RCH. This eventually leads to low-hop communication between SNs and RCHs therefore to reduced energy consumption.

The MS *Trajectory Estimation* algorithm (Algorithm 3) loops up to a specified number of times as long as the value of the best solution is not improved. Inside the loop, the insertion step is applied until a local optimum is reached. If the current solution's value is larger than that of the best so far solution, the current solution is kept as the best solution. In the sequel, the shake step is applied, where the algorithm tries to escape from a local optimum by removing a number of nodes from the current solution, in search of non-included nodes that improve the value of the value of the solution. The shake step takes as input two integers: (a) $RNum$ that determines the number of the consecutive nodes to be removed from the current solution and (b) $SNum$ that indicates where to start removing nodes on the tour of the current solution. If, throughout this process, the base station is reached, then the removal continues with the nodes following the base station. After the application of the shake step, the values of its parameters are adapted as follows: the value of $SNum$ is increased by the value of $RNum$ and the value of $RNum$ is increased by one. If $SNum$ is greater than l to the size of the current solution, then $SNum$ is decreased by this size. If $RNum$ equals to $\frac{|C|}{3k}$ where C is the set of all CHs elected at Phase 1, then it is reset to one.

Phase 3: Estimation of routing paths toward RCHs
After Phase 2, the subset of RCHs residing on the MS trajectory has been determined. All other SNs should send their data over multi-hop paths toward these CHs which, then, will relay the buffered data when MS will be within their transmission range. The proposed approach for creating these paths runs in two stages. The first stage takes place in the MS, and it uses the clusters created in the first stage. Specifically an overlay graph is created $G_C(V_C, E_C)$ where V_C is the set of CHs and E_C is the set of edges between CHs. Two CHs are linked by an edge when their corresponding clusters are neighboring in the sense described above. Moreover, each edge e_{ij} is associated with a distance d_{ij} which is the euclidean distance between CHs i and j. On this overlay graph, MS runs the *Cluster Tree Building* algorithm (Algorithm 4) which creates a number of routing trees, one per each RCH. The algorithm is similar to Dijkstra's or Prim's algorithm and starts by first "compressing" all RCHs into one fictitious node. At each step, the algorithm fixates the routing path of an additional node. Let d_i denote the sum of edge distances along the routing path of CH i which the algorithm has already decided about. Note that for nodes not yet processed, the value of that parameter is infinite from the initialization of the algorithm. Now, the next node whose route is going to be fixated, is the node j which is connected to the so far constructed routing tree via an edge (i, j) such that the ratio $\frac{E_i}{(d_i+d_{ij})^2}$ is the minimum over all other nodes which are not in the tree. The rational behind this criterion is to minimize the distance of each CH toward the MS trajectory. This is expected to lead to reduced energy consumption since the packets from this cluster will likely be relayed through relatively low number of hops. On the other hand, in order to ensure long network lifetime, the relay nodes along the paths toward the MS trajectory should have sufficient energy for successfully handling the increased data volume coming from upstream nodes. So, apart from minimizing the distance to the MS trajectory, the energy of the immediate relay node (node i) of the node j should have plenty of energy and this factor is considered when selecting the next node to insert into the routing tree.

After the first stage, MS has a synoptic view of how data are routed across the network. Specifically, for each cluster, it has been determined which neighboring clusters will receive the data from this cluster on the way to the MS trajectory. This set includes not only the parent of the cluster in the output tree of the Cluster Tree Building algorithm but also all neighboring clusters located at levels of the tree higher than that of this cluster. Consequently, the traffic coming

Algorithm 3: Trajectory Estimation(maxIterations)

```
begin                                              /* runs at the MS */
    ValBestSol ⟵ 0;
    R ⟵ ∅;
    NotImproved ⟵ 0;
    SNum ⟵ 1;
    RNum ⟵ 1;
    local optimum ⟵ false;
    while NotImproved < maxIterations do
        while not local optimum do
            Insert Step(R);
        if (TotalEnergy(R)*TotalDist(R)) / (TotalDistC(R)*Size) > ValBestSol then
            Sol ⟵ R;
            ValBestSol ⟵ (TotalEnergy(R)*TotalDist(R)) / (TotalDistC(R)*Size(R));
            RNum ⟵ 1;
            NotImproved ⟵ 0;
        else
            NotImproved ⟵ NotImproved + 1;
        Shake Step with parameters RNum, SNum ;
        SNum ⟵ SNum + RNum;
        RNum ⟵ RNum + 1;
        if SNum > Size(R) then
            SNum ⟵ SNum − Size(R)
        if RNum > |C|/3 then
            RNum ⟵ 1;
    return Sol;
```

Algorithm 4: Cluster Tree Building

Input: the cluster overlay graph $G_C = (V_C, E_C)$
Output: $T = (V_T, E_T)$: spanning forest of G_C with trees rooted at Rendezvous CHs

```
begin                                              /* runs at the MS */
    V_T ⟵ ∅;
    E_T ⟵ ∅;
    for u ∈ V_C do
        if u is a RCH then
            V_T ⟵ V_T ∪ {u} ;
            d_u ⟵ 0;
        else
            d_u ⟵ ∞
    while |V_T| < n do
        (u, v) ⟵ argmax_{u∈V_T, v∈V_C−V_T} E_u/(d_u+d_uv)^2 ;
        V_T ⟵ V_T ∪ {v} ;
        E_T ⟵ E_T ∪ {(u, v)};
        d_v ⟵ d_u + d_uv
```

from a cluster is dispersed to more than one relay clusters and this leads to better balanced energy consumption across the network. Notice also that by selecting relay clusters for a cluster always from tree levels higher than that of this cluster, we ensure that no rooting loops are created and the traffic from this cluster will reach the MS trajectory eventually.

Algorithm 5: Intra Cluster Route Finding

```
begin                                              /* runs at SNs */
    foreach node u do
        CH_u = the CH of u;
        parent ⟵ nil;
        if u is a gateway or a RCH then
            minD ⟵ 0;
            minE ⟵ E_u;
            u broadcasts (u,CH_u,minD,minE) to its
            neighbors;
        else
            minD ⟵ +∞;
            minE ⟵ −∞;
            on u receiving (w,CH_w,minD′,minE′) do
                if CH_u = CH_w then
                    if minE′ > minE or minE′ = minE
                    and minD′ < minD then
                        parent ⟵ w;
                        minD ⟵ minD′ + d_wu;
                        minE ⟵ min{minE′, E_u};
                        u broadcasts (u,CH_u,minD,minE)
                        to its neighbors;
```

Since, the relay cluster information is important for each CH in the network, MS disseminates this information by visiting the CHs up close. This second walk of the MS across the field is much shorter than the first one since now the MS visits only the CHs. Then, each CH broadcasts the relay information to their members. The cluster members with neighbors in the relay clusters will serve as gateways and forward all data from the other cluster members to these neighbors (*relay nodes*). Now, all remaining nodes of each cluster should determine intra-cluster routing paths toward one of these gateway nodes. To this end, the gateways initiate a local distance-vector protocol (Algorithm 5) and each node determines the most energy efficient path for sending its data to a gateway. Specifically, the cost of a path in this protocol is defined as the minimum node energy found along the path and the protocol finds the path maximizing this minimum. For avoiding routing loops, in case of two paths with the same cost, the path with shorter total length is preferred.

The above discussion completes the description of the network organization (setup) stage of the protocol. Now, during the operational data gathering stage of the protocol (Algorithm 6), the gateways of each cluster send the incoming traffic to their relay nodes alternatively according to a certain probability. Specifically, each relay node is selected with a probability proportional to its current energy. Thus, data are always forwarded through energy rich neighbors thereby avoiding rapid energy exhaustion in the local area. Also, the relay nodes send their energy level periodically to gateways so that the above traffic sharing is according to the current local energy distribution.

Algorithm 6: Data Gathering

```
begin                                              /* runs at SNs */
    foreach node u do
        on u sensing or receiving a new message do
            if u is a RN then
                u sends buffered data when MS in range;
            else if u is a gateway then
                RlN_u = the set of relay nodes of u;
                u sends its buffered data to node w ∈ RlN_u
                with probability p_w where p_w = E_w / Σ_{v∈RlN_u} E_v;
            else          /* ordinary cluster member */
                u sends buffered data to its single parent ;
```

In addition, our algorithm runs re-clustering periodically and then build again the MS trajectory and all routing structures for data gathering at RNs (Algorithm 7). The aim of this periodic rebuilding is to relieve the nodes at higher tree levels from the heavy data traffic, which rapidly reduces their residual energy. These nodes are moved lower in the trees and replaced with nodes with higher residual energy. Once again, MS should learn the new CHs after re-clustering and then, after the cluster tree building, MS should also inform CHs about their relays. In order to speed up this information transfer among MS and CHs, the existing routing infrastructure is employed. Specifically, the MS trajectory, the RNs and the remaining routing structure are kept throughout the new rebuilding and they are disposed only after the end of the on-going rebuilding. Specifically, the new CHs after re-clustering send their IDs and their neighborhood to their old RCHs using the old trees. Thus, MS need only visit the CHs along its old trajectory to find this information. Then, after the completion of the Cluster Tree Building Algorithm, MS uses again the old RCHs along

Algorithm 7: Proposed Protocol

```
begin
    foreach node u do
        E_u = the current energy of u ;
    Clustering();
    MS visits all nodes to find the new CHs ;
    MS Trajectory Estimation();
    Cluster Tree Building();
    MS visits and gives each CH its Relay Clusters;
    foreach CH u do
        u sends Relay Cluster information to its members;
    Intra Cluster Route Finding();
    while true do
        Data Gathering() ; /* normal network operation
        till the next rebuilding is triggered */
        foreach node u do
            E_u = the predicted energy of u just after the
            current rebuilding;
        Clustering();
        foreach CH u do
            u sends its ID and neigboring clusters to its old
            RCH;
        MS collect the above information from the old
        RCHs;
        MS Trajectory Estimation();
        Cluster Tree Building();
        MS gives the Cluster Tree Building output to old
        RCHs;
        foreach old RCH u do
            u sends all new CHs in its tree their relay
            clusters;  /* New CHs learn through the old
            routing tree */
        Intra Cluster Route Finding();
```

the old trajectory to disseminate the relay information for each new CH. Each old RCH knows which of the new CHs are its descendants in the old trees and forward the received relay information to these CHs.

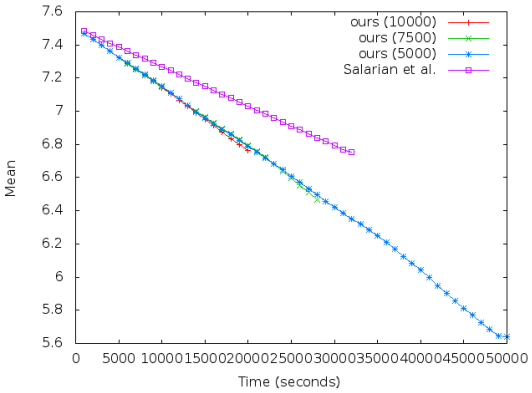
Due to these two relatively slow update rounds, rebuilding may not be a short procedure. However, the normal operation of the network (i.e. sensory data gathering) can be seamlessly extended during the rebuilding phase, since the old routing structure still exists until rebuilding ends. Essentially, rebuilding is run in the background while the network continues its data gathering operation. But now, due to the duration of rebuilding and the parallel data gathering operation of the network, the node energies at the end of rebuilding may differ a lot from those when rebuilding started. Thus, the newly constructed routing structure may not be the best-suited for the energy distribution across the network as there exists at the end of rebuilding. For solving this issue, re-clustering and all other steps of rebuilding should work with predicted node energies, namely with those at the end of rebuilding. For a precise prediction, each SN should know the duration of rebuilding and should also estimate its energy consumption rate since the last rebuilding. Each SN can find the rebuilding duration by simply estimating the time required for MS to traverse its trajectory twice, since this time largely determines the whole rebuilding time. This travel time can be easily found since the old MS trajectory is still followed during the on-going rebuilding. For the energy consumption rate, again the old routing structure is used for sensory data gathering and as each SN receives and sends a constant number of packets (assuming a low number of retransmissions) along this routing structure since the last rebuilding, the consumption rate is expected to be constant on average and thus can be approximated by the formula (current SN energy - SN energy after last rebuilding)/ (current time - time of last rebuilding).

5. EXPERIMENTAL RESULTS

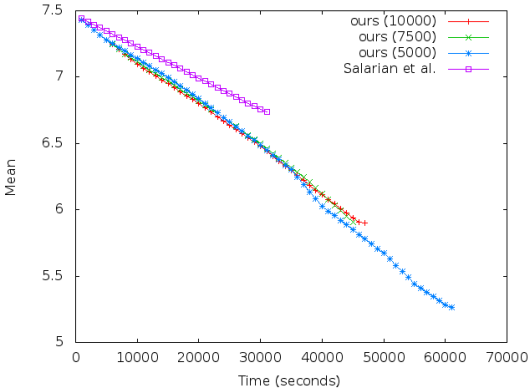
We have carried out a number of simulations tests (Figures 1-4) on the Castalia simulator [3] where our protocol has been compared against the protocol of Salarian et al. [8]. We selected this work because it is very recent and appears to be the best among other competitive approaches in many respects. In simulation tests, we assume 600 SNs spread over a terrain of 500m×500m. We have also set the radio bandwidth at 20 MHz, and the transmission power at one of the following levels 0, -1, -3, -5, -7, -10, -15, -25 dbm depending on the distance of the intended recipient. We have run experiments where the initial battery power of each SN is uniformly selected from 50 to 100 joules as well as experiments with the same initial battery, namely set at 75 joules. We also assume that each SN sends one packet per 10 seconds and each experiment runs on a different random topology. In addition, we considered different frequencies for the periodic rebuilding, specifically every 10000, 7500, and 5000 secs. The speed of the MS was set at 1 m/s and the length limit L at 600m in Figures 1-3. It is worth mentioning that the curves in Figures 1-3 are drawn up to certain time moment which may be different for each curve. These time moments is the time when the first SN dies, that is the network lifetime which differs for each simulated scenario.

Figure 1 illustrates the average residual energy of SNs over the time. [8] achieves higher residual energy at each SN on average than our algorithm. However, this should be expected because the objective in [8] is to minimize the total energy consumption across the network which in turn leads to higher average node energy. However, despite the higher average energy levels of [8], balanced energy consumption over all SNs and hence long lifetime may not be attainable. It may be the case that many SNs distant from the MS trajectory have plenty of energy while a relatively smaller number of SNs near the trajectory have very low energy. In this scenario, the network lifetime is very short despite the higher average energy over all SNs. This will be confirmed in Figures 2 and 3. Another interesting conclusion drawn from Fig. 1 is that the average energy in our protocol is virtually not affected by the rebuilding frequency, which implies that this phase can be executed with relatively low energy cost.

Figure 2 shows the deviation of the residual node energy over time for the two protocols. With different initial node energies, our protocol achieves lower deviation levels than [8], while for equal energies, when rebuilding is relatively frequent, our algorithm excels after a while. The better performance of our algorithm for different initial node energies should be expected since the full potential of our approach is attained in this setting. Notice also that a low deviation

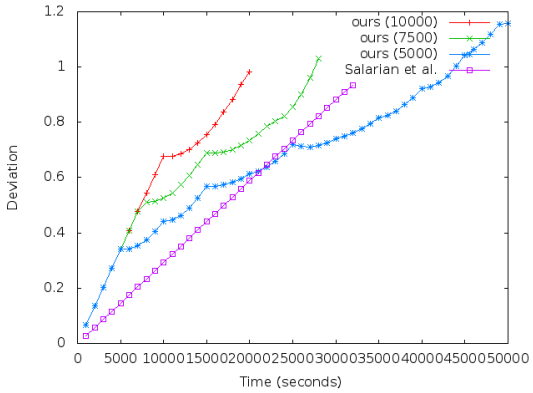


(a) Same initial node energies

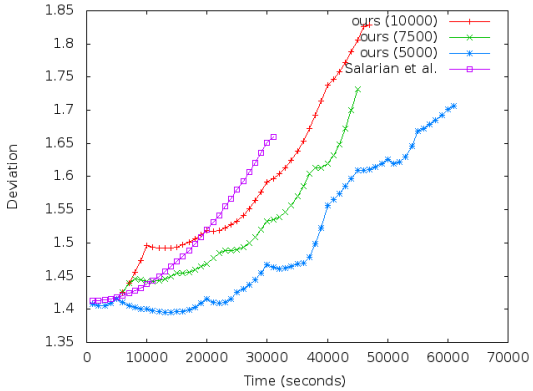


(b) Different initial node energies

Figure 1: Average node residual energy



(a) Same initial node energies



(b) Different initial node energies

Figure 2: Deviation of node residual energy

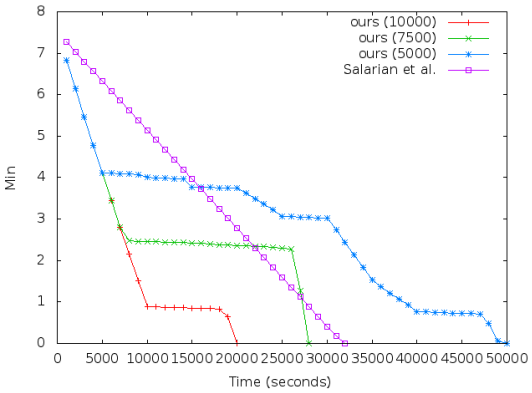
level clearly indicates balanced energy consumption across the network which in turn leads to prolonged network lifetime. Figure 2 also reveals an interesting pattern for the energy deviation. Specifically, after each rebuilding, the deviation falls as the last rebuilding drives packets through energy-rich SNs. Thus, the high energy level of these SNs is gradually decreasing and hence their difference from the nodes of lower energy. Thus, the deviation decreases until the energy of the formerly energy rich nodes falls under the average energy level across the network. After that point, the deviation starts to increase again. Clearly, the right moment for the next rebuilding is at the start of the deviation increase. However, this cannot be easily predicted and thus rebuilding is done periodically in our scheme.

Figure 3 shows the energy of the most energy deprived node of the network over time. This SN may not be the same all the time especially after rebuilding. This can be easily seen by looking at how the gradient of the curve of our algorithm is varying. If there was one node (actually not too many nodes) that constantly had lower energy than all the others, the curve would be a straight line, as in [8]. However, the fact that the curve in our scheme is not linear clearly indicates that the SN with minimum available energy is changing over time. Notice also that the time when this minimum node energy is zeroed out always signifies the network lifetime. Regarding the relative performance of the two protocols, it can be clearly seen that for the case of the same initial node energies, rebuilding every 5000 seconds keeps the minimum node energy at higher levels than [8]. For different initial node energies, regardless of rebuilding frequency, our scheme outdoes [8], which is expected since node energy is an important parameter in our algorithm and is taken into account during RN selection and route building. In contrast, in [8], node energies are not considered. Figure 4 illustrates the network lifetime of the network as a function of the maximum length of the route L. Again we see that our algorithm performs better in almost all cases for both the same and different initial node energies. As is expected, the performance gap between our algorithm and the algorithm in [8] is wider in the case of different initial node energies and our algorithm outperforms [8] regardless of the rebuilding frequency in that case.

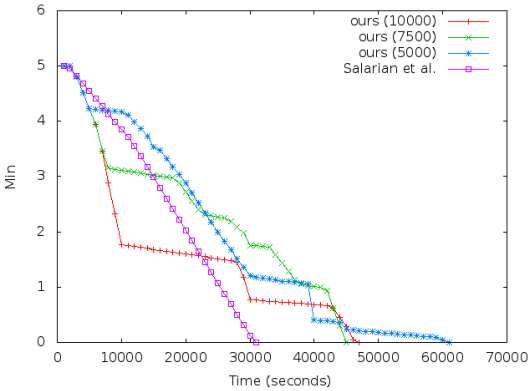
Acknowledgments. This research has been co-financed by the EU and Greek national funds through the NSRF Research Funding Program: Archimedes III.

6. REFERENCES

[1] K. Almi'ani, A. Viglas, and L. Libman. Energy-efficient data gathering with tour length-constrained mobile elements in wireless sensor

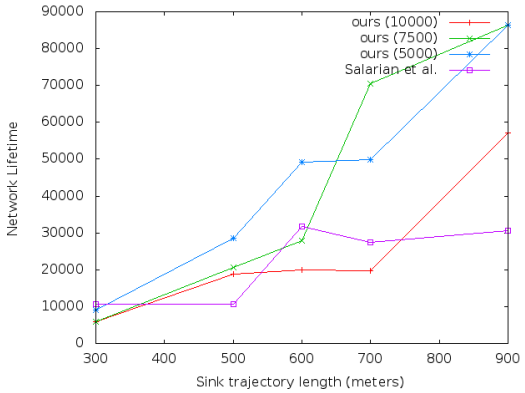


(a) Same initial node energies

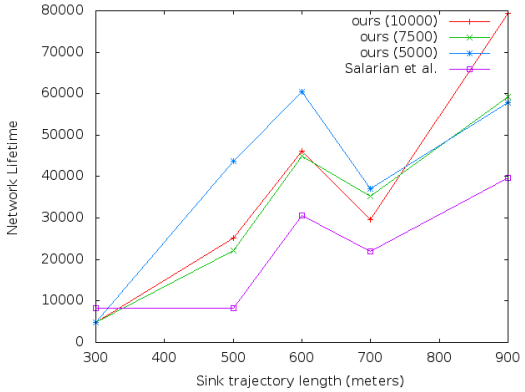


(b) Different initial node energies

Figure 3: Minimum node residual energy



(a) Same initial node energies



(b) Different initial node energies

Figure 4: Network lifetime

networks. In *35th IEEE Conf. on Local Computer Networks (LCN)*, pages 582–589. IEEE, 2010.

[2] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z. M. Wang. Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wireless Networks*, 14(6):831–858, 2008.

[3] Castalia. https://castalia.forge.nicta.com.au.

[4] C. Konstantopoulos, B. Mamalis, G. Pantziou, and V. Thanasias. Watershed-based clustering for energy efficient data gathering in wireless sensor networks with mobile collector. In *Euro-Par 2012 Parallel Processing*, pages 754–766. Springer, 2012.

[5] J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM 2005. 24th Conference of the IEEE computer and communications societies. Proc. IEEE*, volume 3, pages 1735–1746. IEEE, 2005.

[6] L. Mai, L. Shangguan, C. Lang, J. Du, H. Liu, and Z. Li. Load balanced rendezvous data collection in wireless sensor networks. In *8th IEEE Intl.l Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 282–291. IEEE, 2011.

[7] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.

[8] H. Salarian, K. Chin, and F. Naghdy. An energy efficient mobile sink path selection strategy for wireless sensor networks. *IEEE Transactions on Vehicular Technology*, PP(99):1–1, 2013.

[9] R. Sugihara and R. Gupta. Improving the data delivery latency in sensor networks with controlled mobility. In *IEEE Intl. Conf. DCOSS*, volume 5067 of *LNCS*, pages 386–399. Springer, 2008.

[10] R. Sugihara and R. K. Gupta. Optimal speed control of mobile node for data collection in sensor networks. *IEEE Trans. Mobile Computing*, 9(1):127–139, 2010.

[11] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984.

[12] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.

[13] G. Xing, T. Wang, W. Jia, and M. Li. Rendezvous design algorithms for wireless sensor networks with a mobile base station. In *9th ACM Intl. Symp. MOBIHOC' 2008*, pages 231–240. ACM, 2008.

[14] G. Xing, T. Wang, Z. Xie, and W. Jia. Rendezvous planning in wireless sensor networks with mobile elements. *IEEE Transactions on Mobile Computing*, 7(12):1430–1443, 2008.