Computer Networks 57 (2013) 1124–1138

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Active node determination for correlated data gathering in wireless sensor networks

Efe Karasabun, Ibrahim Korpeoglu *, Cevdet Aykanat

Bilkent University, Department of Computer Engineering, 06800 Ankara, Turkey

ARTICLE INFO

Article history:
Received 4 January 2011
Received in revised form 5 July 2012
Accepted 8 November 2012
Available online 22 December 2012

Keywords:
Wireless sensor networks
Correlated data gathering
Active sensor node determination

ABSTRACT

In wireless sensor network applications where data gathered by different sensor nodes is correlated, not all sensor nodes need to be active for the wireless sensor network to be functional. Given that the sensor nodes that are selected as active form a connected wireless network, the inactive sensor nodes can be turned off. Allowing some sensor nodes to be active and some sensor nodes inactive interchangably during the lifecycle of the application helps the wireless sensor network to have a longer lifetime. The problem of determining a set of active sensor nodes in a correlated data environment for a fully operational wireless sensor network can be formulated as an instance of the connected correlation-dominating set problem. In this work, our contribution is twofold; we propose an effective and runtime-efficient iterative improvement heuristic to solve the active sensor node determination problem, and a benefit function that aims to minimize the number of active sensor nodes while maximizing the residual energy levels of the selected active sensor nodes. Extensive simulations we performed show that the proposed approach achieves a good performance in terms of both network lifetime and runtime efficiency.

1. Introduction

Wireless sensor networks (WSNs) are composed of a large number of spatially distributed sensor nodes that are limited in power. These sensor nodes are equipped with three main components to cooperatively collect information about a monitored region. These three main components of a sensor node are a processing unit with limited capability, environment sensors and a short-range wireless transceiver. By the use of these components, sensor nodes can form a multi-hop wireless network and transmit the sensed data about the monitored environment to a data gathering node. Sensors are able to obtain information about the monitored environment including but not limited to temperature, humidity, pressure, sound and motion. Some WSN applications include environment and habitat monitoring, health-care assistance, home automation, industrial process monitoring and control, and battlefield and border surveillance.

Limited energy availability in sensor nodes makes network lifetime an important issue in WSN applications. To extend the network lifetime, energy efficient wireless sensor network protocols and algorithms have been devised in the literature. Node clustering, in-network data processing, data fusion and network coding are some of the measures taken to reduce the amount of data that is processed, sensed or transmitted. Minimization of energy spent in processing, sensing and transmitting of data allows sensor nodes to save energy. Such energy savings help to extend the lifetime of WSN applications.

In some WSN applications, not all sensor nodes are required to be active (turned on, thus spending energy) in order for the WSN application to be fully functional. One example to these types of applications can be environment monitoring applications. In these types of applications, exploiting the inherent data correlations among the sensor devices may help to prolong the network lifetime extensively. The data correlations between the sensor

* Corresponding author. Tel.: +90 3122902599.
E-mail address: korpe@cs.bilkent.edu.tr (I. Korpeoglu).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.11.018

devices may exist due to the characteristics of a sensor region and sensor node deployment such as the proximity of the sensor nodes. Moreover, correlations may be static (not changing over time) for some applications, especially for applications where data sensed by sensor nodes depend on the location of the nodes. In such applications, as long as sensor node are static (not moving), the data sensed will be location-dependent and there will be static correlations among the data sensed by nearby nodes.

The data correlations among sensor nodes can be modeled as a set of two-tuples, where each tuple contains a source set of nodes that infers a sensor node. When a source set is selected into the active sensor node set, the sensor node inferred by that source set may stay inactive. In these types of WSN applications, since the data of some sensor nodes can be inferred from the data of some other nodes, it is crucial to determine the set of active sensor nodes that can be sufficient to infer the data of inactive sensor nodes. In such scenarios, only the active sensor nodes need to sense, process and transmit data. The inactive nodes will be turned off and therefore they will not spend any energy as long as they remain inactive.

In this work, we aim to find an effective and runtime-efficient centralized active sensor node selection heuristics for correlated data gathering in WSNs to prolong the sensor network lifetime. For this purpose, we model the active node determination problem as an instance of the connected correlation-dominating set problem [1]. In connected correlation-dominating set problem, given a network and correlation information about which node subsets, i.e., *source sets*, infer which other nodes, we are interested in finding a set of (dominating) nodes that can infer the (correlated) data of the rest of the nodes. Gupta et al. [1] proposes a sophisticated but time-consuming constructive *L*-hop centralized heuristic. The objective of the *L*-hop centralized heuristic is to construct a connected correlation-dominating set with minimum number of active sensor nodes. In order to achieve this task, the *L*-hop centralized heuristic uses a benefit function that only takes into account the number of active sensor nodes while not considering the remaining energy levels of selected active nodes. Hence, our contribution in this work is twofold:

- We propose an iterative active sensor node determination (IAND) heuristic, which is both effective and runtime-efficient. The IAND heuristic is composed of a greedy constructive heuristic followed by an iterative improvement heuristic to find an effective and runtime-efficient correlation-dominating set for WSNs.
- We define an energy-aware benefit function that is used by both the greedy constructive heuristic and the iterative improvement heuristic of IAND.

Given a large correlation data set as the input, the purpose of the greedy constructive heuristic is to construct a correlation-dominating set in a runtime-efficient manner. The iterative improvement heuristic is executed after the greedy constructive heuristic to improve the energy quality of the active sensor nodes selected by the greedy constructive heuristic. By energy quality we mean the residual energy levels of the active sensor nodes. The bigger is the residual energy, the better the energy quality is. The basic move operation in the iterative improvement heuristic is the swap of an already selected sensor node in the current correlation-dominating set with a set of unselected source sets. The objective in a swap operation is to find a set of unselected source sets that achieves the maximum amount of improvement in the energy quality of the WSN under the constraint of preserving the correlation-dominating set property. We formulate the problem of finding a good unselected source set for swapping a given sensor node as a subproblem of the original correlation-dominating set problem. The iterative improvement heuristic uses the 0-hop centralized heuristic of Gupta et al. [1] to construct a solution to this swap subproblem. Although the 0-hop centralized heuristic is slow with a large correlation data set as the input, it generates a better selection of active sensor nodes, in terms of sensor network lifetime, compared to that of the greedy constructive heuristic with small-scale correlation data as the input in the swap subproblem.

A correlation-dominating set constructed by the IAND heuristic does not necessarily have to result in a connected wireless network. To achieve wireless connectivity among active sensor nodes, we use the minimum Steiner tree construction heuristic [2]. The objective of the minimum Steiner tree heuristic is to construct a connected wireless network by adding the minimum number of additional nodes into the active sensor nodes set. Thus, the minimum Steiner tree forms the connected correlation-dominating set from the correlation-dominating set constructed by the IAND heuristic.

We performed extensive simulations to observe the performance of the IAND heuristics in Section 5. Furthermore, we compared our results with a recent and state-of-the-art solution to the active sensor node determination problem proposed in [1]. We evaluate the heuristics in terms of sensor network lifetime and runtime efficiency and show that we are able to achieve considerably better results than the existing solution to the problem.

The rest of the paper is organized as follows: In Section 2, we discuss the related work and in Section 3 we give a formal definition of the problem. In Section 4, we describe our solution approach and detail our IAND heuristic. In Section 5, we provide the results of our simulation experiments done to evaluate the performance of our IAND approach. Finally, in Section 6 we conclude our work.

2. Related work

In WSNs having data correlations between sensor nodes, reducing the total number of bits transmitted to the data gathering node is a common approach to reduce energy consumption and prolong the network lifetime. Some approaches to achieve a longer network lifetime in a correlated data environment include using clusters for data aggregation, constructing data aggregation trees, utilizing network coding, using probabilistic models and constructing correlation-dominating sets.

Clustering in WSNs is a rather well studied topic [3]. On one hand, there are generic clustering algorithms for WSNs such as HEED [4] and LEACH [5] that do not consider data

correlations between sensor nodes. On the other hand [6], studies the effect of partially correlated data on the performance of clustering algorithms. It uses random geometry methodologies [7] to analyze the energy consumption for forwarding data in a multi-hop sensor network. Furthermore the authors combine the result they obtain with rate distortion theory [8]. This way the authors provide a mathematical analysis framework to study the energy consumption and network lifetime when there are arbitrary data correlations between sensor nodes. The analysis framework allows to determine the optimal tuning of the cluster-head selection probability to balance the trade-off between energy consumption and network lifetime in clustering algorithms for WSNs.

To reduce the number of transmissions performed in the network [9], devises the Clustered Aggregation (CAG) mechanism, which provides approximate results to aggregate queries using the spatial data correlations among sensor nodes. CAG selects a set of cluster-heads, which correspond to a correlation-dominating set, using a simple localized scheme during the query propagation phase. The main pitfall of CAG is that it uses a simple notion of correlation, where the edges of the forwarding tree constitute the correlations for the selection of cluster-heads and connecting sensor nodes.

A recent work on the subject, GRASS [10], provides exact and heuristic approaches to find a minimum number of aggregation points while routing data to the data gathering node such that the network lifetime is maximized. In GRASS, correlations refer to sensor nodes' readings, which overlap statistically as they monitor the same event. These overlappings are used in GRASS to represent the relations among the gathered data. GRASS solves the aggregator selection and routing problems jointly at the data gathering node and then sends the results to the sensor nodes. This way, an optimal solution that is obtained by the data gathering node will result in an optimal routing and aggregation strategy.

Constructing data aggregation trees [11–14] is another approach to reduce the amount of data transmitted by the sensor nodes and prolong the network lifetime. This approach allows data aggregation at the intermediate nodes of the data aggregation tree. The proposed methods by Goel and Estrin [11] construct efficient data aggregation trees that are rooted at the data gathering node. Similarly, work by Enachescu et al. [12] propose a randomized tree construction algorithm that achieves a constant factor approximation of the optimal tree for grid network topologies. In both works, the correlations are specific to aggregation, where multiple data values can be compressed into a data value of defined size. The correlation structure that we consider is more general in the sense that the data of the given set of sensor nodes can be compressed depending on the correlation structure available in the network. Another approach is taken by Luo et al. [13] where a randomized approximation algorithm, namely the minimum fusion Steiner tree (MFST), takes into account not only the data transmission cost but also the data fusion cost.

Utilizing network coding to efficiently gather correlated data has been investigated by several studies [15–17]. Foreign-coding and self-coding are proposed by Rickenbach and Wattenhofer [15]. In these coding techniques, they devise algorithms to construct optimal (minimum weighted number of bit transmissions) and near-optimal data-gathering trees. Work by Chou et al. [16] devise a method to reduce the number of bits transmitted where the data gathering node is informed about the data correlations between sensor nodes. The data correlations that are realized by the data gathering node are then used to inform the sensor nodes about the number of bits they should use for encoding their sensed data. A pitfall of this approach is that it assumes a star topology and does not aim to reduce the number of bits transmitted in the network. Two approaches to optimize the transmission structure and the rate allocation determination at the sensor nodes is proposed by Baltasar et al. [17]. Their first approach allows nodes to use joint coding of correlated data without explicit communication where routing and coding are separated. This results in complex data coding and also global network knowledge is needed for an optimal solution. Their second approach allows nodes to exploit the data correlation only by receiving explicit side information from other nodes. This way, the correlation structure is exploited through communication and joint aggregate coding/decoding locally at each node. This results in easy data coding and relies only on locally available data as side information. However, it should be noted that optimizing the routing structure in the approach becomes quite complex.

Another approach for correlated data gathering in WSNs is use of probabilistic WSN models. The work by Deshpande et al. [18] develop correlation models for WSNs based on initial sensor readings and by using learning algorithms. According to the proposed model, for a given query, possible costs of data acquisition strategies are predicted. Then, the authors propose a simple polynomial-time heuristic for choosing the best way of collecting data by the minimization of the associated costs. This approach provides a novel framework for realizing and predicting the correlations among the sensor nodes. However, our work differs from [18] by utilizing correlations instead of deriving correlations. The work of Deshpande et al. [18] focuses on how correlations are realized and utilized in a WSN. Our work, on the other hand, focuses on prolonging the lifetime of WSNs by effectively and efficiently selecting active nodes when a set of correlations are already defined and given. By taking correlation information as input, we focus on an active node selection solution that is energy-aware, effective and runtime-efficient.

A very recent solution to the connected correlation-dominating set problem in the context of WSNs is given by Gupta et al. [1]. The authors propose a centralized approximation algorithm called the L-hop centralized heuristic. The objective of the L-hop centralized heuristic is to find a correlation-dominating set with minimum number of nodes. The L-hop centralized heuristic is composed of two phases. The first phase constructs a correlation-dominating set and the second phase runs a Steiner tree approximation algorithm [2] to connect the correlation-dominating set constructed in the first phase. The complexity of the L-hop centralized heuristic is $O(nm^2g^L)$, where n is the number of sensor nodes in the network, m is the number of correlations, g is the maximum degree of a

sensor node in the intersection graph of source sensor nodes and L is the hop count used in the heuristic.

There are two main drawbacks of the L-hop centralized heuristic algorithm. The first drawback is its high computational complexity. In a dense WSN, the execution time of the algorithm becomes unexpectedly high. Gupta et al. [1] suggests that best results that are closest to the optimum solution set are obtained by taking the L value as 1. However our simulation results in Section 5.3 report that choosing the L value as 1 as opposed to 0, only produces a small increase in the network lifetime while having a dramatically better runtime performance. The second drawback is the limited energy awareness of the L-hop centralized heuristic. The heuristic tries to increase the sensor network lifetime by only selecting the minimum number of sensor nodes. However, it does not consider the residual energy levels of the sensor nodes while constructing the correlation-dominating set. In this work, we develop an iterative improvement heuristic as a solution to the first drawback by achieving an effective and runtime-efficient correlation-dominating set and we devise an energy-aware benefit function as a solution to the second drawback. Therefore, we solve the same problem as in [1], but in a more runtime-efficient and network lifetime effective way, as our simulations show. Furthermore, to the best of our knowledge, the only proposal for solving the connected correlation-dominating set problem as defined by Gupta et al. [1] is provided by the work of Gupta et al. [1], therefore we believe that our contribution is an improvement for solving the connected correlation-dominating set problem in a runtime-efficient and energy-aware manner.

3. Problem definition

We represent a WSN as a two-tuple $\mathcal{W} = (\mathcal{N}, \mathcal{C})$. Here, $\mathcal{N}$ represents the set of sensor nodes and $\mathcal{C}$ represents the set of correlations among sensor nodes. In $\mathcal{C}$, each correlation is represented as a two-tuple $C = (S,s)$, where source set S contains the source sensor nodes and s is the inferred node. The correlation $C = (S,s)$ means that when source sensor nodes in set S are active nodes in the WSN, sensor node s may stay inactive. This would result in energy saving in node s as it will not need to process, sense or transmit any data. In our work we assume that sensor node correlations (i.e., the source sets and their inferred nodes) are given a priori and are time-invariant (static).

Let $Nodes(S)$ denote the set of sensor nodes constituting the source set S. We extend the $Nodes(\cdot)$ operator to denote the sensor nodes that constitute a set $\widetilde{S}$ of source sets, i.e.,

$$Nodes(\widetilde{S}) = \bigcup_{S \in \widetilde{S}} Nodes(S). \tag{1}$$

Let $Infer(S)$ denote the set of sensor nodes that are inferred by the source set S, i.e.,

$$Infer(S) = \{s : (S,s) \in \mathcal{C}\}. \tag{2}$$

We extend the $Infer(\cdot)$ operator to denote the set of nodes inferred by a set $\widetilde{S}$ of source sets, i.e.,

$$Infer(\widetilde{S}) = \bigcup_{S \in \widetilde{S}} Infer(S). \tag{3}$$

Let $SrcSet(s)$ denote the set of source sets that contain node s, i.e.,

$$SrcSet(s) = \{S : (S,s) \in \mathcal{C}\}. \tag{4}$$

It should be noted that the correlations are not transitive. That is, $Infer(S_1) = S_2$ and $Infer(S_2) = S_3$ does not imply $Infer(S_1) = S_3$.

The problem of selecting the minimum number of sensor nodes while keeping the WSN fully operational can be formulated as an instance of the connected correlation-dominating set problem [1]. For a given sensor network $\mathcal{W} = (\mathcal{N}, \mathcal{C})$, a set $\mathcal{M}$ of source sets is called a connected correlation-dominating set if the following two conditions hold:

1. For each sensor node $s \notin Nodes(\mathcal{M})$, there is a source set $S \subseteq \mathcal{M}$ such that (S,s) is a correlation in $\mathcal{C}$.
2. The communication subnetwork induced by $Nodes(\mathcal{M})$ is connected, and $Nodes(\mathcal{M})$ contains the data-gathering node.

Here, $Nodes(\mathcal{M})$ denotes the set of sensor nodes that form the connected correlation-dominating set, i.e.,

$$Nodes(\mathcal{M}) = \bigcup_{S \in \mathcal{M}} Nodes(S). \tag{5}$$

The connected correlation-dominating set problem is NP-hard, since the less general minimum dominating set problem is well-known to be NP-hard [19]. Therefore, we should use heuristics for solving the problem.

4. Iterative active sensor node determination (IAND) heuristic

In order to effectively and efficiently solve the connected correlation-dominating set problem for wireless sensor networks, we devise a fast energy-aware greedy constructive heuristic that is followed by an iterative improvement heuristic. The proposed approach is referred to here as the iterative active node determination (IAND) heuristic. Both the greedy constructive heuristic and iterative improvement heuristic use an energy-aware benefit function for the determination of which nodes to keep active in the WSN.

4.1. Energy aware benefit function

The benefit function $B(S,\mathcal{M})$ used by Gupta et al. [1] determines the number of newly inferred nodes per new source node added to set $Nodes(\mathcal{M})$. Therefore the benefit function tries to select the highest number of newly inferred nodes while keeping the number of newly added source nodes to $Nodes(\mathcal{M})$ the smallest. This way set $Nodes(\mathcal{M})$ is constructed by selecting the minimum number of nodes, while inferring the maximum number of nodes. The benefit function $B(S,\mathcal{M})$ is as follows:

$$B(S,\mathcal{M}) = \frac{\text{Number of newly inferred nodes by } S}{\text{Number of new source nodes added to } Nodes(\mathcal{M})} = \frac{|Infer(S) - Infer(\mathcal{M})|}{|Nodes(S) - Nodes(\mathcal{M})|}. \tag{6}$$

Rather than defining a totally different benefit function, we extend the benefit function in Eq. (6) by adding energy awareness. For this purpose, we introduce an energy awareness function $E(S,\mathcal{M})$:

$$E(S,\mathcal{M}) = \frac{\text{Energy average of new source nodes added to } Nodes(\mathcal{M})}{\text{Energy average of newly inferred nodes by } S} = \frac{E_{avg}(Nodes(S) - Nodes(\mathcal{M}))}{E_{avg}(Infer(S) - Infer(\mathcal{M}))}. \quad (7)$$

We obtain the new energy aware benefit function by combining $B(S,\mathcal{M})$ and $E(S,\mathcal{M})$, where the primary benefit value is considered as $B(S,\mathcal{M})$ and the secondary benefit value is considered as $E(S,\mathcal{M})$. The energy aware benefit function is outlined in Algorithm 1. The source set with the higher primary benefit value is assumed to have a higher benefit value. If two source sets have primary benefit values that are close to each other, i.e., their absolute difference is smaller than ϵ, then the secondary benefit value determines which source set has the higher benefit value. Consider a benefit value comparison of two source sets S_1 and S_2 for possible inclusion into $\mathcal{M}$. If $abs(B(S_1,\mathcal{M}) - B(S_2,\mathcal{M})) < \epsilon$ then the source set with higher $E(S,\mathcal{M})$ is assumed to have a higher benefit value. Otherwise, source set with higher $B(S,\mathcal{M})$ value is assumed to have a higher benefit value. The purpose of the energy-aware benefit function is to select the minimum possible number of sensor nodes while preserving the energy quality of the selected nodes as much as possible.

Algorithm 1. Energy aware benefit function

```
input : S1, S2, M
if abs(B(S1, M) − B(S2, M)) ≤ ε then
    if E(S1, M) ≥ E(S2, M) then
        return S1
    else
        return S2
else
    if B(S1, M) ≥ B(S2, M) then
        return S1
    else
        return S2
```

We prefer geometric averaging scheme in the computation of $E(S,\mathcal{M})$. The geometric average of a given a set $\{e_1, e_2, \ldots, e_n\}$ of data is computed as $\sqrt[n]{e_1, e_2, \ldots, e_n}$. Another approach could have been using arithmetic averaging scheme. Furthermore, instead of averaging, a min–max approach could have also been taken where $E(S,\mathcal{M})$ would be the minimum energy value of the new sensor node in the source set divided by the maximum energy value of the new sensor node in the newly inferred nodes set.

For a given dataset with a fixed arithmetic average, geometric averaging gives higher results for lower variations in the data values. That is why we prefer using the geometric averaging scheme rather than the arithmetic averaging scheme. For example, consider a source set S_1 with two new source nodes whose energy values are 1 and 19. Also consider a second source set S_2 with again two new source nodes whose energy values are 10 and 10. Assume that both source sets infer one new node whose energy value is 20. Because $B(S_1,\mathcal{M}) = B(S_2,\mathcal{M})$, the secondary metric will decide which source set to be selected. If arithmetic averaging would be used, this would have resulted in $E(S_1,\mathcal{M}) = E(S_2,\mathcal{M}) = 0.5$. However, it is obvious that S_1 should definitely have a lower benefit value since source set S_2 will likely be able to live longer than S_1. If geometric averaging would be used, this would have resulted in $E(S_1,\mathcal{M}) \simeq 0.2175$ and $E(S_2,\mathcal{M}) = 0.5$ which is desirable as selection of S_2 would likely result in a longer network lifetime.

When compared with the max–min approach, geometric averaging performs better in such cases. Consider a source set S_3 with two new source nodes whose energy values are 10 and 40. Also consider a second source set S_4 with again two new source nodes whose energy values are 10 and 10. Assume both source sets infer one new node whose energy value is 20. Because $B(S_3,\mathcal{M}) = B(S_4,\mathcal{M})$, the secondary metric will decide which source set to be selected. If max–min approach would be used, this would have resulted in $E(S_3,\mathcal{M}) = E(S_4,\mathcal{M}) = 0.5$. However, if geometric averaging would be used, this would have resulted in $E(S_3,\mathcal{M}) = 1$ and $E(S_4,M) = 0.5$ which is desirable as selection of S_3 would likely result in a longer network lifetime.

During simulations, we observe that using geometric averaging in computing the $E(S,\mathcal{M})$ values prolongs the network lifetime most when combined with the iterative improvement heuristic. The details of the trade-off between these three benefit functions are given in Section 5.3.

4.2. Greedy constructive heuristic

We introduce the greedy constructive heuristic, which generates acorrelation-dominating set from the given set $\mathcal{C}$ of data correlations as the input. The constructed correlation-dominating set will be an input to the iterative improvement heuristic for refinement. The purpose of the greedy constructive heuristic is to perform the active sensor node selection as fast as possible for a large data correlation input. The purpose of the greedy constructive heuristic is not to find the best or the minimum set of active sensor nodes. It is intended to be used together with the iterative improvement heuristic so that the energy quality of the selected active sensor nodes can be further improved. The greedy constructive heuristic uses the energy-aware benefit function for computing the benefit values of source sets.

Our constructive heuristic briefly works as follows. It first computes the energy-aware benefit values for each source set through a single sequential pass over the given source sets. Then the source sets are sorted using a quicksort-based algorithm [20] into decreasing order according to the energy-aware benefit values. Finally, source sets with higher benefit are added to set $\mathcal{M}$ until $\mathcal{M}$ becomes

a correlation-dominating set. The pseudocode of the heuristic is given in Algorithm 2.

Algorithm 2. greedyConstructiveHeuristic

```
input : 𝒩, 𝒞, dataGatheringNode d
output: ℳ
1  ℳ ← ∅;
2  S_List ← ∅;
3  Nodes(ℳ) ← d;
4  foreach correlation C = {S, s} ∈ 𝒞 do
5      S.benefit1 ← B(S, ℳ);
6      S.benefit2 ← E(S, ℳ);
7      S_List ← S_List ∪ (S, S.benefit1, S.benefit2);
8  //sort in descending order;
9  S_SortedList ← Sort(S_List);
10 while IsCorrelationDom(ℳ) = FALSE do
11     S ← next source set in S_SortedList;
12     ℳ ← ℳ ∪ {S};
```

For the sake of runtime efficiency, the source sets are maintained in compressed form in two one-dimensional arrays *srcNodeIndexArray* and *srcNodeArray*. The IDs of the source sensor nodes that belong to the source set *S* are stored in *srcNodeArray* at the indices beginning from *srcNodeIndexArray*[*S*] to *srcNodeIndexArray*[*S* + 1] − 1. The inferred nodes are also maintained in compressed form in two one-dimensional arrays*inferredNodeIndexArray* and *inferredNodeArray*. The IDs of the inferred sensor nodes by the source set *S* are stored in *inferredNodeArray* at the indices beginning from *inferredNodeIndexArray*[*S*] to*inferredNodeIndexArray*[*S* + 1] − 1. The data structures that are output of the greedy constructive heuristic are the *setMArray* that corresponds to $\mathcal{M}$ and the *nodesInSetMArray* that corresponds to *Nodes*($\mathcal{M}$). The *setMArray* stores 1 in its *i*th index if the source set with ID *i* is in set $\mathcal{M}$ or 0 otherwise. Similarly the *nodesInSetMArray* stores 1 in its *j*th index if the node with ID *j* is inside a source set that is in *Nodes*($\mathcal{M}$).

4.3. Iterative improvement heuristic

The selected set of active sensor nodes that constitute the correlation-dominating set found by the constructive heuristic is an initial solution to our iterative improvement heuristic. The purpose of our iterative improvement heuristic is to go through the initial solution and try to improve the quality of the selected active sensor nodes while preserving the correlation-dominating set property. Our iterative improvement heuristic is outlined in Algorithm 3.

The iterative improvement heuristic is composed of 4 phases;

1. Induction of source sets that are not in $\mathcal{M}$ due to the sensor nodes of source sets in $\mathcal{M}$.
2. Identification and removal of redundant nodes in *Nodes*($\mathcal{M}$).
3. Performing a sequence of swaps between selected sensor nodes and unselected source sets to improve the energy quality of $\mathcal{M}$.
4. Identification and removal of redundant nodes in *Nodes*($\mathcal{M}$).

Algorithm 3. Iterative Improvement Heuristic

```
1 //First phase;
2 sourceSetsInduction()
3 //Second phase;
4 eliminateRedundantNodes()
5 //Third phase;
6 performSwaps()
7 //Forth phase;
8 eliminateRedundantNodes()
```

For the first phase, a subset $\widetilde{S}$ of source sets in $\mathcal{M}$ may already contain the sensor nodes of another source set S_j that is not in $\mathcal{M}$. Source sets such as S_j are said to be induced by $\mathcal{M}$. That is,

$$\bigcup_{S_i \in \widetilde{S}} Nodes(S_i) \supseteq Nodes(S_j), \text{ where } S_j \notin \mathcal{M}. \quad (8)$$

These induced source sets are the ones that are not selected by the constructive heuristic but do exist. Such an induced source set exist when the source sensor nodes in *Nodes*($\mathcal{M}$) are considered, but its nodes are probably in different source sets. Therefore, without adding any further nodes to *Nodes*($\mathcal{M}$), more source sets can be considered to exist in $\mathcal{M}$. Induction of new source sets increases the number of source sets in $\mathcal{M}$ and the number of sensor nodes inferred by $\mathcal{M}$. This increases the degrees of freedom of the iterative improvement heuristic, which in turn increases the possibility of identification and deletion of redundant sensor nodes in phases 2 and 4, and increases the possibility of performing more swaps in phase 3 to enhance the energy quality of $\mathcal{M}$. For example, consider $S_1 \in \mathcal{M}$, $S_2 \in \mathcal{M}$ and $S_3 \notin \mathcal{M}$. Let *Nodes*(S_1) = {s_1,s_4,s_5} and *Infer*(S_1) = {s_7}, and *Nodes*(S_2) = {s_2,s_9} and *Infer*(S_2) = {s_{10}}. Let *Nodes*(S_3) = {s_1,s_2} and *Infer*(S_3) = {s_3}. Since S_1 and S_2 induce S_3, S_3 can be added to $\mathcal{M}$ without any cost, and s_3 can be considered as a new induced node. This phase is outlined in Algorithm 4.

Algorithm 4. sourceSetsInductionFunction

```
input : ℳ
output: ℳ
1  foreach source set S ∉ ℳ do
2      existenceFlag ← TRUE;
3      foreach source node s ∈ S do
4          if s ∉ Nodes(ℳ) then
5              existenceFlag ← FALSE;
6              break;
7      if existenceFlag = TRUE then
8          ℳ ← ℳ ∪ {S}
```

In the second phase of the algorithm, the redundant sensor nodes in *Nodes*($\mathcal{M}$) are identified and removed from $\mathcal{M}$. A sensor node s is said to be redundant in *Nodes*($\mathcal{M}$) if the following two conditions hold:

1. There is a source set $S \in \mathcal{M}$ where S infers sensor node s and does not contain s as its source sensor node. That is, $\exists\, S \in \mathcal{M}$ such that $s \notin S$ and $s \in \mathit{Infer}(S)$.
2. The sensor nodes that are inferred by the source sets that contain s are already inferred by other source set (s) in $\mathcal{M}$. That is, $\exists\, \overline{S} \subseteq \mathcal{M}$ such that $\mathit{Infer}(\overline{S}) \supseteq \mathit{Infer}(\mathit{SrcSet}(s))$ and $\overline{S} \cap \mathit{SrcSet}(s) = \emptyset$.

The number of active sensor nodes in wireless sensor networks is a very important factor on the network lifetime. If a sensor network has a large number of active sensor nodes, these sensor nodes will need to transmit their sensed data to the data gathering node. Due to multi-hop data routing, each forwarded data item will consume some energy from the relaying sensor node. This will affect the overall network lifetime since having more active sensor nodes will cause a faster reduction in the energy levels of the sensor nodes. Therefore it is very important to keep the number of active sensor nodes in the WSN as small as possible. For this purpose, in the iterative improvement heuristic, this phase allows to delete redundant sensor nodes from *Nodes*($\mathcal{M}$). This phase deletes these redundant sensor nodes while preserving the correlation-dominating set property of the selected active sensor node set. Deletion of redundant sensor nodes will cause less network traffic without sacrificing the correct operation of the sensor network. This will help the sensor network to have a longer lifetime. The pseudocode of this phase is provided in Algorithm 5.

In Algorithm 5, the first two for loops (lines 1–5) compute the inference count for each sensor node. Here, the inference count for sensor node s denotes the number of source sets that infer s. Then, the algorithm checks whether each sensor node s in *Nodes*($\mathcal{M}$) can be eliminated. For this purpose, the algorithm checks the inference count of each sensor node r that is inferred by the source sets that contain s. If the inference count of any sensor node r is smaller than or equal to 1, it means that there is at most one source set that infers r. Therefore, elimination of s from *Nodes*($\mathcal{M}$) should not be allowed as it will leave r as an uninferred node. If r would remain as uninferred, set $\mathcal{M}$ would no longer be a correlation-dominating set. The if statement (lines 16–22) is executed if s can be removed from *Nodes*($\mathcal{M}$). In that case, s is removed from *Nodes*($\mathcal{M}$), the source sets that contain s as a source sensor node are removed from $\mathcal{M}$ and finally the inference count of each sensor node that is no longer inferred is decremented. It should be noted here that the processing order of the sensor nodes in *Nodes*($\mathcal{M}$) for elimination might affect the solution quality. Finding the maximum number of sensor nodes that can be eliminated from *Nodes*($\mathcal{M}$) seems to be a hard problem. Therefore, for the sake of runtime efficiency, we prefer using a simple yet effective solution scheme.

Algorithm 5. eliminateRedundantNodesFunction

```
1  foreach sensor node s ∈ 𝒩 do
2      count[s] ← 0;
3  foreach source set S ∈ ℳ do
4      foreach sensor node s ∈ Infer(S) do
5          count[s] ← count[s] + 1;
6  foreach sensor node s ∈ Nodes(ℳ) do
7      foreach source set S ∈ SrcSet(s) do
8          removeFlag ← TRUE;
9          if S ∈ M then
10             foreach sensor node r ∈ Infer(S) do
11                 if count[r] ≤ 1 then
12                     removeFlag ← FALSE;
13                     break;
14             if removeFlag = TRUE then
15                 break;
16     if removeFlag = TRUE then
17         Nodes(ℳ) ← Nodes(ℳ) − {s};
18         foreach source set S ∈ SrcSet(s) do
19             if S ∈ M then
20                 ℳ ← ℳ − S;
21                 foreach sensor node r ∈ Infer(S) do
22                     count[r] ← count[r] − 1;
```

In the third phase of the algorithm, in order to improve the energy quality of selected nodes, the iterative improvement heuristic tries to perform swaps between the selected active sensor nodes and unselected source sets. For each sensor node s whose residual energy level is less than the geometric average of the sensor nodes in *Nodes*($\mathcal{M}$), the heuristic finds the set of sensor nodes that will remain uninferred if s is removed from set *Nodes*($\mathcal{M}$). Then the heuristic tries to find a "good" subset of unselected source sets that can replace sensor node s in order to infer the uninferred sensor nodes when s is removed from set *Nodes*($\mathcal{M}$). Here, goodness of a subset of source sets refers to containing a small number of additional sensor nodes that have high residual energy levels.

Here we show that the solution to this swapping problem can be formulated as a subproblem of the original problem in a much smaller scale. We are again trying to select source sets for the inference of some nodes, but this time in a smaller scale. In a given correlation-dominating set solution $\mathcal{M}$ to the original problem $\mathcal{W} = (\mathcal{N}, \mathcal{C})$, finding a "good" subset of unselected source sets to replace a source node s from *Nodes*($\mathcal{M}$) can be formulated as finding a "good" correlation-dominating set of the following subproblem

$$\mathcal{W}_{sub}(s) = (\mathcal{N}_{sub}(s), \mathcal{C}_{sub}(s)), \tag{9}$$

where

$$\mathcal{C}_{sub}(s) = \{(S, r) : S \notin \mathcal{M} \wedge r \in \mathit{Infer}(\mathit{SrcSet}(s)) \wedge r \notin \mathit{Infer}(\mathcal{M} - \mathit{SrcSet}(s))\}. \tag{10}$$

$$\mathcal{N}_{sub}(s) = \bigcup_{(S,r)\in\mathcal{C}_{sub}} Nodes(S). \quad (11)$$

In the subproblem $\mathcal{W}_{sub}(s)$, $\mathcal{C}_{sub}(s)$ consists of the correlations among unselected source sets that infer the sensor nodes of already selected source sets that contains s and that are not inferred by the remaining source set in $\mathcal{M}$ and, $\mathcal{N}_{sub}(s)$ contains the sensor nodes of source sets in $\mathcal{C}_{sub}(s)$.

We use the 0-hop centralized constructive heuristic of Gupta et al. [1] with our energy-aware benefit function defined in Section 4.1 for solving the above problem. The 0-hop centralized constructive heuristic is outlined in Algorithm 6. The reason why we use the 0-hop centralized constructive heuristic rather than the greedy constructive heuristic (Algorithm 3) is because of the small scale of the subproblem. The scale of swapping problem is small because we are only trying to find source sets for inferring a small set of sensor nodes. Although 0-hop centralized constructive heuristic takes much more time than the greedy constructive heuristic for large scale problems, the running time of 0-hop centralized constructive heuristic is expected to be in acceptable levels for the small scale of the subproblem, and hence amortizing its better solution quality compared to the greedy constructive heuristic. The swapping of sensor nodes in $\mathcal{M}$ with unselected source sets is outlined in Algorithm 7.

It should be noted here that the notion of "L-hop" refers to the number of hops that are considered for the formation of source sets that are added to set $\mathcal{M}$. For example, for 0-hop centralized heuristic, only the source sets that are the input of the algorithm are added to set $\mathcal{M}$ one by one while computing the benefit function, whereas for 1-hop centralized heuristic, the original source sets that are input of the algorithm and their 1-hop correlation neighboring source sets are added to set $\mathcal{M}$ (forming a temporary new source set) while computing the benefit function. Since our simulations showed that there is no significant performance difference in network lifetime performance between 0-hop and 1-hop centralized heuristics, but there is a very high runtime performance difference between the two, we have only used the 0-hop centralized heuristic. Therefore, here we are only giving the simplified algorithm for the 0-hop centralized heuristic saving the reader from the details of the generalized L-hop centralized heuristic.

Algorithm 6. 0-HopCentralizedHeuristic

```
   input  : N_sub, C_sub
   output: M
1  M ← ∅;
2  foreach correlation C_cur = (S_cur, s_cur) ∈ C_sub do
3      if S_cur ∉ M then
4          S_max ← S_cur;
5          foreach correlation C_tmp = (S_tmp, s_tmp) ∈ C_sub do
6              if S_tmp ∉ M then
7                  S_max ← EnergyAwareBenefit(S_max S_tmp, M);
8          M ← M ∪ {S_max};
9          if IsCorrelationDom(M) = TRUE then
10             break;
```

In Algorithm 7, sensor nodes in $Nodes(\mathcal{M})$, whose residual energy levels are smaller than the average residual energy level of $\mathcal{M}$, are considered for swapping, starting from the sensor node with the minimum residual energy level. We need to maintain a priority queue Q for the selection of sensor nodes with low energy levels because new sensor nodes that are added to $Nodes(\mathcal{M})$ due to the swap operations might be considered for swapping in the future iterations. The first two inner for loops (lines 7–14) construct the correlation-dominating set subproblem that will be solved for the swap of the current sensor node s from $Nodes(\mathcal{M})$. The subproblem is solved at line 15 using 0-hop centralized constructive heuristic. At line 16, this subproblem solution is checked in order to see whether it improves the current quality of $\mathcal{M}$ in terms of average residual energy level. If the newly selected sensor nodes improve the overall solution quality of set $Nodes(\mathcal{M})$, then s is swapped with the newly selected source sets. The 0-hop centralized constructive heuristic may fail to find a solution for the subproblem, in which case the resulting M_{sub} is not swapped for the current solution $\mathcal{M}$. The last two for loops (lines 19–27) realize the swap operation together with inserting the new sensor nodes in $Nodes(\mathcal{M})$ into Q. It should be noted that the *energy*(s) function gives the residual energy level of sensor node s.

Algorithm 7. performSwapsFunction

```
   input  : N, C, M
   output: M
1  //Priority queue Q keyed with residual energy;
2  Q ← Nodes(M);
3  s ← ExtractMin(Q);
4  initialEnergyAvgM ← E_avg(M);
5  while energy(s) < initialEnergyAvgM do
6      C_sub ← ∅;
7      foreach correlation (S, r) ∈ C do
8          if E_avg(S) > initialEnergyAvgM then
9              if source set S ∉ M then
10                 if sensor node r ∈ Infer(SrcSet(s)) then
11                     C_sub ← C_sub ∪ (S, r);
12     N_sub ← ∅;
13     foreach correlation (S, r) ∈ C_sub do
14         N_sub ← N_sub ∪ Nodes(S)
15     M_sub ← 0HopCentralizedHeuristic(N_sub, C_sub)
16     if E_avg(M ∪ M_sub) > E_avg(M) then
17         //Swap s with Nodes(M_sub);
18         Nodes(M) ← Nodes(M) − {s};
19         foreach correlation (S, r) ∈ C do
20             if S ∉ M then
21                 if r ∈ Infer(SrcSet(s)) then
22                     M ← M − {S};
23         M ← M ∪ {M_sub};
24         foreach sensor node r ∈ Nodes(M_sub) do
25             if r ∉ M then
26                 Nodes(M) ← Nodes(M) ∪ {r};
27                 Insert(Q, r);
28     s ← ExtractMin(Q);
```

The fourth phase of the algorithm is the same as the second phase. The improved solution $Nodes(\mathcal{M})$ is pruned by identifying and deleting the nodes that become redundant after the swap phase.

4.4. Minimum Steiner tree construction

After the execution of the iterative active sensor node determination heuristic, the correlation-dominating set is established. The correlation-dominating set is unaware of the network connectivity of the sensor nodes. It only guarantees that the constructed set is able to fully sense the necessary data of the WSN. In order for this data to be successfully collected at the data gathering node, the correlation-dominating set has to be fully connected. To establish the connected correlation-dominating set, we construct a minimum Steiner tree [2] rooted at the data gathering node, which connects the sensor nodes in $Nodes(\mathcal{M})$ by adding a number of sensor nodes not in $Nodes(\mathcal{M})$ to achieve wireless connected active node set. The sensor nodes that are added to $Nodes(\mathcal{M})$ by the minimum Steiner tree construction step are called Steiner sensor nodes. The objective of minimum Steiner tree algorithm is to keep the number of Steiner nodes as small as possible. Note that the Steiner sensor nodes will not need to sense data even though they will be active. The Steiner nodes will only be responsible for forwarding data packets towards the data gathering node.

5. Simulations

In this section, we report and discuss the results of the simulations we performed to test the validity of our proposed approach to the active sensor node determination problem in wireless sensor networks. For this purpose, we first discuss the energy consumption model used in our simulations. Then, we report simulation results in different network topologies with different parameters to observe the performance of the proposed approach.

5.1. Energy consumption model

In order to determine the amount of energy depleted from each selected sensor node, we define the following energy consumption model. After each configuration of set $\mathcal{M}$ as a connected correlation-dominating set, there are R data gathering rounds until the next configuration. During any given round, each selected active sensor node generates one packet towards the data gathering node. We assume that data aggregation is not applied at intermediate nodes. Let P be the amount of energy that is spent for transmitting or receiving one packet. We assume sensor nodes use short-range radios and therefore we assume that the energy spent for transmitting a packet is nearly the same for receiving a packet. We ignore the energy spent at the transmit amplifier, since it is very small for short range radios. Therefore we ignore the effect of the distance between the transmitter and receiver on the transmit energy consumption because of short distance. Let G denote the number of descendants of an active sensor node. The total amount of energy spent by a selected active sensor node in a round is $P \times 2G + 1$ and the total amount of energy spent by a Steiner sensor node in a round is $P \times 2G$. Note that Steiner nodes do not generate data packets. They only act as routers for packets from other nodes. The total amount of energy that is consumed between two successive configurations is $R \times (P \times 2G + 1)$ for a selected active sensor node and is $R \times (P \times 2G)$ for a Steiner sensor node.

5.2. Assumptions and parameter values

The P value, which is the amount of energy that is spent for transmitting or receiving one packet, is selected as 0.01 energy units, where the initial energy of a sensor node is set to be 100 energy units. The R value, which is the number of data gathering rounds between two configurations, is selected as 100. The communication range between sensor nodes is assumed to be 20 m. The two main criteria for making performance comparisons between different approaches is sensor network lifetime and runtime of the active sensor node determination heuristics. We assume that the WSN is dead and cannot further operate once a connected correlation-dominating set cannot be constructed. Unless otherwise stated, the WSN topology is modeled according to Gaussian distribution with standard deviation (σ) set to 1 on a $150 \times 150\ m^2$ area, where the data gathering node is selected from the center of the network. It should be noted that in a WSN topology modeled according to Gaussian distribution, more sensor nodes are placed around the center of the network as the σ value becomes smaller. The σ value indicates the variation of node positions around the data gathering node position.

The correlations that define the inference relationship among nodes are generated randomly in the simulations. The three parameters that affect the random correlation generation process are C_{per}, C_{maxsrc} and C_{maxhop}. A candidate correlation is accepted as a valid correlation if the correlation percentage value that is randomly selected in the scale of $[0, 100]$ is smaller than the defined C_{per} value for correlation generation. The C_{maxsrc} parameter defines the maximum number of source sensor nodes that is allowed to be in a given correlation. Lastly, the C_{maxhop} parameter defines the maximum hop-count in the WSN for sensor nodes to infer one another. Unless otherwise stated, the correlation generation parameter values are $C_{per} = 50, C_{maxsrc} = 5$ and $C_{maxhop} = 3$.

For each simulation experiment, 10 different correlation sets are generated with different random seeds. The average of the 10 simulations on these correlation sets is reported in the following figures. This simulation scheme is used in order to provide average case results in the comparison of various active node determination heuristics.

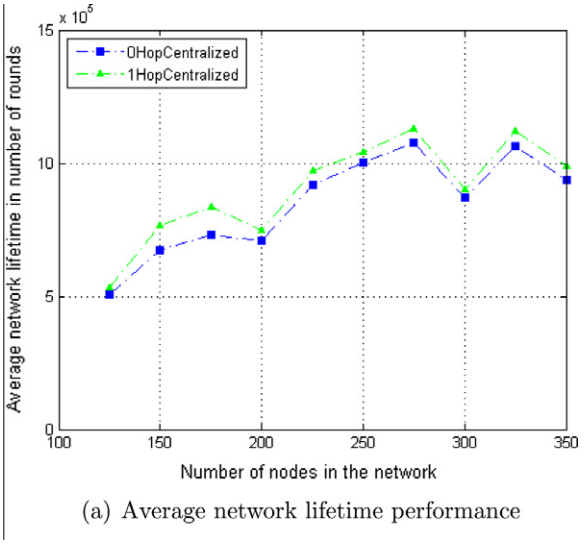
It should be noted here that in this work we assume that correlations among sensor nodes may exist due to many reasons which may or may not depend on the communication topology of the network. The focus of our paper is not characterizing correlations, but using given correlations to efficiently and effectively decide on nodes to be active.

The correlations that are randomly generated in our simulations are for showing the performance of our proposed heuristics to solve the connected correlation-dominating set problem. It is up to the specific WSN application to decide whether or not to consider the actual network topology or other factors while deciding on the correlation model that is best for the developed WSN application. Moreover, our correlation generation model is not a framework for generating the best set of possible correlations among the sensor nodes that reflect the real data correlations in a WSN. Instead, it provides an example set of correlations that are used to evaluate our proposals for solving the connected correlation-dominating set problem.

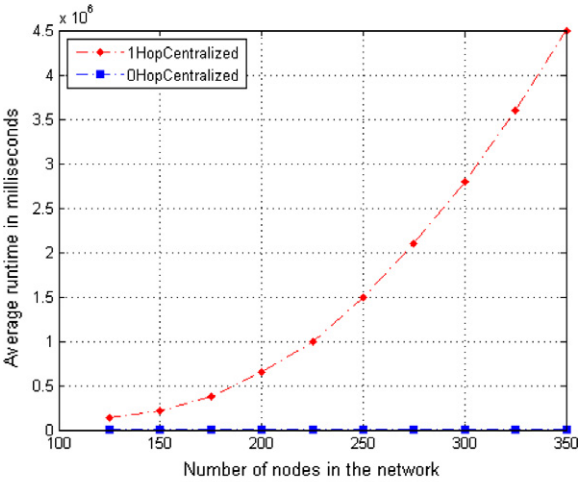
5.3. Simulation results

We first performed simulations to determine the parameters to be used in the comparison of IAND and L-hop centralized heuristics. Once we determined the parameter values to be used in these heuristics, we compared them in different network topologies. Finally, we changed the correlation generation parameter values to further observe and report the performance of the compared heuristics.

Fig. 1 compares the performance of the 0-hop and 1-hop centralized heuristics with increasing number of nodes. The 0-hop and 1-hop heuristics are L-hop heuristics where L is 0 and 1, respectively. Fig. 1a shows that the network lifetime performance of the 1-hop centralized heuristic is slightly better than that of the 0-hop centralized heuristic. The reason for the slightly better network lifetime performance of the 1-hop centralized heuristic is because of the fact that it includes a larger set of source sets into $\mathcal{M}$ through 1-hop union of source sets. However, in terms of runtime performance, Fig. 1b shows that the runtime of the 1-hop centralized heuristic dramatically increases as the network

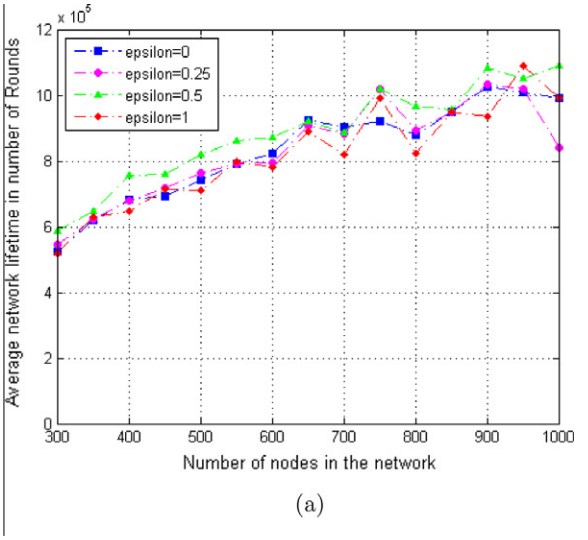


(a) Average network lifetime performance

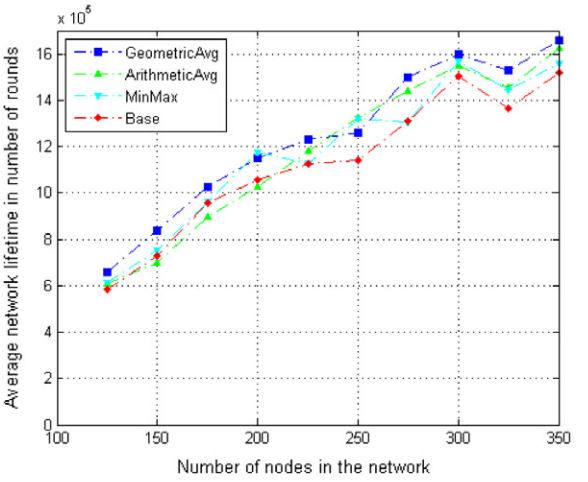


(b) Average runtime performance

Fig. 1. Performance comparison of 0-hop and 1-hop centralized heuristics.



(a)



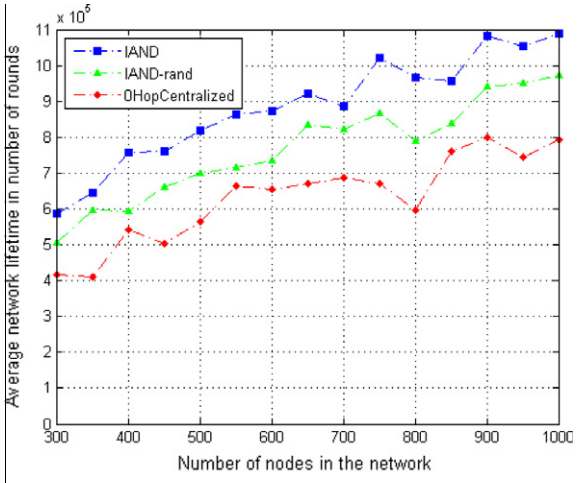
(b)

Fig. 2. Effect of (a) ϵ parameter and (b) benefit function scheme on the performance of the IAND heuristic.

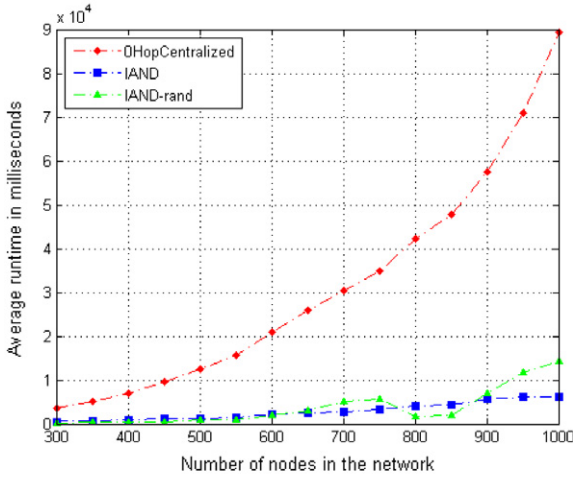
becomes denser. Therefore, it becomes impractical to use the 1-hop centralized heuristic even for medium scale WSNs (and any L-hop heuristic with L larger than 1). For this reason, we compare our IAND heuristic against the 0-hop centralized heuristic in the rest of the experiments. The 0-hop centralized heuristic achieves a solution of reasonably good quality with very short running time. It should be noted that the runtime of the 0-hop centralized heuristic is very small compared to that of the 1-hop centralized heuristic, so that its running time seems to lie on the x-axis of Fig. 1b. Because of the extremely long runtime of the 1-hop centralized heuristic, this simulation is performed on a 110×110 m^2 area in which the number of sensor nodes in the network is varied between 125 and 350.

Fig. 2a shows the effect of the ϵ parameter on the performance of our IAND heuristic. As seen in Fig. 2a, the network lifetime performance of IAND heuristic increases with increasing ϵ until $\epsilon = 0.5$, and then it begins to decrease for higher ϵ values. This experimental finding is as expected. For small ϵ values, the energy-aware benefit function gives more emphasis to the $B(S, \mathcal{M})$ function, which considers the number of source nodes and inferred nodes. For large ϵ values, the energy-aware benefit function gives more emphasis to $E(S, \mathcal{M})$ function, which considers the residual energy levels of the source nodes and inferred nodes. We have performed this experiment in order to show that selecting the ϵ value affects the WSN lifetime that is obtained using our heuristics. The difference between different ϵ values might seem small, however, as it can be seen from Fig. 2a, there is a value of $\epsilon(\epsilon = 0.5)$ for which WSN lifetime is further improved. This information is useful to fix ϵ at best possible value and do all other simulation experiments with this fixed value.

Fig. 2b shows the effect of three different energy averaging schemes proposed in Section 4.1 for our energy-aware benefit function in the performance of IAND

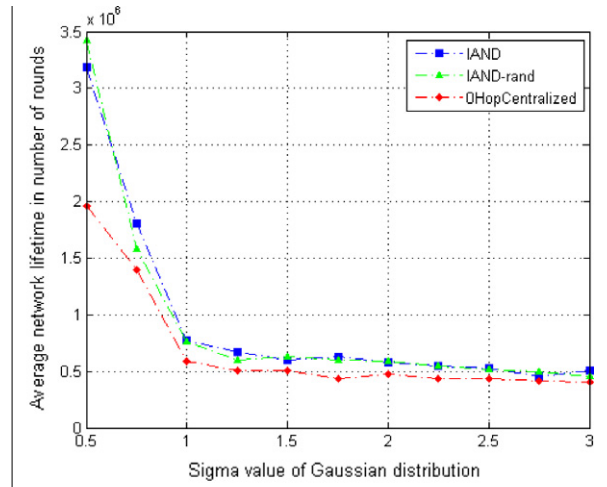


(a) Average network lifetime performance

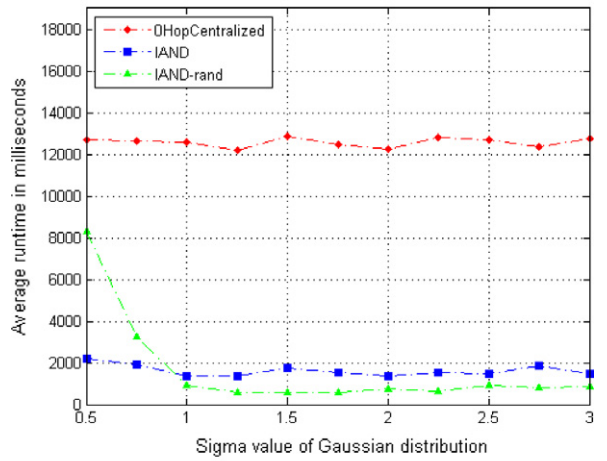


(b) Average runtime performance

Fig. 3. Performance comparison of IAND, IAND-rand and 0-hop centralized heuristic with increasing number of nodes and Gaussian distribution with $\sigma = 1$.



(a) Average network lifetime performance



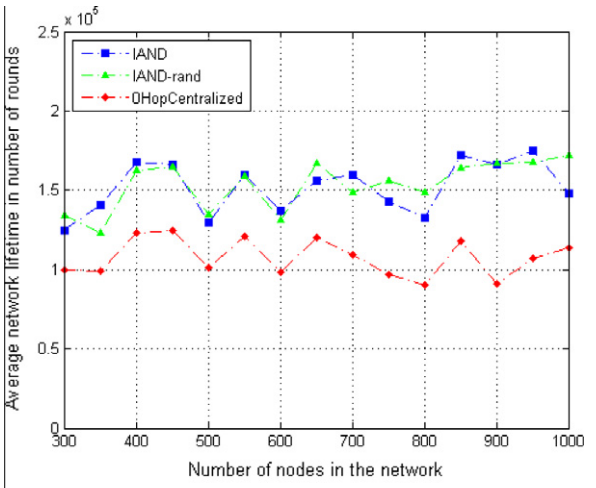
(b) Average runtime performance

Fig. 4. Performance comparison of IAND, IAND-rand and 0-hop centralized heuristic with increasing Gaussian distribution σ.

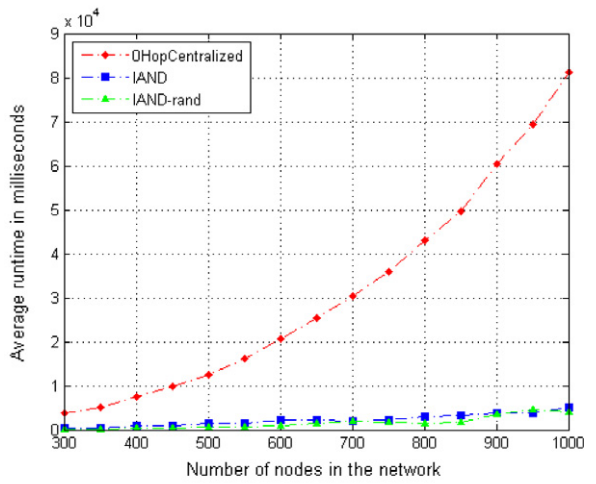
heuristic. We compare the performance of our energy-aware benefit function against the benefit function of Gupta et al. [1] which is referred to as "base" in Fig. 2b. As seen in the figure, although the difference is not large, all proposed energy-aware benefit function schemes perform better than the benefit function of Gupta et al. [1], in terms of their effect on network lifetime. Furthermore, Fig. 2b confirms our expectation that the geometric averaging scheme performs better than the arithmetic averaging scheme and min–max scheme.

Fig. 3 shows the IAND heuristic compared with the 0-hop centralized heuristic as the WSN becomes denser. In Fig. 3, we also display the simulation results for an IAND version in which a random constructive heuristic is used instead of the greedy constructive heuristic given in Algorithm 2. This random constructive heuristic selects source sets randomly until the correlation-dominating set is constructed. IAND-rand results are given here to show the effectiveness of iterative improvement heuristic even when a simple constructive heuristic is used for finding an initial solution. Thus, the IAND-rand heuristic is composed of the random constructive heuristic followed by the iterative improvement heuristic. As seen in Fig. 3a, both IAND and IAND-rand perform considerably better than the 0-hop centralized heuristic in terms of average network lifetime, while IAND performs better than IAND-rand. As seen in Fig. 3b, the proposed IAND heuristics run drastically faster than the 0-hop centralized heuristic and the runtime performance gap increases considerably with the increasing network density in favor of IAND heuristics.

Fig. 4 compares the performance of the IAND and IAND-rand heuristics with the 0-hop centralized heuristic as the σ value of the Gaussian distribution of the given WSN topology increases while the number of nodes stays as 500 in the same area. It should be noted here that the

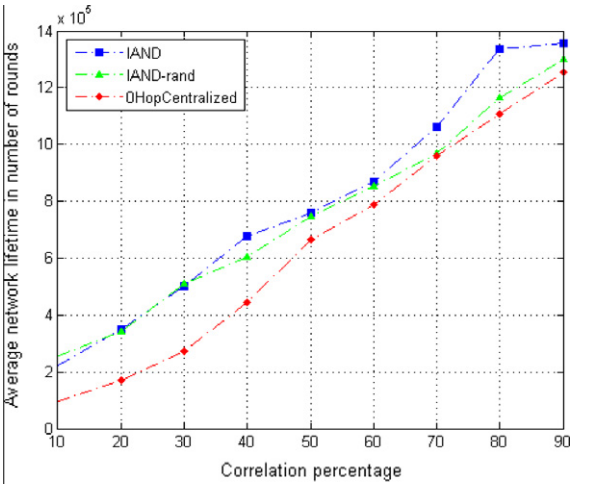


(a) Average network lifetime performance

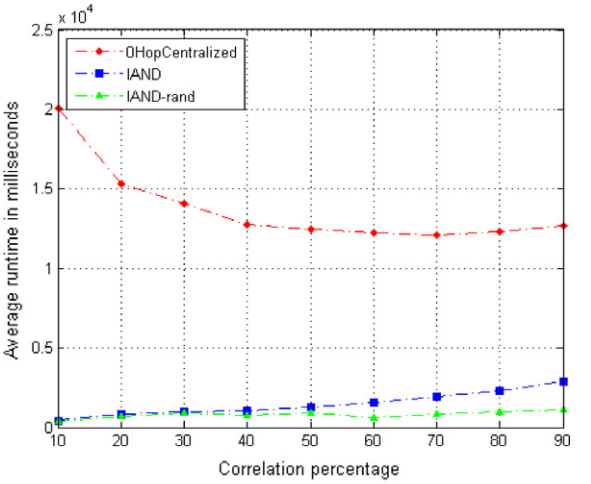


(b) Average runtime performance

Fig. 5. Performance comparison of IAND, IAND-rand and 0-hop centralized heuristic in a uniform topology.



(a) Average network lifetime performance

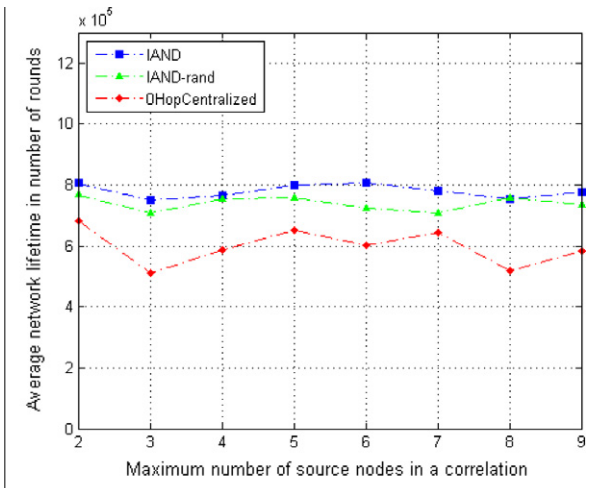


(b) Average runtime performance

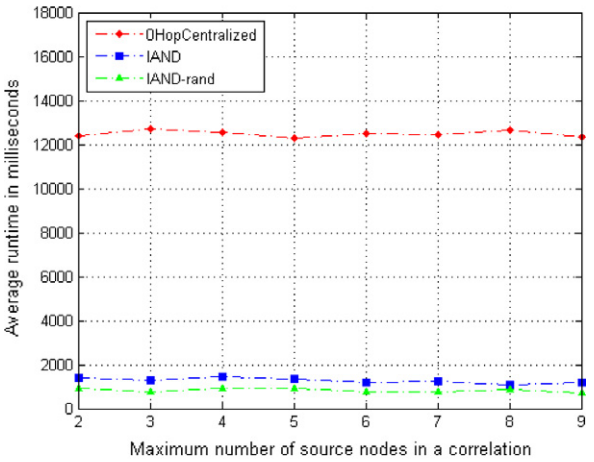
Fig. 6. Performance comparison of IAND, IAND-rand and 0-hop centralized heuristics as the C_{per} value is increased, where $C_{maxsrc} = 5$ and $C_{maxhop} = 3$.

WSN topology becomes more uniform with increasing σ. Similarly, Fig. 5 compares the performance of the IAND and IAND-rand heuristics against the 0-hop centralized heuristic in a uniform network topology as the number of nodes in the same area increases. Fig. 4b compares the heuristics in terms of their run-time performance. We can see from this figure that the IAND approach is able to achieve better network lifetime performance for small values of σ where the network topology is skewed and denser around the data gathering node. As also seen in Fig. 4a, the performance gap between the IAND heuristics and the 0-hop centralized heuristic becomes smaller with increasing σ. However, as seen in Fig. 5a, there is still considerable network lifetime performance difference between IAND approach and the 0-hop centralized heuristic in the uniform WSN topology. As seen in Fig. 4b, the runtime performance gap between IAND and the 0-hop centralized heuristic stays the same with increasing σ. As seen in Fig. 5b, the IAND heuristics run drastically faster than the 0-hop centralized heuristic as the number of nodes in the uniform WSN topology increases.

The main reason for the decrease in the network lifetime performance gap between IAND heuristics and 0-hop centralized heuristic with increasing σ is because of the fact that when the network topology is uniform, the nodes around the data gathering node constitute a bottleneck for the performance of all heuristics. The energy levels of the sensor nodes around the data gathering node deplete faster than other nodes in the network because the sensor nodes around the data gathering node have more descendant sensor nodes and therefore they need to forward more packets than other sensor nodes. Since the sensor nodes around the data gathering node deplete their energy levels and become unusable, it becomes harder to construct a connected correlation-dominating set in a uniform topology. That is why all the approaches were not

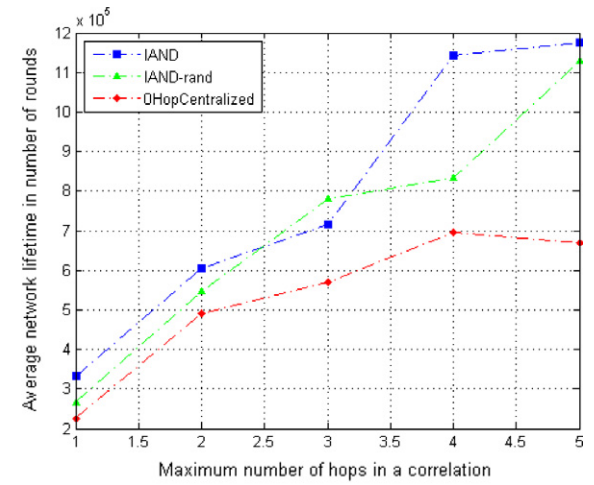


(a) Average network lifetime performance

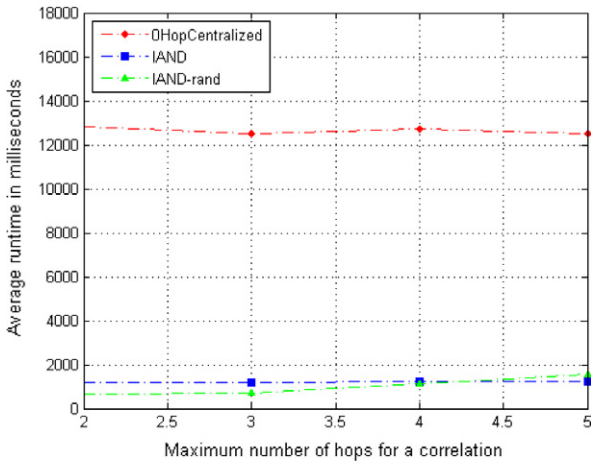


(b) Average runtime performance

Fig. 7. Performance comparison of IAND, IAND-rand and 0-hop centralized heuristics as the C_{maxsrc} value is increased, where $C_{per} = 50$ and $C_{maxhop} = 3$.



(a) Average network lifetime performance



(b) Average runtime performance

Fig. 8. Performance comparison of IAND, IAND-rand and 0-hop centralized heuristics as the C_{maxhop} value is increased, where $C_{per} = 50$ and $C_{maxsrc} = 5$.

able to increase the network lifetime after a certain point. This experimental finding was also reported in [21,22]. Fig. 5a also shows that in uniform WSN topology, the network lifetime performance gap between the IAND and IAND-rand heuristics is quite small.

Figs. 6–8 are presented to show the effect of the correlation-set generation parameters C_{per}, C_{maxsrc} and C_{maxhop} in the performance comparison of heuristics. Figs. 6–8 show the performance variation of IAND and 0-hop centralized heuristic when varying one of the parameters while fixing the other two. The simulations were performed with a 500 node WSN network.

As seen in Fig. 6a, with increasing C_{per} value, the network lifetime performance of all heuristics increase while the performance gap between IAND and 0-hop centralized heuristic remains nearly the same. Fig. 6b shows that the runtime performance gap between IAND and 0-hop centralized heuristic becomes smaller as the correlation percentage increases. This is because, as the C_{per} value increases, the number of available candidate source sets increases and hence the number of passes over the candidate source sets performed by the 0-hop centralized heuristic decreases.

As seen in Fig. 7, the C_{maxsrc} value does not affect the network lifetime and runtime performance of the heuristics considerably. This behavior might be attributed to the possibility that the size of the union of source sets that constitute the connected correlation-dominating set remains nearly the same with varying C_{maxrsc} value.

As seen in Fig. 8a, the increase of the C_{maxhop} value increases the performance gap between IAND and 0-hop centralized heuristic in favor of IAND. As the C_{maxhop} value increases, more sensor nodes that are distant from each other are able to infer one another. This allows the iterative improvement heuristic to perform more swap operations which in turn increases the network lifetime performance. As seen in Fig. 8b, the runtime performance gap between IAND heuristics and 0-hop centralized heuristic remains nearly the same when C_{maxhop} is increased.

6. Conclusion

In wireless sensor network (WSN) applications where data gathered by different sensor nodes is correlated, all sensor nodes need not be active for the WSN to be functional. In such WSN applications, selecting a set of active sensor nodes in the network is a critical issue for the performance of the WSN. In this work, we considered the problem of finding an active subset of nodes that are connected and can infer the correlated data of the inactive sensor nodes. This problem was formulated as an instance of the connected correlation-dominating set problem. In order to solve the connected correlation-dominating set problem in the context of WSNs, we have proposed and developed an Iterative Active sensor Node Determination (IAND) heuristic that is composed of a fast constructive heuristic followed by an effective and runtime-efficient iterative improvement heuristic. The constructive heuristic is a fast algorithm that provides an initial solution for the iterative improvement heuristic. This initial solution is composed of selected active sensor nodes that constitute a correlation-dominating set for the given network.

The iterative improvement heuristic performs a sequence of swap operations to further improve the quality of active sensor nodes while preserving the correlation-dominating set property of the set of active sensor nodes. The swap operations take place between the selected sensor nodes in the current correlation-dominating set and the unselected source sets. The problem of finding a "good" swap for a given selected source node was formulated as a subproblem of the original correlation-dominating set problem. We used the 0-hop centralized heuristic of Gupta et al. [1] for solving this swap subproblem due to the small-size of the subproblem.

The extensive simulations that we performed showed that the proposed approach can efficiently compute an active sensor node set and can be effective in prolonging the network lifetime. We also compared our approach with a state-of-the-art approach. The simulation results showed that our approach can perform considerably better in terms of WSN lifetime than the existing approach, while achieving drastically better runtime efficiency.

References

[1] H. Gupta, V. Navda, S. Das, V. Chowdhary, Efficient gathering of correlated data in sensor networks, ACM Transactions on Sensor Networks 4 (1) (2008) 1–31. doi:http://doi.acm.org/10.1145/1325651.1325655.

[2] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs, Information Processing Letters 27 (3) (1988) 125–128. doi:http://dx.doi.org/10.1016/0020-0190(88)90066-X.

[3] A.A. Abbasi, M.F. Younis, A survey on clustering algorithms for wireless sensor networks, Computer Communications 30 (14–15) (2007) 2826–2841.

[4] O. Younis, S. Fahmy, Heed: a hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks, IEEE Transactions on Mobile Computing 3 (4) (2004) 366–379.

[5] W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Communications 1 (4) (2002) 660–670. URL http://dx.doi.org/10.1109/TWC.2002.804190.

[6] M. Lotfinezhad, B. Liang, Effect of partially correlated data on clustering in wireless sensor networks, IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON) (2004) 172–181.

[7] S.G. Foss, S. Zuyev, On a voronoi aggregative process related to a bivariate poisson process, Advances in Applied Probability 28 (1996) 965–981.

[8] T. Berger, Rate Distortion Theory: A Mathematical Basis for Data Compression, Prentice-Hall, 1972.

[9] Y. SunHee, C. Shahabi, Exploiting spatial correlation towards an energy efficient clustered aggregation technique (cag), IEEE International Conference on Communications (ICC) 5 (2005) 3307–3313.

[10] J.N. Al-Karaki, R. Ul-Mustafa, A.E. Kamal, Data aggregation and routing in wireless sensor networks: optimal and heuristic algorithms, Computer Networks 53 (7) (2009) 945–960. doi:http://dx.doi.org/10.1016/j.comnet.2008.12.001.

[11] A. Goel, D. Estrin, Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk, in: SODA'03 Conference, 2003, pp. 499–505.

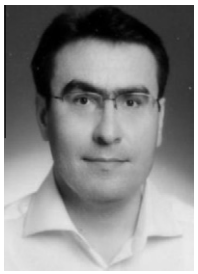
[12] M. Enachescu, A. Goel, R. Govindan, R. Motwani, Scale-free aggregation in sensor networks, Theoretical Computer Science 344 (1) (2005) 15–29. doi:http://dx.doi.org/10.1016/j.tcs.2005.06.023.

[13] H. Luo, Y. Liu, S. Das, Routing correlated data with fusion cost in wireless sensor networks, IEEE Transactions on Mobile Computing 5 (11) (2006) 1620–1632. doi:10.1109/TMC.2006.171.

[14] H.O. Tan, I. Korpeoglu, I. Stojmenovic, Computing localized power efficient data aggregation trees for sensor networks, IEEE Transactions on Parallel and Distributed Systems 22 (3) (2011), 489–500. http://dx.doi.org/10.1109/TPDS.2010.68.

[15] P.V. Rickenbach, R. Wattenhofer, Gathering correlated data in sensor networks, 2004, pp. 60–66.
[16] J. Chou, D. Petrovic, K. Ramchandran, A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks, in: IEEE Infocom Conference, 2003.
[17] R.C. Baltasar, R. Cristescu, B. Beferull-lozano, M. Vetterli, On network correlated data gathering, in: IEEE Infocom Conference, 2004, pp. 2571–2582.
[18] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong, Model-driven data acquisition in sensor networks, in: Proceedings of the Thirtieth international conference on Very large data bases, VLDB Endowment, vol. 30, 2004, pp. 588–599.
[19] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387. doi:http://dx.doi.org/10.1007/PL00009201.
[20] D.R. Musser, Introspective sorting and selection algorithms, Software-Practice and Experience 8 (1997) 983–993.
[21] R. Subramanian, F. Fekri, Sleep scheduling and lifetime aximization in sensor networks: Fundamental limits and optimal solutions, in: IPSN '06: Proceedings of the 5th international Conference on Information Processing in Sensor Networks, ACM, New York, NY, USA, 2006, pp. 218–225. doi:http://doi.acm.org/10.1145/1127777.1127813.
[22] Y. Ding, C. Wang, L. Xiao, An adaptive partitioning scheme for sleep scheduling and topology control in wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems 20 (9) (2009) 1352–1365. doi:http://doi.ieeecomputersociety.org/10.1109/TPDS.2008.230.

Efe Karasabun received his M.S. and B.S. degrees from the Department of Computer Engineering of Bilkent University in 2009 and 2007, respectively. He worked in Siemens Corporate Research in New Jersey, USA, as an intern. His research interests include computer networks, algorithms, wireless networks and p2p networks.



Ibrahim Korpeoglu is an associate professor in Department of Computer Engineering of Bilkent University, Ankara. He received his Ph.D. and M.S. degrees from University of Maryland at College Park, both in Computer Science, in 2000 and 1996, respectively, and B.S. degree from Bilkent University Computer Engineering Department in 1994. He joined Bilkent University as a faculty member in 2002. Before that, he worked in several research and development companies in USA, including Ericsson, IBM T.J. Watson Research Center, Bell Laboratories, and Bell Communications Research (Bellcore). He received Bilkent University Distinguished Teaching Award in 2006 and IBM Faculty Award in 2009. He is a member of ACM and a senior member of IEEEE.

Cevdet Aykanat received the BS and MS degrees from Middle East Technical University, Ankara, Turkey, both in electrical engineering, and the PhD degree from Ohio State University, Columbus, in electrical and computer engineering. He was a Fulbright scholar during his PhD studies. He worked at the Intel Supercomputer Systems Division, Beaverton, Oregon, as a research associate. Since 1989, he has been affiliated with the Department of Computer Engineering, Bilkent University, Ankara, Turkey, where he is currently a professor. His research interests mainly include parallel computing, parallel scientific computing and its combinatorial aspects, parallel computer graphics applications, parallel data mining, graph and hypergraph partitioning, load balancing, neural network algorithms, high performance information retrieval systems, parallel and distributed web crawling, parallel and distributed databases, and grid computing. He has (co) authored over 40 technical papers published in academic journals indexed in SCI. He is the recipient of the 1995 Young Investigator Award of The Scientific and Technological Research Council of Turkey. He is a member of the ACM and the IEEE Computer Society. He has been recently appointed as a member of IFIP Working Group 10.3 (Concurrent Systems).