Global Journal of Computer Science and Technology
Network, Web & Security
Volume 13 Issue 11 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Data Gathering with Tour Length-Constrained

By Mohammad A. Almahameed, Mohammed Aalsalem, Khaled Almi'ani & Ghazi Al-Naymat

Al Hussein Bin Talal University, Jordan

Abstract - In this paper, given a single mobile element and a time deadline, we investigate the problem of designing the mobile element tour to visit subset of nodes, such that the length of this tour is bounded by the time deadline and the communication cost between nodes outside and inside the tour is minimized. The nodes that the mobile element tour visits, works as cache points that store the data of the other nodes. Several algorithms in the literature have tackled this problem by separating two phases; the construction of the mobile element tour from the computation of the forwarding trees to the cache points. In this paper, we propose algorithmic solutions that alternate between these phases and iteratively improves the outcome of each phase based on the result of the other. We compare the resulting performance of our solutions with that of previous work.

Keywords : component; wireless sensor networks; data gathering; mobile sensing.

GJCST-E Classification : C.2.1

Data Gathering with Tour Length-Constrained

Mohammad A. Almahameed [α], Mohammed Aalsalem [σ], Khaled Almi'ani [ρ] & Ghazi Al-Naymat [ω]

Abstract - In this paper, given a single mobile element and a time deadline, we investigate the problem of designing the mobile element tour to visit subset of nodes, such that the length of this tour is bounded by the time deadline and the communication cost between nodes outside and inside the tour is minimized. The nodes that the mobile element tour visits, works as *cache points* that store the data of the other nodes. Several algorithms in the literature have tackled this problem by separating two phases; the construction of the mobile element tour from the computation of the forwarding trees to the cache points. In this paper, we propose algorithmic solutions that alternate between these phases and iteratively improves the outcome of each phase based on the result of the other. We compare the resulting performance of our solutions with that of previous work.

Keywords : component; wireless sensor networks; data gathering; mobile sensing.

I. Introduction

In wireless sensor network, data gathering using *Mobile Elements* (MEs) [1][2][3] is one of the applications that gives rise to a fundamental problem in networks: given a network of sensor locations, design a path(s) for the mobile element(s) that will enable efficient gathering of all data from the network. Many variations of this problem have been studied in the literature. In some cases [4][5][6][7][8], the investigated problem is described as given mobile element(s), design a path for the mobile element(s) to collect the data of the nodes. The objective of such problems is normally minimizing the length of the mobile element path or minimizing the number of used mobile elements. In this case we have a multiple or single travelling salesman problem instance to solve[9]. Some other problems [10][11][12][13][14][15][16], investigate the scenario where only one mobile element is used and the data of each node must be delivered to the sink within a pre-define time deadline. This time deadline is either due to timeliness constraints on the sensor data or a limit on the amount of energy available to the mobile element itself.

With the presence of this time deadline, constructing the single mobile element tour to collect the data of the sensors via single-hop communication is not expected to obtain a feasible solution. The typical speed of the mobile element can be about 0.1-2 m/s[17][18], resulting in substantial travelling time for the ME and, correspondingly, delay in gathering the sensors' data, and eventually violating the time deadline.

To address this problem, several proposals presented a hybrid method, which merges multi-hop forwarding with the use of mobile elements. In this method, some nodes behave as *caching points* (CPs) for data from other sensors. When an ME reaches a CP, it polls the data stored in the CP as well as data in other sensors. The ME is thus able to collect data from the network without having to physically visit all the sensors. This hybrid method arises the generic optimization problem; that is: determine the set of CPs such that the ME tour length is below a given constraint, that insures a minimum communication needs of the sensor nodes. The nature of the minimization objective can vary according to the application requirements, for example. it can be based on the number of hops or Euclidean distance, and target either the total or the maximum sum. The focus is on the problem of minimizing the total number of forwarding hops from all sensors to their respective nearest CPs; in other words, we target a solution where the forwarding requirements of all nodes are balanced as much as possible (subject to the constraint on the ME tour length). We can say that the variation with the most practical importance, as each node's hop counts to the closest CP is very correlated to the energy consumption it imposes on its peers, and, ultimately, to the network lifetime.

In this paper, we investigate this optimization problem, which we refer to as the *Periodic Rendezvous Data* Collection (PRDC). Accordingly, we present two algorithmic solutions that address two application gathering scenarios. In the first scenario, that can be describe as the general one, we assume that each node forward exactly the number of packets as it received without employing any aggregation model. In the second model, we assume that the network adopt the *n-to-1* aggregation model. In this model, each node wait until it receives all of the packets from its children in the routing tree, then aggregate these packets and send only packet. Considering the adopted aggregation model during the designing of the algorithmic solution is major issue in order to further increase the lifetime of network. "Roughly speaking" this is established since in the first scenario, the number of forwarding hops will play an important factor in determining the network lifetime, where in the second scenario, the degree of the nodes the routing trees will be the factor.

Authors α ρ : Faculty of Information Technology Al Hussein Bin Talal University, Maan, Jordan. E-mails : mahameed@ahu.edu.jo., k.almiani@gmail.com
Author σ : Faculty of Computer and Information System Jazan University, Saudi Arabia. E-mail : aalsalem.m@gmail.com
Author ω : College of Computer Science and IT University of Dammam, Saudi Arabia. E-mail : ghalnaymat@ud.edu.sa

We begin with describing the first algorithm that we refer to as the Cluster-Based algorithm. This algorithm groups the network into a number of balanced-size clusters with a single caching point in each cluster, and iteratively improves the solution by alternating between the mobile element tour building phase and the caching points forwarding tree computation phase. We then describe the second algorithm that we refer to as the degree-based algorithm. The main step of this algorithm is to structure the routing trees in a way to avoid having nodes with high number of routing trees branch routed at this node (in other words high degree nodes). This work significantly extends our earlier results [14], by providing the second algorithm. We evaluate the algorithms experimentally on a wide range of practical scenarios, showing that it consistently outperforms the algorithm of [12].

The rest of the paper is organised as follows. Section II provides a formal definition of the problem, and Section III presents the related work in this research area. Section V presents the CB and DB algorithms, which are then evaluated in Section VI. Finally, Section VII concludes the paper.

II. Problem Definition

An instance of the PRDC problem consists of an undirected graph $G = (V, E)$, where V is the set of vertices representing the locations of the sensors in the network, and E is the set of edges that represents the communication network topology, i.e. $(v_i, v_j) \Sigma E$ iff vi, vj are within each other's communication range. A distinguished vertex $v_s \Sigma V$ denotes the location of the sink. In addition, the complete graph $G' = (V, E')$, where $E' = V \times V$, represents the possible movements of the ME. Each edge $(v_i, v_j) \Sigma E'$ has a length r_{ij}, which represents the time needed by the ME to travel between sensor v_i and v_j. The data of all sensors must be uploaded to the ME periodically at least once in L time units, where L is determined from the application requirements and the sensors' buffer size. In other words, we assume the ME conducts its tour periodically, with L being a constraint on the maximum tour length. In this paper, for simplicity, we assume that the ME travels at constant speed, and that, therefore, the travelling times between sensors (r_{ij}) correspond directly to their respective Euclidean distances; however, this assumption is not essential to the solution algorithm and can be easily alleviated if necessary.

A solution to the PRDC problem in the general application scenario consists of a tour (i.e. a path in G') that starts and ends in v_s, where the length of the tour is bounded by L, such that the sum of hop-distances between every sensor and the tour is minimized. Where once the *n-1* aggregations model the goal become reducing the upper bound of the highest degree possible for the nodes.

III. Related Work

This work is categorised as a merging multi-hop forwarding with data collection by MEs. Earlier research, [20][21], assumed the mobile route to be predefined and mainly concerned with the timing of transmissions, to minimize the need for in-network caching by timing the transmissions to coincide with the passing of the tour. In [12][11], the minimum-energy Rendezvous Planning Problem (RPP) was introduced. This problem deals with determining the set of *rendezvous* points constructing the ME tour. In RPP, the target is to have the *Euclidean* distance, between the source nodes and the tour, as minimum as possible. Unlike the PRDC problem that we consider in this paper, where we aim to minimize the *hop* distance, as the Euclidean distance is not a reliable indicator of the true communication cost between nodes, because the existance of physical barriers. In addition, the method of [12][11] requires that a sensor is able to aggregate packets from multiple sources into a single transmission, which thereby limits the extent that it can be used in practice. From an algorithmic perspective, the solutions in [12][11] is performed by first computing the maximal tour under the constraint, and then building the forwarding trees around that tour. This separation reduces the solution search space, but our proposed algorithm repeats the tour building and forwarding tree computation phases in a way that iteratively improve the solution.

Path finding algorithms based on max flow computations have been considered by [16]. However the problem they consider is finding a path through the network area, which does not need to move from a sensor location to another. In our case this is a restriction for the mobile element. Also the mobile element moves along the determined path in the case of [16] while in our case we are looking for tour that does not revisit any node. The problem presented in this work shares some similarities with the mobile element navigation problem defined by [22]. As a solution for this problem, the authors presented an integrated mobile element navigation and data routing framework. This framework aims to achieve a desired trade-off between energy consumption and delay in the network. The objective of this framework is to determine the mobile element tour such that each node is at most k-hops away from the tour, where k is given. The proposed process starts by identifying the nodes, which will be involved in the mobile element tour. Then the tour of the mobile element is constructed by employing the TSP-solver developed by Bonabeau et al.[23]. To identify the nodes involved in the mobile element tour, the authors proposed a heuristic-based approach. This approach starts by representing the network as a tree, and then the process works by dividing this tree into sub-trees, where the width of each sub-tree is bounded by k. By bounding the number of hops, the authors aim to achieve a desired balance between the lifetime of the network and End-to-End delay.

The PRDC problem shares some similarities with the Vehicle Routing Problem (VRP)[24]. Given a fleet of vehicles assigned to a depot, VRP deals with determining the fleet routes to deliver goods from a depot to customers while minimizing the vehicles' total travel cost. Among the VRP variations, the Vehicle Routing Problem with Time Windows (VRPTW) [25]is the closest to PRDC. In VRPTW, each customer must be visited by exactly one vehicle and within a pre-defined time interval. PRDC also shares some similarity with the Deadline Travelling Salesman Problem (Deadline-TSP)[26], which can be described as seeking the minimum tour length for a salesman to visit a set of cities, where each city must be visited before a pre-determined time deadline. In particular, the special case when all cites have the same deadline reduces Deadline-TSP to the well known Orienteering problem[27]; PRDC can thus be considered as a generalization of the Orienteering problem.

IV. Algorithmic Solutions

The proposed algorithm is based on the insight that, in trying to maximize the network lifetime, the problem of finding the most efficient ME tour and that of finding the best forwarding trees from sensors to the tour are dependent; the solution of each has a intense impact on the outcome of the other. Hence, the two problems should ideally be solved jointly. Since a joint solution is intractable for all except the smallest instance sizes, we propose two algorithms, the Cluster-Based (CB) and the Degree-Based (DB).

a) The Cluster-Based Algorithm

This algorithm iteratively obtains the mobile element tour and the routing trees, this improves the solution in each iteration based on the result of the previous one. To accomplish this, the algorithm partitions the network into energy-aware clusters, such that in each cluster a single CP is finally chosen.

The CB algorithm comprises of two phases: tour-building and final-tour-improvement. During the tour-building phase, the algorithm finds a tour that visits as many clusters as possible, where clusters are obtained by partitioning the network into groups of approximately the same number of nodes. This type of construction balances the forwarding traffic inside the clusters. As soon as this tour is obtained, the final-tour-improvement phase starts to enhance the quality of this tour.

Figure 1 shows the structure of the CB algorithm. Lines 1-11 correspond to the tour-building phase; the second phase (final-tour-improvement) is given in line 12 and is described in detail in subsection IV.C. The tour-building phase follows a process similar to the binary-search mechanism. In each round, the process selects number of clusters to be established (c) from the middle of the range of possible values so far, which initially starts from 1 to the total number of sensors in the network. In case the choice of produces a tour that satisfies the length constraint L, then the lower half of the range is deleted and the process is repeated; conversely, if the length constraint is not satisfied, the upper range is deleted instead. The search stops if the maximum number of clusters is found that is able to satisfy the tour length constraint.

In line 4, we show how to construct a tour for a given number of clusters c. This construction comprises two steps: first, partitioning the network into clusters, followed by finding a shortest tour that visits one node in each cluster. These steps are explained in more detail in the following sections.

```
     Input: $G$ (network topology graph), $G'$ (ME movements graph), $L$ (tour length constraint)
     Output: $final\_tour$ (ME tour)
1    $l \leftarrow 0, u \leftarrow n$
2    $c \leftarrow \lfloor (u-l)/2 \rfloor$
3    While $u - l > 1$
4        do $tour \leftarrow build\_Tour(G, G', c, L)$
5            If $tour\ travelling\ time \leq L$
6                Then $l \leftarrow c$
7                    $last\_valid\_tour \leftarrow tour$
8                    If $c = n$
9                        Then break
10           Else $u \leftarrow c$
11           $c \leftarrow \lfloor (u+l)/2 \rfloor$
12   $final\_tour \leftarrow tour\_improvement(valid\_tour, clusters)$
```

Figure 1 : The steps of the CB algorithm

i. *Clustering Step*

The clustering step finds a given number of clusters such that the sum of hop-distances is minimized among nodes belonging to the same cluster. Hence, in a network of homogeneous node density, this results to a balanced number of nodes in the clusters. To that end, the clustering step works by bounding the distance between a node and its cluster's centre node (c), that is defined as the node that has the minimum total hop-distance to all other nodes in the cluster.

Figure 2 shows the process of the clustering step. At the start, c nodes are selected randomly as the initial clusters centre nodes. Then, based on the hop-distance to the cluster's centre node every other node is assigned to its closest cluster. If all nodes have been assigned, the centre node for each cluster is recalculated, and the process is repeated from the beginning based on the new c cluster centres. The clustering step stops when the identity of the clusters' centre nodes does not change between two consecutive iterations.

```
     Input: $G$ (topology graph), $c$ (number of established clusters)
     Output: a set of clusters
1    Randomly choose $c$ nodes as initial cluster centres
2    **Do**
3        **for** all nodes in $G$
4            assign each node to its nearest cluster centre (in terms of hop-distance)
5        Recalculate the centre nodes of the resulting clusters
6    **Until set of centre nodes is unchanged from previous iteration**
```

Figure 2 : The Clustering Step

ii. *Tour-Finding Step*

When all clusters are found in the clustering step, the next phase is to find a tour that traverses exactly one node (the CP) from each cluster. To guarantee an ideal communication energy consumption, the CP in each cluster should be the cluster centre node, since it has the minimum hop-distance to all other nodes in the same cluster. Inappropriately, this will usually result to have a tour with a much longer travelling time than many other possible tours. However, the objective of the tour-finding step is to find the tour with the shortest overall length that traverses exactly one node from each cluster. Even though such a tour will maybe end up with sub-optimal CPs for the given set of clusters, it allows the overall algorithm to finally attain a larger number of clusters while satisfying the tour length constraint. Certainly, the number of clusters has a greater impact on the overall quality of the solution than the choice of a particular CP inside a cluster.

The problem solved in the tour-finding step is thus an instance of One-of-a-Set TSP [28]. Its solution consists of two parts, namely: nodes identification, during which the identity of the CPs is found, and *tour construction* which finds the optimal tour among the chosen points. In our algorithm, the nodes identification simply iterates over the clusters and selects the nearest node to the set of CPs so far. If the nodes are found, the optimal tour connecting them is identified; for example, using Christofides algorithm [29]. Figure 3 provides a psaudocode of the process performed in tour-finding step.

```
     Input: $G$ (topology graph), $clusters$
     Output: $CPs, tour$
1    $CPs \leftarrow v_s$
2    $tagged \leftarrow \emptyset$
3    **While** $tagged \neq clusters$
4        find the node in $\{clusters\} \backslash \{tagged\}$ that is closest to any of the nodes in $CPs$
5            Add the cluster of the selected node to $tagged$
6            Add the selected node to $CPs$
7    $tour \leftarrow tour\_construction(CPs)$
```

Figure 3 : The tour-finding step

iii. *Final Tour Improvement Phase*

In this section, we describe a final tour phase to enhance the quality of the tour found in the first phase. Up to this stage, the network is partitioned to the maximum possible number of clusters such that the resulting tour does not violate the length constraint L. Yet, the tour itself is the minimum-length for this set of clusters, and naturally, the travelling time of the tour at this stage is strictly less than L. This gap raises the possibility of amending the tour by choosing different CPs (closer to the respective cluster centres), as long as the tour length constraint stays satisfied, so as to reduce the total hop-distance between CPs and other nodes in their respective clusters.

The final tour improvement phase is given in Figure 4. For every cluster, unless the corresponding CP already happens to be the cluster's centre node, an alternative CP is considered (denoted CP') which is the next node on the shortest path from CP to the cluster's centre in G. The algorithm calculates how much the ME tour length would increase if CP were replaced by CP' in this cluster only, and performs the change to the alternative CP' for the cluster where the length increase is minimal. Repeatedly the process keeps running until it is no longer possible to change the CP without violating the tour length constraint L.

```
  Input: $T$ (tour), $CPs$ (clusters' caching points), $L$ (tour length constraint), clusters
  Output: $T$ (tour to be assigned to the ME)
1 $l \leftarrow$ find the closest cluster to the tour (the distance between the cluster centre node and the tour)
2 $T' \leftarrow T$
3 While $travelling\ time\ for\ T' < L$
4     do $T \leftarrow T'$
5         If all clusters' centre nodes are already the CPs, Then exit
6         For every cluster $l$ where $\mathrm{CP}(l)$ is not the centre node, denote $\mathrm{CP}'(l)$ to be the next node on the shortest path from $\mathrm{CP}(l)$ to the cluster's centre
7         Set $l^*$ to be the cluster where swapping $\mathrm{CP}(l)$ for $\mathrm{CP}'(l)$ in $T$ causes the minimal increase in the tour length of $T$
8         In $T'$ swap $\mathrm{CP}(l^*)$ for $\mathrm{CP}'(l^*)$, update edges accordingly
9 Return $T$ (the final tour)
```

Figure 4 : The final tour improvement phase

b) *The Degree-Based Algorithm*

In this algorithm, since the application is assumed to use the *n-to-1* aggregation model, minimizing the total number of hops is no longer an important issue. With such model, having the same number of nodes in different routing tree structure (regardless the number of hops) will result in the same network lifetime, as long as the degree of the nodes in all cases are the same. This is established, since in this scenario, the energy consumed by receiving is the main issue in determining the lifetime of the node. For instance, imagine that you are given a sub-tree consists of ten nodes rooted at node n. Now designing the routing tree for this sub-tree in a way that makes each node directly transmit its data to the node n results in significantly reducing the lifetime of node n, compared to the situation where the nodes are connected in a chain-structure (each node receive from maximum one node).

To this end, the DB algorithm is designed with the objective of reducing the maximum degree possible for each node in the routing tree. The DB algorithm starts by constructing the Shortest Path Tree (SPT) rooted at the sink. Once this tree is obtained, the algorithm proceeds by eliminating the nodes (except the sink) that only have one child in the SPT. Once any node is eliminated, the tree connectivity will be maintained by add an edge between the parent and the child of the eliminated node. Once this elimination process is performed, the resultant tree will consist of the following type of nodes:

i. *Leaf Nodes*

Nodes with no children.

ii. *High Degree Nodes (HD-Nodes)*

Nodes that have more than one child in the tree.

iii. *Intermediate Nodes*

Nodes with high degree nodes their parents. In addition, they must have one leaf node as a child.

Now, if we select the intermediate nodes as the caching points, the resultant network lifetime will be optimal, since each node in the routing will receive data from only one node. However, the length of such a tour might violate the time transit constraint. In this direction, the caching point identification step works by selecting subset of the intermediate nodes to be the caching points with the objective of increasing the lifetime of the network. In this step, each HD-node (n_i) will be assigned a value p_i that represents the number of children for this node. This value is used to prioritize selecting the caching points to be nodes that have HD-nodes as their parents, since reducing the number of children for these HD-nodes is a major factor in increasing the lifetime of the network. This step works by iteratively reduce the degree of such nodes. Now, in each iteration, the step works by adding the nearest child of the HD-node with the biggest p_i to the current constructed tour.

This step starts by assigning a value l_i for each intermediate node v_i. This value is the number of children for this node parent in the original SPT. If this addition result in a tour satisfies the time transit constraint, the added node as well as its child will be removed from the SPT, and the p_i value for this node will be subtracted by one. Then, the caching point identification step will be re-triggered to select caching point as the child of the HD-node that have the biggest p_i. This process stops when no new caching point (intermediate node) can be added to the tour without violating the transit constraint. Figure 5 shows the steps of the DB algorithm. The tour is obtained using Christofides algorithm [29].

```
  Input: $G$ (Graph topology), $L$ (tour length constraint), clusters
  Output: $T$ (tour to be assigned to the ME)
1 $SPT \leftarrow SPT(G)$
2 $SPT' \leftarrow eliminate(SPT)$
3 $tour \leftarrow sink$
4     while $tour < L$ and nodes not empty
5         Do for each node calculate $p_i$
6         $tour \leftarrow$ cachingPointsIdentification()
7     If $tour > L$
8         Then undo the last step
9 Return tour
```

Figure 5 : The Degree-based Algorithm

V. Simulation Evaluation and Results

To validate the performance of the CB and the DB algorithms, we have conducted an extensive set of experiments using the J-sim simulator for WSN[30]. Due to space constraints, we only present a sample of our results here, based on a few representative scenarios that are described henceforth. Unless mentioned otherwise, the network area is $160{,}000\ m^2$. The radio parameters are set according to the MICAz data sheet [31], namely: the radio bandwidth is 250 $kbps$, the transmission power is 21 mW, the receiving power is 15 mW, and the initial battery power is 20 Joules. Each sensor node sends one packet per ME tour, where the packet has a fixed size of 100 bytes. Each experiment is an average of 10 different random topologies. We are particularly interested in investigating the following metrics:

- Network lifetime
- Number of CPs
- Total size of routing trees

The deployment of sensors is typically application-dependant and the evaluation results will depend on the deployment characteristics. In this evaluation, we consider the following scenarios:

- Uniform deployment: in this scenario, we assume that the nodes are uniformly deployed in a square area of $400 \times 400\ m^2$.
- Multi-level: in this scenario, we divide the network into a 5×5 grid of squares, where each square is $80 \times 80\ m^2$. Then, we randomly choose 10 of the squares, and in each one of those we fix the node density to be 5 times the density in the remaining squares.

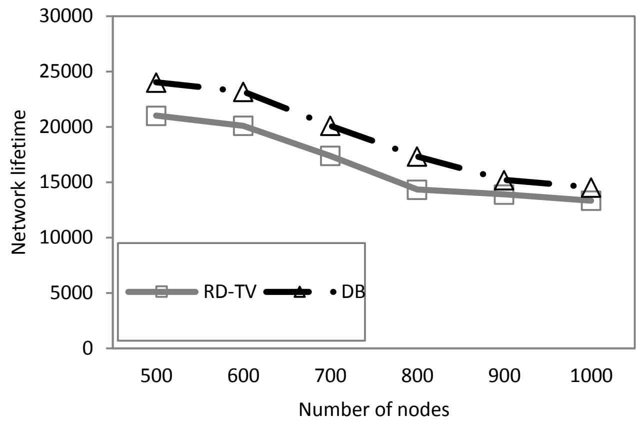

Figure 6 : Network lifetime vs number of nodes for the uniform deployment scenario (n-to-1 aggregation model)

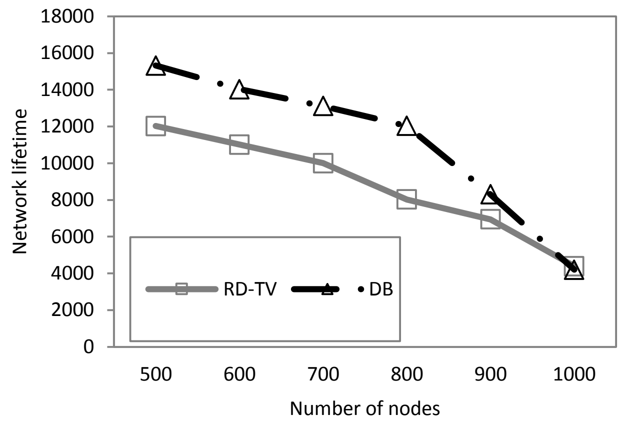

Figure 7 : Network lifetime vs number of nodes for the multi-level deployment scenario.(n-to-1 aggregation model)

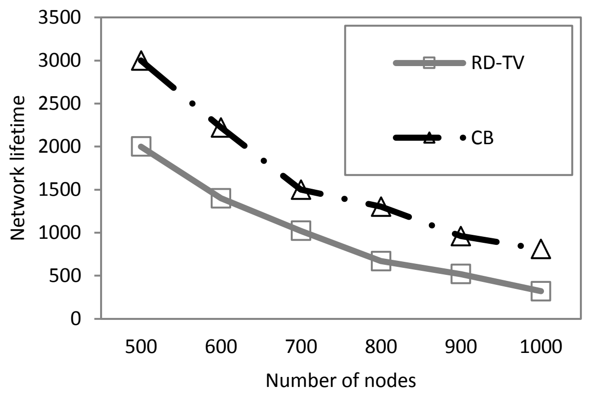

Figure 8 : Network lifetime vs number of nodes for the uniform deployment scenario

To benchmark our algorithm, we compare it against the Rendezvous Design for Variable Tracks (RD-VT) algorithm presented in[12]. The RD-VT algorithm works by constructing the MST of the sensor network and traversing it in preorder, until a tour of the nodes covered so far can no longer be found without violating the length constraint. To ensure the fairness of the comparison, we use the Christofides algorithm [29], i.e. the same algorithm we use in our tour-finding step (see subsection V.B), to find a tour for a given set of nodes in every iteration of RD-VT as well. Eventually, each sensor is connected to the nearest point of the tour via the shortest path.

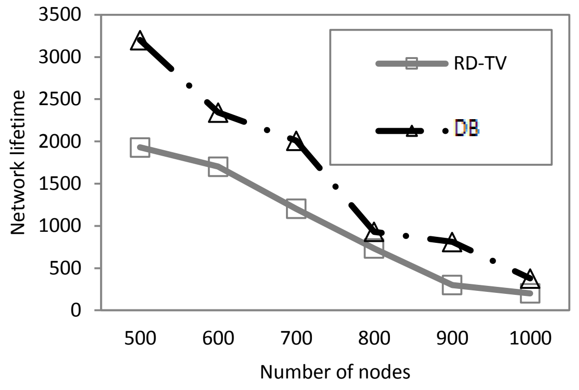

Figure 9 : Network lifetime vs number of nodes for the multi-level deployment scenario

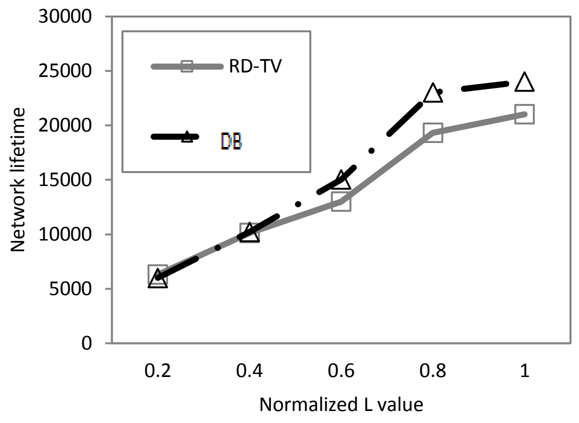

Figure 10 : Network lifetime vs number of nodes for the uniform deployment scenario (n-to-1 aggregation model)

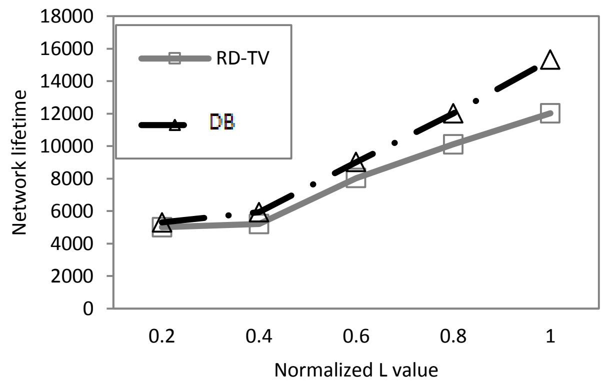

Figure 11 : Network lifetime vs number of nodes for the multi-level deployment scenario.(n-to-1 aggregation model)

a) Network Lifetime

In this evaluation, we consider the 1% network lifetime metric, which is defined as the time until 1% of nodes run out of energy. For simplicity, we only account for the radio receiving and transmitting energy. Figures 5-8 show the results for both deployment scenarios as a function of the number of nodes (equivalently, network density), for a value of L that is set to 0.15 $\cdot s \cdot T_L$, where $s = 1$ m/s is the speed of the mobile element, and T_L is the length of the minimum spanning tree (MST) that connects all the nodes for 1000-nodes network. Figures 6 and 7 show the result for the scenario where the application adopts the 1-to-n aggregation model, and figures 8 and 9 shows the results when there is no aggregation model used by the application. From figures 6 and 7 we can see that the DB algorithm constantly outperforms the RD-VT algorithm. Also, we can see from these figures that increasing the number of nodes results in increasing the gap between the DB and the RD-VT algorithms, especially in the multi-level deployment scenarios. This is mainly due to the mechanism both algorithms used. The DB algorithm works by reducing the degree of the nodes (number of tree branches) inside the routing trees, and as we discussed before, the number of routing tree branches is the main factor in determining the lifetime of the network. The RD-VT algorithm construct it solution by traversing the MST in preorder, and such traversing does not take into account the degree of the nodes during the construction of the tours. These are the main factors behind the shown performance.

Also, from figures 8 and 9, we can see that the CB consistently outperforms RD-VT. This also due to how each algorithm constructs its tour. The RD-VT algorithm constructs its solution by traversing the MST, and such traversing does not take the distribution of the nodes during the construction of the solution. On the other hand, the CB algorithm works by dividing the network into similarly sized clusters and building the tour to visit one node from each cluster; this results in a relatively balanced communication load of the clusters, and thereby increases the network lifetime. The influence of tis factor is more obvious in the multi-level deployment scenario.

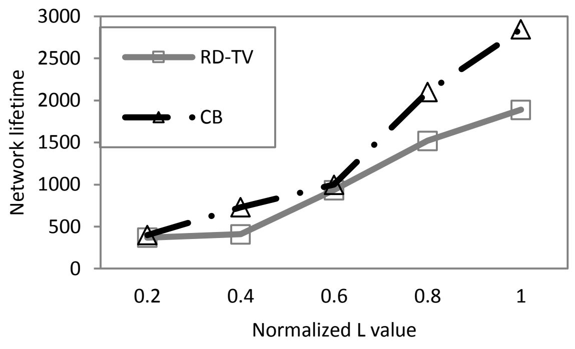

Figure 12 : Network lifetime vs number of nodes for the uniform deployment scenario

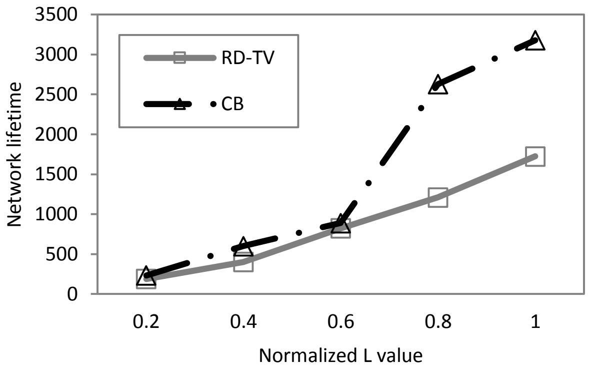

Figure 13 : Network lifetime vs number of nodes for the multi-level deployment scenario

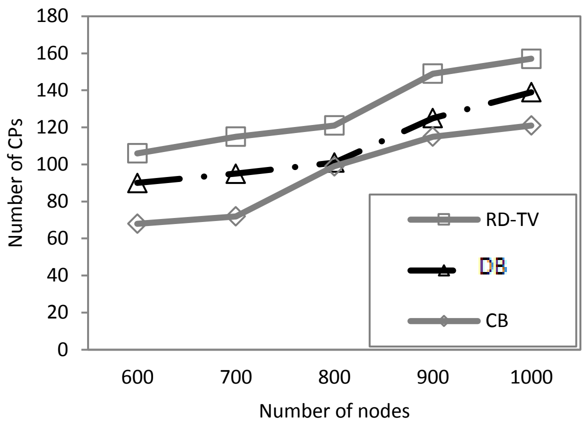

Figure 14 : Number of caching points vs number of nodes for the uniform deployment scenario

We proceed to show how the network lifetime depends on the value of the tour length constraint L. Figures 10-13 show the results for both deployment scenarios with 500 nodes. Figures 10 and 11 show the result for the 1-to-n aggregation model, and figures 12 and 13 show the results where there is no aggregation model in used. Here, the horizontal axis shows the value of L normalized as a fraction of $\cdot T_L$. We observe that, reducing the value of the transit constraint results in reducing the gap between these algorithms performances. This is mainly due to the fact that reducing the transit constraint results in significantly reducing the solution space. Such reduction results in reducing the importance of the algorithms key factors, since it will significantly limit the number of feasible solutions.

b) Number of CPs

Figures 14 and 15 show the impact of the network density on the number of CPs each algorithm obtains. The figures show that for the same tour length, RD-VT includes more CPs than DB and CB. As one would expect, this is due to RD-VT mechanism of traversing the tree. However, as previously shown in the discussion on network lifetime, the number of CPs in itself has no impact on the resulting performance; it is the *location* of the CPs that is the main factor that influences the network lifetime.

c) Size of Routing Trees

Figures 16 and 17 show the impact of the number of nodes on the total size of the routing trees (i.e. the sum of hop-distances from all sensor nodes to their respective CPs) that each algorithm obtains. The figures show that the size of the routing trees obtained by the DB algorithm is smaller than the one obtained by the RD-VT algorithm and bigger than the one obtained by the CB algorithm. Although the RD-VT algorithm obtained more CPs than the CB and the DB algorithms, the latter two algorithms nevertheless achieves smaller routing trees overall. This is because the tree traversal process used by RD-VT results in a set of CPs that includes many mutual neighbors, which is not useful when those nodes are used as roots for separate trees; in other words, many of the resulting trees are very small, while only a limited number of CPs end up being roots of large trees (i.e. serve as cache points for a large number of sensors). On the other hand, the CPs selected by the CB and the DB algorithms are distributed more evenly across the network, resulting in trees whose size is better balanced and therefore lowering the total sum of hop-distances of all nodes.

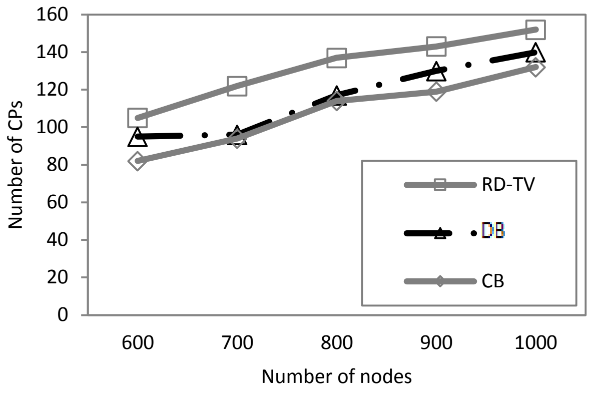

Figure 15 : Number of caching points vs number of nodes for the multi-level deployment scenario

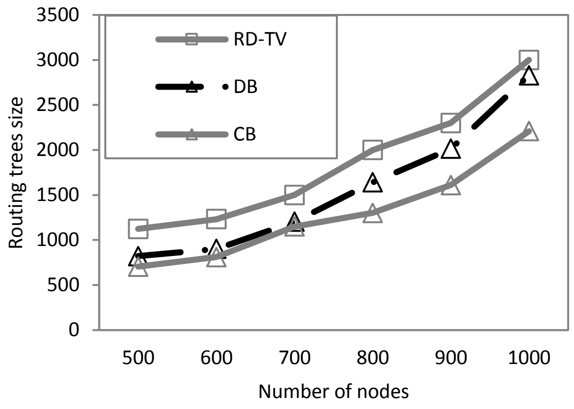

Figure 16 : Size of routing trees vs number of nodes for the uniform deployment scenario

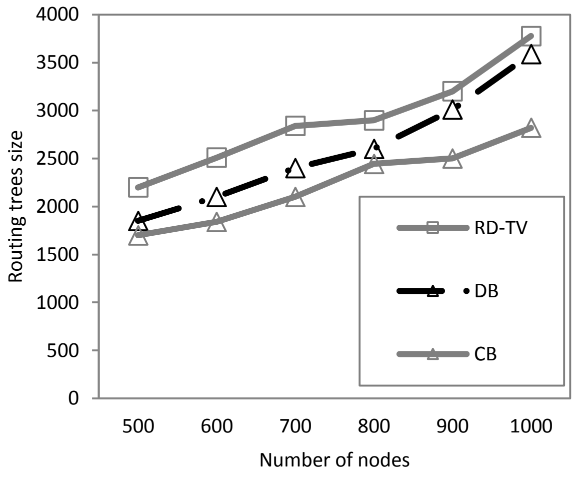

Figure 17 : Size of routing trees vs number of nodes for the multi-level deployment scenario

VI. Conclusion

In this paper, to find efficient tours for mobile elements in WSNs, we presented two algorithmic solutions. The difference between the proposed solutions is in the adopted assumption for the application scenario. In the first algorithm, we assumed that the n-to-1 aggregation model is employed, and in the second algorithm, no assumption about the availability of aggregation was made. Such information helped by emphasizing the main factors behind increasing the lifetime of the network during the construction of the tours. Through a wide range of simulation scenarios, we showed that the proposed algorithms increase the resulting network lifetime significantly compared to the previously best known solutions.

References Références Referencias

1. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. B. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello, "Making sensor networks practical with robots," Lecture notes in computer science, pp. 152–166, 2002.
2. J. Butler, "Robotics and ssMicroelectronics: Mobile Robots as Gateways into Wireless Sensor Networks," Technology@Intel Magazine.
3. E. Ekici, Y. Gu, and D. Bozdag, "Mobility-based communication in wireless sensor networks," IEEE Communications Magazine, vol. 44, no. 7, pp. 56 – 62, 2006.
4. K. Almi'ani, S. Selvadurai, and A. Viglas, "Periodic Mobile Multi-Gateway Scheduling," in Proceedings of the Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2008, pp. 195–202.
5. Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C. G. Lee, "Partitioning based mobile element scheduling in wireless sensor networks," in Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on, 2005, pp. 386–395.
6. S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in Proceedings of IEEE Globecom, 2003, vol. 1, pp. 377–381.
7. A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," IEEE transactions on Mobile Computing, vol. 6, no. 4, pp. 395–410, 2007.
8. A. Somasundara, A. Ramamoorthy, and B. Srivastava, "Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines," in Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International, 2004, pp. 296 – 305.
9. S. S. Skiena, "Traveling Salesman Problem," The Algorithm Design Manual, pp. 319–322, 1997.
10. Y. Xu, Zichuan and Liang, Weifa and Xu, "Network Lifetime Maximization in Delay-Tolerant Sensor Networks With a Mobile Sink," in Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems, 2012, pp. 9–16.
11. G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in mobility-assisted wireless sensor networks," in proceedings of the 28th IEEE InternationalReal-Time Systems Symposium (RTSS), 2007, pp. 311 – 320.
12. G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), 2008, pp. 231–240.
13. G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," IEEE Transactions on Mobile Computing, vol. 7, no. 12, pp. 1430–1443, Dec. 2008.
14. K. Almi'ani, A. Viglas, and L. Libman, "Energy-efficient data gathering with tour length-constrained mobile elements in wireless sensor networks," in IEEE 35th Conference on Local Computer Networks (LCN), 2010, pp. 582 – 589.
15. K. Almi'ani, A. Viglas, and M. Aalsalem, "Mobile Element Path Planning for Gathering Transit-Time Constrained Data," in 12th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011, pp. 221–226.
16. M. Ma and Y. Yang, "SenCar: An Energy-Efficient Data Gathering Mechanism for Large-Scale Multihop Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 10, pp. 1476–1488, Oct. 2007.
17. R. Pon, M. A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W. J. Kaiser, and others, "Networked infomechanical systems: a mobile embedded networked sensor platform," in Proceedings of the 4th international symposium on Information processing in sensor networks, 2005, pp. 376 – 381.
18. K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, "Robomote: enabling mobility in sensor networks," in Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN), 2005, pp. 404 – 409.
19. Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, "Exploiting sink mobility for maximizing sensor networks lifetime," in Proceedings ofthe 38th Annual Hawaii International Conference on System Sciences (HICSS), 2005, vol. 9, pp. 03–06.
20. D. Jea, A. Somasundara, and M. Srivastava, "Multiple controlled mobile elements (data mules) for data collection in sensor networks," Lecture Notes in Computer Science, vol. 3560, pp. 244–257, 2005.
21. A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, "Controllably mobile

infrastructure for low energy embedded networks," IEEE Transactions on Mobile Computing, vol. 5, no. 8, pp. 958–973, 2006.

22. J. Rao and S. Biswas, "Joint routing and navigation protocols for data harvesting in sensor networks," in 2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008, pp. 143–152.
23. E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm Intelligence," Oxford university press, 1999.
24. P. Toth and D. Vigo, "The Vehicle Routing Problem," Society for Industrial & Applied Mathematics (SIAM), 2001.
25. M. M. Solomon, "ALGORITHMS FOR AND SCHEDULING PROBLEMS THE VEHICLE ROUTING WITH TIME WINDOW CONSTRAINTSS," Operations Research, vol. 35, no. 2, pp. 254–265, 2010.
26. N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Approximation algorithms for deadline-TSP and vehicle routing with time-windows," in Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC), 2004, pp. 166 – 174.
27. B. Golden, L. Levy, and R. Vohra, "The orienteering problem," Naval Research Logistics, vol. 34, no. 3, pp. 307–318, 2006.
28. J. Mitchell, "Geometric shortest paths and network optimization," in Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, 1998, pp. 633– 701.
29. N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem," 1976.
30. J. C. Hou, L. Kung, N. Li, H. Zhang, W. Chen, H. Tyan, and H. Lim, "J-Sim: A Simulation and emulation environment for wireless sensor networks," IEEE Wireless Communications Magazine, vol. 13, no. 4, pp. 104–119, 2006.
31. J. L. Hill and D. E. Culler, "Mica: A wireless platform for deeply embedded networks," IEEE micro, vol. 22, no. 6, pp. 12–24, 2002.

This page is intentionally left blank