PATH-CONSTRAINED DATA GATHERING SCHEME FOR WIRELESS SENSOR NETWORKS WITH MOBILE ELEMENTS

Bassam A. Alqaralleh and Khaled Almi'ani
Al-Hussein Bin Talal University, Jordan

ABSTRACT

Wireless Sensor Networks (WSNs) have emerged as a promising solution for variety of applications. Recently, in order to increase the lifetime of the network, many proposals have introduced the use of Mobile Elements (MEs) as a mechanical carrier to collect data. In this paper, we investigate the problem of designing the mobile element tour to visit subset of the nodes, termed as caching points, where the length of the mobile element tour is bounded by pre-determined length. Caching can be implemented at various points on the network such that any node in the network is at most k-hops away from one of these caching points. To address this problem, we present heuristic-based solution. Our solution works by partitioning the network such that the depth of each partition is bounded by k. Then, in each partition, the minimum number of required caching points is identified. We compare the resulting performance of our algorithm with the best known comparable schemes in the literature.

KEYWORDS

Wireless Sensor Networks, Mobile Element

1. INTRODUCTION

Wireless Sensor Networks (WSNs) composed of multifunctional miniature devices with sensing, computation and wireless communication capabilities. Such devices are normally battery operated. Thus, recharging a sensor node is impractical because they are typically designed for hostile environments; therefore, energy efficiency is critical. In typical static multi-hop data communication paradigm, the sensors around the sink are likely the first to run out of energy. This is due to the fact that these sensors carry heavier traffic loads. Once these sensor nodes fail, the operational lifetime of the networks ends and the network stops working because the entire network becomes unable to communicate with the sink. Therefore, reducing energy consumption in WSNs is becoming a major design challenge.

In order to significantly increase the lifetime of the network via considerably reducing the energy consumption, many proposals [1][2][3] have introduced the use of the Mobile Elements (MEs) as an efficient approach. A mobile element roams through the network and collects data from sensor nodes via short range (single-hop) communications. Therefore, in comparison with multi-hop communications the energy consumption can be considerably reduced. However, the speed of the mobile element is typically low[4][5], and therefore reducing the data gathering latency is important to ensure the efficiency of such approach.

To address this problem, several proposals presented a hybrid approach, which combines multi-hop forwarding with the use of mobile elements. In this approach, each mobile element visits subset of the nodes termed as caching points. These caching points store the data of the nodes that are not included in the tour of the mobile element. Once a mobile element becomes within the transmission range of a caching point, the caching point transmits its data to the mobile element. By adopting such an approach, the mobile element gathers the data of the entire network without the need for visiting each node physically.

In this paper, we focus on the problem of reducing the latency of the mobile elements tours. Accordingly, we investigate the K-Hop tour planning (KH-tour) problem. This problem deals with designing the shortest possible tour for the mobile, such that the tour start and end at the sink, and the maximum number of hops between any node and the tour is bounded by pre-determined value.

To address this problem, we present an algorithmic-based solution. This solution is based on partitioning the network into groups of partitions, where the depth of each partition is selected to ensure that one node in each partition can be identified as a caching point without violating the hop constraints. Then, the tour of the mobile element will be constructed to visit all nodes that have been identified as caching points.

The rest of the paper is organized as follows. Section 2 provides formal definition of the presented problem. Section 3 presents the related work in this area. Section 4 presents the heuristic solution of the presented problem. In Section 6, the evaluation for the proposed heuristics is presented. Finally, Section 7 concludes the paper.

2. RELATED WORKS

Many proposals in the recent literature have studied the use of mobile elements to reduce the energy consumption in sensor networks. In this section, we discuss the most related work in the literature.

The problem presented in our work can be recognized as an extended version of the tour scheduling problem presented by Somasundara et al. [5][6]. In this tour scheduling problem, the authors assume that each node must be visited within a time-deadline to avoid buffer overflow. Accordingly, the authors proposed different heuristics to determine the path of the mobile elements.

Guney et al. [7] formulated the problem of finding the optimal sink trajectory and the data flow routs as mixed integer programming formulations. Accordingly, they presented several heuristics to address this problem. Liang et al. [10]also presented a mixed integer programming formulation for similar problem, where they incorporate the constraint of the travelling to the formulation.

In [9], the authors proposed a data collection scheme aims to increase the network throughput. In this scheme, the mobile element is assumed to move periodically in pre-determined path, which is expected to visit a subset of the nodes called subsinks. The main goal of this scheme is optimizing the assignment of the sensor nodes to subsinks in order to increase the network throughput.

Zhao et al [11][12]investigated the problem of maximizing the overall network utility. Therefore, they presented two distributed algorithms for data gathering where the mobile sink stays at each anchor point (gathering point) for a period of sojourn time and collects data from nearby sensors via multi-hop communications. They considered both variable and fixed sojourn time.

The problem presented in this paper share some similarities with the Vehicle Routing Problem (VRP) [13]. 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 vehicle routing problem variations, the Vehicle Routing Problem with Time Windows (VRPTW) [14]is the closest to our problem. In VRPTW, each customer must be visited by exactly one vehicle and within a pre-defined time interval. The problem presented in this work also share some similarities with the Deadline Travelling Salesman Problem (Deadline-TSP)[14], which seeks 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 where all cites have the same deadline reduces Deadline-TSP to the well-known orienteering problem (OP) [15]; thus, our problem can be considered as a generalization of the orienteering problem.

The problem presented in this work shares some similarities with the problem proposed by Ma et al. [16]. In their proposal, they explored two settings where each polling point is either the location of a sensor or a point in the network area. On the other hand, the mobile element can communicate with one or more identified nodes in order to collect their data. Their goal is to design the mobile element tour which consists of polling points in order to find the shortest possible tour where each node is either on the tour or one-hop away from the tour.

The problem presented in this share some similarities with minimum-energy Rendezvous Planning Problem (RPP) [17][18]. In this problem, the objective is to determine the mobile element path such the total Euclidean distance between nodes not included in the tour and the tour is minimized. In [17], the authors presented the Rendezvous Design for Variable Tracks (RD-VT) algorithm. The process of this algorithm starts by construction the Steiner Minimum Tree (SMT) that connects the source nodes. Then, the obtained tree will be traversed in pre-order until no more nodes can be visited without violating the deadline constraint. In this algorithm the visited nodes is identified as the caching points. Xing et al. [3] provided a utility-based algorithm and address the optimal case for restricted version of the problem. Many proposals [19] have also investigated this problem.

3. PROBLEM DEFINITION

An instance of the KH-tour problem consists of an undirected complete graph $G = \langle V, E \rangle$, v_s. V is the set that represents the location of the sensor nodes in the network.v_s represents the location of the sink. E is the set of edges that represents the communication pattern in the network. For each pair of nodes $v_i, v_j \in V$ there is an edge between these two nodes, if they are within each other communication range. $d(v_i, v_j)$ represents the Euclidean distance that the mobile element will travel between nodes v_i and v_j. In addition, we use a pre-determined value k that represents the maximum number of hops allowed between any node and the tour.

A solution to the KH-tour problem consists of a tour (a path in G) that starts and ends in v_s, where the objective is to minimize the Euclidean distance of this tour, such that each node $v_i \in V$ is at most k-hops away from the tour.

4. INTEGER LINEAR PROGRAM FORMULATION

Let $V = \{v_0, v_1, \dots, v_n\}$ be the set the represents the nodes in the network($v_s \in V$). Also, let $d(u, v)$ be the distance which the mobile element needs to travel to reach node v from node u. We use the $y_{u,v}$ as an (0-1) indicator for each pair of nodes $(u, v) \in V$ such that $y_{u,v} = 1$, if the mobile element travels between node v and node u, and 0 otherwise. Also, let $z_u$be an integer variable for each node $u \in V$, taking only positive values, showing the order in which the nodes are visited in the resulting tour. In addition, let $x_{u,v}$be an integer variable for each pair of nodes $v, u \in V$ such that $x_{u,v} = 1$, if nodes v and u are one hop away from each other, and node u

stores node v data; 0 otherwise. Also, let $h(v,u)$ be the number of hops between nodes v and u. Given the Integer Linear Program for our problem can be given as follows:

Minimize
$$\sum_{i,j \in V} y_{i,j} \cdot d(i,j) \tag{1}$$

Subject to:
$$\sum_{i \in V} y_{j,i} \leq 1 \quad \forall j \in V \tag{2}$$
$$\sum_{i \in V} y_{i,j} \leq 1 \quad \forall j \in V \tag{3}$$
$$\sum_{i \in V} y_{i,j} - \sum_{i \in V} y_{j,i} = 0 \quad \forall i \in V \tag{4}$$
$$x_{i,j} \cdot h(v_i, v_j) \leq k \quad \forall i,j \in V \tag{5}$$
$$\sum_{i \in V} x_{i,j} + \sum_{i \in V} y_{j,i} = 1 \quad \forall j \in V \tag{6}$$
$$z_i - z_j + n \cdot y_{i,j} \leq n - 1 \quad \forall j \in V/\{v_s\} \; \forall i \in V \tag{7}$$

Constraints (2-4) ensure that the load at each node is balanced. Constraint (5) ensures that the number of hops between any node and the tour is at most. Each node must be either involved in the mobile element tour or connected to a node involved in this tour, and this is represented by constraint (6). Constraint (7) is the sub-tour elimination constraint. Typically, it is hard to solve such problem. However, the ILP is given to take close look at the problem.

5. ALGORITHMIC-BASED SOLUTION

Our objective is to determine the shortest mobile element tour such that the number of hops between any node not included in the tour and the tour is at most k. Towards this end, in this section, we present the Graph Partitioning (GP) algorithm. The main idea of this algorithm is to divide the graph of the network into partitions, such that the depth of each partition is at most $2 \cdot k$ hops. As we will discuss in this section, this algorithm is designed to address the situation where the nodes are uniformly deployed, and the partitioning process aims to simplify the step of identifying the caching points.

The Graph Partitioning (GP) algorithm starts by identifying the center node of the graph. This node is the closest to all other nodes (in term of number of hops) inside the graph. Then the process iterates to identify the nodes belong to each partition. Once the nodes in each partition are identified, they are flagged by their partition number, to make sure that they will not be reconsidered during the partitioning steps. Then, the caching point identification step starts the process of identifying the caching points in each partition. This process aims to identify the minimum number of required caching points, such that the number of hops between any node and one of these nodes is at most k. As we will see next, this process works by establishing a path from the nodes, which have the highest number of neighbors. Two nodes are neighbors, if they are at most k-hops away from each other. Establishing this path aims to select the minimum number of required caching points. The last step of this algorithm is to construct the mobile element tour from the identified caching points. **Algorithm** 1 illustrates these steps. Now, we discuss the steps of this algorithm in more details.

Algorithm 1. The Graph Partitioning (GP) algorithm

```
1: procedure GP$(G, G', k)$
        Inputs: the graphs $G$ and $G'$
$k$, maximum hops constraints
        Outputs: A tour $(T)$ and forwarding trees $(R)$
2: $cn$ ←CenterNode$(G)$
3: $P$ ←Partition$(G, cn)$
4: $C$ ←CachingPoints$(P, G)$
5:$T$ ←TourBuild(C)
6:$R$ ←BuildRouting$(T, G)$

7: procedure CenterNode ($G$)
8:          for each vertex $v$ in $G$ do
9:              dist$(v)$ ← sum of hop-distances from all nodes
10:         end for
11:         return node with minimum dist(v)
12: end procedure

13: procedure Partition ($G, cn$)
14:   for each vertex $v$ in $G$ do
15:     hops$(v)$ ←number of hops in the shortest path between $v$ and $cn$
16:     $i \leftarrow \lfloor \text{hops}(v)/2 \cdot k \rfloor + 1$
17:     assign $v$ to $P_i$
18:   end for
19:  return $P$
20:end procedure

21: procedure CachingPoints ($G, P, k$)
22:          for each $P_i$ in $P$ do
23:               for each $v_i, v_j$ in $P_i$ do
24:                    if path between $v_i$ and $v_j$ at most k hops do
25:                        add edge between $v_i$ and $v_j$ in $G$
26:               end for
27:          end for
28:          for each $P_i$ in $P$ do
29:               add node with highest degree to $C_i$
31:               Remove this node and its one-hope neighbors from $P_i$
32:          end for
33:          return $P$
34:end procedure
```

5.1 The partitioning step

This step starts by identifying the center node of the graph. Then, starting by the first partition, the process iterate to identify the nodes belong to each partition. The identity of the nodes that are assigned to each partition is selected based on the partition number, and the number of hops between any node and center node. For partition i, the set of the nodes that belong to this partition consists of the nodes that are not assigned to any previous partition, and can reach the center node within $i \cdot 2 \cdot k$ hops. As we can see, by performing such partitioning process, the network will be divided into partitions, where the depth of each partition is $2 \cdot k$. Now, in each partition there must be a set of nodes that can reach all other nodes inside the partition within k-hops. By performing this partitioning step, we attempt to reduce the solution space for selecting the caching points in an efficient manner in order to simplify the process of identifying the caching nodes.

5.2 The caching point identification step

In this step, the process iterates to determine the caching points for each partition. Starting at partition number1, for each pair of nodes $v_i, v_j \in V$ belong to this partition, we add an edge between these two nodes to E, if they are k-hops away from each other. Only the nodes that belong to the same partition will be considered during the process of determining the k-hops connectivity. Once the k-hops connectivity for all partitioned are examined, the process proceed by constructing a set of the caching points for each partition $C = [C_1, C_2, \dots, C_i]$. In each partition($i$), based on the degree of the nodes, subset of the partition nodes will be added to C_i. In each iteration, the node with the highest degree will be add to C_i. This node and all of its one-hop neighbors (inside the partition) will be flagged to ensure that they are considered in consecutive iterations. This process stops when all nodes inside each partition are flagged. Now, the set of the caching points consists of the collection of caching points in all partitions.

5.3 The routing trees construction and tour building steps

Once the caching points set are identified, each node not included in this set will be assigned to its nearest caching point. Then, for each caching point and the nodes which are assigned to this caching point, a Minimum Spanning Tree (MST) is created to establish multi-hop forwarding trees. The tour of the mobile element consists of the caching points and the sink. This is established using Christofides algorithm [18].

6. EXPERIMENTAL EVALUATION

To evaluate the presented algorithm's performance, we conducted an extensive set of experiments using the J-sim simulator [19]. the area of the network is 250,000m^2, and The radio parameters are set according to the MICAz data sheet [20], 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 10 Joules. The packet has a fixed size of 100 bytes. Each experiment is taken over an average of 10 different realizations of random topologies. We are particularly interested in investigating the following metrics:

- The lifetime of the network
- Number of caching points.

The parameters we consider in our experiments look at varying the number of nodes. We consider the following deployment scenarios:

- Uniform density deployment: in this scenario, we assume that the nodes are uniformly deployed in a square area of 500× 500m^2.
- Variable density deployment: in this scenario, we divide the network into a 10×10 grid of squares, where each square is 50×50m^2. We randomly choose 30 of the squares, and in each one of those squares we fix the node density to be x times the density in the remaining squares. x is a density parameter, which in most experiments (unless mentioned otherwise) is set to $x = 5$.

To benchmark the presented algorithm performance, we compared it against the Spanning Tree Covering Algorithm, which is proposed by Ma et al.[16], we will refer to this algorithm as the T-Covering algorithm. This algorithm starts by selecting the sink as the initial point in the mobile element tour. Then, in each iteration and based on a cost function, the polling point with the lowest cost will be added to the tour. Ma et al.[18] defines the polling point as a point in the network, where the mobile element can communicate with one or more sensor nodes via a single hop transmission. For a given polling point, this cost function returns the shortest distance between this polling point and a node belong to the tour, divided by the number of new nodes will be covered if this polling point is selected. This process iterate until all nodes are covered by the tour. To ensure the fairness of the comparison, we restrict the

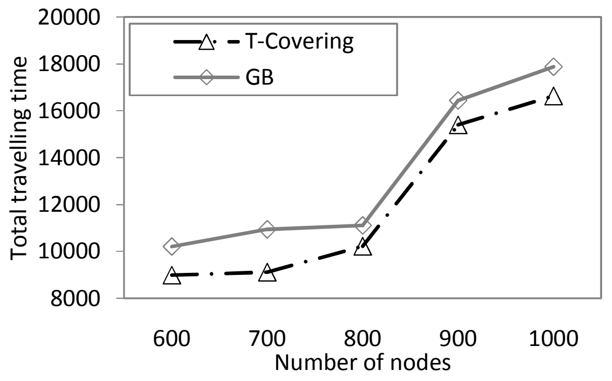

Figure 1: Total travelling time against the number ofnodes, for the uniform density deployment scenario

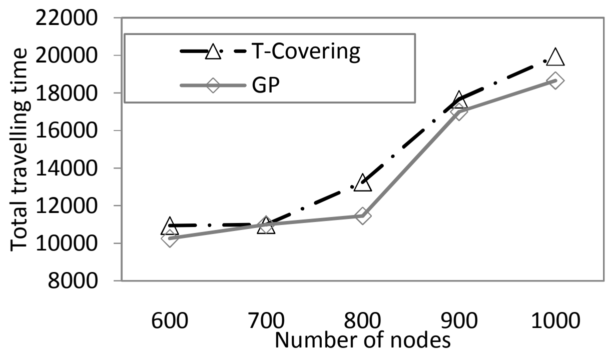

Figure 2: Total travelling time against the number of nodes, for the variable density deployment scenario

polling points in the T-covering algorithm to be the actual sensor nodes, and in the T-covering algorithm, we add a direct edge between any two nodes, if they are k-hops from each other.

First, we evaluate the impact of the number of nodes on the number of tours each algorithm obtains. Figure 1 and Figure 2 show the results for deployment scenarios. From the figures, we can see that in the uniform deployment scenario, the T-Covering algorithm outperforms the GP algorithm, where in the variable deployment scenario, the GP algorithm outperform the T-Covering algorithm. To understand this behavior, let us clarify the relationship between the deployment scenario and the expected performance for each algorithm. In the T-Covering algorithm, the employed cost function selects the node with the lowest cost to be included in the tour. The cost value for each node is obtained by dividing the distance between this node and the tour, by the number of nodes which will be covered if this node is included in the tour. Such cost function results in degrading

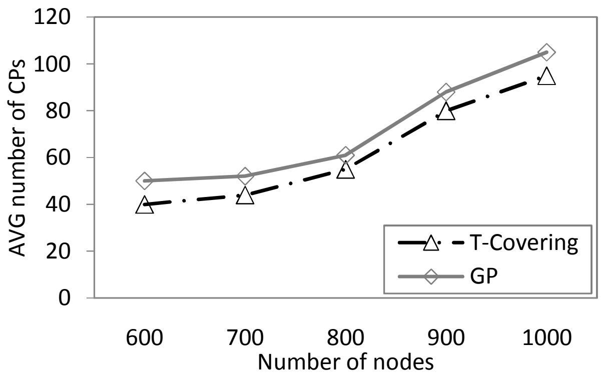

Figure 3: Average number of caching points against the number of nodes, for the uniform density deployment scenario

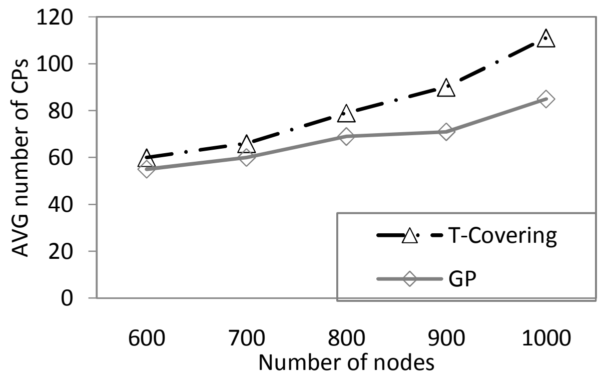

Figure 4: Average number of caching points against the number of nodes, for the variable density deployment scenario

the performance of the T-Covering algorithm in the variable deployment scenario compared to the uniform deployment scenario. This occurs because in the uniform deployment scenario, we expect that the T-Covering algorithm will select caching points uniformly located across the entire network. Furthermore, in the variable deployment scenario, this cost function is expected to result in selecting caching points mainly in the dense areas in the network and this is expected to increase the length of the obtained tour. In the GP algorithm, the process of partitioning the network results in disconnecting the process of selecting the caching points from the deployment distribution. These are the main factors which explain the performance of our algorithm.

We proceed to investigate the impact of the number of nodes on the number of caching points that each algorithm obtains. Figures 3 and 4 show the results for both the uniform density and the variable density deployment scenarios; respectively. From the figures, we can see that in the uniformly deployment scenario, the T-Covering algorithm obtains slightly lower number of caching points compared to the GP algorithm. On the other hand, in variable deployment scenario, the GP algorithm obtains significantly lower number of caching points. This is also due to the factors mentioned above, since in the variable deployment scenario, we expect that the T-covering algorithm obtains longer tour compared to the GP algorithm.

7. CONCLUSIONS

In this work, we address the problem of designing the mobile element tour, such that the length of this tour is below a pre-determined value, and each node is at most k-hops away from the tour. Accordingly, we propose a heuristic-based solution which is based on dividing the network into partitions, with pre-determined depth. Then, a subset of the nodes in each partition which will act as caching points for data must be identified such that the length of the tour that consists of these caching points and the sink node is below *L*.

An interesting open problem is considering some application scenarios where the latency requirements of data gathering may vary from one network's location to another. For example, some areas in the network need to be visited more frequently than others. In this case the tour length constraints would be different for different areas.

REFERENCES

[1] 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.

[2] G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," IEEE Trans. Mob. Comput., vol. 7, no. 12, pp. 1430–1443, Dec. 2008.

[3] 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.

[4] 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.

[5] 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.

[6] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," IEEE Trans. Mob. Comput., vol. 6, no. 4, pp. 395–410, 2007.

[7] E. Güney, I. K. Altmel, N. Aras, and C. Ersoy, "Efficient integer programming formulations for optimum sink location and routing in wireless sensor networks," in Proceedings of the 23rd International Symposium on Computer and Information Sciences, 2008, pp. 1–6.

[8] W. Liang, J. Luo, and X. Xu, "Prolonging Network Lifetime via a Controlled Mobile Sink in Wireless Sensor Networks," in Proceedings of Global Telecommunications Conference (GLOBECOM 2010), 2010, pp. 1–6.

[9] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: a distributed mobile sensor computing system," in Proceedings of the 4th international conference on Embedded networked sensor systems, 2006, pp. 1–14.

[10] M. Zhao and Y. Yang, "Optimization-Based Distributed Algorithms for Mobile Data Gathering in Wireless Sensor Networks," IEEE Trans. Mob. Comput., vol. 11, no. 10, pp. 1464–1477, 2012.

[11] M. Zhao and Y. Yang, "efficient data gathering with mobile collectors and space-division multiple access technique in wireless sensor networks," IEEE Trans. Comput., vol. 60, no. 3, pp. 400–417, 2011.

[12] P. Toth and D. Vigo, "The Vehicle Routing Problem," Soc. Ind. Appl. Math., 2001.

[13] M. M. Solomon, "ALGORITHMS FOR AND SCHEDULING PROBLEMS THE VEHICLE ROUTING WITH TIME WINDOW CONSTRAINTSS," Oper. Res., vol. 35, no. 2, pp. 254–265, 2010.

[14] B. Golden, L. Levy, and R. Vohra, "The orienteering problem," Nav. Res. Logist., vol. 34, no. 3, pp. 307–318, 2006.

[15] M. Ma, Y. Yang, and M. Zhao, “Tour Planning for Mobile Data-Gathering Mechanisms in Wireless Sensor Networks,” IEEE Trans. Veh. Technol., 2013.
[16] 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.
[17] 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.
[18] N. Christofides, “Worst-case analysis of a new heuristic for the traveling salesman problem,” 1976.
[19] 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 Wirel. Commun. Mag., vol. 13, no. 4, pp. 104–119, 2006.
[20] 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.

AUTHORS

Bassam A. Y. Alqaralleh received his BSc degree in Computer Science in 1992, graduate Diploma in Computer Information Systemsin 2002 and Masters Degree in Computer Science in 2004. After that, he received Ph. D. in Computer Science from University of Sydney in 2010. His research interests are in the areas of Distributed Systems, Load-Balancing, Networking and Security Systems. He has published a number of conference and journal papers. Currently, he is an assistant professor at the Computer Science Department / Faculty of Information Technology - Al-Hussein Bin Talal University. Since 2013, he is the dean for the Faculty of Information Technology.

Khaled Almiani is currently an assistant professor at the school of computer Science, Al-Hussein Bin Talal University, Jordan. His main research interests include designing efficient algorithms to improve the performance of WSNs and game-theoretical modellingfor WSNs. Khaled received his PhD degree in Information technology from the University of Sydney in 2010.