On Delay-minimized Data Harvesting with Mobile Elements in Wireless Sensor Networks[1]

R. Moazzez-Estanjini[a], I. Ch. Paschalidis[b]
[a]Center for Information and Systems Engineering, Boston University, MA, USA, Email: reza2007@bu.edu
[b]Department of Electrical and Computer Engineering, and Division of Systems Engineering, Boston University, MA, USA, Corresponding author, Email: yannisp@bu.edu, URL: http://ionia.bu.edu

Abstract

We consider the problem of routing and scheduling a set of mobile elements that act as mechanical carriers of data, harvesting them from sensor nodes and delivering them to a sink. The objective is to minimize the data delivery latency. Most of the existing work has focused on designing delay minimizing routes for the mobile nodes by leveraging variants of the *Traveling Salesman Problem (TSP)*. We show that TSP-based routes can lead to delay that is arbitrarily worse than the optimal. The main insight is that as data generation rates of sensors may vary, some sensors need to be visited more frequently than others. To that end, we consider a network with a single sink and develop a Path Splitter Algorithm that "splits" a TSP-based route into several loops intersecting at the sink. Numerical results show that our algorithm can improve average delay by more than 40% in some instances while requiring a modest computational effort to modify the TSP-based route. The work is useful in prolonging sensor network lifetime and in relaying data in partitioned networks.

Keywords: Wireless Sensor Networks, Mobile Elements, Data Harvesting, Traveling Salesman Problem, Routing.

1. Introduction

The traditional approach for data relaying in Wireless Sensor NETworks (WSNETs) involves multi-hop communications from data sources to destinations (sinks). Such an approach faces obvious challenges in sparse —disconnected— networks but can also substantially reduce a connected network's lifetime, especially when relaying over many hops is needed. Further, as the number

[1]Research partially supported by the NSF under grant EFRI-0735974, by the DOE under grant DE-FG52-06NA27490, by the ARO under grant W911NF-11-1-0227, and by the ODDR&E MURI10 program under grant N00014-10-1-0952.
A brief conference paper containing some preliminary results appeared in [1].

of sinks in WSNETs is relatively small, the nodes close to the sinks may run out of energy earlier since all traffic is funneled through them. As an alternative, it is known that the use of mobile elements to transport data from node to node can significantly reduce the energy consumption at each sensor, elongating the functional lifetime of the network, in exchange for increased data delivery latency [2, 3, 4, 5, 6].

These mobile elements with onboard memory act as relays, reaching to the nodes, downloading the data, mechanically carrying them to their destination, and uploading them when in close proximity to the sink. The mobile elements can easily be recharged periodically at a base station as opposed to sensor nodes which do not possess energy replenishment capabilities, hence, the battery lifetime of individual sensors can be increased by shifting the energy consumption burden for communication to the mobile elements.

There are some recent applications appreciating the effectiveness of using such mobile elements, e.g., the ZebraNet project [7], the Manatee project [8], the DakNet project [9], unmanned vehicles in underwater environmental monitoring [10], and a UAV (Unmanned Aerial Vehicle) in structural health monitoring [11]. In some applications, such as urban traffic monitoring using stationary sensors, vehicles that can act as mobile elements already exist in the environment (e.g., buses, police cars, municipal vehicles, etc.). In other applications such as the surveillance of large and inhospitable areas with sparse (mostly disconnected) sensor networks, mobile elements like UAVs may already be used for monitoring with high bandwidth sensor modalities (e.g., cameras) and can also be tasked with collecting more complete monitoring data from the stationary sensors. We do not elaborate further on the applications of these mobile elements; the reader is referred to [12, 13, 2, 14, 15] for more settings where their use is appropriate and effective. In the literature, data-carrying mobile elements are sometimes referred to with different terms (e.g., *mobile routers* [16], *mobile elements* [17, 18], *data MULE*s [2], *Message Ferry* [12, 13], *Actor* [19], *Actuator* [20], and *mobile sinks* [21, 22, 23, 24, 25]). In this paper, we adopt the abbreviation *ME* to refer to these *Mobile Elements.*

Hereafter, we will assume that data in the WSNET are exclusively transported by the MEs. In practice, one would typically see disconnected clusters of sensors communicating via the MEs. Representing each such cluster by a single node yields a WSNET using only MEs for data transport. In such a setting, the data transfer efficiency depends heavily on the path followed by the MEs. The problem, which we term the *Data Harvester Problem (DHP)*, can be stated as follows. Consider a set of N static nodes located at fixed positions, each of which is generating data with some known/estimated rate r_i $(i = 1, \ldots, N)$. The data generated at the nodes are destined to a static sink. A single ME or a group of MEs are responsible for visiting the nodes periodically, collecting any buffered data and carrying them to the sink. For simplicity, we assume that the speed of the MEs is a constant v. Let d_i be the average delay time for delivery of the data generated at node n_i. The goal is to find a schedule for the MEs to

visit the nodes and the sink such that the average delay

$$d = \frac{\sum_{i=1}^{N} \lambda_i d_i}{\sum_{i=1}^{N} \lambda_i}, \tag{1}$$

is minimized, where the choice of λ_i's represents the importance of the delay for data generated at node n_i. A reasonable choice would be to set $\lambda_i = r_i, \quad \forall i = 1, \ldots, N$, and this is what we do in our experiments reported in Section 6. The average delay d_i consists of the *waiting time* of the data at the node before they are transmitted to an ME, and the *carrying time* of the data by the ME before they are delivered to the sink.

We note that the problem description above is general enough to encompass a wide variety of applications. For instance, consider environmental surveillance and monitoring where a set of sites are required to be visited on a regular basis in order to detect any potential malfunction/threats as soon as they occur (e.g., gas leakage, fire, etc.). Different sites might have a different likelihood for occurrence of a potential malfunction/threat, hence, the λ_i's should be chosen accordingly. Another example would be the problem of scheduling the visits of an ME to sensor nodes so that data loss due to sensor node buffer overflow is minimized. In such a scenario, choosing λ_i's accordingly may mean that $\lambda_i = r_i/b_i$ where b_i is the buffer capacity of node n_i. We assume the data transmission time between the nodes and the ME to be negligible compared to the ME's travel times. Even though there are works that accommodate non-negligible transmission times [26, 27], our assumption is justified in many settings, especially when the area covered by the WSNET is large and the ME travel times dominate [17].

We also note that the problem is fundamentally different from any variant of the *Pickup and Delivery Problem* (PDP) [28, 29, 30]. In PDP, the quantities of goods (here, the data) are considered to be discrete; that is, some fixed (certain in *static* cases and *uncertain* in dynamic cases) quantities of goods are waiting in their origin locations to be picked up and delivered to a depot. Once the tour is completed and the vehicle returns to the depot where it started its trip, the problem is over. In DHP, on the other hand, the goods (the data) at each origin location (nodes) are constantly being generated according to some rates and the objective is to minimize the *average* data delivery delay. The data are buffered in each node until the location is visited by the ME, hence the amount of data to be carried depends on the time that the node is visited by the ME.

In this paper, we prove the sub-optimality of the DHP solutions which are based on the well-known *Traveling Salesman Problem (TSP)*. [2] In fact, we formally establish that *any Hamiltonian cycle* [3] can result in data delivery delay that is *arbitrarily* worse than that of the optimal solution. In the sequel we will

[2] In such solutions, the ME's route is the shortest route to visit each of the nodes exactly once.

[3] A Hamiltonian cycle is a cycle in an undirected graph which visits each node exactly once and returns to the starting node.

call *Hamiltonian solution* the solution that follows a Hamiltonian cycle. Instead, we propose a *Path Splitter Algorithm (PSA)* for the case of a single ME and further extend it to cases where multiple MEs are at our disposal. The algorithm improves upon an available TSP-based route by "splitting" the route into several loops, hence, transforming a Hamiltonian solution to a non-Hamiltonian solution. The proposed solution, which we term *periodic visit schedule* (or *schedule* for short), is an ordered list of node indices each ME should visit in each "period." This ordered list of nodes is in fact a *walk* [4] [31] in a graph whose vertices represent sensor nodes and edges represent physical routes among them. The schedule repeats itself in time, thus defining a periodic structure. The schedule is then loaded to the ME's memory, and the ME visits the nodes and the sink according to the schedule. In general, a node or the sink can be visited more than once in each period.

To provide an example, consider the single ME case, and let F denote the number of nodes (including the sink) to be visited in a period. $F \geq (N+1)$ since each node and the sink must be visited *at least* once in each period. F is upper-bounded by the size of the ME's memory. An example of a schedule would be $\{0, n_1, n_3, n_1, n_2\}$, where $F = 5$, the network includes three nodes n_1, n_2, and n_3 and the sink identified by 0. Then, the ME would visit the nodes in the following order

$$\{0 \to n_1 \to n_3 \to n_1 \to n_2 \to 0 \to n_1 \to n_3 \to n_1 \to n_2 \to \cdots\}.$$

It should be mentioned that we do not claim that solutions in the above form (periodic schedules) are necessarily optimal. The motivation of introducing them comes from the fact that they do not require extensive computational capabilities at the MEs (i.e., there is no need for the ME to continuously compute the next node to be visited). Moreover, they can be constrained to accommodate restrictions on the MEs' memory size (as we mentioned, F is upper-bounded by the size of the ME's memory).

In what follows, Section 2 presents a review of the related work. Section 3 presents some observations about the problem, describes the shortest-cycle solutions, and establishes their sub-optimality. In Section 4, we propose the PSA for the single ME case. Section 5 extends the PSA to the case of multiple MEs. In Section 6, we evaluate effectiveness of our PSA using simulation. Conclusions are drawn in Section 7.

2. Related work

Most of the existing work in the literature focuses on TSP-based solutions in which the ME visits each node exactly once in its route (see, e.g.,

[4] A walk in a graph $\mathscr{G}$ is a sequence of nodes $(n_1, n_2, \ldots, n_l)$ such that each pair of nodes $(n_1, n_2), (n_2, n_3), \ldots, (n_{n-1}, n_l)$ is an edge of $\mathscr{G}$.

[32, 23, 6, 13, 33, 34]). [14] proposed a heuristic algorithm to find the delay-optimal Hamiltonian cycle rather than the distance-optimal Hamiltonian cycle. Regardless, the proposed solution is still in the form of a Hamiltonian cycle.

As the amount and frequency of data generation in sensor nodes may vary based on the event occurrence frequency (which is generally a function of the sensor location), it is intuitive that some sensors need to be visited more frequently than others. For instance, in [35] sensor networks are used to sense air pollution levels in large urban areas. Since the variation of pollution levels is expected to be higher in industrial areas rather than in residential areas, sensors in industrial areas generate data at higher rates and need to be visited more frequently than others in order to minimize the average delay. The recent work in [36] is an effort to depart from the traditional Hamiltonian cycle approach; the authors proposed two-path planning algorithms based on stochastic modeling and machine learning. Although the non-Hamiltonian solution provided there is an improvement over the traditional Hamiltonian solution approach, their solution is impractical even for very small networks. (They mentioned that finding a route takes about 1 hour for 3 nodes, 6 hours for 4 nodes and about 20 hours for 5 nodes.)

It should also be mentioned that, although [35] and [17] considered non-Hamiltonian solutions, the objective considered there is to avoid buffer-overflow rather than to minimize average delay. We also note that our model is different from work in the context of *polling systems* [37]. Polling systems consist of a single server with N queues where the server visits the queues in a fixed order specified by a *polling table* in which each queue occurs at least once, and the objective considered is to find a polling table which minimizes the mean waiting cost. It is typically assumed that the time needed to switch the server from queue i to any other queue j is independent of j whereas in our case the time an ME needs to reach a node depends on the location of that node; that is, in our setting the topology of the network plays an important role.

3. The sub-optimality of the Hamiltonian solutions

In this section, we prove that a Hamiltonian solution may be arbitrarily worse than the optimal. Let $c_{i,j}$ be the distance from node n_i to node n_j (we represent the sink as node 0). A periodic schedule can be represented by a row vector containing the indices of nodes visited by the ME in each period. Let $(s_1, s_2, \ldots, s_M)$ be some schedule, where s_i denotes the ith node (or sink) visited in the schedule ($M \geq N+1$ since each node and the sink must be visited *at least* once in each period). Define $|T| = c_{s_M,s_1} + \sum_{i=1}^{M-1} c_{s_i,s_{i+1}}$ to be the *length of the period*. Assume that node n_i is visited only once in each period, i.e., $M = N+1$. As mentioned earlier, the average delay d_i for data generated at node n_i consists of the *waiting time* at node n_i before transmitted to the ME, and the *carrying time* by the ME before delivered to the sink. With the assumption of constant bit rate, the waiting time equals to $|T|/2v$, where v denotes the ME's speed. The carrying time equals l_i/v, where l_i is the distance the ME travels according

to the schedule from node n_i to the sink. Then, for d_i in Eq. (1) we have

$$d_i = \frac{|T|}{2v} + \frac{l_i}{v}. \tag{2}$$

However, if $M > (N+1)$, i.e., when a node n_i is visited more than once in the schedule, calculating the average delay is not as straightforward. We now derive a closed-form formula to calculate d_i for such situations.

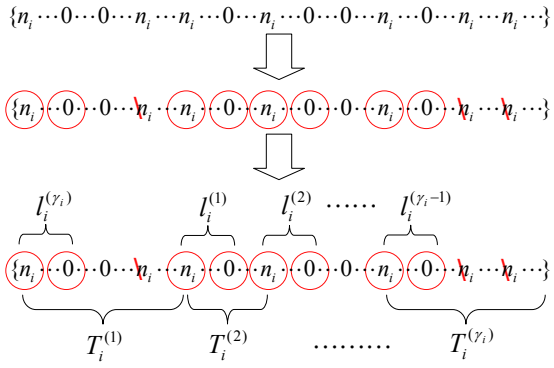

Figure 1: Calculating d_i. Only the n_i's and the 0's are shown, where an n_i (resp., 0) represents a visit by the ME to node n_i (resp., sink).

Since the schedule is periodic and repeats itself in time, any shift in the schedule does not affect the solution. To calculate d_i, let us shift the schedule such that the first node to be visited is node n_i as illustrated in Fig. 1. Obviously, since n_i is visited more than once in the schedule, this shift is not unique. Also, note that multiple visits to n_i between any two consecutive visits to the sink are useless; we can simply buffer the data and collect them at the last visit. Therefore, we assume that the schedule only includes a single visit to n_i between any two consecutive visits to the sink. Circle now in Fig. 1 the n_i's and the very next 0 (sink) after each circled n_i. Let γ_i denote the number of circled n_i's. Let $l_i^{(r)}$ denote the distance the ME travels according to the schedule from the $(r+1)$st circled n_i to the $(r+1)$st circled 0, and $|T_i^{(r)}|$ denote the distance the ME travels according to the schedule from the rth circled n_i to the $(r+1)$st circled n_i. ($l_i^{(\gamma_i)}$ denotes the distance the ME travels from the first circled n_i to the first circled 0, and $|T_i^{(\gamma_i)}|$ denotes the distance the ME travels from the γ_ith circled n_i to the first circled n_i). We have

$$\begin{aligned} d_i &= \sum_{r=1}^{\gamma_i} \frac{|T_i^{(r)}|}{\sum_{q=1}^{\gamma_i} |T_i^{(q)}|} \left(\frac{|T_i^{(r)}|}{2v} + \frac{l_i^{(r)}}{v} \right) \\ &= \frac{1}{2v|T|} \left(\sum_{r=1}^{\gamma_i} |T_i^{(r)}|^2 + 2 \sum_{r=1}^{\gamma_i} |T_i^{(r)}| l_i^{(r)} \right), \end{aligned} \tag{3}$$

where $|T| = \sum_{q=1}^{\gamma_i} |T_i^{(q)}|$ is the length of the period and it is identical for all i. Eq. (3) is the weighted average of average delays (Eq. (2)) of γ_i pieces of data generated at node n_i in each period, where the weight $|T_i^{(r)}|/(\sum_{q=1}^{\gamma_i} |T_i^{(q)}|)$ is the fraction of time delay $|T_i^{(r)}|/2v + l_i^{(r)}/v$ occurs.

Substituting d_i's from Eq. (3) in Eq. (1), we have the following closed-form formula for the total average delay:

$$d = \frac{1}{2v|T|\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i \Big(\sum_{r=1}^{\gamma_i} |T_i^{(r)}|^2 + 2 \sum_{r=1}^{\gamma_i} |T_i^{(r)}| l_i^{(r)} \Big). \tag{4}$$

Theorem 3.1 *In DHP, the ratio of delay of the delay-optimal Hamiltonian solution to the delay of the optimal solution can be arbitrarily large.*

Proof : Consider a simple ad hoc network with two static nodes n_1 and n_2 placed on two corners of a triangle with node 0 (the sink) placed on the other corner. Assume $c_{1,0} = c_{0,1} = x, c_{0,2} = c_{2,0} = \beta x, c_{1,2} = c_{2,1} = \beta x, \lambda_1 = \beta^2 r_2, \lambda_2 = r_2, v = 1$, where $\beta > 6$ is any arbitrarily large integer. Two Hamiltonian solutions exist:

$$H_1 : \{0 \to n_1 \to n_2 \to 0\}, \quad H_2 : \{0 \to n_2 \to n_1 \to 0\}.$$

Now consider the following set of non-Hamiltonian solutions:

$$NH(\gamma) : \{0 \to n_1 \to \ldots \to 0 \to n_1 \to 0 \to n_2 \to 0\},$$

where "$0 \to n_1 \to$" is consecutively repeated γ times in the schedule. Let T_{H^*} and T^* denote the delay-optimal Hamiltonian solution and the optimal solution, respectively. Using Eq. (1) and (2) and after some algebra we obtain

$$d^{T_{H^*}} = \min(d^{H_1}, d^{H_2}) = d^{H_2} = (\beta + \frac{3}{2} + \frac{\beta}{\beta^2 + 1})x > \beta x.$$

Using Eq. (1), (2) and (3) we have

$$d^{NH(\beta^2)} = \frac{4\beta^2 + 2\beta}{\beta^2 + 1} x < \frac{6\beta^2}{\beta^2} x = 6x,$$

hence,

$$d^{T_{H^*}} > \frac{\beta}{6} d^{NH(\beta^2)} \geq \frac{\beta}{6} d^{T^*}.$$

□

4. Path splitter algorithm

As we showed earlier, the optimal solution to the problem may not be a Hamiltonian cycle. In fact, even if it was, finding such a Hamiltonian cycle solution is still difficult. Solutions based on the well-known Traveling Salesman

Problem (TSP) where the ME's route becomes the shortest route to visit all nodes exactly once in each period (which we term *distance-optimal Hamiltonian solution*) do not necessarily result in the smallest average delay among all Hamiltonian cycle solutions. The following theorem states this fact.

Theorem 4.1 *The distance-optimal Hamiltonian solutions do not necessarily result in the smallest average delay among all Hamiltonian cycle solutions.*

Proof : Consider an instance of the DHP: a simple ad hoc network with three static nodes n_1, n_2, and n_3 placed on vertices of a unit square as shown in Fig. 2 (node 0 represents the sink and is placed on the other vertex). Assume $v = 1$ and

$$\lambda_1 = \lambda_3 = \lambda > 0, \quad \lambda_2 = \beta\lambda.$$

A total of two shortest Hamiltonian solutions exist:

$$H_1 : \{0 \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow 0\},$$

$$H_2 : \{0 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \rightarrow 0\}.$$

Using Eq. (1) and (2) we have

$$d^{H_1} = d^{H_2} = 4.$$

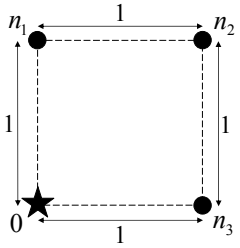

Figure 2: An instance of DHP; the star represents the sink.

Now, consider the following Hamiltonian cycle solution which is longer than H_1 and H_2:

$$H_3 : \{0 \rightarrow n_1 \rightarrow n_3 \rightarrow n_2 \rightarrow 0\}.$$

Eq. (1) and (2) yield

$$d^{H_3} = \frac{(2\sqrt{2}+1)\beta + (5\sqrt{2}+4)}{\beta + 2}.$$

We have

$$d^{H_3} < \min(d^{H_1}, d^{H_2}),$$

for large enough β (e.g., $\beta > 20$). □

We mention that even though the TSP solutions (the distance-optimal Hamiltonian solutions) are not necessarily the best Hamiltonian solutions in terms of average delay, they are relatively good Hamiltonian solutions to the DHP, as stated in the following theorem.

Theorem 4.2 *Let T^* be the minimum-delay solution within the class of all Hamiltonian solutions for the DHP. Let T be a TSP solution generated by an algorithm with approximation ratio α, and let T' be the reverse of T; that is, in T' the ME visits the nodes in the reverse order as in T. Then,* $\min(d^T, d^{T'}) \leq 2\alpha d^{T^*}$*, where d^T, $d^{T'}$ and d^{T^*} are the average delays if route T, T', or T^* is chosen, respectively.*

Proof : Note that for all i,

$$l_i + l'_i = |T|,$$

where l_i (resp., l'_i) is the distance from node n_i to the sink in route T (resp., T'). Using Eq. (1) and (2), we have

$$d^T + d^{T'} = \frac{2|T|}{v}.$$

Thus either d^T or $d^{T'}$ is no more than $\frac{|T|}{v}$. Hence,

$$\min(d^T, d^{T'}) \leq \frac{|T|}{v}. \tag{5}$$

From Eq. (1) and (2), for any TSP route T we have

$$d^T = \frac{1}{\sum_{i=1}^N \lambda_i}\left[\sum_{i=1}^N \lambda_i\left(\frac{|T|}{2v} + \frac{l_i}{v}\right)\right] = \frac{|T|}{2v} + \frac{\sum_{i=1}^N \lambda_i l_i}{v\sum_{i=1}^N \lambda_i} \geq \frac{|T|}{2v},$$

therefore,

$$\min(d^T, d^{T'}) \geq \frac{|T|}{2v}. \tag{6}$$

Now, let |TSP| be the length of the optimal TSP solution. From Eq. (6),

$$d^{T^*} \geq \frac{|\text{TSP}|}{2v}.$$

Thus, from Eq. (5) we have

$$\min(d^T, d^{T'}) \leq \frac{|T|}{v} \leq \alpha\frac{|\text{TSP}|}{v} \leq 2\alpha d^{T^*}.$$

□

There are efficient algorithms for the TSP (see [38] for an overview of exact and approximate algorithms), hence, given the good quality of the TSP solution among all Hamiltonian solutions, we start with the TSP solution. As we have

argued, it may be possible to further decrease the average delay by visiting some nodes more than once in each period. Next, we propose a *Path Splitter Algorithm (PSA)*. The PSA is an improvement step that can be applied to any Hamiltonian solution. It essentially splits the Hamiltonian solution into multiple *loops*, each of which starts from and ends at the sink and determines a schedule for visiting the loops (the order in which the loops are traveled by the ME within a period). The loops are constructed so that they do not share any nodes other than the sink. Note that a loop may be visited more than once within a period. We first develop a method that, given the loops, determines the order in which they are visited. We then use this as a procedure to form the loops. Fig. 3(h) shows an example of the PSA solution with three loops. We note that it may be possible that in the optimal solution one node participates in more than one loops (i.e., loops intersect at some nodes), however, considering such schemes remarkably raises the complexity of the solution approach.

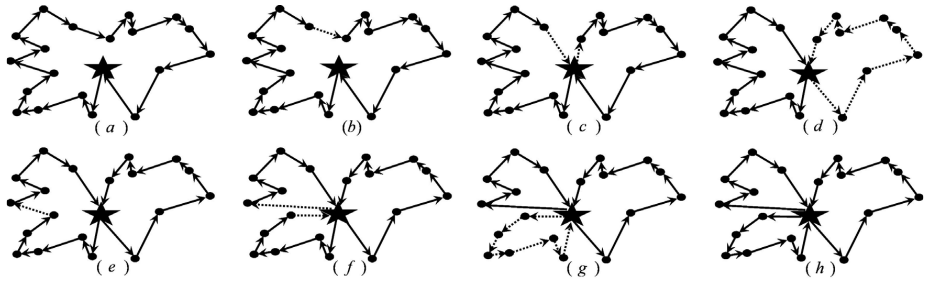

Figure 3: (a) The Hamiltonian cycle solution. (b) Selecting an edge. (c) Replacing the edge with the two edges. (d) Reversing the direction of one of the new loops. (e) Selecting an edge. (f) Replacing the edge with the two edges. (g) Reversing the direction of one of the new loops. (h) The PSA solution at the end of the 2nd iteration.

4.1. Scheduling visits to the loops

Assuming that the loops are given, let K denote the number of loops. Let $\{P_i\} = \{n_i(1), n_i(2), \ldots, n_i(N_i), 0\}$ represent loop i ($i = 1, \ldots, K$), where $n_i(j)$ denotes the index of the jth node visited in loop i starting at the sink and N_i is the total number of nodes in loop i not counting the sink. Given the way we construct loops, we have $\{P_i\} \cap \{P_j\} = \{0\}$ for all $i \neq j$, $\bigcup_{i=1}^{K} \{P_i\} = \{S_N, 0\}$ where S_N is the set of all nodes, and $\sum_{i=1}^{K} N_i = N$ where N is the total number of nodes (excluding the sink). For all i let $L_i = c_{0,n_i(1)} + c_{n_i(N_i),0} + \sum_{j=1}^{N_i-1} c_{n_i(j),n_i(j+1)}$ denote the length of loop i and $\Lambda_i = \sum_{j=1}^{N_i} \lambda_{n_i(j)}$. Let $\{P_{s_1}, P_{s_2}, \ldots, P_{s_M}\}$ be the PSA solution, where s_j is the index of the jth loop repeated in the ME's periodic schedule ($M \geq K$ is the maximum allowable number of loops to be repeated in the schedule; M is determined by limitations in the size of the ME's memory). Let ρ_i denote the number of repetitions of loop i in the schedule. Finally, recall from Sec. 3 that γ_i denotes the number of

times node n_i is visited by the ME in each period, $l_i^{(r)}$ is the distance the ME travels from the $(r+1)$st visit to n_i to the $(r+1)$st visit to 0 in each period, and $|T_i^{(r)}|$ is the distance the ME travels from the rth visit to n_i to the $(r+1)$st visit to n_i in each period.

Theorem 4.3 *For any $|T| > 0$, if for all $i = 1, \ldots, N$, $l_i^{(r)}$ are equal for all $r = 1, \ldots, \gamma_i$, then d in Eq. (4) is minimized if for each n_i the $|T_i^{(r)}|$ are set equal for $r = 1, \ldots, \gamma_i$.*

Proof : Observe that if d_i in Eq. (3) is minimized for all i, then the right hand side of Eq. (4) is also minimized. Since for all i, $\sum_{q=1}^{\gamma_i} |T_i^{(q)}| = |T|$, minimizing d_i in Eq. (3) amounts to minimizing

$$\sum_{q=1}^{\gamma_i} |T_i^{(q)}|^2$$

subject to

$$\sum_{q=1}^{\gamma_i} |T_i^{(q)}| = |T|,$$

for which $|T_i^{(q)}| = |T|/\gamma_i$, $q = 1, \ldots, \gamma_i$ yields the minimum. □

Now, observe that in the PSA solution, $|T| = \sum_{i=1}^{K} \rho_i L_i$, and that for any node $n_i(j)$ in the solution, we have $\gamma_{n_i(j)} = \rho_i$. Moreover,

$$\begin{aligned} l_{n_i(j)}^{(r)} &= l_{n_i(j)} \\ &= \begin{cases} c_{n_i(N_i),0} + \sum_{q=j}^{N_i-1} c_{n_i(q),n_i(q+1)}, & \text{if } j < N_i, \\ c_{n_i(N_i),0}, & \text{if } j = N_i, \end{cases} \end{aligned}$$

for $r = 1, \ldots, \gamma_{n_i(j)}$. Using Thm. 4.3, in order to minimize d in Eq. (4), we set

$$|T_{n_i(j)}^{(r)}| = \frac{|T|}{\gamma_{n_i(j)}} = \frac{\sum_{i=1}^{K} \rho_i L_i}{\rho_i}, \tag{7}$$

for $r = 1, \ldots, \gamma_{n_i(j)}$. Substituting in Eq. (4), we have

$$\begin{aligned} d &= \frac{1}{2v|T| \sum_{i=1}^{K} \Lambda_i} \left[\sum_{i=1}^{K} \sum_{j=1}^{N_i} \lambda_{n_i(j)} \left(\frac{|T|^2}{\rho_i} + 2|T| l_{n_i(j)} \right) \right] \\ &= \frac{1}{2v \sum_{i=1}^{K} \Lambda_i} \left[\sum_{i=1}^{K} \frac{\Lambda_i}{\rho_i} \left(\sum_{q=1}^{K} \rho_q L_q \right) + 2 \sum_{i=1}^{K} \sum_{j=1}^{N_i} \lambda_{n_i(j)} l_{n_i(j)} \right]. \end{aligned} \tag{8}$$

To minimize the above expression, we need to minimize

$$\sum_{i=1}^{K} \frac{\Lambda_i}{\rho_i} \left(\sum_{q=1}^{K} \rho_q L_q \right) \tag{9}$$

with respect to $\rho_1, \rho_2, \ldots, \rho_K$. Taking partial derivatives of the above with respect to the ρ_i's, setting them to 0, and solving the resulting equations yields that the expression in (9) takes its minimal value when

$$\rho_i = C\sqrt{\Lambda_i/L_i}, \quad \text{for some constant } C$$

hence, the fraction of the time that loop i should be repeated in each period is

$$\frac{\rho_i}{\sum_{j=1}^K \rho_j} = \frac{\sqrt{\Lambda_i/L_i}}{\sum_{j=1}^K \sqrt{\Lambda_j/L_j}}.$$

Eq. (7) states that in order to minimize the delay, the loops should be visited "uniformly" throughout the period; that is, the inter-visit times to each loop should be equalized. To do so, assuming the loops are given, the procedure in Fig. 4 is proposed to obtain a good schedule (not necessarily optimal).

---

1. For each $i = 1, \ldots, K$, select $\hat{\rho}_i$ such that

$$\begin{array}{ll} \min & \sum_{i=1}^K \left(\hat{\rho}_i - \frac{M\sqrt{\Lambda_i/L_i}}{\sum_{j=1}^K \sqrt{\Lambda_j/L_j}} \right)^2 \\ \text{s.t.} & \sum_{i=1}^K \hat{\rho}_i = M, \\ & \hat{\rho}_i \geq 1, \quad \forall i, \\ & \hat{\rho}_i \text{ integer}, \quad \forall i. \end{array} \tag{10}$$

2. For each $i = 1, \ldots, K$, define a set A_i containing $\hat{\rho}_i$ elements, where the jth element of A_i is

$$\frac{\sum_{q=1}^K \hat{\rho}_q L_q}{2\hat{\rho}_i} + \frac{(j-1)\sum_{q=1}^K \hat{\rho}_q L_q}{\hat{\rho}_i}.$$

3. Sort the points in $\bigcup_{i=1}^K A_i$ and populate a set A. The schedule is $\{P_{s_1}, P_{s_2}, \ldots, P_{s_M}\}$, where s_j, the index of the jth loop visited by the ME in the periodic schedule, is

$$s_j = \{i \mid \text{the } j\text{th smallest element of } A \text{ belongs to } A_i\}.$$

Figure 4: A procedure to generate a schedule given that the loops have already been identified.

Since $\sqrt{\Lambda_i/L_i}/(\sum_{j=1}^K \sqrt{\Lambda_j/L_j})$ is the fraction of the time that loop i should be repeated in each period, $M\sqrt{\Lambda_i/L_i}/(\sum_{j=1}^K \sqrt{\Lambda_j/L_j})$ is the number of repetitions of loop i. The latter quantity is not necessarily integer and Step (1) of the procedure in Fig. 4 "rounds" these values to integer, returning $\hat{\rho}_i$ for each i. Note that each loop should be repeated at least once in the schedule, hence, this rounding scheme should ensure that $\hat{\rho}_i \geq 1$ for each i. Problem (10) is a simple

quadratic integer programming problem which can be solved fast by standard solvers for moderate values of M. (This will be the case in our application as M is limited by the ME's memory and by the complexity of the schedule we would like to derive). Alternatively, and if the overhead of solving (10) becomes an issue, one can employ a heuristic procedure which first assigns 1 to each i (to guarantee $\hat{\rho}_i \geq 1$) and then allocates the remaining budget of $M - K$ to each i proportionally to $\sqrt{\Lambda_i/L_i}/(\sum_{j=1}^{K} \sqrt{\Lambda_j/L_j})$, rounding off when necessary to obtain integer $\hat{\rho}_i$'s.

Next, for each i, Step (2) of the procedure in Fig. 4 splits the period $|T| = \sum_{q=1}^{K} \hat{\rho}_q L_q$ into $\hat{\rho}_i$ equal pieces and stores in the set A_i the starting points of each such piece. We can think of these starting points as the points in the period when loop i "requests" to start ignoring that the other loops should be performed as well. Without loss of generality, and to avoid multiple loops requesting the same points, we have shifted the first starting point of loop i from 0 to $(\sum_{q=1}^{K} \hat{\rho}_q L_q)/(2\hat{\rho}_i)$.

Finally, in Step (3), we determine the order in which the ME performs the various loops based on their requested starting times obtained in Step (2).

4.2. Forming the loops

So far, we assumed that the loops are given. To determine the loops, PSA uses an iterative algorithm which at each iteration splits one of the current loops into two loops, thus, increasing the total number of loops by one.

The algorithm starts with the shortest Hamiltonian cycle solution (or a good approximation of that solution) and divides it into two loops by removing an edge $n_i \rightarrow n_j$ and replacing it with two edges $n_i \rightarrow 0 \rightarrow n_j$ (Fig. 3 (c)). For this first iteration, there are a total of $(N - 1)$ possible edges to be tried (edges starting from or ending at the sink cannot be removed). To decide which edge of the current loop will be removed, the algorithm tries all possible edges and applies the procedure in Fig. 4 to obtain the schedule for each trial. The algorithm then chooses the best edge greedily; that is, it selects the one whose removal results in the lowest average delay. The algorithm follows the same routine for the next iteration until no further improvement is achieved. Note that at each iteration k, there are a total number of k loops and $N - k$ possible edges to be tried, hence, the total number of edge removal trials is bounded by $\sum_{k=1}^{N}(N-k) = N(N-1)/2$. This upper bound on the number of trials specifies the low complexity of PSA.

One last note is that reversing the direction of a loop does not change the waiting time at the nodes (since the time it takes for the ME to travel along each loop would still remain unchanged), yet it may decrease the carrying times and yield a lower average delay. Therefore, at each iteration, after a loop is split into two smaller loops, the algorithm further verifies whether reversing the direction of either of the two newly formed loops results in the lower delay, and if it does it adopts the new direction (Fig. 3 (d)). Fig. 3 (a)-(d) illustrate the first iteration and Fig. 3 (e)-(h) illustrate the second iteration of PSA in a certain graph.

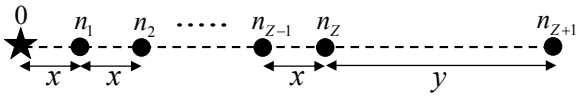

Figure 5: An instance of DHP; the star represents the sink.

5. Multiple MEs

When there are more than one ME in our disposal, the problem is much more difficult since the MEs could cooperate and relay data among themselves (see [14]). However, this kind of cooperation requires synchronization as the MEs need to meet in order to relay the data among themselves. In real scenarios, this might bring up some technical difficulties. For simplicity, we assume that each node is assigned to one ME and once an ME picks up data from the nodes assigned to it, it is responsible to relay the data to the sink. Therefore, the goal is to assign the nodes to Z MEs and schedule the MEs to visit the nodes and the sink periodically such that the delay in Eq. (1) is minimized.

Consider a set of Z tours (Z is the number of MEs) that collectively cover all the nodes where each tour covers at least one node (and the sink) and each node is visited by only one tour. Let us call such a set a *Z-Hamiltonian cycle* (for instance, Fig. 3(h) shows a 3-Hamiltonian cycle). Let us call Z-TSP the problem of finding a Z-Hamiltonian cycle such that the sum of lengths of the Z tours is minimized (i.e., the *shortest* Z-Hamiltonian cycle solution).

Theorem 5.1 *In DHP with Z MEs, for any $Z \geq 2$ the ratio of delay of the optimal Z-TSP solution to the delay of a Z-Hamiltonian solution which has the smallest average delay among all Z-Hamiltonian solutions can be arbitrarily large.*

Proof : Consider an instance of the DHP with $Z+1$ static nodes n_1 up to n_{Z+1} on a 2-dimensional plane as shown in Fig. 5 (node 0 represents the sink). Assume

$$c_{i,j} = c_{j,i} = |(i-j)x|, \quad \forall i,j = 0,1,\ldots,Z,$$
$$c_{i,Z+1} = c_{Z+1,i} = y + (Z-i)x, \quad \forall i = 0,1,\ldots,Z,$$
$$\lambda_i = 1, \quad \forall i = 1,\ldots,(Z-1), \qquad \lambda_{Z+1} = 1, \quad \lambda_Z = \beta,$$

where $\beta > 1$ is an arbitrarily large number. Let $v = 1$ and $y = \sqrt{\beta}x$.

We first show that the optimal Z-TSP solution to the problem consists of the following Z loops: $\{0 \to n_1 \to 0\}, \{0 \to n_2 \to 0\}, \ldots, \{0 \to n_{Z-1} \to 0\}$, and $\{0 \to n_{Z+1} \to n_Z \to 0\}$. Any Z-TSP solution to this problem must have a loop that includes n_{Z+1}, hence, there is at least one loop with length no less than $2(Zx + y)$. There are $Z-1$ loops that do not include n_{Z+1}. Each of these $Z-1$ loops would have at least one node from the set of remaining nodes $\{n_1, n_2, \ldots, n_{Z-1}, n_Z\}$. Since n_Z is the farthest node to the sink in this set, the

total length of the $Z-1$ loops is minimized if we do not include n_Z and that minimal value is equal to

$$2x + 4x + 6x + \ldots + 2(Z-1)x = x(Z-1)Z,$$

hence, the sum of length of Z loops in any Z-TSP solution to this problem is no less than $x(Z-1)Z + 2(Zx+y)$. The sum of the length of the loops $\{0 \to n_1 \to 0\}, \{0 \to n_2 \to 0\}, \ldots, \{0 \to n_{Z-1} \to 0\}$, and $\{0 \to n_{Z+1} \to n_Z \to 0\}$ is $x(Z-1)Z + 2(Zx+y)$, thus, this solution is optimal.

Now, using Eq. (2) we have

$$d_i^{\text{Z-TSP}} = 2ix, \quad \forall i = 1, \ldots, Z-1,$$

$$d_Z^{\text{Z-TSP}} = 2Zx + y, \quad d_{Z+1}^{\text{Z-TSP}} = 2(Zx+y),$$

and by using Eq. (1) we obtain

$$\begin{aligned} d^{\text{Z-TSP}} &= \frac{(\sum_{i=1}^{Z-1} 2ix) + \beta(2Zx+y) + (2Zx+2y)}{Z+\beta} \\ &= \frac{x\beta\sqrt{\beta} + 2Zx\beta + 2x\sqrt{\beta} + Zx + Z^2x}{\beta+Z}. \end{aligned}$$

Now, consider the following Z-Hamiltonian cycle solution (that is not Z-TSP optimal), denoted by H, consisting of the following Z loops: $\{0 \to n_1 \to 0\}, \{0 \to n_2 \to 0\}, \ldots, \{0 \to n_{Z-2} \to 0\}, \{0 \to n_Z \to 0\}$, and $\{0 \to n_{Z+1} \to n_{Z-1} \to 0\}$. Using Eq. (2) we have

$$d_i^{\text{H}} = 2ix, \quad \forall i = 1, \ldots, Z-2,$$

$$d_{Z-1}^{\text{H}} = y + 2Zx - x, \quad d_Z^{\text{H}} = 2Zx, \quad d_{Z+1}^{\text{H}} = 2(Zx+y).$$

Using Eq. (1) it follows

$$\begin{aligned} d^{\text{H}} &= \frac{(\sum_{i=1}^{Z-2} 2ix) + (y + 2Zx - x) + \beta(2Zx) + 2(Zx+y)}{Z+\beta} \\ &= \frac{2Zx\beta + 3x\sqrt{\beta} + 4Zx - x + x(Z-1)(Z-2)}{\beta+Z}. \end{aligned}$$

Obviously, for large enough β, the ratio $d^{\text{Z-TSP}}/d^{\text{H}}$ can be arbitrary large. □

The Theorem above states that Z-TSP solutions to the DHP can be far from optimal. Before suggesting our proposed method for finding an effective Z-Hamiltonian cycle solution to the DHP, we state the following theorem.

Theorem 5.2 *Assume a Z-Hamiltonian cycle with Z loops $\{T_1\}, \ldots, \{T_Z\}$. Let for any $j = 1, \ldots, Z$, $\{T_j\} = \{n_j(1), n_j(2), \ldots, n_j(N_j), 0\}$ represent loop j, where $n_j(l)$ denotes the index of the lth node visited in loop j starting at the sink and N_j is the total number of nodes in loop j not counting the sink. Let T_j' be the reverse of T_j for any j; that is, in T_j' the loop visits the nodes in the reverse*

order as in T_j. Let for all j, H_j be equal to T_j or T'_j depending on which of them yields a lower average delay for the nodes in the loop. Let, for all j, $|T_j|$ denote the length of loop T_j, and define $\Lambda_j = \sum_{i=1}^{N_j} \lambda_{n_j(i)}$. Then, the Z-Hamiltonian cycle solution, denoted by H, consisting of the Z loops $\{H_1\}, \ldots, \{H_Z\}$ yields average delay d^H such that

$$\frac{\sum_{i=1}^{Z} |T_i|\Lambda_i}{2v \sum_{i=1}^{N} \lambda_i} \leq d^H \leq \frac{\sum_{i=1}^{Z} |T_i|\Lambda_i}{v \sum_{i=1}^{N} \lambda_i}. \tag{11}$$

Proof : Let d_i^H be the delay of node n_i in the Z-Hamiltonian cycle solution H. From Eq. (1), (5) and (6) we have

$$\frac{|T_j|}{2v} \leq \frac{\sum_{i=1}^{N_j} \lambda_{n_j(i)} d^H_{n_j(i)}}{\sum_{i=1}^{N_j} \lambda_{n_j(i)}} \leq \frac{|T_j|}{v}.$$

Multiplying all sides of the above inequality by Λ_j, summing up over all j's, and then dividing each side by $\sum_{i=1}^{N} \lambda_i$ (to use Eq. (1)) completes the proof. □

Motivated by this result, we seek to obtain loops $\{T_1\}, \ldots, \{T_Z\}$ such that the right hand side of (11) is minimized. Since $v \sum_{i=1}^{N} \lambda_i$ is constant, we turn our attention to finding loops $\{T_1\}, \ldots, \{T_Z\}$ such that $\sum_{i=1}^{Z} |T_i|\Lambda_i$ is minimized. We next propose Z-PSA, the extension of PSA for the multiple ME case.

We start with a single shortest Hamiltonian loop including all nodes. We then split the single loop into Z loops, each of which starts from and ends at the sink. This is done by increasing the number of loops by 1 at each step and splitting a loop into two loops (by selecting and then replacing an edge with two edges as shown in Fig. 3) in a *greedy* way. In particular, at each step k we select the edge which when replaced with two corresponding edges results in the lowest value for $\sum |T_i|\Lambda_i$ where the sum is over the $k+1$ loops. Let $\tilde{d}_{(k+1)}$ denote this lowest value. The algorithm terminates after $Z-1$ steps and at the end of the $(Z-1)$st step we have a total of Z loops, i.e., a Z-Hamiltonian cycle.

Theorem 5.3 $\tilde{d}_1 \geq \tilde{d}_2 \geq \tilde{d}_3 \geq \ldots \geq \tilde{d}_Z$.

Proof : We show that $\tilde{d}_i \geq \tilde{d}_{(i+1)}$ for any $i = 1, \ldots, Z-1$. Let $\{T_1\}, \ldots, \{T_i\}$ denote i loops to be the solution obtained by Z-PSA before step i is performed. Using the same notation defined in Theorem 5.2, assume without loss of generality, that Z-PSA selects the edge $(n_j(l), n_j(l+1))$ from loop j to be replaced at step i $(1 \leq l \leq (N_j - 1))$. This results in updating loop j, $\{T_j\}$, with loop $\{T_j^{\text{new}}\} = \{n_j(1), n_j(2), \ldots, n_j(l), 0\}$, and creating a new loop $i+1$,

$\{T_{i+1}\} = \{n_j(l+1), n_j(l+2), \ldots, n_j(N_j), 0\}$. We have,

$$\begin{aligned}
\tilde{d}_i - \tilde{d}_{(i+1)} &= |T_j| \sum_{p=1}^{N_j} \lambda_{n_j(p)} - |T_j^{\text{new}}| \sum_{p=1}^{l} \lambda_{n_j(p)} - |T_{i+1}| \sum_{p=l+1}^{N_j} \lambda_{n_j(p)} \\
&= (c_{0,n_j(1)} + \sum_{p=1}^{N_j-1} c_{n_j(p),n_j(p+1)} + c_{n_j(N_j),0}) \sum_{p=1}^{N_j} \lambda_{n_j(p)} \\
&- (c_{0,n_j(1)} + \sum_{p=1}^{l-1} c_{n_j(p),n_j(p+1)} + c_{n_j(l),0}) \sum_{p=1}^{l} \lambda_{n_j(p)} \\
&- (c_{0,n_j(l+1)} + \sum_{p=l+1}^{N_j-1} c_{n_j(p),n_j(p+1)} + c_{n_j(N_j),0}) \sum_{p=l+1}^{N_j} \lambda_{n_j(p)}.
\end{aligned}$$

From the triangle inequality we have that

$$c_{n_j(l),0} \leq \sum_{p=l}^{N_j-1} c_{n_j(p),n_j(p+1)} + c_{n_j(N_j),0},$$

and

$$c_{0,n_j(l+1)} \leq c_{0,n_j(1)} + \sum_{p=1}^{l} c_{n_j(p),n_j(p+1)},$$

which yields

$$\tilde{d}_i - \tilde{d}_{(i+1)} \geq 0.$$

□

Note that at each step k, there are a total number of k loops and $N-k$ possible edges to be tried, hence the total number of edge removal trials is bounded by $\sum_{k=1}^{Z-1}(N-k) < ZN$. This upper bound on the number of trials specifies the low complexity of the proposed Z-PSA algorithm. We then start with such a solution. As we have argued, it may be possible to further decrease the average delay by visiting some nodes more than once within each loop in each period. The PSA for the single ME case is then applied to each of the Z loops to further decrease the delay by splitting each ME's loop into sub-loops independently.

6. Performance evaluation

In this section, we evaluate the effectiveness of our algorithms via simulation under various scenarios for both the single and the multiple ME case. The simulation was developed in the MATLAB environment and run on a computer with an 1.40GHz CPU.

We start with the single ME case. A random graph generator forms the random sensor network topology and determines the sensor data generation

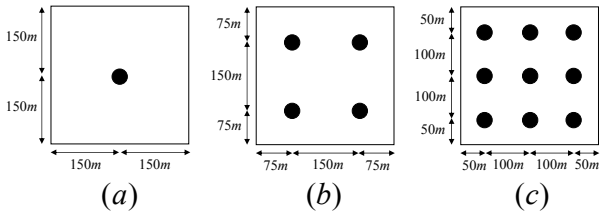

Figure 6: (a) Location of the cluster center in topology A. (b) Locations of the cluster centers in topology B. (c) Locations of the cluster centers in topology C.

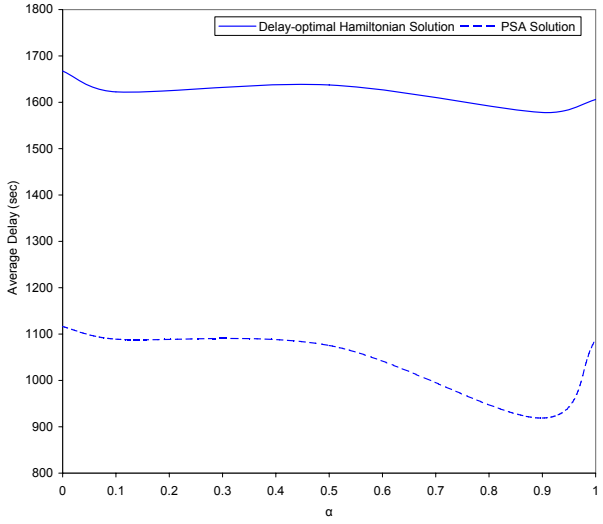

Figure 7: Comparison of the solutions for the scenario with topology A and the sink located at the center of the area.

rates according to a given scenario. Given the geographical topology of the nodes, as well as their data generation rates, a schedule for the ME to visit the nodes is generated by the PSA. We consider 180 nodes distributed on a $300m \times 300m$ area. Four topologies, A, B, C, and U, are constructed in our simulations. Topology U corresponds to uniformly distributed nodes over the coverage area. For topologies A, B, and C, we assume that the nodes are concentrated at certain locations called *cluster centers*. The cluster centers have the highest density of nodes, which drops radially outward. Topologies A, B, and C have one, four, and nine cluster centers, respectively. Fig. 6 shows the locations of the cluster centers for each topology. The lower left corner and the upper right corner of the area have coordinates $(x, y) = (0, 0)$ and $(x, y) = (300, 300)$, respectively. To generate topologies A, we generate x and y coordinates of the nodes from a Normal distribution with mean equal to

Table 1: Results for the single ME case - the sink located at the center of the area.

Scenario	Delay-opt. Hamilt. Delay (sec)	PSA Delay (sec)	Extra Comp. Time (sec)	Number of Loops	Improvement (%)
A_0	1667.4	1116.3	11.09	6.3	33%
$A_{0.1}$	1622.4	1089.1	10.21	6.1	33%
$A_{0.5}$	1637.5	1075.5	14.41	8.6	34%
$A_{0.9}$	1578.1	918.9	19.22	11.7	42%
A_1	1606	1088	9.77	5.7	32%
B_0	2033.8	1476.6	6.96	4	27%
$B_{0.1}$	2082.1	1521.6	7.38	4.3	27%
$B_{0.5}$	2037.3	1486.2	9.37	5.4	27%
$B_{0.9}$	1889.3	1360.9	9.43	5.6	28%
B_1	2017.9	1459.5	6.90	4	28%
C_0	2256.7	1662	8.57	4.7	26%
$C_{0.1}$	2272.2	1670.9	9.71	5.7	26%
$C_{0.5}$	2245.3	1691.7	11.83	7	25%
$C_{0.9}$	2216.6	1618.3	12.42	7.5	27%
C_1	2250.7	1681.3	9.01	5.2	25%
U_0	3182.4	2202.6	8.54	5	31%
$U_{0.1}$	3184.5	2236.5	8.60	5	30%
$U_{0.5}$	3150.9	2206.6	11.52	6.8	30%
$U_{0.9}$	3053.3	2025.1	14.55	8.9	34%
U_1	3109.5	2187	8.60	5	30%

150m and standard deviation equal to 35m. To generate topologies B, x and y coordinates of each of the 45 nodes belonging within each cluster are generated from a Normal distribution with mean equal to the corresponding coordinate of their cluster center and standard deviation equal to 20m. To generate topologies C, x and y coordinates of each of the 20 nodes belonging within each cluster are generated from a Normal distribution with mean equal to the corresponding coordinate of their cluster center and standard deviation equal to 15m.

We consider two locations for the sink, one in the center of the area, and the other in the lower left corner. For the node data generation rates, we assume that an α fraction of the nodes generate data at a rate of $1Kbps$ and the rest generate data at a higher rate of $100Kbps$. We let the value of α vary in order to generate various scenarios. Note that $\alpha = 1$ corresponds to a scenario where all nodes generate data at the same rate of $1Kbps$. As we mentioned in the Introduction, throughout this section we set $\lambda_i = r_i, \quad \forall i = 1, \ldots, N$. Let also set $v = 1$ (m/s). Fig. 7 illustrates the results of the simulation for the scenario where nodes are distributed in the network according to topology A and the sink is located at the center of the area. The results shown are the average of 100 independent runs. Graphs were generated for all other scenarios and exhibited the same qualitative behavior; we omit them in the interest of space.

In addition, Tables 1 and 2 present some details regarding the solutions. The first column of the tables identifies the scenarios. We use the value of

Table 2: Results for the single ME case - the sink located at the corner of the area.

Scenario	Delay-opt. Hamilt. Delay (sec)	PSA Delay (sec)	Extra Comp. Time (sec)	Number of Loops	Improvement (%)
A_0	1844.7	1701.7	3.67	2	8%
$A_{0.1}$	1864.9	1683	3.88	2	10%
$A_{0.5}$	1884	1697	4.34	2.4	10%
$A_{0.9}$	1812.7	1617.9	4.69	2.6	11%
A_1	1881.4	1700.7	3.66	2	10%
B_0	2057.5	1870.7	3.95	2.1	9%
$B_{0.1}$	2058.6	1881.7	4.02	2.2	9%
$B_{0.5}$	2079.1	1878.5	4.53	2.5	10%
$B_{0.9}$	2080.3	1859.7	5.43	3.1	11%
B_1	2104.1	1895.6	3.74	2.1	10%
C_0	2312.5	2038.6	4.21	2.3	12%
$C_{0.1}$	2330.9	2044.3	3.79	2.1	12%
$C_{0.5}$	2312.5	2015.9	4.48	2.5	13%
$C_{0.9}$	2216.9	1974.3	5.42	3	11%
C_1	2316.8	2002.4	3.61	2	14%
U_0	3210.2	2593.8	5.09	2.8	19%
$U_{0.1}$	3150	2550.7	4.87	2.7	19%
$U_{0.5}$	3179.8	2567.3	6.68	3.8	19%
$U_{0.9}$	3072.8	2502.9	7.12	4.1	19%
U_1	3227.5	2559.7	5.06	2.8	21%

α as a subscript in the topology identifier (A, B, C, or U). For instance, $B_{0.9}$ corresponds to a scenario where $\alpha = 0.9$ and the nodes are distributed according to topology B. The second and third columns list delays of the delay-optimal Hamiltonian solution and the proposed PSA solution, respectively. The fourth column reports the additional computation time that PSA takes on top of the computational cost to compute the Hamiltonian solution. The fifth column shows K, the average number of loops in the final PSA schedule (the numbers are not necessarily integer since each number is the average of 100 independent runs). The sixth column represents the percentage of improvement that PSA brings in comparison to the delay-optimal Hamiltonian solution, i.e., the values of the quantity

$$\frac{\text{Delay-optimal Hamiltonian Solution Delay} - \text{PSA Solution Delay}}{\text{Delay-optimal Hamiltonian Solution Delay}}.$$

The results show that the average delay of topology A is less than that of topology B, which is less than that of topology C, which is less than topology U. As it can be seen, by applying PSA the improvement is at least about 10% and in some cases more than 40% compared to the delay-optimal Hamiltonian solutions. The improvement is more significant when the sink is located closer to the nodes (the center of the area) rather than the corner of the area which might seem intuitive. The simulation results also show that the additional computational time PSA requires on top of the delay-optimal Hamiltonian solution

Table 3: Results for the multiple ME case - the sink located at the center of the area.

Scenario	Z	MURA Delay (sec)	Z-PSA Delay (sec)	Improvement (%)
$A_{0.9}$	2	784.3	472.9	40%
A_1	2	812.6	552.1	32%
$A_{0.9}$	4	405.2	237.4	41%
A_1	4	417.8	278.9	33%
$A_{0.9}$	10	178.5	104.5	41%
A_1	10	183.2	120.7	34%
$C_{0.9}$	2	1126.5	831.7	26%
C_1	2	1142.7	859.2	25%
$C_{0.9}$	4	568.4	417.9	26%
C_1	4	581.3	429.6	26%
$C_{0.9}$	10	242.4	173.1	29%
C_1	10	256.9	181.3	29%
$U_{0.9}$	2	1530.5	1035.3	32%
U_1	2	1559.2	1078.1	31%
$U_{0.9}$	4	773.9	504.7	35%
U_1	4	789.2	518.1	34%
$U_{0.9}$	10	315.7	218.2	31%
U_1	10	329.4	231.1	30%

is on the order of a few seconds. This computational overhead is correlated with the number of loops, and in general as the number of loops increases, so does the improvement. We note that the average numbers of loops tends to be larger in the scenarios where the sink is located at the center compared to the scenarios where the sink is located at the corner.

For the multiple ME case, we use Multi-Route Algorithm (MURA) proposed in [14] as the basis to evaluate the performance of the proposed algorithm. MURA was shown in [14] to be very effective in minimizing the delay. Briefly, MURA starts with N MEs (called "ferries" in [14] by the authors) and each node is assigned to an ME. That is, each ME route consists of one node. MURA refines the node assignment and reduces the number of MEs to Z by using four types of operations. In each step, MURA estimates the weighted delay of the resulting node assignment for each operation and chooses to perform the best one until the number of MEs is Z and no further improvement can be found. In our simulations we let Z, the number of MEs, to take values from 2, 4, and 10 to represent different levels of available resources. Similar results as in the single ME case are obtained for the multiple ME case and are summarized in Tables 3 and 4. Again, our algorithm outperforms the alternative by at least 10% and in some cases the improvement exceeds 40%.

Table 4: Results for the multiple ME case - the sink located at the corner of the area.

Scenario	Z	MURA Delay (sec)	Z-PSA Delay (sec)	Improvement (%)
$A_{0.9}$	2	908.5	807.7	11%
A_1	2	921.2	831.5	10%
$A_{0.9}$	4	467.3	416.7	11%
A_1	4	481.4	428.2	11%
$A_{0.9}$	10	208.3	173.7	17%
A_1	10	221.4	188.1	15%
$C_{0.9}$	2	1124.8	973.2	13%
C_1	2	1138.6	997.3	12%
$C_{0.9}$	4	560.9	497.3	11%
C_1	4	571.8	505.1	12%
$C_{0.9}$	10	232.4	204.5	12%
C_1	10	246.7	212.9	14%
$U_{0.9}$	2	1551.8	1271.2	18%
U_1	2	1573.5	1292.6	18%
$U_{0.9}$	4	787.9	630.2	20%
U_1	4	802.5	648.7	19%
$U_{0.9}$	10	314.1	256.3	18%
U_1	10	327.6	268.2	18%

7. Conclusions

We presented algorithms aimed at minimizing the average delay in harvesting data using mobile elements in wireless networks. Our approach transforms Hamiltonian solutions to potentially better non-Hamiltonian solutions. The algorithm improves upon the available Hamiltonian solution by "splitting" it into several loops. Numerical results show that our algorithms can improve average delay by more than 40% in some instances while requiring a modest computational effort to modify the TSP-based route. The improvement is more significant when the sink is located closer to the nodes (closer to the center of the area).

There are several directions future work could consider. Focusing on sensor networks in urban areas, one can note that MEs travel on a network of roads and a Euclidean distance is not the most appropriate metric to be used. Instead, one could assume that MEs travel on a graph and use edge distances. Further, in many settings MEs do not need to reach the exact location of sensor nodes but only approach them closely. Thus, it is natural to consider a variant of TSP where the route needs to reach neighborhoods of points – knows as TSP with neighborhoods [39]. Other interesting directions include handling multiple sinks and incorporating ME energy efficiency into the objective.

References

[1] R. Moazzez-Estanjini, I. C. Paschalidis, Improved delay-minimized data harvesting with mobile elements in wireless sensor networks, in: Proceedings of the 9th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Princeton, New Jersey, 2011, pp. 49–54.

[2] R. Shah, S. Roy, S. Jain, W. Brunette, Data MULEs: Modeling a three-tier architecture for sparse sensor networks, Elsevier Ad Hoc Networks Journal 1 (2) (2003) 215–233.

[3] A. Chakrabarti, A. Sabharwal, B. Aazhang, Using predictable observer mobility for power efficient design of sensor networks, in: IPSN, 2003, pp. 129–145.

[4] X. Li, A. Nayak, I. Stojmenovic, Exploiting actuator mobility for energy-efficient data collection in delay-tolerant wireless sensor networks, in: 2009 Fifth International Conference on Networking and Services, 2009, pp. 216–221.

[5] W. Wang, V. Srinivasan, K. Chua, Using mobile relays to prolong the lifetime of wireless sensor networks, in: MobiCom, 2005, pp. 270–283.

[6] O. Tekdas, V. Isler, J. Lim, A. Terzis, Using mobile robots to harvest data from sensor fields, in: IEEE Wireless Communications in Robotic Networks, 2009.

[7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, D. Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet, in: International Conference on Architectural support for programming languages and operating systems, 2002.

[8] A. Beafour, M. Leopold, P. Bonnet, Smart tag based data dissemination, in: ACM Workshop on Wireless Sensor Networks and Applications, 2002.

[9] Daknet project, http://web.media.mit.edu/ amir/daknet/.

[10] M. Dunbabin, P. Corke, I. Vasilescu, D. Rus, Data muling over underwater wireless sensor networks using an autonomous underwater vehicle, in: Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA), 2006, pp. 2091–2098.

[11] M. Todd, D. Mascarenas, E. Flynn, T. Rosing, B. Lee, D. Musiani, S. Dasgupta, S. Kpotufe, D. Hsu, R. Gupta, G. Park, T. Overly, M. Nothnagel, C. Farrar, A different approach to sensor networking for SHM: Remote powering and interrogation with unmanned aerial vehicles, in: Proceedings of the 6th International workshop on Structural Health Monitoring, 2007.

[12] W. Zhao, M. Ammar, Message Ferrying: proactive routing in highly-partitioned wireless ad hoc networks, in: Proceedings of the 9th IEEE workshop on future trends of distributed computing systems, 2003, pp. 308–314.

[13] W. Zhao, M. Ammar, E. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks, in: Proceedings of the 5th ACM international symposium on mobile ad hoc networking and computing, 2004, pp. 187–198.

[14] W. Zhao, M. H. Ammar, E. W. Zegura, Controlling the mobility of multiple data transport ferries in a delay-tolerant network, in: Proceedings of IEEE INFOCOM 2005, Vol. 2, 2005, pp. 1407–1418.

[15] A. Pandya, A. Kansal, G. Pottie, Goodput and delay in networks with controlled mobility, in: 2008 IEEE Aerospace Conference, 2008, pp. 1–8.

[16] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, D. Estrin, Intelligent fluid infrastructure for embedded networks, in: Proceedings of MobiSys, 2004, pp. 111–124.

[17] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, C. Lee, Partitioning based mobile element scheduling in wireless sensor networks, in: Proceedings of IEEE SECON, 2005, pp. 386–395.

[18] G. Xing, T. Wang, W. Jia, M. Li, Rendezvous design algorithms for wireless sensor networks with a mobile base station, in: Proceedings of ACM MobiHoc, 2008, pp. 231–239.

[19] I. Akyildiz, I. Kasimoglu, Wireless sensor and actor networks: research challenges, Ad Hoc Networks Journal 2 (2004) 351–367.

[20] E. Ngai, Y. Zhou, M. Lyu, J. Liu, Reliable reporting of delay-sensitive events in wireless sensor-actuator networks, in: Proceedings of the 3rd IEEE MASS, 2006.

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

[22] S. Hashish, A. Karmouch, Deployment-based solution for prolonging lifetime in sensor networks with multiple mobile sinks, Ad Hoc and Sensor Wireless Networks.

[23] S. Nesamony, M. Vairamuthu, M. Orlowska, On optimal route of a calibrating mobile sink in a wireless sensor network, in: Proceedings of INSS, 2007, pp. 61–64.

[24] Z. Vincze, D. Vass, R. Vida, A. Vidacs, A. Telcs, Adaptive sink mobility in event-driven densely deployed wireless sensor networks, Ad Hoc and Sensor Wireless Networks 3 (2) (2007) 255–284.

[25] X. Li, A. Nayak, I. Stojmenovic, Sink mobility in wireless sensor networks, in: X. Li, H. Liu, A. Nayak, I. Stojmenovic (Eds.), Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication, Wiley, 2009.

[26] R. Sugihara, R. Gupta, Improving the data delivery latency in sensor networks with controlled mobility, Distributed Computing in Sensor Systems (2008) 386–399.

[27] I. C. Paschalidis, R. Moazzez-Estanjini, The capacity of sparse networks under controlled mobility, in: Proceedings of the 49th IEEE Conference on Decision and Control, Atlanta, Georgia, 2010, pp. 5610–5615.

[28] S. Parragh, K. Doerner, R. Hartl, A survey on pickup and delivery problems: Part I: Transportation between customers and depot, Journal fr Betriebswirtschaft 58 (1) (2008) 21–51.

[29] S. Parragh, K. F. Doerner, R. Hartl, A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations, Journal fr Betriebswirtschaft 58 (2) (2008) 81–117.

[30] G. Berbeglia, J. Cordeau, I. Gribkovskaia, G. Laporte, Static pickup and delivery problems: a classification scheme and survey, TOP 15 (1) (2007) 1–31.

[31] R. Diestel, Graph Theory, 3rd Edition, Springer-Verlag, 2005.

[32] B. Yuan, M. Orlowska, S. Sadiq, On the optimal robot routing problem in wireless sensor networks, IEEE Trans. Knowl. Data Eng. 19 (9) (2007) 1252–1261.

[33] R. Suganthel, P. Balasubramanie, Efficient routing for intermittently connected mobile ad hoc network, IJCSNS International Journal of Computer Science and Network Security 8 (11) (2008) 184–191.

[34] Z. Zhang, Z. Fei, Route design for multiple ferries in delay tolerant networks, in: Proc. of IEEE Wireless Communications and Networking Conference, 2007, pp. 3460–3465.

[35] A. Somasundara, A. Ramamoorthy, M. Srivastava, Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines, in: Proceedings of the 25th IEEE International Real-Time System Symposium, 2004, pp. 296–305.

[36] D. Henkel, T. Brown, Towards autonomous data ferry route design through reinforcement learning, in: World of Wireless, Mobile and Multimedia Networks, 2008, pp. 1–6.

[37] O. Boxma, H. Levy, J. Weststrate, Efficient visit frequencies for polling tables: minimization of waiting cost, Queueing Systems 9 (1) (1991) 133–162.

[38] G. Laporte, The traveling salesman problem: An overview of exact and approximate algorithms, European Journal of Operational Research 59 (2) (1992) 231–247.

[39] A. Dumitrescu, J. Mitchell, Approximation algorithms for TSP with neighborhoods in the plane, in: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2001, pp. 38–46.