Partitioning Based Mobile Element Scheduling in Wireless Sensor Networks

Yaoyao Gu, Doruk Bozdağ, Eylem Ekici, Füsun Özgüner, Chang-Gun Lee

Department of Electrical and Computer Engineering
Ohio State University, Columbus, OH 43210
{guy,bozdagd,ekici,ozguner,cglee}@ece.osu.edu

Abstract— **In recent studies, using mobile elements (MEs) as mechanical carriers of data has been shown to be an effective way of prolonging sensor network life time and relaying information in partitioned networks. As the data generation rates of sensors may vary, some sensors need to be visited more frequently than others. In this paper, a partitioning-based algorithm is presented that schedules the movements of MEs in a sensor network such that there is no data loss due to buffer overflow. Simulation results show that the proposed Partitioning Based Scheduling (PBS) algorithm performs well in terms of reducing the minimum required ME speed to prevent data loss, providing high predictability in inter-visit durations, and minimizing the data loss rate for the cases when the ME is constrained to move slower than the minimum required ME speed.**

I. Introduction

Advances in VLSI and radio transceiver technologies as well as evolution in power efficient methods have enabled realization of wireless sensor networks, which especially find use in critical areas such as battlefield surveillance, environmental monitoring [1], nuclear, biological and chemical attack detection [2], and traffic monitoring [3]. The traditional approach for data delivery in wireless sensor networks involves multi-hop communication from data sources to sinks. However, multi-hop communication may not always be possible due to network partitioning. Furthermore, relaying data over a large number of hops also reduces the life time of sensor nodes. As the number of sinks in wireless sensor networks is relatively small, the nodes close to the sinks may run out of energy before the others since all traffic is funnelled through these nodes. Although battery replenishment and power harvesting techniques are under development, reducing energy consumption of individual sensor nodes still plays the most important role in maximizing the network lifetime.

Recently, mobile elements have been proposed as mechanical carriers of data in wireless networks [4], [5]. Using mobile elements, the battery life time of individual sensors can be increased. Low density networks also benefit from utilization of mobile elements. The ZebraNet project [6] and Manatee project [7], [8] are among the first to explore the idea of using mobility in sensor networks. In [9], a three-tier MULE (Mobile Ubiquitous LAN Extensions) architecture has been proposed, where the mobile elements are vehicles (cars, buses) outfitted with transceivers and move randomly to collect data from the sensor nodes as they approach them. As an application of controlled mobility, Message Ferrying is introduced in [10], [11], [12], where a set of special mobile elements (called message ferries) provide communication service for nodes in sparse ad-hoc networks. Since each node communicates only with the message ferries, any long distance communication is avoided, resulting in increased node lifetime. In Wireless Sensor and Actor Networks (WSANs) [13], a set of mobile nodes, called *Actors*, are employed to perform appropriate actions based on the data collected by sensor nodes.

A mobile element capable of short-range communication can collect data from the nearby sensor nodes as it approaches during its motion. Controlling the mobile element motion leads to the *Mobile Element Scheduling (MES)* problem [14], which is defined as the problem of scheduling the visits of a mobile element to sensor nodes so that there is no data loss due to sensor node buffer overflow. It is assumed that all sensors have limited capacity buffer, and once a mobile element visits a node, it transfers the data from the sensor to its own memory and the sensor's memory is freed.

The amount and frequency of data generation in sensor nodes varies based on the event occurrence frequency, which is generally a function of the sensor location. In the example given in [14], sensor networks can be used to sense air pollution levels in large urban areas. Since the variation of pollution levels are expected to be higher in industrial areas than in residential areas, sensors in industrial areas generate data at higher rates than sensors in residential areas. As a result, sensors must be visited at different frequencies, and a sensor may need to be visited multiple times before all other sensors are visited to avoid buffer overflow. As soon as a node is visited and data is transferred, its visit deadline is updated. Therefore, the deadlines are "dynamically" updated as the mobile element performs its job as a data collector.

In this paper, we investigate the MES problem and propose a *Partition Based Scheduling (PBS)* algorithm which tackles the problem by dividing it into two sub-problems: Partitioning and Scheduling. First, all nodes are partitioned into several groups with respect to their data generation rate and location. Then, within a single group, the scheduling algorithm generates node visiting schedules for the ME by minimizing the overhead of moving back and forth across far-away nodes. Finally, the scheduling solutions of the groups are concatenated forming the entire Mobile Element path so that all nodes can be visited at adequate frequencies to prevent buffer overflows. In this paper, the investigated performance metrics are the data loss rate for a given ME speed and the minimum speed to completely avoid buffer overflow. We confirm through simulations that our PBS algorithm performs well in terms of both metrics as well as providing high predictability in nodes inter-visit schedule.

The remainder of the paper is organized as follows: The related work is presented in Section II, followed by a detailed description of the MES problem and our PBS scheduling algorithm in Section III. Simulation results are presented and discussed in Section IV. Finally, the paper is concluded in Section V.

II. Related Work

The use of mobile elements to carry data has recently been considered in the literature. Data MULES [9] focuses on utilization of mobile elements (called MULEs) in sparse sensor networks. The MULEs move randomly and collect data opportunistically from sensor nodes. The movement of data gathering elements are not controlled in this framework. In the message ferrying (MF) [10], [11] approach, message ferries are used to route data from one node to another in a sparse ad hoc network. Based on a given traffic matrix, the goal of message ferrying approach is to find the optimal route of a ferry so that the average delay from source to destination is minimized while meeting the bandwidth requirement of flows.

Related to the MES problem are the Orienteering Problem (OP) [15], the Prize Collecting Traveling Salesman Problem (PC-TSP) [16], as well as the original TSP. These problems deal with routing a vehicle to visit each city at most once. However in our problem, a node may need to be visited more than once before all other nodes are visited because of the difference in buffer overflow deadlines. In OP and Prize Collecting TSP, each city has an associated nonnegative *prize* and the vehicle aims to collect the maximum total prize. Although the mobile element in the MES problem also collects data that can be considered as prize, the value of the prize is dynamic and depends on the time of the visit.

The Vehicle Routing Problem (VRP) [17] is defined as finding a route for a vehicle that minimizes the total travel cost to deliver cargo between a depot and customers. Unlike TSP, VRP considers more than one vehicle and nodes can be visited more than once. Among many variants of VRP, VRP with deadline [18], [19] and Periodic VRP (PVRP) [17] are relevant to the MES problem. The goal of *VRP with Deadline* is to schedule a vehicle to visit as many nodes as possible by their deadlines. Different from our problem, each node in Deadline VRP is visited at most once. Furthermore, the deadline of the visit to a node in Deadline VRP is fixed, whereas the deadline of a node changes periodically in our case.

Periodic VRP is the problem of designing routes for delivery vehicles for a given T-day period where not all customers require delivery on every day in the period. Customers are associated with a set of feasible schedules that are some combinations of days they can be visited. In PVRP, the feasible solution set consists of a finite number of possibilities. However, in MES, the feasible solution set consists of an infinite number of possibilities such that the time difference between any two consecutive visits scheduled to the same node is smaller than the associated buffer overflow time. Moreover, in the MES problem, the vehicle does not need to go back to a certain node at the end of every cycle whereas the vehicles in PVRP go back to the depot every day. Although the MES problem can be discretized in the time domain, the resulting size of the feasible solution set does not scale well with the range of data generation rates.

The MES problem in wireless sensor networks is proved to be NP-complete and three heuristic algorithms are presented in [14]. The first one is the Earliest Deadline First (EDF) algorithm, where the node with the closest deadline is visited first. To improve EDF, the second algorithm, EDF with k-lookahead, is proposed. Instead of visiting a node whose deadline is the earliest, this algorithm considers the $k!$ permutations of the k nodes with smallest deadlines, and chooses the next node which leads to the earliest finish time. Consequently, the EDF with k lookahead algorithm performs better than pure EDF. The third algorithm is the Minimum Weight Sum First (MWSF) algorithm, which accounts for the weights of deadlines as well as distances between nodes in determining the visiting schedule. The MWSF algorithm performs the best among the three proposed algorithms.

Even though the MWSF solution considers both deadlines as well as distances, "back and forth" movement between far away nodes occurs frequently. In our proposed PBS algorithm, we consider the deadline and distances of *all nodes* simultaneously and utilize a two-layer scheduling approach to reduce the back-and-forth movement behavior. This is achieved by partitioning the set of all nodes according to deadlines as well as their geographic locations. The resulting schedules and paths are usually shorter, which reduces the minimum required speed of the ME to prevent buffer overflow.

III. Partitioning-Based Scheduling Algorithm

Algorithm 1. PBS($\{w_{ij}\}$,$\{o_i\}$)

```
1: Partition the nodes $\{n_i\}$ into M bins according to the buffer overflow times $\{o_i\}$.
2: Geographically partition each bin $B_j$ into $2^{j-1}$ sub-bins using partitioning algorithm.
3: Calculated a TSP path for each sub bin.
4: Concatenate all TSP paths to build the overall schedule.
```

Our proposed Partitioning-Based Scheduling (PBS) algorithm is designed to solve the Mobile Element Scheduling problem, which aims to schedule the visits of the mobile element to each sensor to avoid data loss due to sensor buffer overflow. With the PBS algorithm, we first partition all nodes into several groups, called *bins*, such that nodes in the same bin have similar deadlines and are geographically close to each other. Then, to solve the scheduling problem of the mobile element within a single bin, we solve the Traveling Salesman Problem, which calculates a minimum cost tour that visits each node exactly once. Finally, the schedules for individual groups are concatenated to form the entire schedule. In this section, we first outline our notation and problem formulation. Then, we present a detailed description of our solution to the MES problem.

A. Problem Formulation and Notation

In this paper, wireless sensor networks composed of homogeneous sensor nodes are considered. The nodes are equipped with wireless communication interfaces with limited ranges. Sensor nodes capture the events in their surroundings and record them to their buffers. The following assumptions are also made regarding the sensor nodes and the mobile element.

- The physical sizes of sensor nodes and mobile elements are negligible.
- Mobile elements can move in any direction without any latency of making any turns.
- Data transfer time between sensor nodes and mobile elements is negligible compared to the delay due to ME movement.
- All sensors have the same finite buffer size, and at time $t = 0$, all sensor node buffers start in an empty state. Mobile elements have infinite data buffers and do not suffer from buffer overflow.

We denote the number of nodes in the network by N and the set of nodes by $\{n_i\}$, where $i = 1, \ldots, N$. Let w_{ij} denote the distance between nodes n_i and n_j. The buffer overflow time and data generation rate of each node are denoted by o_i and f_i, respectively. For a buffer of size b, $o_i = \frac{b}{f_i}, i = 1, \ldots, N$. We assume that the data generation rate f_i is directly related to event occurrence rate.

The MES problem $(\{w_{ij}\}, \{o_i\})$ is to find a sequence of visits to nodes $\{n_i\}$, for $i = 1, \ldots, N$, and calculate the minimum speed v_{min} of the ME so that no node buffers overflows.

B. The Proposed PBS Algorithm

Let B_j, $j = 1, \ldots, M$, denote bin j, where M is the total number of bins. In PBS, nodes are first partitioned into bins in such a way that overflow times of the nodes in bin B_i is smaller than those in B_j, for $j > i > 0$. Moreover, the range of overflow times for nodes in B_{i+1} is twice that of B_i. This allows the nodes in B_i to be visited twice more frequently than the nodes in B_{i+1} during generation of the visit schedules. Then, each bin again is partitioned into sub-bins so that nodes in the same sub-bin are geographically close to each other. This two level partitioning results in groups of nodes with similar deadlines and locations. Therefore, in each sub-bin, node visits can be scheduled using a solution for the Traveling Salesman Problem. Finally, the schedule for individual groups are concatenated to form the entire schedule that guarantees all deadline constraints are satisfied. In the following, we describe the details of PBS algorithm outlined in Algorithm 1.

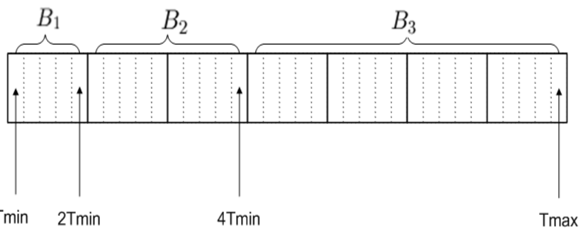


Fig. 1. The diagram of step 1: Partition according to overflow times.

1) Bin Partitioning according to overflow times: Let o_{min} and o_{max} denote the minimum and maximum overflow time of all nodes. Nodes are assigned to bins according to the following equation:

$$n_i \in B_j, \begin{cases} \text{if } 2^{j-1}o_{min} \leq o_i \leq 2^j o_{min}, j = 1, \ldots, M-1; \\ \text{if } 2^{j-1}o_{min} \leq o_i \leq o_{max}, j = M. \end{cases} \tag{1}$$

Fig. 1 shows an example of partitioning all nodes into three bins. The overflow times of nodes in B_1 range from o_{min} to $2o_{min}$, the overflow times of nodes in B_2 range from $2o_{min}$ to $4o_{min}$, and the overflow times of nodes in B_3 range from $4o_{min}$ to o_{max}. Therefore, all nodes in a bin B_j are considered as if they are assigned an overflow time of $2^{j-1}o_{min}$. Every bin is then visited at different frequencies: All nodes in B_1 are visited every cycle, nodes in B_2 are visited every other cycle, and nodes in B_3 are visited every four cycles, where we define a *cycle* as a closed path among a set of nodes, such that no node is included more than once in the same cycle. We also define a *supercycle* as a closed path composed of concatenated cycles such that every node is included at least once in a supercycle. In our algorithm, a supercycle is equivalent to the period of the ME schedule.

2) Sub-bin partitioning according to locations: Each bin obtained in Step 1 is then partitioned into sub-bins according to the node locations such that the nodes in the same sub-bin are geographically close to each other. The number of sub-bins of a bin B_j is calculated based on the index j. As an example, the nodes in B_2 need to be visited only half as frequently as the nodes in B_1. Hence, B_2 is partitioned into two sub-bins: B_2^1 and B_2^2, where B_2^1 is visited every even cycle, and B_2^2 is visited every odd cycle. Following the same rule, B_3 is partitioned into four sub-bins: B_3^1, B_3^2, B_3^3 and B_3^4, and in general, bin B_i is partitioned into 2^{i-1} sub-bins: $B_i^1, \ldots, B_i^{2^{i-1}}$. The KD-tree algorithm [20] is utilized to realize this partitioning. KD-tree is a k-dimensional binary search tree for information retrieval by associative searches. In our case, we use the 2D-tree where the two dimensions are the length and width of the sensor deployed field.

An example of 2D-tree partitioning is shown in Fig. 2 for N randomly deployed nodes in a given region. As shown in the figure, the nodes are first geometrically divided into two balanced parts by the cut A with respect to their x-coordinates. The nodes having x coordinates smaller than the average of

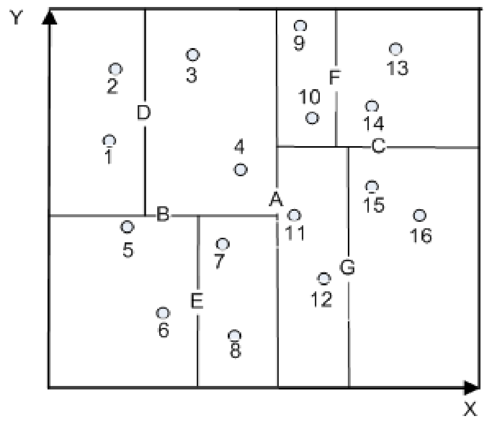


Fig. 2. Nodes in a 2 dimension space cut by KD-tree.

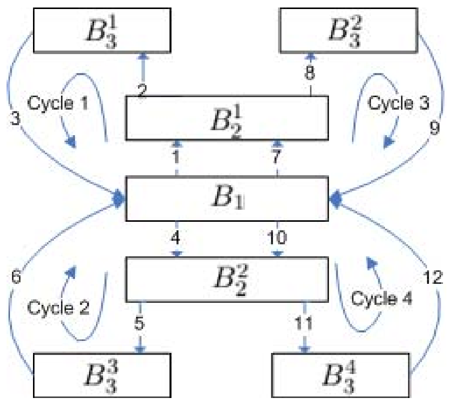


Fig. 3. An example of the visiting sequence of sub-bins in a 'supercycle'

Algorithm 2. 2D-tree(S, cutonx, depth, d)

```
1:  **if** d=depth **then**
2:      return
3:  **end if**
4:  **if** cutonx **then**
5:      sort S according to x coordinate
6:  **else**
7:      sort S according to y coordinate
8:  **end if**
9:  split S into S1 and S2
10: cutonx ← !cutonx
11: KD-tree(S1, cutonx, depth, d+1)
12: KD-tree(S2, cutonx, depth, d+1)
```

the x coordinates $\frac{\sum_{i=1}^{N} x_i}{N}$ is assigned to one part and rest to the other.

As the cut A partitions the region vertically, cut B and C horizontally partition the resulting two parts considering the y-coordinates of the nodes. This process is repeated alternatingly until the desired number of partitions is obtained. The number of partitions, which is also the number of sub-bins in our problem, decides the depth of the 2D-tree. In our PBS solution, bin B_j is partitioned into 2^{j-1} sub-bins. Therefore, the depth of 2D-tree for B_j is:

$$depth_j = log_2 2^{j-1} = j - 1. \qquad (2)$$

The pseudo-code of the 2D-tree partitioning algorithm is shown in Algorithm 2. The input S is the set of nodes in the bin to be partitioned. $cutonx$ is a boolean flag showing the cut criteria: x coordinate (true) or y coordinate (false). d is the current cut level and $depth$ is the depth of the KD-tree, which is determined by the desired number of sub-bins. In each procedure call, the nodes are first sorted with respect to x or y coordinates and partitioned into two based on value of $cutonx$. Then, $cutonx$ flag is changed and the procedure is called recursively on resulting partitions until the desired 2D-tree depth is reached.

3) Forming a TSP solution on each sub-bin: The two level partitioning results in groups of nodes with similar deadlines and locations. Therefore, the ME scheduling problem is reduced to the Traveling Salesman Problem (TSP) for each sub-bin. In the literature, several algorithms to calculate the TSP paths are proposed such as the nearest neighbor, LKH [21], and Prim's algorithm [22]. In our solution, we adopt the Prim's algorithm to first calculate the minimum spanning tree and then make a pre-order tree walk. In PBS, the path of the ME differs from the TSP path by not returning back to the first visited node after visiting the last node in the sub-bin. Instead, the ME visits the first node of the next sub-bin. Then, it follows the ME path in that sub-bin and proceeds to the following sub-bin. As a special case, in B_1, which is the first bin in every cycle, the node with the minimum overflow time is taken as the start node.

4) Forming the supercycle: After the ME paths within the sub-bins are calculated, the visit order of the sub-bins should be decided to form the complete ME path. At the end of partitioning, there are 2^{i-1} sub-bins of bin B_i, each composed of nodes with deadlines at least twice the deadlines of the nodes in sub-bins of B_{i-1}, $i = 1, \ldots, M$. Therefore, in a reasonable ME schedule, sub-bins in the same bin should be visited with the same frequency and a sub-bin in B_{i-1} should be visited twice more frequently than a sub-bin of bin B_i. This heuristic choice results in a sub-bin of B_i to be visited 2^{M-i} times for each visit to a sub-bin of bin B_M. Recall that in a supercycle each sub-bin, hence each node, is visited at least once. In other words, each sub-bin in the least frequently visited bin B_M is visited at least once. Without loss of generality, let the ME visit each sub-bin in B_M exactly once in a supercycle. As a result, a sub-bin of bin B_i should be visited exactly 2^{M-i} times in a supercycle according to our heuristic choice.

Let $I_{i,j}$ be defined as the maximum duration between two consecutive visits to a node in sub-bin B_i^j. Then, the sufficient condition to avoid buffer overflow for all nodes of B_i^j is given as

$$I_{i,j} \leq o_{min} \times 2^{i-1} \qquad (3)$$

Let $L_{i,j}$ denote the longest ME path between two consecutive visits to a node of sub-bin B_i^j, i.e., $L_{i,j} = I_{i,j} \times v$, where v is the speed of the ME. Hence, to avoid buffer overflow in B_i^j, $v \geq \frac{L_{i,j}}{o_{min} \times 2^{i-1}}$ should be satisfied. This can be achieved by either increasing the ME speed or decreasing $L_{i,j}$. Our objective of minimizing the ME speed for a lossless schedule can be achieved by minimizing $L_{i,j}$ for each sub-bin and setting v to the largest required value to satisfy Inequality 3 for every sub-bin.

Since all sub-bins are formed according to geographical proximity as well as overflow deadlines, the TSP tours of sub-bins B_i^j of a bin B_i are approximately the same length. In order to have a predictable visiting schedule of sub-bins, we form cycles such that only one sub-bin from each bin is contained in a cycle. Furthermore, all cycles preserve the order of bins B_i from which sub-bins are selected. In particular, sub-bins are visited starting from B_1 in increasing bin number order and a sub-bin in B_i is always visited after the same sub-bin in B_{i-1}. This ensures that in every cycle that a sub-bin B_i^j is visited, the nodes in B_i^j are visited at exactly the same time as they were visited in the previous cycle relative to the start times of the cycles. The rationale behind this choice will be discussed in more detail in the following subsection.

There are two times more sub-bins in bin B_i than B_{i-1} and each sub-bin in B_i is always visited after the same sub-bin in B_{i-1}. As a result, exactly two sub-bins in B_i follow each sub-bin in B_{i-1} for $i = 2, \ldots, M$. In PBS, the two sub-bins to follow a particular sub-bin is decided by considering the geographic locations of the sub-bins. We use the center of gravities of sub-bins to measure the distances between sub-bins. For each sub-bin of B_{i-1}, we greedily schedule the closest two sub-bins from B_i to follow it. Note that more complicated partitioning algorithms can be used to further minimize the distances between consecutively visited sub-bins.

In Fig. 3, a visiting sequence example of sub-bins in a supercycle is given for $M = 3$. The supercycle in this case consists of four cycles and the visit schedules of the sub-bins are as follows: B_1 is visited every cycle, B_2^1 is visited in cycles 1 and 3, and B_2^2 is visited in cycles 2 and 4 and the sub-bins of the last bin, B_3, are visited in cycles 1, 2, 3, and 4, one at a time.

C. Discussion of Minimum Required Speed

In order to minimize the power consumption of the ME, its speed should be minimized by a MES solution. In this paper, we calculate a lower bound for the PBS solutions on the ME speed, denoted as v_{min}, such that no buffer overflow occurs in the network nodes. In PBS algorithm, although the length of the paths that the ME traverses in different cycles are tried to be made close to each other, they are still not exactly the same. Let L_{max} and T_{max} denote the path length and total visit duration of the maximum length cycle, respectively.

As mentioned in the previous subsection, in every cycle a particular node is visited, it is always visited at the same time relative to the start time of the cycle. Since each sub-bin, therefore node, in B_i is visited every 2^{i-1} cycles, the maximum inter-visit time for a node in bin B_i is upper bounded by $2^{i-1} \times T_{max}$. In order to avoid buffer overflow, maximum inter-visit time to a node in B_i should be less than or equal to $o_{min} \times 2^{i-1}$, resulting in

$$T_{max} = \frac{L_{max}}{v} \leq o_{min} \tag{4}$$

where v is the speed of the ME. By letting $\frac{L_{max}}{v} = o_{min}$ in order to solve for the minimum ME speed for no buffer overflow, v_{min} is calculated as

$$v_{min} = \frac{L_{max}}{o_{min}}. \tag{5}$$

In general, L_{max} cannot be approximated easily except for certain special deployments of sensor nodes. Once an ME schedule is generated by the PBS algorithm, L_{max}, therefore v_{min}, can be computed numerically. If the ME is constrained to move at a smaller speed than v_{min}, there will be data losses due to buffer overflow. Besides trying to minimize v_{min} for a lossless schedule, the proposed algorithm is also designed to minimize the data loss rate in such cases. Although not discussed in this paper, further optimizations can be done for the latter objective by trading off performance of the former one. In the performance analysis section, we also present the performance of our PBS algorithm as a function of the ME speed.

D. Time Complexity Analysis

Assume that N and M denote the number of sensor nodes and number of bins, respectively. Then, the complexity for each step of the PBS algorithm is as follows:

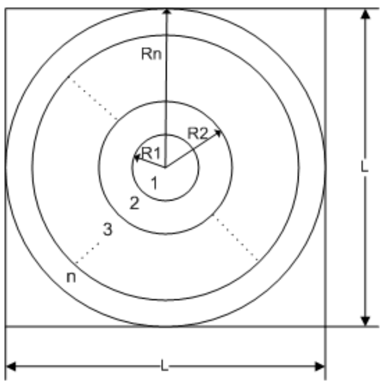
- *Step 1:* The bin partitioning in Step 1 is dominated by sorting of the nodes with respect to the buffer overflow times resulting in $O(NlogN)$ complexity.
- *Step 2:* In the geographical partitioning step, 2D-tree algorithm has a time complexity of $DNlogN$ where D is the depth of the tree. Since 2D-tree algorithm is applied on all bins and D is equal to $i-1$ for bin B_i, the overall complexity of this step is $O(M^2 NlogN)$.
- *Step 3:* The complexity of calculating the TSP path for S nodes is $O(S^2)$. Since there are a total of $2^M - 1$ sub-bins, the time complexity of this step is $O(N^2 2^M)$ in the worst case. Note that 2^M is $O(o_{max})$.
- *Step 4:* Time complexity of Step 4 is $O(NlogN)$.

As a result, the PBS algorithm have an overall time complexity of $O(M^2 NlogN + N^2 2^M)$.

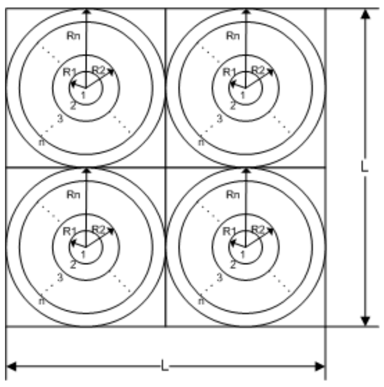
IV. Performance Evaluation

To evaluate the performance of our proposed PBS algorithm, we have run an extensive set of simulations. In this simulation study, the following scenarios are considered for performance evaluation.

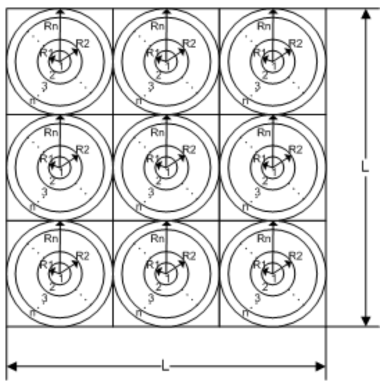
- Simulation I: We observe the data loss rates as a function of the mobile element speed.
- Simulation II: We observe the effect of the node density on the minimum required speed to avoid data loss.



(a) Topology of type A



(b) Topology of type B



(c) Topology of type C

Fig. 4. Three types of topologies considered in simulation. Each topology A, B, and C have one, four, and nine eyes, respectively.

- Simulation III: The effect of number of bins on the PBS performance is evaluated for different network sizes and properties.
- Simulation IV: Sensor visit predictability is investigated as a function of overflow time and node density through inspection of the standard deviation of inter-visit times.

Simulations I, II, and IV are also run for the MWSF [14] algorithm to compare the performance of our PBS algorithm. The details of the simulation setup as well as the detailed discussion of the simulation results are presented in the following sections.

A. Metrics and Methodology

We observe the following metrics to evaluate the performance of the PBS algorithm.

- *Data loss occurrence rate* is defined as the ratio of number of sensors missing their deadlines to the total number of nodes visited. It presents the performance in terms of the number of sensors missing their deadlines rather than the amount of data lost.
- *Data loss rate* is defined as the ratio of the data lost due to buffer overflow to the total amount of data generated.
- *Minimum required speed* is defined as the minimum speed of the mobile element to guarantee no sensor buffer overflow.
- *Predictability* is defined as the standard deviation of the inter-visit duration which is the duration between two visits to the same node.

In our simulations, we use the following default settings unless specified otherwise. Each simulation is run on a network with 150 sensor nodes randomly placed following the uniform distribution on a 100×100 unit^2 area. Each sensor node is equipped with same size buffers. The overflow time of each sensor differs due to different events occurrence rate of different regions. To simulate the different event occurrence rates based on regions, we assume that events are concentrated at certain locations, which we call $Eyes$. The nodes in the eye centers have highest data generation rate, which drops radially outwards. Four topologies[1], A, B, C, and D, are considered in our simulations. Topologies A, B, and C have one, four, and nine eyes, respectively. Topology D correspond to uniformly distributed data generation rates over the sensor network. It can also be considered as having infinite number of eyes. As shown in Figure 4(a), to model topology A, a sequence of concentric circles divides the given area into several ring shaped regions; Region 1 to Region n. The radius of each concentric circle is denoted by R_1, R_2, R_3,..., R_n, where $R_1 = 2\ units$. The value of each radius is calculated as:

$$R_i = i \cdot R_1, \quad i = 1, \ldots, n$$

The nodes in the innermost region are assigned the smallest overflow time, which is called the *base_time*, and overflow times for nodes in regions radially outwards are calculated as:

$$overflowTime_i = base\_time + (i-1) \cdot step,\ i = 1, \ldots, n$$

where $overflowTime_i$ is the overflow time assigned to nodes in Region i and $step$ is the size of the increments. Through the simulations, we take 20 $units$ for $base\_time$ and 20 $units$ for $step$. Similarly, we consider the grids with four eyes and nine eyes as shown in Figure 4(b) and Figure 4(c), respectively. In order to ensure that all topologies have the same $overflowTime$ distribution, overflow times are distributed as follows: In Topology B with four eyes, the smallest circle has a radius of $1unit$, and radius increases by $\frac{1}{2}step$. Similarly, in Topology C, the smallest radius is $\frac{2}{3}units$ and radius increases by $\frac{1}{3}step$. In Topology D, nodes are first generated in the same way as in topology A and then deployed randomly in a space.

Each run of simulations lasts 100000 time units. A mobile element with a speed of one unit covers one unit length in one time unit. For each simulation presented, results are average values for 100 independent runs. We run the experiments for both PBS and MWSF algorithms on all four topologies in order to make the comparisons. The MWSF algorithm is run with weight $\alpha = 0.1$, where α is the weight of deadline, and $1-\alpha$ is the weight of distance. According to the experimental

[1]'Topologies' A, B, C, D refer to the distribution of data generation rates over the sensor field, where all nodes are placed randomly.

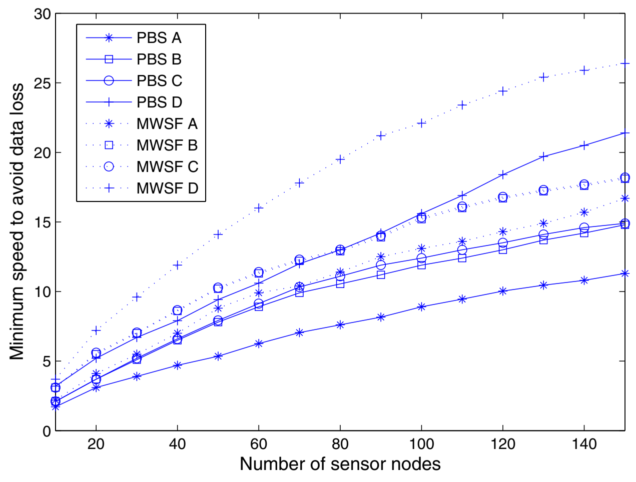


Fig. 6. The minimum required speed of the mobile element to avoid data loss for PBS and MWSF algorithm on topologies A, B, C and D with different node densities.

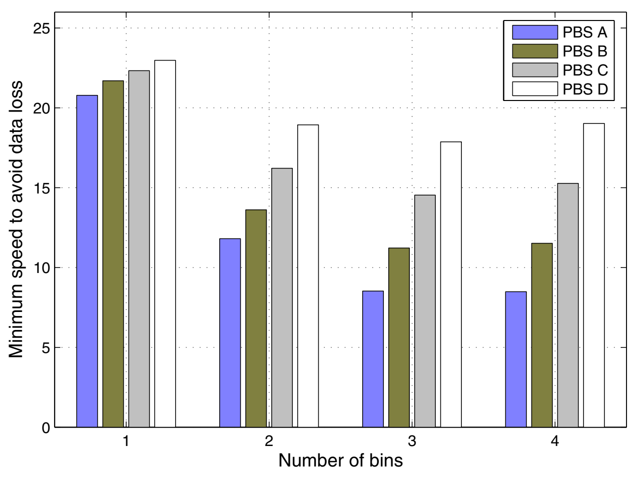


Fig. 8. Impact of number of bins and node density on minimum speed to avoid data loss with PBS algorithm.

results in [14], MWSF algorithm yields the best performance when the value of α is around 0.1. We considered PBS with the default bin number $M = 3$ unless specified otherwise.

B. Impact of the Speed of the Mobile Element

Figures 5(a) and 5(b) show data loss occurrence rates and data loss rates for simulations run with both PBS and MWSF algorithms on the four topologies A, B, C, and D. We have collected experimental results for different speeds of the mobile element ranging from 0 to 20. Loss rates range from 0 to 1, which correspond to the no loss and complete loss cases, respectively. In both figures, loss rates decrease with the increased speed. Furthermore, the performances on topology A, B, C, and D are in a decreasing order. This behavior is expected since the ME covers larger distances when nodes with similar overflow times are not located closely together. The simulation results shown in Fig. 5(a) and Fig. 5(b) also match in the sense that the data loss occurrence rate and data loss rate drop to zero at the same ME speeds. The loss rate of PBS scheduling algorithm decreases at a higher rate as the speed increases. We can conclude that our PBS algorithm is more effective in terms of reducing the loss rate, and provides a better performance than MWSF algorithm in general.

C. Impact of Node Density

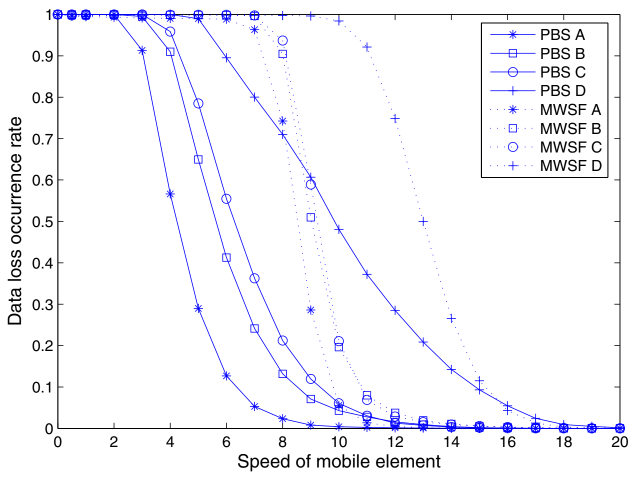
In this section, the impact of the node density is evaluated to observe the scalability of PBS algorithm. In this simulation the number of nodes deployed on a 100×100 area is varied between 10 and 150. With the increasing node density, we measure the minimum speed of mobile elements to avoid data loss. Figure 6 shows the results of running this simulation on four topologies. The minimum required speed increases with the increasing node density. This is expected because when the node density increases, the path length in a supercycle increases, as well. This leads to increased minimum speed of mobile elements to avoid buffer overflows. With the same node density, the minimum required speed for Topology A is lower than B, which is lower than C. Minimum required speed for Topology D is the highest. This behavior also matches previous results in Section IV.B. The path the mobile element covers in a supercycle is longer when the eye number is larger. The minimum required speeds in Fig.6 also match results in Fig. 5(a) and Fig. 5(b) which is evaluated when node density is 150.

D. Impact of Number of Bins

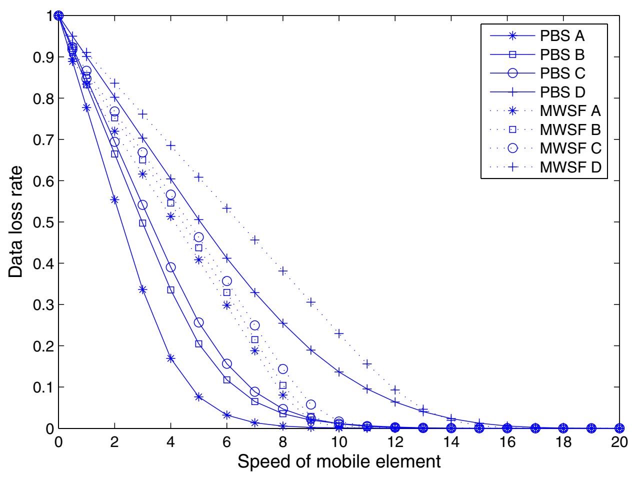
In this section, we study how the number of bins M affects the performance of the PBS algorithm. $M = 1$ corresponds to the case where all the nodes are contained in one bin and are visited every cycle. No geographical partitioning is used in this case. When the bin number is larger than one, nodes are first partitioned according to overflow times and then partitioned geographically. According to Fig. 7, in lower speed regions, small number of bins delivers lower loss rates. In higher speed regions, higher bin numbers are favored. This is due to the fact that the TSP path in the one bin case traverses all nodes, favoring nodes with large overflow times and sacrificing performance in low overflow time nodes. The effect of nodes with low overflow times will dominate the overall performance when the ME speed is not very low.

It can also be observed that a large number of bins provides much smaller minimum speed to avoid buffer overflows. As an example, in Fig. 7(d), when the data loss rate of the four bins case drops to zero, the curve of one bin case still asymptotically approaches x axis. Scheduling with small bin numbers sacrifices performance on nodes with low overflow time severely. Therefore, although the performance of scheduling with one bin scheme has smaller data loss when mobile element moves at lower speed, its advantage disappears for moderate and high speeds.

Fig. 8 shows the minimum required speed for simulations run with the PBS algorithm on four topologies with bin

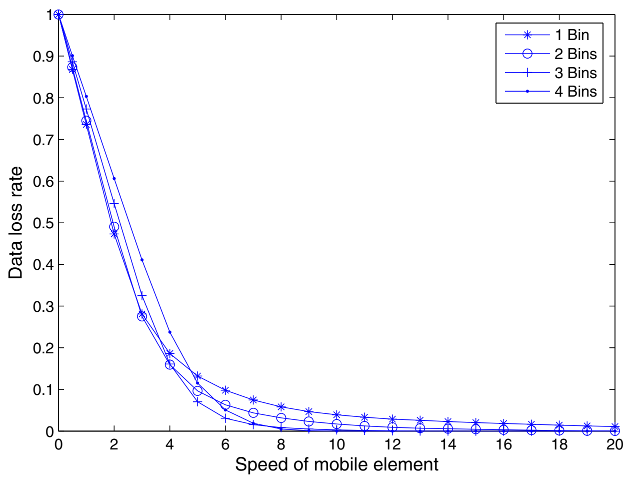


(a) Fraction of nodes which miss deadlines

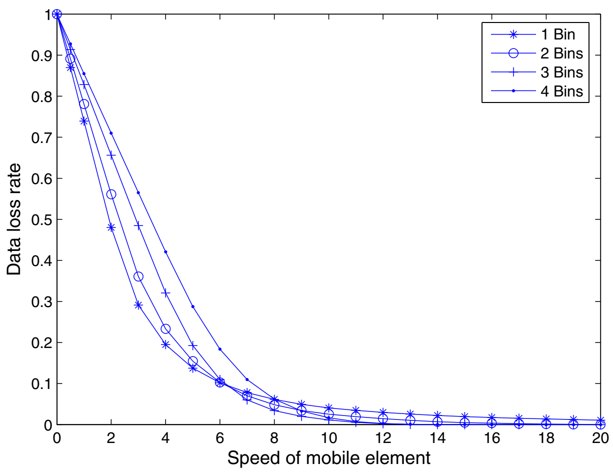


(b) Fraction of data lost to data generated

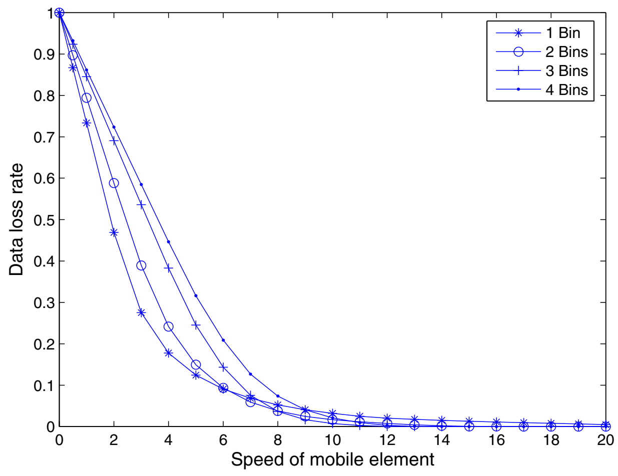
Fig. 5. Sensor loss rate under different mobile element speed for Partitioning-Based Scheduling Algorithm and Minimum Weighted Sum First on topologies A, B, C and D.



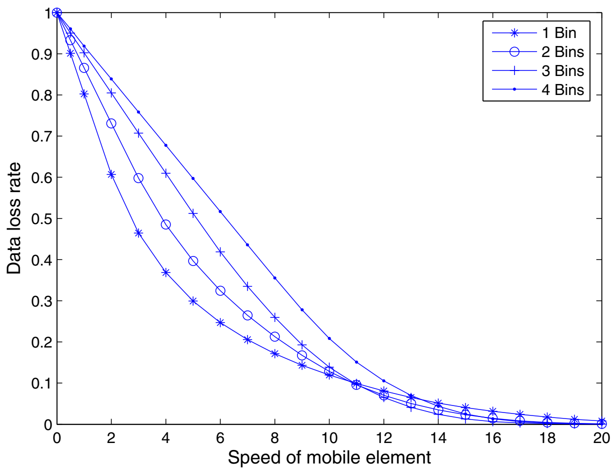
(a) Experiment on topology A with one eye



(b) Experiment on topology B with four eyes



(c) Experiment on topology C with nine eyes



(d) Experiment on uniform distribution

Fig. 7. Impact of number of bins in PBS Algorithm on data loss rate on topologies A, B, C and D.

number ranging from one to four. With same number of bins, the minimum required speed of mobile element increases with the increased eye number. Furthermore, minimum required speed decreases when number of bins increases on same topology.

E. Sensor Visit Predictability

In Section III, we claim that our PBS algorithm provides periodic supercycle scheduling. Furthermore, cycles in a supercycle are almost periodic, as well. To analyze this periodicity, we consider the variation of the duration between visits to a given sensor node. The standard deviation of inter-visit durations is used to measure the predictability of visits to sensor nodes, which shows the periodical feature of our scheduling scheme. We first observe the predictability of sensor node visit times as a function of different data generation rates. We focus on the standard deviations of inter-visit durations and group the results according to overflow times. Fig. 9(a) shows the results of running PBS and MWSF algorithms on the four Topologies A, B, C and D. We choose 6 units as the speed of the mobile element, where buffer overflows still occur. As shown in Fig. 9(a), the standard deviation of inter-visit times for PBS is much smaller than MWSF for all overflow time values. For Topology A with one eye, while MWSF sacrifices the performance in data generation rate nodes, PBS provides very high predictability in these areas.

Note that the standard deviation of inter-visit times of PBS decreases as the overflow time of sensor nodes increases. Especially for nodes with very high overflow times, the standard deviation is nearly zero. This is due to the fact that the nodes with high overflow times mostly belong to the last bin, which is visited once every supercycle. Since the period of the supercycle is always the same, they are visited exactly periodically. Thus, the standard deviation of inter-visit times for these nodes is zero.

To observe the impact of the node density on sensor visit predictability, we sampled three set of nodes with overflow times A1=150, A2=250 and A3=350. The number of nodes in the network ranges from 10 to 150, with an increment of 10. In Fig. 9(b), the six curves are grouped in two: The upper three for MWSF algorithm and the lower three for our PBS algorithm. It can be observed that our PBS algorithm have much smaller standard deviation for inter-visit duration, which shows better predictability. Moreover, although both MWSF and PBS have a decreasing predictability as sensor node density increases, PBS shows better scalability. Another observation is that the standard deviation values increase for the MWSF algorithm as the overflow times increase for all sensor node densities. For our PBS algorithm, the opposite behavior is observed: As the node density increases, the standard deviation decreases.

V. Conclusion and Future Work

Using a controlled mobile element is a promising approach to collect data from sparsely deployed sensor nodes. The sensor nodes may have different data generation rates, which leads to the Mobile Element Scheduling Problem. In this paper, we propose a Partitioning-Based Scheduling (PBS) algorithm to address this problem. We compare our algorithm with Minimum Weighted Sum First algorithm and showed that our PBS algorithm provides higher performance in terms of decreasing loss rate, reducing the minimum required speed, and providing high predictability. Our future work includes investigation of methods to utilize more than one mobile element for data collection and to cater to the needs of urgent real-time communication events.

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in *Proceedings of ACM Wireless Sensor Networks and Applications Workshop (WSNA 2002)*, 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor network: a survey,” *Computer Networks(Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, March 2002.

[3] M. Vahdat and D. Becker, “Epidemic routing for partially connected ad hoc networks,” *Technical report, Duke University*, 2000.

[4] Q. Li and D. Rus, “Sending messages to mobile users in disconnected ad-hoc wireless networks,” in *MobiCom ’00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 44–55, ACM Press, 2000.

[5] M. Grossglauser and D. N. C. Tse, “Mobility increases the capacity of ad hoc wireless networks,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 477–486, 2002.

[6] P. Juang and et. al., “Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet,” in *Int’l. Conf. on Architectural support for programming languages and operating systems*, 2002.

[7] “Manatee website, http://distlab.dk/manatee/,” 2002.

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

[9] R. Shah, S. Roy, S. Jain, and W.Brunette, “Data mules: Modeling a three-tier architecture for sparse sensor networks,” in *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[10] W. Zhao and M. H. Ammar, “Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks,” in *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS’03)*, 2003.

[11] W. Zhao, M. Ammar, and E. Zegura, “A message ferrying approach for data delivery in sparse mobile ad hoc networks,” in *MobiHoc ’04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pp. 187–198, ACM Press, 2004.

[12] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin, “Intelligent fluid infrastructure for embedded networks,” in *MobiSYS ’04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pp. 111–124, ACM Press, 2004.

[13] I. F. Akyildiz and I. H. Kasimoglu, “Wireless sensor and actor networks: Research challenges,” *Ad Hoc Networks Journal (Elsevier)*, vol. 2, pp. 351–367, October 2004.

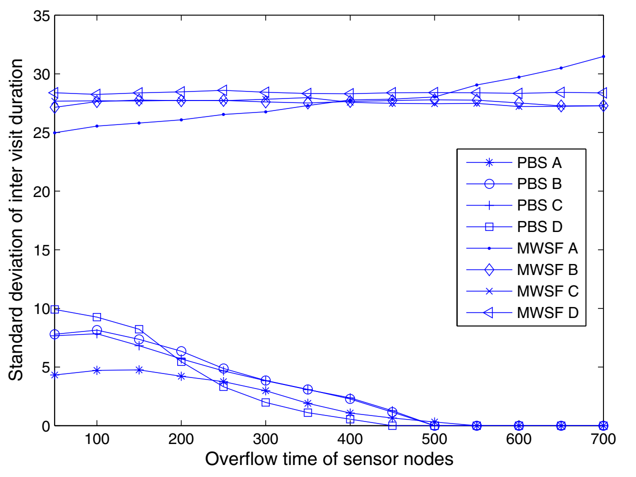
[14] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, “Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines,” in *25th IEEE International Real-Time Systems Symposium (RTSS’04)*, pp. 296–305, 2004.

[15] B.L.Golden, L. Levy, and R. Vohra, “The orienteering problem,” *Naval Research Logistics*, vol. 34, pp. 307–318, 1987.

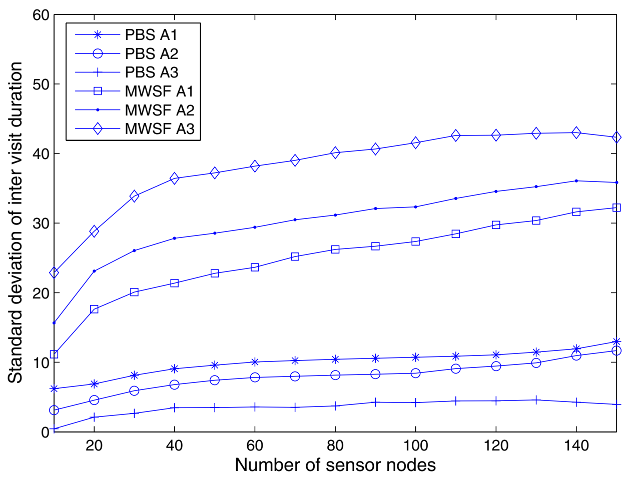
[16] R. Bar-Yehuda, G. Even, and S. M. Shahar, “On approximating a geometric prize-collecting traveling salesman problem with time windows,” *Journal of Algorithm*, vol. 55, pp. 76–92, April 2005.

[17] P. Toth and E. C. Vigo, “The vehicle routing problem,” *Society for Industrial and Applied Mathematics (SIAM)*, 2001.

[18] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, “Approximation algorithms for deadline-tsp and vehicle routing with time-windows,” in *STOC ’04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 166–174, ACM Press, 2004.



(a) Standard deviation of inter-visit duration at different sample regions



(b) Standard deviation of inter-visit duration at different node density. A1,A2,A3 are three group of nodes with overflow time of 150, 250,350, respectively

Fig. 9. Comparison of predictability between Partitioning-Based Scheduling Algorithm and Minimum Weighted Sum First.

[19] M. Solomon, "Algorithms for the vehicle routing and scheduling problem with time window constraints," in *Operations Research*, vol. 35, 1985.

[20] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[21] J.Bentley, "Fast algorithms for geometric traveling salesman problem," *ORSA Journal on Computing*, vol. 4, pp. 387–411, 1992.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and et.al, *Introduction to Algorithms*. The MIT Press, 2002.