11

Effective Heuristics for Route Construction of Mobile Data Collectors

Samer Hanoun & Saeid Nahavandi
Centre for Intelligent Systems Research Deakin University
Australia

1. Introduction

Wireless sensor networks (WSN) are composed of large numbers of small sensing self-powered nodes which are densely deployed either inside the phenomenon or very close to it (Akyildiz et al., 2002; Culler et al., 2004). The capabilities of these inexpensive, low-power communication devices ensure serving in a wide range of application domains (Chong and Kumar, 2003; Haenggi, 2004). While their potential benefits are clear, a number of open problems must be solved in order for wireless sensor networks to become viable in practice. These problems include issues related to deployment, security, calibration, failure detection and power management.

Recently, significant advances have been accomplished in the field of mobile robotics (Engelberger, 1999), and robots have become increasingly more feasible in practical system design. Therefore, a number of problems with wireless sensor networks can be solved by including a mobile robot as an integral part of the system. Specifically, the robot can be used to deploy and calibrate sensors, detect and react to sensor failure, deliver power to sensors, and otherwise maintain the overall health of the wireless sensor network.

Energy consumption is one of the most important requirements for many WSN applications. Since energy resources are scarce and battery replacement is not an option for networks with thousands of physically embedded sensor nodes, energy optimization for individual nodes is required as well as the entire network. In static wireless sensor networks, it is observed that, as data traffic must be concentrated towards the sink (base station), the nodes around that sink have to forward data for other nodes whose number can be very large; this problem always exists, regardless of what energy conserving protocol is used for data transmission. As a result, those bottleneck nodes around the sink deplete their batteries much faster than other nodes and, therefore, their lifetime upper bounds the lifetime of the whole network.

Mobile collectors (mobile robots) are utilized to act as mechanical data carriers taking advantage of mobility capacity (Grossglauser and Tse, 2002) and physically approaching the sensors for collecting their data using single hop communication. This approach trades data delivery latency for the reduction of energy consumption of sensors; however, it shows remarkable enhancement in the network lifetime. Still, the data delivery latency depends mainly on the mobility regime applied by the collector.

In random mobility regimes (Burrell et al., 2004; Jain et al., 2006; Shah et al., 2003), random moving humans and animals act as "data mules", collect data opportunistically from sensor nodes when entering their communication ranges. A grid type topology is assumed for the network deployment with sensors located at the grid intersection points, where the mobile entities (data mules) can move in any of the four directions with equal probability (Shah et al., 2003) or move up and down the rows (Burrell et al., 2004). The random mobility regime shows improved data capacity (Grossglauser and Tse, 2002), however, in all cases, the worst-case latency of data delivery cannot be bounded. This unbounded latency may lead to excessive data caching at mobile entities, resulting in buffer overflows and data in transit may have to be dropped before being delivered to the destination, making it harder to provide transport layer reliability.

In predictable mobility regimes (Baruah et al., 2004; Chakrabarti et al., 2003), vehicles moving on a predesigned path collect sensors data when they move near them. The sensor nodes learn the times at which they have connectivity with the vehicle, and wake up accordingly to transfer their data. The trajectory of the mobile vehicle is known to the sensor nodes, which helps the sensors to save energy by sleeping until the predicted time of data transfer comes. A queuing model (Chakrabarti et al., 2003) is introduced to accurately model the data collection process. Using this queuing system model, the success rate of data collection and power consumption are analysed. Also, applying reinforcement learning to locate the vehicle efficiently at any point of time along its path is studied in (Baruah et al., 2004). The predictable mobility regime provides efficient solutions for saving the sensors energy consumption; however, it lacks flexibility and scalability as the sensors have to relearn the new data transfer times if the vehicle's path change and the need for redesigning the vehicle's path when transplanted to other networks.

In contrast, the controlled mobility regime adapts the motion strategy of the mobile collector according to the network runtime conditions to balance the sensors energy consumption and the data delivery latency. The message ferrying approach introduced in (Kansal et al., 2004; Zhao and Ammar, 2003; Zhao et al., 2004, 2005) controls the motion of a mobile relay to route messages between nodes in sparse networks. The idea is studied on networks with stationary sensor nodes (Kansal et al., 2004; Zhao and Ammar, 2003) and networks with mobile nodes (Zhao et al., 2004, 2005). The message ferry moves proactively to meet nodes wishing to send or receive packets and most communication involves short range radios for avoiding excessive energy consumption. The approach proves its strength in improving data delivery and energy efficiency; however, the collector acts only as a mobile relay between sparse sensor nodes.

Controlling the mobile collector motion for efficient data collection is presented in (Gu et al., 2005; Ngai et al., 2007; Somasundara et al., 2004; Tirta et al., 2006). The motion strategy of the mobile collector (element) is formulated as a scheduling problem based on knowing in advance the sensors sampling intervals and the rate by which the events in the environment occur. Offline solutions are provided in (Gu et al., 2005; Ngai et al., 2007; Tirta et al., 2006) based on having advance knowledge regarding the deployment locations of the sensors, their data generation rates, and buffer sizes. The solutions presented minimize the data latency by optimizing the collector inter-arrival times while using single hop communication for data transfer to minimize the sensor's energy consumption, remains that all operate offline and maladaptive to the network operational conditions. An online solution is presented in (Somasundara et al., 2004) that overcomes some of the former

problems. A mobile data collector is scheduled in real time to visit sensors such that no sensor buffer overflow occurs. The algorithm considers buffer overflow deadlines as well as distances between nodes in determining the visiting schedule. The new deadline for the node's future visit is updated, once it is visited and depends mainly on knowing the node's buffer size and sensing rate to compute its next overflow deadline. The solution works online; still, there is a requirement on having full knowledge about the operational parameters of the sensor node.

This work is concerned with the controlled mobility regime. The former presents the solutions lying in this domain of research; however, more research remains and requires deep attention and addressing. The following are among these research points:

- Offline solutions (Gu et al., 2005; Ngai et al., 2007; Tirta et al., 2006) produce optimized results but depend mainly on having in advance full knowledge about the network operational parameters, which may not be always feasible. Solutions working online and in real-time are needed to accommodate changes in the network operation.
- The time required for the sensor's buffer to become full (overflow time) is assumed to be fixed and constant with time (Somasundara et al., 2004, 2007). Intelligent algorithms running locally on the sensor, performing local fusion and compression, impact the sensor's overflow time as this depends mainly on the data sensed and the quality threshold measures applied by the fusion and compression algorithms. Additionally, the overflow time can change with time according to the dynamics in the phenomenon which the sensors are sensing. Therefore, the change in the buffer overflow time with time has to be encountered in the solution.
- In scenarios where sensor nodes form clusters (Younis and Fahmy, 2004), the mobile data collector can visit the centroids of these clusters (cluster heads). An early arrival would force the mobile collector to wait until enough data is aggregated at the cluster head, and a late arrival may cause missing some of the aggregated data. Adaptive schedule considering the runtime conditions of the sensors is required to handle such cases.
- In event-driven sensor network applications, the dynamics in the phenomenon which the sensor nodes are sensing changes with time and the rate and times the events occur are not known ahead and even unpredictable. Moreover, in query-driven applications, the network operator may query the network to perform certain tasks, in irregular patterns. This requires that the motion strategy of the mobile collector to act based on these unpredictable situations.
- Deploying thousands of sensors manually and setting their locations is not a feasible process and in most situations the sensors are deployed randomly in the environment. Moreover, some sensors may be carried by low mobile platforms and their locations change from time to time. These factors impact the mobile collector schedule and should be adopted properly. The network performance is considered to be dependent only on the sensor nodes in the network. However, the embedded capabilities of the mobile collector (i.e speed) can be utilized to enhance the network performance. Also, using multiple mobiles with appropriate cooperation strategies can add more benefit to the data collection operation. Adaptive decentralized solutions are required to take advantage of these capabilities.

Accordingly, in this work a dynamic scheme for data collection in wireless sensor networks is presented. The scheme aims to consider most of the research points described above. First of all, the mobile element utilized for collecting the sensors' data buffers works online and constructs its collection route dynamically according to the collection requests received from the sensors requiring the collection service. Secondly, the mobile collector does not require any prior knowledge about the sensors deployment positions, their data generation rates and buffer size. This is achieved by making the sensor node disseminates its deployment position in the collection request sent to the mobile collector. Additionally, when the sensor's buffer becomes full and after disseminating the collection request, it sleeps and waits the arrival of the collector, which relieves the collector the requirement of knowing ahead the sensor's data generation rate and buffer size to schedule its collection before the buffer overflow time comes. Therefore, each sensor in the network can operate with different sensing frequencies and its buffer size may vary according to the type of application the network is serving (i.e. event-driven and query-driven sensor network applications). However, this achieves no dependency between the sensor network and the mobile collector; it requires optimizing the sensor sleeping time waiting the arrival of the collector to avoid missing changes in the observed phenomenon and to increase the network activity.
Experimental testing and simulations are performed to evaluate the presented heuristics with real world data parameters. Performance metrics are selected for comparing the proposed heuristics. The results show that the presented heuristics are capable of reducing the data collection time and ensure high network activity along its operational lifetime. Also, online and dynamic solutions can provide high flexibility and scalability and lose coupling between the sensor network and the mobile collectors can be achieved.

2. Related Work

This section presents some relevant literature in routing and scheduling theory.

2.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is one of the classical challenging combinatorial optimization problems. The objective of the TSP is to minimize the total distance travelled by visiting all locations once and only once and then returning to the depot point. A common application of the TSP is the movement of people, equipment and vehicles around tours of duty to minimize the total travelling cost. For example, in a school bus routing problem, it is required to schedule the bus to pick up waiting students from the pre-specified locations. Post routing is another application of the TSP. The postman problem is modelled as traversing a given set of streets in a city, rather than visiting a set of specified locations.
Besides the above mentioned applications, some other seemingly unrelated problems are solved by formulating them as the TSP. The genome sequencing problem occurs in the field of bio-engineering. The aim of this problem is to find the genome sequence based on the markers that serve as landmarks for the genome maps. The drilling problem is another application of the TSP with the objective of minimizing the total travel time of the drill. In the electronic industry, the printed circuit board normally has a very large number of holes used for mounting components or integrated chips. These holes are typically drilled by

automated drilling machines that move between specified locations to drill a hole one after another. Therefore, the locations in the drilling problem correspond to the cities in the TSP. The applications of the TSP are not limited to the examples described above. A detailed review of the applications of the TSP can be found in (Lawler et al., 1990).
It has been proved that TSP is NP-hard in (Garey and Johnson, 1979), which implies that a polynomial bounded exact algorithm for TSP is unlikely to exist. Several heuristic based tour construction algorithms were proposed for solving the TSP. Among these, the nearest neighbour procedure (Rosenkrantz et al., 1977), the Clarke and Wright savings' algorithm (Clarke and Wright, 1964), the partitioning approach (Karp, 1977) and the minimal spanning tree approach (Christofides, 1976). Comprehensive review of the techniques developed for the TSP can be found in (Bodin et al., 1983; Laporte, 1992).
It is important to clearly outline the differences between our problem and the conventional Travelling Salesman Problem. In TSP, the goal is to find a minimum cost tour that visits each node exactly once. However in our problem, a sensor node may need to be visited multiple times before all other nodes are visited depending on the rate of its data generation. In addition, the mobile collector downloads the data once it is in the communication range of the sensor node and need not to be exactly at the sensor location. Also the transfer of the data buffer can be done while the mobile collector is in motion.

2.2 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) (Toth and Vigo, 2001) can be described as the problem of designing least cost routes from one depot to a set of geographically scattered points. There are nodes each has a service request in terms of demand for a certain quantity of goods, and vehicles available for servicing these requests, which are stationed at the depot. Each vehicle has a certain capacity in terms of quantity of goods it can carry. The goal is to find the number of vehicles and the sequence of nodes each vehicle has to visit such that sum of the distances travelled by each vehicle is minimum, subject to the following constraints:

- Each vehicle starts and ends at the depot.
- Each node is serviced exactly by one vehicle.
- Capacity constraints are met, i.e. sum of demands from the nodes on a vehicle's list does not exceed the vehicle's capacity.

The Traveling Salesman Problem (TSP) mentioned earlier is a special case of VRP, where there is a single vehicle that visits the nodes, and there are no capacity constraints.
There are many variants to the basic VRP mentioned above. The VRP with time Windows (VRPTW) (Solomon, 1987) has an added constraint in which there is a time window within which each node has to be visited. The VRPTW can be defined as follows. Let G = (*V, E*) be a connected digraph consisting of a set of $n + 1$ nodes, each of which can be serviced only within a specified time interval or time window, and a set E of arcs with non-negative weights d_{ij} and with associated travel times, t_{ij}. The travel time t_{ij} includes a service time at node i, and a vehicle is permitted to arrive before the opening of the time window, and wait at no cost until service becomes possible, but it is not permitted to arrive after the latest time window. Node 0 represents the depot. Each node i, apart from the depot, imposes a service requirement q_i that can be a delivery from, or a pickup for the depot. The main objective is

to find the minimum number of tours, K^*, for a set of identical vehicles such that each node is reached within its time window and the accumulated service up to any node does not exceed a positive number Q (vehicle capacity). A secondary objective is often either to minimize the total distance travelled or the duration of the routes. All problem parameters, such as customer demands and time windows, are assumed to be known with certainty. Moreover, each customer must be served by exactly one vehicle, thus prohibiting split service and multiple visits. The tours correspond to feasible routes starting and ending at the depot.

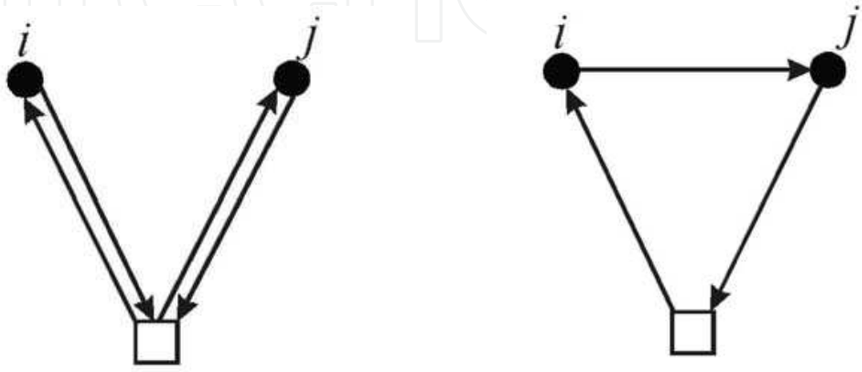


Fig. 1. The savings heuristic. In the left part, customers *i* and *j* are served by separate routes; in the right part, the routes are combined by inserting customer *j* after *i*

There is a dynamic version of VRP, known as Dynamic Vehicle Routing Problem (DVRP) (Ghiani et al., 2003). Here the information (input) is revealed to the decision maker online concurrently with the determination of routes. For instance, the VRP mentioned above is solved for the initial requests (nodes to be visited) and routes assigned to vehicles. Then, as the vehicles start their scheduled routes, new requests come, which need to be accommodated.

There are some differences between our problem and the VRPTW. First of all, some nodes may need to be visited more than once before visiting any other node once depending on its data generation rate. Secondly, the number of mobile collectors servicing the network is fixed, thus the goal is no longer to find the number of mobiles but to find the most feasible mobile collector for servicing the request. Also, the time window (i.e. the time for the data collection) is not constraint; however, the objective is to minimize the data collection time among all sensors and for all generated collection requests. Thus our problem reduces from the VRPTW to the problem of designing the route for the mobile collector to follow. This route must be optimized to provide the minimum data collection time among the sleeping sensors waiting for the arrival of the collector.

The savings method (Clarke and Wright, 1964), originally developed for the classical VRP, is probably the best-known route construction heuristic. It begins with a solution in which every customer is supplied individually by a separate route. Combining the two routes serving respectively customers *i* and *j* results in a cost savings of $S_{ij} = d_{i0} + d_{0j} - d_{ij}$. The arc *(i, j)* linking customers *i* and *j* with maximum S_{ij} is selected subject to the requirement that the combined route is feasible. With this convention, the route combination operation is

applied iteratively. In combining routes, one can simultaneously form partial routes for all vehicles or sequentially add customers to a given route until the vehicle is fully loaded. To account for both the spatial and temporal closeness of customers, a limit to the waiting time of the route is set. The savings method is illustrated in Figure 1.

The second heuristic, a time oriented nearest-neighbour, starts every route by finding an unrouted customer closest to the depot. At every subsequent iteration, the heuristic searches for the customer closest to the last customer added into the route and adds it at the end of the route. A new route is started any time the search fails to find a feasible insertion place, unless there are no more unrouted customers left. The metric used to measure the closeness of any pair of customers attempts to account for both geographical and temporal closeness of customers.

The most successful methods among the sequential insertion heuristics is called I1 (Solomon, 1987). A route is first initialized with a "seed" customer and the remaining unrouted customers are added into this route until it is full with respect to the scheduling horizon and/or capacity constraint. If unrouted customers remain, the initializations and insertion procedures are then repeated until all customers are serviced. The seed customers are selected by finding either the geographically farthest unrouted customer in relation to the depot or the unrouted customer with the lowest allowed starting time for service. After initializing the current route with a seed customer, the method uses two subsequently defined criteria $c_1(i,u,j)$ and $c_2(i,u,j)$ to select customer u for insertion between adjacent customers i and j in the current partial route.

Let $(i_0, i_1, i_2,..., i_m)$ be the current route with i_0 and i_m representing the depot. For each unrouted customer u, compute first its best feasible insertion cost on the route as:

$$c_1(i(u),u,j(u)) = \min_{p=1,\ldots,m} c_1(i_{p-1},u,i_p) \tag{1}$$

Next, the best customer u* to be inserted in the route is the one for which:

$$c_2(i(u^*),u^*,j(u^*)) = \max_u \{c_2(i(u),u,j(u)),u \text{ is unrouted and route is feasible}\} \tag{2}$$

Client u^* is then inserted into the route between $i(u^*)$ and $j(u^*)$. When no more customers with feasible insertions can be found, the method starts a new route, unless it has already routed all customers. More precisely $c_1(i,u,j)$ is calculated as:

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j), \text{ where } \alpha_1 + \alpha_2 = 1, \alpha_1 \geq 0, \alpha_2 \geq 0, \tag{3}$$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, \quad \mu \geq 0, \tag{4}$$

$$c_{12}(i,u,j) = b_{ju} - b_j, \tag{5}$$

and d_{iu}, d_{uj} and d_{ij} are distances between customers i and u, u and j and i and j respectively. Parameter μ controls the savings in distance and b_{ju} denotes the new time for service to

begin at customer j, given that u is inserted on the route and b_j is the beginning of service before insertion. The criterion $c_2(i, u, j)$ is calculated as follows:

$$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j), \quad \lambda \geq 0. \tag{6}$$

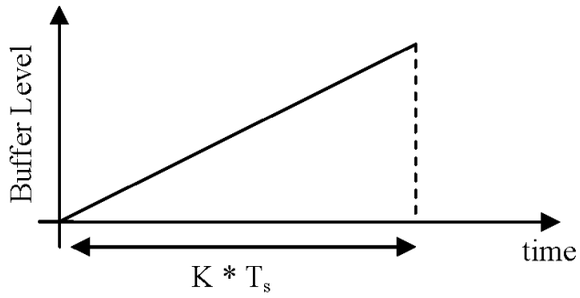
Parameter λ is used to define how much the best insertion place for an unrouted customer depends on its distance from the depot and on the other hand how much the best place depends on the extra distance and extra time required to visit the customer by the current vehicle. Other types of the insertion heuristics such as Type (I2) aims to select customers whose insertion costs minimize a measure of total route distance and time and Type (I3) accounts for the urgency of servicing a customer.

3. Problem Formulation

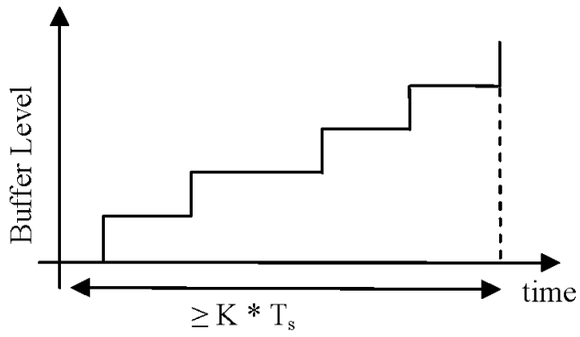
This section clearly models our problem and outlines all assumptions. A formal definition of the problem is presented to give an insight into the problem objectives. The problem is modeled as follows:

- The sensor network is modeled as a fully connected graph $G(V, E)$ of n nodes, where every edge (s_i, s_j) has an associated cost $c(i,j)$, and all the costs form a matrix $C = \left[c(i,j)\right]_{i,j=1}^{n}$.
- Sensor nodes remain stationary and are deployed uniformly at random in the sensing field where the phenomenon of interest is to be monitored. The sensing field dimension is $A = L * L$ ($m2$).
- Each sensor node has a deployment location (X, Y), buffer size of K bytes, radio communication radius R in meters and generates a sample every T_s seconds.
- The time required for the buffer to be full is $\geq K * T_s$, where the next time the buffer becomes full is unknown and differs from time to time. This is modeled by generating a random number between 0.0 and 1.0 to specify whether the current sample is qualified to be added to the buffer or not. If the random number is greater than a pre-specified threshold φ then the sample is added to the buffer, otherwise, it is discarded. This ensures that the sensor's buffer full time is variable which reflects different network operational modes (i.e. periodic sensing, event driven, query based). This also simulates the operation of data aggregation algorithms operating locally on the sensor for reducing the redundancy in the sensor measurement which alters the buffer overflow time. Figure 2 shows the buffer level accumulation with time for different models.
- Once the sensor's buffer is full, the sensor disseminates a collection request Q with fields {ID, X, Y} and goes to an idle state and sleeps waiting for the collector arrival. While idle the sensor does not sense any new samples but it relays collection requests sent by others.
- The mobile collector starts at a central position in the sensing field, has a reasonably high amount of energy that can last beyond the network lifetime, and its memory size can accommodate the data generated by the sensors during the network operational

time. The mobile collector moves with a speed $v \leq S_{max}$ and it can change its speed during the data collection operation.



(a) Fixed with time



(b) Variable with time

Fig. 2. The sensor buffer level accumulation rate

The following assumptions are made:

- At time t = 0 all the buffers of the sensor nodes start filling up.
- The actual data transfer time from the sensor node to the mobile element is negligible.

The **Mobile Collector Route Design (MCRD)** problem is the problem of finding a sequence of visits to nodes requesting the collection of their data buffers such that to minimize the collection time required. Once a node is visited, its buffer is transferred to the mobile collector internal memory and its normal sensing operation is resumed. This is mathematically formulated as follows:
N: is the set of sensor nodes in the network, where $N = \{s_1, s_2, ..., s_n\}$,
S: is the set of sensor sites, i.e. deployment locations of the sensor nodes, where $S = \{(x_1,y_1),...(x_n,y_n)\}$,
d_{ij}: Euclidean distance (meters) between sensor sites, where $i,j \in S$,
R: is the set of requests to be serviced, where $R = \{r_1,r_2,...,r_m\}$,
A_k: is the time taken for servicing request k, where $k \in R$, i.e. time taken for collecting request k generated by sensor i, where $k \in R$, $i \in N$,
v: is the moving speed of the mobile collector in m/sec,

Mobile Collector Route Design (MCRD):

$$\min T = \sum_{\forall k} A_k \tag{7}$$

where T is the overall route period, and each A_k is computed based on the distance cost between requests k and $(k+1)$ and the speed selected by the mobile collector for servicing this request.
The problem of finding the optimum order of arranging the requests to form the least cost route is *NP*-complete. The way is to simply enumerate each possible route, and pick the one with minimum total cost. Having m requests to service, gives $(m - 1)!$ possible route. This leads to an $O(m!)$ algorithm to produce the optimum route, which is not efficient to use.

4. Heuristic Based Algorithms

The problem of designing the mobile collector route is NP-complete. This section presents some heuristic based algorithms for designing the route to be followed by the mobile collector for collecting the data buffers of the sensor nodes requesting the collection service.
A heuristic algorithm provides feasible solution to an optimization problem, which may or may not be optimal. Good heuristics give solutions that are close to the optimal solution, and usually are efficient in terms of theoretical or practical running time. A desirable property of a heuristic is that the worst-case quality of the solution provided can be guaranteed. For example, some heuristics guarantee that the solution they identify is at worst η-optimal; that is, if the optimal cost is C*, the heuristic finds a solution no greater than ηC* for minimization problems. In our problem, the route constructed by ordering the requests based on a timely manner according to their arrival is considered as an upper bound on the solution provided by the heuristic.

4.1 Minimum Spanning Tree Route Construction

In the Traveling Salesman Problem (TSP) (Lawler et al., 1990) context, a vertex can be interpreted as a sensor and the edge weight can be the distance between the sensors or the time of travel between any two sensors. With these notations, the mobile collector waits until a certain number of requests m are received, and then the route construction is interpreted as the problem of finding a minimum-cost tour that visits each of the sleeping sensors exactly once and returns to the starting point (center of the sensing field). The route objective is expressed as follows:

$$\min \sum_{i=1}^{m} c(R(i), R(i+1)) \tag{8}$$

where $R(i)$ is the i^{th} request on the route; and $c(R(i), R(i + 1))$ is the distance cost from request i to request $(i + 1)$.
Defining $x_{ij} = 1$, if the edge from request i to request j is on the route, $i,j \in \{1,..., m\}$ and 0 otherwise, maps the objective to:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} * \chi_{ij} \tag{9}$$

subject to:

$$\sum_{j=1}^{m} \chi_{ij} = 1, \forall i \tag{10}$$

$$\sum_{i=1}^{m} \chi_{ij} = 1, \forall j \tag{11}$$

$$y_1 = 1, \tag{12}$$

$$2 \le y_i \le m, \forall i \ne 1 \tag{13}$$

$$y_i - y_j + 1 \le (m-1)(1-x_{ij}), \forall i \ne 1, \forall j \ne 1 \tag{14}$$

$$x_{ij} \in \{0,1\}, \forall i, j \tag{15}$$

where
m: number of collection requests; i: current request, $i \in \{1,..., m\}$; j: next request, $j \in \{1,..., m\}$; and y_i : extra variable to exclude sub tours, $i \in \{1,..., n\}$.
Constraints (10) and (11) are called the degree constraints, which enforce that every sensor reached is left exactly once. Constraints (12), (13) and (14) are subtour elimination constraints, which prohibit the formation of subtours having less than m vertices.
It is necessary to show that a route with a Hamiltonian cycle has an overall time period $A_{\max}$ less than that without. Let r_0, r_1, r_2,... ,r_m be the requests on a route, where r_0 is the center of the sensing field from which the collector route starts and ends. c_{01}, c_{12}, c_{23},..., c_{m0} are the cost between consecutive requests. The period of the route is $T = \sum_{\forall c} c / v$, where v is the mobile collector speed. If the route is a Hamiltonian cycle, $A_{\max}$ of any request location r is always equal to T. On the contrary, considering a route that is not a Hamiltonian cycle, the mobile collector will need to move from one end to another end, then back to the beginning request location to complete a cycle. The $A_{\max}$ of the request locations at two ends will then be $2T$, which is much longer then T.
Algorithm 1 presents the steps followed by the mobile collector for constructing its route. The minimum spanning tree is computed based, as presented by Algorithm 2, on partitioning the graph of the collection requests into two independent subsets, S and T, such that, $S \cup T = N$ and $S \cap T = \varnothing$ and the set (S, T) is given by all arcs (i,j), where $i \in S$, $j \in T$, or $i \in T$, $j \in S$.

Algorithm 1 Minimum Spanning Tree Route Construction Algorithm (MST-R)
Input:
m collection requests, each with fields {*ID, X, Y*}
Body:
1. Build a fully connected graph *G* for the m received collection requests.
2. Add the center of the sensing field to *G*.
3. Compute a minimum spanning tree *T* for *G*.
4. Generate using Depth First Search (DFS) a list *L* of vertices on *T*.
5. Construct the Hamiltonian cycle H for visiting the vertices *L*.
6. Follow the Hamiltonian cycle *H* as the constructed route.

Algorithm 2 Minimum Spanning Tree Algorithm
Initialization:
$MST = \varnothing$
$S = \{1\}$, $T = N \setminus S$ (the set S contains node 1, while the set T contains all nodes except node 1)
Iterations:
while $| MST | < m$, do
1- Find the arc (i, j) in (S, T) with minimum cost c_{ij}
2- Add (i, j) to MST
3- Remove j from T and add j to S

It is important to note that the list generated by the Depth-First Search (DFS) will produce each node twice as the DFS will visit each edge twice, once going down the tree when exploring it and once going up after exploring the entire sub tree. For example, in the depth-first search of Figure 3, the vertices in order 1-*2-1-3-5-8-59-5-3-6-3-1-4-7-10-7-11-7-4-1* are visited, thus using every tree edge exactly twice. Therefore, this tour has weight twice that of the minimum spanning tree, and hence at most twice optimal. To remove the extra vertices, at each step a shortest path to the next unvisited vertex is taken. The shortcut tour for the tree in Figure 3 is *1-2-3-5-8-9-6-4-7-10-11-1*. This ensures that the tour gets shorter and within weight twice that of optimal.

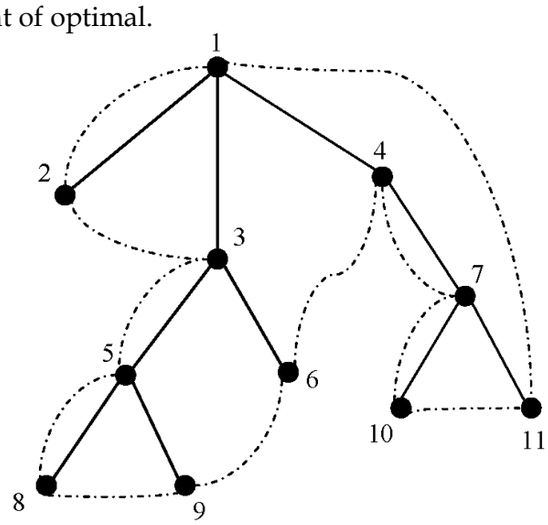


Fig. 3. A depth-first traversal of a spanning tree, with the shortcut tour

4.2 Dynamic Insertion Route Construction

The minimum spanning tree route construction algorithm considers only the distance cost when constructing the collection route. Instead, an algorithm can be designed which gives weights to the distance cost and the time which the sensor remains sleeping until the collection is done. Based on Equation 3, each collection request is inserted dynamically in its minimum insertion position on the collection route. Let r_1 ,r_2,... ,r_m be the requests to be collected by the mobile collector on its route R, and, C_{jk} is the extra cost of placing request k after request j on R, then it is required to:

$$\min \sum_{j=1}^{m} C_{jk} * X_{jk} \tag{16}$$

where X_{jk} = 1, if the placement of request k follows request j; 0 otherwise, and

$$\sum_{j=1}^{m} X_{jk} = 1 \qquad \text{for any k} \tag{17}$$

Algorithm 3 presents the procedure of finding the best insertion point on the mobile collector route for the current received collection request. The sensor sleeping time ST based on the point of insertion along the collection route is not computed absolute, but relative to the current time. Table 1 shows the effect of different α values. To illustrate the contents of the table, consider the scenario presented in Figure 4. Suppose the mobile collector just serviced node A and the next node on its route is node B. Before initiating the service to node B, a request of collection arrives from node C. The two nodes B and C have distance costs 5 and 15 respectively, and the sleeping time of node B is 10 time units. The insertion position of node C is either before node B or after node B. When α ≤ 0.5, node C is inserted before node B and when α > 0.5, node C will be inserted after node B. When α = 0, the mobile collector will always favour newly arriving requests, which can result in high delays in servicing older requests, causing those nodes to wait long times until they are serviced.

Algorithm 3 Dynamic Insertion Route Construction Algorithm

Input:

Request (*r*) with fields {*ID, X, Y*}

Define:

R: current route consisting of *k* requests; mp: mobile collector position;

ST : sensor sleeping time according to the insertion point;

r_f , r_l: first and last requests in R;

$C_b^a = \sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2}$

Initialize:

Cost[1..(k+1)] = 0

Body:

IF (R = ∅) THEN *R* = *r*

ELSE

$ST_1 = C_{m_p}^{r_f}$

Cost[1] = $\alpha * (C_r^{m_p} + C_{r_f}^{r} - C_{r_f}^{m_p}) + (1-\alpha) * ST_1$

Repeat for $i = 2...k$

$ST_i = ST_i + C_{k_i}^{k_{i-1}}$

Cost[i] = $\alpha * (C_r^{k_i} + C_{k_{i-1}}^{r} - C_{k_i}^{k_{i-1}}) + (1-\alpha) * ST_i$

$ST_{k+1} = ST_k + C_{r_l}^{r}$

Cost[k+1] = $\alpha * C_{r_l}^{r} + (1-\alpha) * ST_{k+1}$

Set j = index of minimum in Cost [1..(k+1)]
Insert r in R at position j
END

4.3 Discussion

The complexity of the MST-Route construction algorithm is upper bounded by the complexity of computing the MST. The MST algorithm requires (m-1) iterations in which a single arc is identified in (S, T) and moved into the MST. Each iteration, scans each arc to determine whether it is in (S, T), and if so, if it has cost less than the current minimum. Such an implementation would scan n arcs each iteration, for a total complexity of O(mn). The DFS constructs the list of nodes in an O(m). Thus the MST-R requires O(m2) running time. The DI-Route construction requires an O(m) to compute the cost of insertion along the current route and an O(log n) to find the minimum insertion point, resulting in an O(m) running time complexity. It should be noted that the DI-R construction works totally online and adapts the current route according to each received request, while the MST-R construction groups m received requests to construct the collection route.

$\alpha \in [0,1]$	Result
1	Insertion point is determined only by the distance cost.
0	Insertion point is determined only by the sensor sleeping time.
> 0.5	Distance cost contributes higher in the insertion point.
≤ 0.5	Sensor sleeping time contributes higher in the insertion point.

Table 1: Effect of a on the insertion point

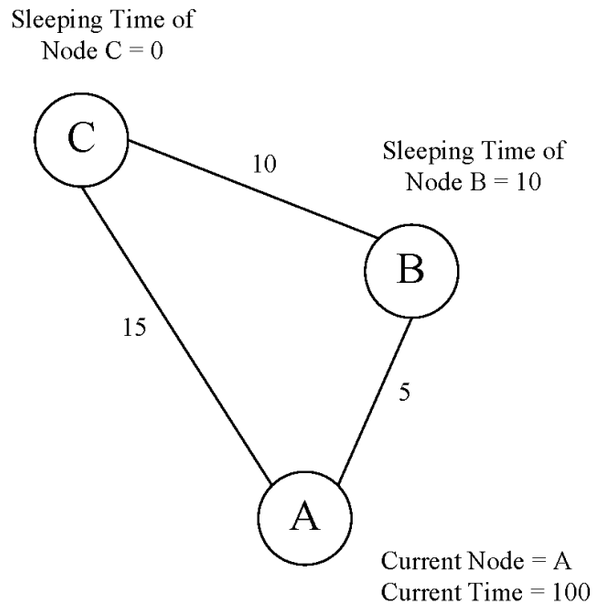


Fig. 4. An example to illustrate the effect of a on the mobile collector route

5. Experimental Methodology and Results

The previous section described some heuristics for constructing the route of the mobile collector. This section presents the evaluation of these algorithms through simulation.

5.1 Methodology

As the parameter space is huge, some parameters are fixed as follows:

- Sensing Field: A square area of dimension 100 x 100 m^2 is considered for the sensors deployment. A number of sensor nodes varying from 50 to 100 sensors are distributed uniformly at random preserving homogenous node density among all areas in the sensing field.
- Sensor Parameters: Each sensor has a radio communication radius of 25m and a buffer of size 1 Kbyte.
- Mobile Collector Parameters: Initially the mobile collector is localized at the center of the sensing field. The mobile collector moves with a fixed speed of 1m/s, and has a communication radius of 50m. The mobile collector achieves communication with the sensor node when it is in the sensor's radio communication range.
- Simulation Time: The simulation run last for 50000 time units, and all results are averaged over 20 different independent networks.

- Sensors Sampling Frequency: The sampling frequency determines the time required for the sensor's buffer to become full and therefore controls the distribution of the collection requests over the simulation time. The first option assumes that events occur regarding some point of interest located at the center of the sensing field. In such a case, a higher number of the requests are coming from a specific part of the network, as the sensors closer to the center sample more frequently. The requests distribution in this case follows Gaussian distribution with small variance. A concentric topology for the sensors sampling rates as shown in Figure 5 is used to model this. A sequence of n concentric circles divides the sensing field 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 = 10m. The value of each radius is calculated as:

$$R_i = i * R_1; \quad i = [1,...,n] \tag{18}$$

where

$$n = \frac{L}{2 * R_1} + 1 \tag{19}$$

The sensors in the innermost region are assigned sensing rates in the range [1, *baserate*] and the sensing rates of sensors in regions radically outwards are calculated as:

$$Sensingrate_i = [\, baserate * (i - 1) + 1 \,,\, baserate * i \,], \quad i = \{1,...,n\} \tag{20}$$

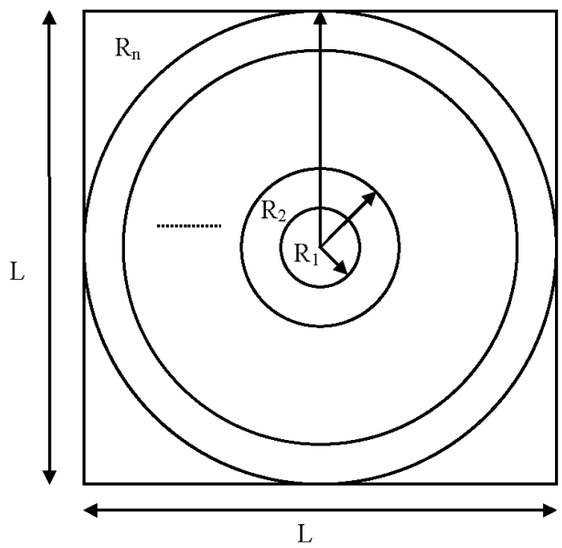


Fig. 5. Concentric Topology: The first type of topology considered in the simulation

Label	*Sensing Rate Topology*	*Number of Sensors*	*Baserate*
A1	Concentric	50	2 sec
A2	Concentric	75	3 sec
A3	Concentric	100	4 sec
B1	Random	50	2 sec
B2	Random	75	3 sec
B3	Random	100	4 sec

Table 2: Set of Experiments

When the baserate is chosen to be 2 secs, the sensors in region R_2 will have a sensing rate in the interval [3,4] which means that some sense a sample every 3 secs and others sense a sample every 4 secs.

The second option assumes random occurrence of events independently from one another in the sensing field. In this case the requests distribution follows Poisson distribution. The sensing rate of each sensor node is chosen randomly in the interval [1, *baserate* * *n*].

Putting everything together, the experiments shown in Table 2 are run. The experiments are labeled for referencing purposes. For each of these, results presented are an average of 20 runs. The parameters that change from run to run are the location of the sensor nodes, and correspondingly the distance cost and the sensing rates.

For the purposes of evaluation, the following performance metrics are considered:

- **Data Collection Time:** This is defined as the period from the time the sensor sends the collection request to the time of arrival of the mobile collector to collect the sensor's buffer. This is averaged across all the nodes in the network.

- **Request Collection Time:** This is computed as the overall data collection time to the number of requests collected from all nodes in the network.

- **Sleeping Sensors Ratio:** This measure is calculated as the number of sleeping sensors waiting the arrival of the mobile collector to the total number of sensors in the network over the simulation time of the experiment.

- **Distance Ratio:** This is defined as the ratio of the distance traveled by the mobile collector to the number of requests collected.

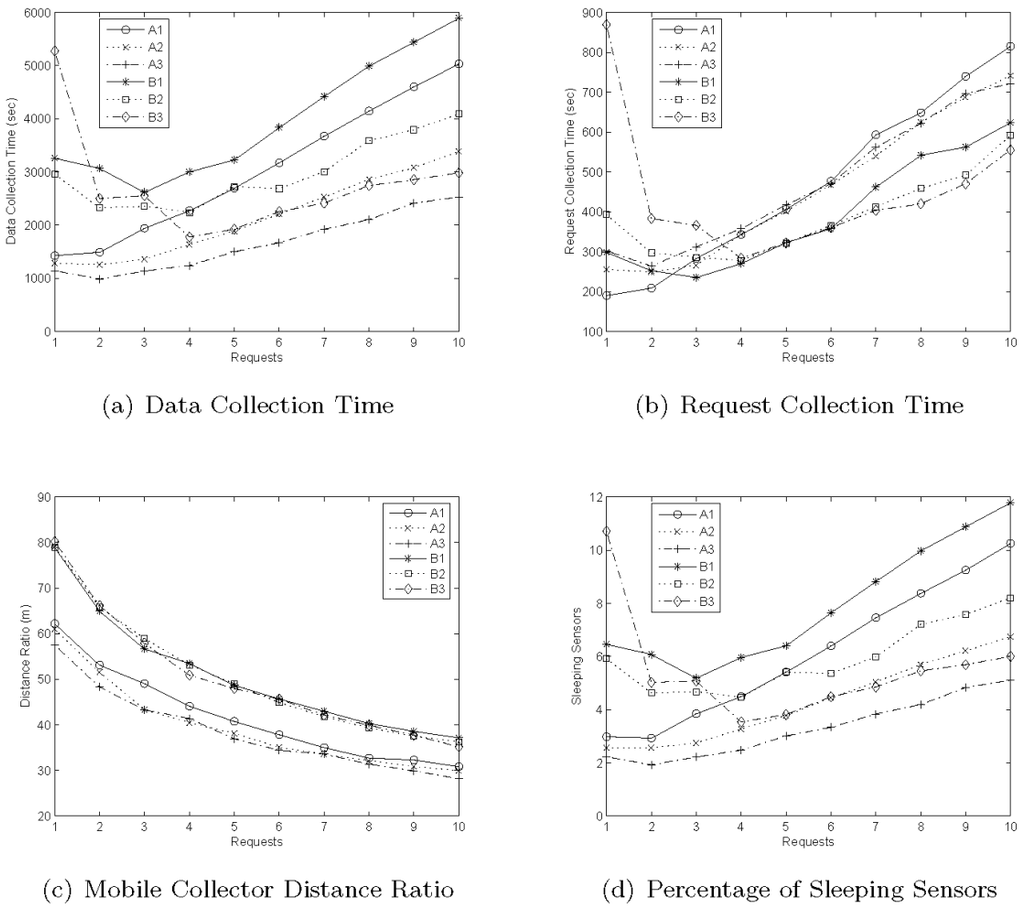
5.2 Performance of the MST-R

Figure 6(a) and 6(b) show the performance results of running the MST-R algorithm on the set of experiments described in Table 2. The number of requests m which the mobile collector uses for constructing its collection route ranged form 1 to 10. The case m = 1, is the case where the mobile collector service the arriving requests one by one in a timely oriented manner. As the mobile collector waits for more requests to arrive this increases the data

collection time however in such a case the mobile collector service more requests on the same collection route which decrease its distance ratio as presented in Figure 6(c). Also, waiting for more requests to arrive would increase the average percentage of sensor nodes requesting the collection service as shown in Figure 6(d). The performance on A3 was better than A2 which was better than A1. This is obvious because these have sensing base rate of 2, 3, 4 secs respectively, and higher this quantity, lesser the requests arrival rate. Similar trends are observed in experiments B1-B3.

5.3 Effect of a on the DI-R

Figure 7(a) and 7(b) show the result of running the DI-R construction algorithm on the set of experiments described in Table 2. a ranged from 0 to 1 with steps of 0.5. The data collection time and the request collection time decrease as a goes near to 1. This is because the request is inserted on the mobile collector route in the position resulting the minimum extra distance, as explained in Table 1. Therefore, the forth and back traveling between nodes is minimized which results in less waiting times for the sensors, thus less sleeping sensors as in Figure 7(d).



(a) Data Collection Time

(b) Request Collection Time

(c) Mobile Collector Distance Ratio

(d) Percentage of Sleeping Sensors

Fig. 6. Performance of MST-R for Experiments A1-A3 and B1-B3

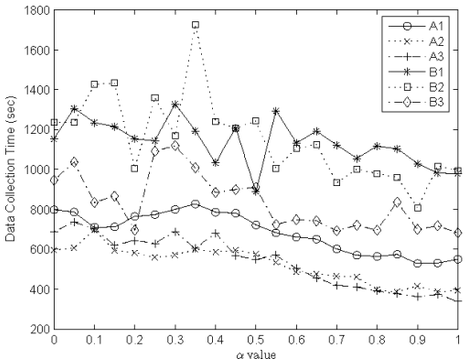
The performance on experiments A1-A3 is better than B1-B3. This is obvious because most of the events come from a specific part of the network which helps the mobile collector to optimize its collection route more than when the events are scattered as in experiments B1-B3. This also appears on the distance ratio of the mobile collector shown in Figure 7(c). As the events are scattered randomly over the sensing field, the mobile collector is required to travel more for the collection which appears on the distance ratio for experiments B1-B3. It is hard to conclude about the dependence of α value on the results, however, α around 0.9 leads to minimum results for the performance metrics described.

5.4 Impact of the Speed of the Mobile Collector

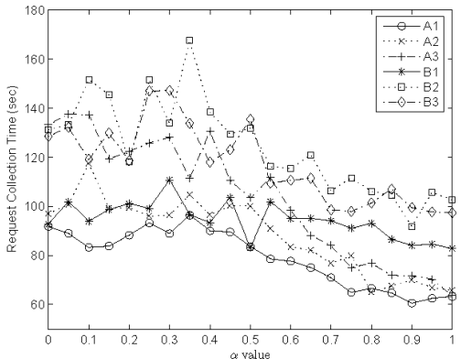
To get an insight into the relative performance of the two algorithms, two constrained topologies are used to show the impact of the mobile collector speed on the data collection time and the request collection time. The topologies considered employ 100 sensor nodes uniformly distributed in a square area of 100 x 100 (m^2). A concentric and random sensing rate topology is used with the sensing base rate equals 2 secs. These are labeled as Topology A and Topology B. Figure 8(a) and 8(b) plot the performance of the two algorithms for different speed values for the mobile collector. The DI-R construction algorithm outperforms the MST-R construction algorithm, however, when the speed of the mobile collector increase, this difference vanishes. Also, previously the algorithms performed better on the concentric sensing rate topology than on the random topology, this does not hold when the speed of the mobile collector is increased.

6. Conclusion and Future Directions

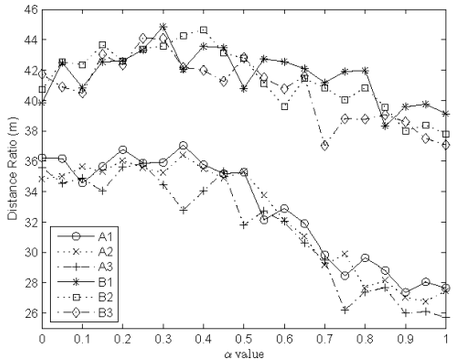
Sensor networks operate under limited energy constraints. Eliminating the relaying overheads can extend the sensor lifetime and prolong the network operational time. In this context, mobile elements (robots) are utilized by acting as mechanical data carriers for collecting the sensory information by approaching physically the sensor node. This chapter presented some heuristics for constructing the mobile collector collection route. The algorithms performance are shown and their impact on the data collection operation is presented. There are many directions in which this work may be pursued further. Statistical measures are required to measure the buffer filling rate and thus the sensor can send its collection request before its buffer is full, which gives an extra advantage for the mobile collector. Applying multiple mobile collectors can enhance the performance. Control schemes for coordinating multiple collectors need to be designed efficiently to maximize the performance.



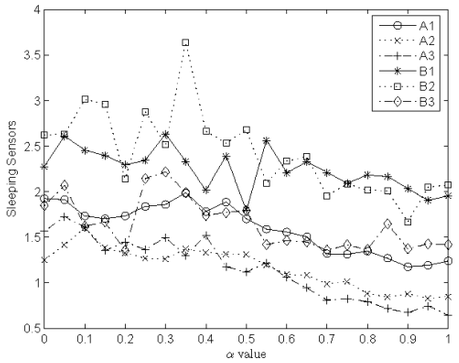
(a) Data Collection Time



(b) Request Collection Time

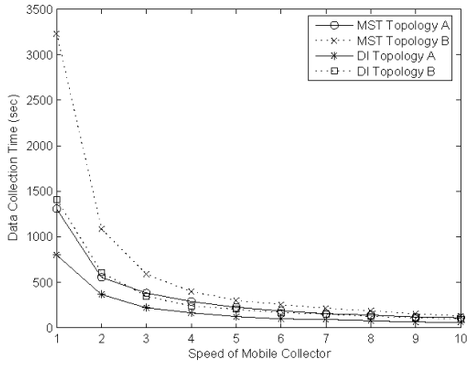


(c) Mobile Collector Distance Ratio

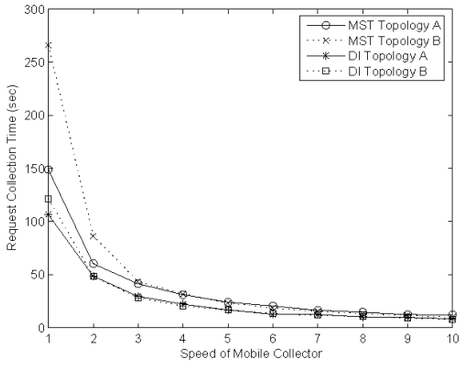


(d) Percentage of Sleeping Sensors

Fig. 7. Performance of DI-R for Experiments A1-A3 and B1-B3



(a) Data Collection Time



(b) Request Collection Time

Fig. 8. Relative Performance for Topologies A and B

References

I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. Computer Networks, 38(4):393-422, 2002.

P. Baruah, R. Urgaonkar, and B. Krishnamachari. Learning enforced time domain routing to mobile sinks in wireless sensor fields. In Proceedings of the 1st IEEE EMNETS, 2004.

L. D. Bodin, B. L. Golden, A. A. Assad, and M. Ball. Routing and scheduling of vehicles and crews, the state of art. Computers and Operations Research, 10:63212, 1983.

J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. IEEE Pervasive Computing, 3(1):38-45, 2004.

A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In 2nd International Workshop on Information Processing in Sensor Networks (IPSN), 2003.

Chee-Yee Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. Proceedings of the IEEE, 91(8):1247-1256, 2003.

N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12:568-581, 1964.

D. Culler, D. Estrin, and M. Srivastava. Introduction: Overview of sensor networks. IEEE Computer, 37(8):41-49, 2004.

G. Engelberger. Services. Shimon Y. Nof, ed., Handbook of Industrial Robotics. John Wiley & Sons, New York, second edition, 1999.

M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: W. H. Freeman, 1979.

G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. European Journal of Operational Research, 151(1):1-11, 2003.

M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Transactions on Networking, 10(4):477-486, 2002.

Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C. Lee. Partitioning based mobile element scheduling in wireless sensor networks. In Proceedings of the 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON), 2005.

M. Haenggi. Opportunities and challenges in wireless sensor networks. Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, pages 1.1-1.14,
2004.

S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. Mobile Networks and Applications, 11(3):327-339, 2006.

A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys), 2004.

R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. Mathematics ofOperations Research, 2:209-224, 1977.

G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. European Journal ofOperational Research, 59:231-247, 1992.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley &
Sons, 1990.

E. C.-H Ngai, J. Lin, and M. R. Lyu. Delay-minimized route design for wireless sensor-actuator networks. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2007.

D. Rosenkrantz, R. E. Sterns, and P. M. Lewis. An analysis of several heuristics for the traveling salesman problem. SIAM Journal on Computing, 6:563-581, 1977.

R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. Ad Hoc Networks, 1(2-3):
215-233, 2003.

M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research, 35(2), 1987.

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, 2004.

A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling with dynamic deadlines. IEEE Transactions on Mobile Computing, 6(4):395-
410, 2007.

Y. Tirta, B. Lau, N. Malhotra, S. Bagchi, Z. Li, and Y. Lu. Controlled mobility for efficient data gathering in sensor networks with passively mobile nodes. Sensor
Network Operations, 3.2:Wiley-IEEE Press, 2006.

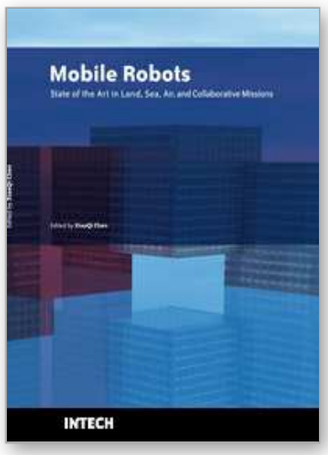
P. Toth and D. Vigo, editors. The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-498-2.

O. Younis and S. Fahmy. Heed: A hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks. IEEE Transactions on Mobile Computing, 3:
366-379, 2004.

W. Zhao and M. H. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In 9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS), 2003.

W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In 5th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), 2004.

W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2005.



Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions

Edited by XiaoQiChen

ISBN 978-953-307-001-8
Hard cover, 335 pages
Publisher InTech
Published online 01, May, 2009
Published in print edition May, 2009

Since the introduction of the first industrial robot Unimate in a General Motors automobile factory in New Jersey in 1961, robots have gained stronger and stronger foothold in the industry. In the meantime, robotics research has been expanding from fix based robots to mobile robots at a stunning pace. There have been significant milestones that are worth noting in recent decades. Examples are the octopus-like Tentacle Arm developed by Marvin Minsky in 1968, the Stanford Cart crossing a chair-filled room without human assistance in 1979, and most recently, humanoid robots developed by Honda. Despite rapid technological developments and extensive research efforts in mobility, perception, navigation and control, mobile robots still fare badly in comparison with human abilities. For example, in physical interactions with subjects and objects in an operational environment, a human being can easily relies on his/her intuitively force-based servoing to accomplish contact tasks, handling and processing materials and interacting with people safely and precisely. The intuitiveness, learning ability and contextual knowledge, which are natural part of human instincts, are hard to come by for robots. The above observations simply highlight the monumental works and challenges ahead when researchers aspire to turn mobile robots to greater benefits to humankinds. This book is by no means to address all the issues associated mobile robots, but reports current states of some challenging research projects in mobile robotics ranging from land, humanoid, underwater, aerial robots, to rehabilitation.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Samer Hanoun and Saeid Nahavandi (2009). Effective Heuristics for Route Construction of Mobile Data Collectors, Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions, XiaoQiChen (Ed.), ISBN: 978-953-307-001-8, InTech, Available from: http://www.intechopen.com/books/mobile-robots-state-of-the-art-in-land-sea-air-and-collaborative-missions/effective-heuristics-for-route-construction-of-mobile-data-collectors

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen