Scheduling under Location and Time Constraints for Data Collection in Sensor Networks

(Work in Progress paper)

Ryo Sugihara Rajesh K. Gupta
Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093

***Abstract*—Unlike the traditional multihop forwarding approach, an alternative approach for data collection in sensor networks is to exploit the mobility: a mobile node travels through the sensor field and collects data from each sensor when it is at the proximity. In this paper, we define *data mule scheduling* problem and formulate it as a scheduling problem that has both location constraints and time constraints. We present some initial results on simple cases of data mule scheduling problem and suggest similarities with speed scaling problems such as DVS (Dynamic Voltage Scaling).**

I. Introduction

An alternative and relatively new approach for energy-efficient data collection in sensor networks is to use a mobile node[2][4][5][7][8][11]. A mobile node, which we call a "data mule", travels inside the field and collects data from each sensor when the distance is short, and later deposits all the data to the base station. In this way, each sensor can conserve a significant amount of energy, since it only needs to send the data over a shorter distance. In addition, as the data mule returns to the base station after the travel, energy issue is not critical.

In this paper, we are interested in the following problem: "how to control a data mule such that it collects data from all the nodes in the minimal amount of time". We call it the *data mule scheduling* problem, as we formulate it as a scheduling problem, viewing communication from each node as a job. We can control the movement of the data mule (path, speed) as well as its communication (i.e., which node it collects data from at certain time duration). The most notable difference from traditional scheduling problems is that data mule scheduling problem has location constraints as well as time constraints. Availability of each job is determined by the range of wireless communication, which primarily depends on the distance from a node and thus serves as a location constraint. On the other hand, by assuming the bandwidth of wireless communication is constant, we also have a time constraint for each node to transmit all the data to a data mule. The movement of data mule determines how the location constraints are transformed into time constraints and produces different real-time scheduling problems.

Due to the space limitations, we focus on the problem definition and some initial results in this paper. For more details, please refer to our technical report [12].

II. Data mule scheduling problem

A data mule is a mobile node that moves inside the field and collects data from each sensor. The problem is how to control the data mule so that it can collect data from the sensors in the minimum amount of time.

As shown in Figure 1, we can decompose the problem into the following three subproblems:

1) Path selection: which trajectory the data mule follows
2) Speed control: how the data mule changes the speed during the travel
3) Job scheduling: from which sensor the data mule collects data at each time point

Path selection is to determine the trajectory of the data mule in the sensor field. To collect data from each particular sensor, the data mule needs to go within the sensor's communication range at least once. Depending on the capability of data mule, there may be some constraints on path selection, such as minimum turning radius.

Speed control is the second subproblem to determine how the data mule changes its speed along the chosen path. The data mule needs to change the speed so that it stays within each sensor's communication range long enough to collect all the data from it.

Final subproblem is job scheduling. Once the time-speed profile is determined, we get a mapping from each location to a time point. Thus we get a real-time scheduling problem by regarding data collection from each sensor as a job. Each job has one or more intervals in which it can be executed. Job scheduling is to determine the allocation of time to jobs so that all jobs can be completed.

In this paper, we focus on the subproblems of speed control and job scheduling, and leave path selection problem to our future work. The primary reason is that these two subproblems constitute 1-D data mule scheduling problem, which is important in many cases including 2-D settings with a fixed path as in [5].

A. Preliminaries

1) Terminology and definitions: Based on standard terminology in real-time scheduling, we define the following terms in data mule scheduling problem:

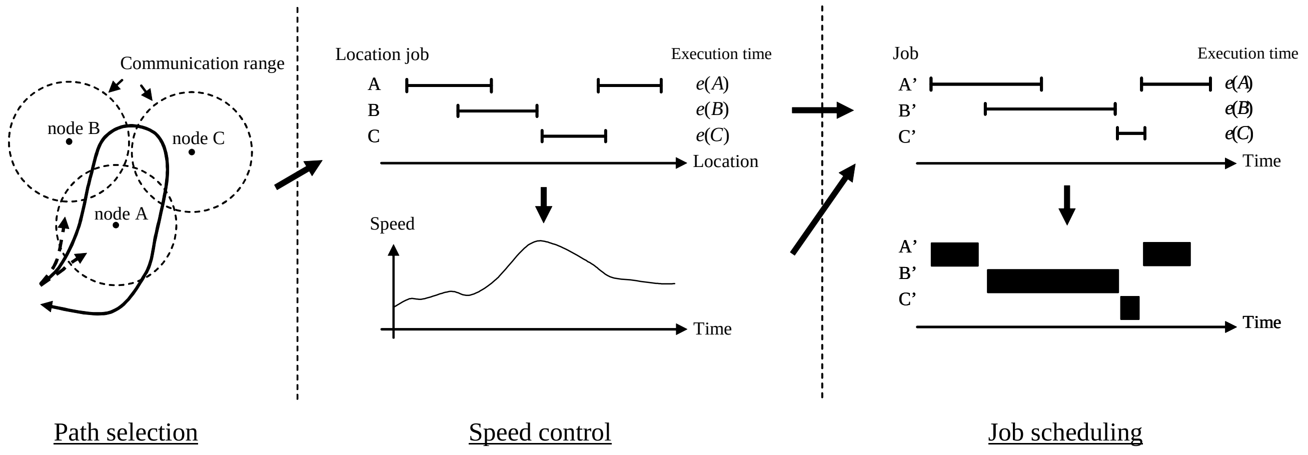

Fig. 1. Subproblems of data mule scheduling

- A location job τ_L has an execution time $e(\tau_L)$ and a set $\mathcal{I}(\tau_L)$ of feasible location intervals, containing one or more feasible location intervals.
  - A simple location job is a location job with one feasible location interval. A general location job can have multiple feasible location intervals.
- A feasible location interval $I_L \in \mathcal{I}(\tau_L)$ is a location interval $[r(I_L), d(I_L)]$, where $r(I_L)$ is a release location and $d(I_L)$ is a deadline location.
  - A location job can be executed only within its feasible location intervals.

We also define "job", "feasible interval", "simple job", "general job", "release time", and "deadline" in the similar way for real-time scheduling.

For a time or location interval $I = [r, d]$, we use the following notations:

- Length: $|I| \equiv d - r$. It is often called "relative deadline" for a simple job.
- Containment (point): $x \in I$ if and only if $r \leq x \leq d$.
- Containment (interval): $I \in I'$ if and only if $r' \leq r$ and $d \leq d'$ where $I' = [r', d']$.

2) Assumptions:

- All constraints and parameters are deterministic.
- All jobs and location jobs are preemptible.
- Communication bandwidth is constant within the communication range and zero out of the range.
- Each sensor has different amount of data to be collected, i.e., execution time of each location job differs.

B. Problem statement

We present the structure of the data mule scheduling problem in its basic form.

1) Instance: Input of the problem is as follows:

- A set $\mathcal{J}_L$ of location jobs
- Total travel interval $[X_s, X_d]$

2) Objective: The objective is to find a time-speed profile and a feasible schedule[1] so that the total travel time is minimized.

3) Constraints: The constraints for each subproblem are as follows. There will be more constraints imposed when we discuss variations of the problem later.

- For speed control:
  - Data mule moves from the start to the destination
  - One-way movement: data mule is not allowed to move backward
- For job allocation:
  - Feasible interval: every job can be executed only within its feasible interval
  - Job completion: every job is allocated time equal to its execution time
  - Processor demand: data mule can collect data from only one node (i.e., execute only one job) at a time

III. Related real-time scheduling problem

Data mule scheduling problem can always be transformed to a real-time scheduling problem once we specify the time-speed profile of the data mule. If there is a feasible schedule for the corresponding real-time scheduling problem, there is also a feasible schedule for the original data-mule scheduling problem. Here we present some related issues in real-time scheduling problem, specifically about feasibility testing algorithm.

The real-time scheduling problem we obtain after we determine time-speed profile is an ordinary preemptive job scheduling, except each job may have multiple feasible intervals in our case. As far as we know there are only few studies on this case: for unit-length, non-preemptible jobs [10] and for preemptible, non-continuable (i.e., jobs must be completed within one feasible interval) jobs [9][3], both of which are different from our problem of interest.

[1]Following the definition in [6], a feasible schedule is a valid schedule in which every job completes by its deadline.

When each job has multiple feasible intervals (i.e. general job), EDF algorithm is not optimal anymore. In fact, we can show it is impossible to have an optimal online scheduling algorithm (proof by an adversary argument. details omitted).

To design an offline scheduling algorithm, we use processor-demand-based feasibility testing by Baruah et al. [1]. Processor demand g in interval $[t_1, t_2]$ is the sum of the execution time of the tasks whose feasible interval is completely contained in the interval, and is defined as follows:

$$g(t_1, t_2) \quad = \sum_{\tau \in \mathcal{J}, I(\tau) \in [t_1, t_2]} e(\tau)$$

Baruah's testing is for periodic tasks with arbitrary relative deadline (i.e., relative deadline of each task can be smaller than its period), but it is also applicable to our case. Under these assumptions, the following theorem holds:

Theorem 1 (Baruah et al. [1])**.** *Let $\tau = \{T_1, ..., T_n\}$ be a task system. τ is not feasible iff there exist natural numbers $t_1 < t_2$ such that $g(t_1, t_2) > t_2 - t_1$*

We can restrict the test points to the sets of release times and deadlines as follows:

Theorem 2. *Taskset is feasible if and only if $g(t'_1, t'_2) \le t'_2 - t'_1$ for any $t'_1 \in \{r_i\}, t'_2 \in \{d_i\}$ satisfying $t'_1 < t'_2$.*

Proof: Omitted from this version. ■

Using Theorem 2, we can formulate the scheduling problem as a linear programming problem (details omitted). Since there is a polynomial time algorithm to solve linear programming, it is a polynomial time optimal offline scheduling algorithm.

IV. Data mule scheduling

We consider two simple cases of data mule scheduling. One is constant speed, where the data mule moves at a constant speed from the start to the destination. The other is variable speed, where the data mule can freely change the speed. We present optimal algorithms for each of the variations and see some interesting similarities with speed scaling schemes such as dynamic voltage scaling (DVS).

A. Constant speed

For constant speed case, the problem is to find the maximum speed v_0 such that all jobs can be finished.

Since there is no optimal online algorithm that minimizes the total travel time without knowledge about the jobs released in the future (proof omitted), we present two optimal offline scheduling algorithms, each for simple location jobs and general location jobs, respectively.

1) Simple location jobs: When each location job has one feasible location interval, following simple algorithm finds the maximum possible v_0 such that all location jobs can be completed. It applies processor-demand based feasibility test (Theorem 2) for all possible pairs of a release location and a deadline location:

```
1 for each location interval $I_L = [r(\tau'_L), d(\tau''_L)]$
                 s.t. $\tau'_L, \tau''_L \in \mathcal{J}_L$, $r(\tau'_L) \le d(\tau''_L)$
2       do   ▷ Calculate processor demand for $I_L$
3           $d = \sum_{\tau_L \in \mathcal{J}_L, \mathcal{I}(\tau_L) \in I_L} e(\tau_L)$
4           $u[I_L] \leftarrow \frac{|I_L|}{d}$   ▷ Max speed allowed for $I_L$
5 return $\min_{I_L} u[I_L]$
```

Note we only consider simple location jobs, each of them having only one feasible location interval. This algorithm runs in $O(n^3)$ time where n is the number of location jobs.

2) General location jobs: For general location jobs case, we formulate the problem as a linear programming problem.

We split the location interval $[X_s, X_d]$ into $(2m+1)$ location intervals $[l_0(= X_s), l_1]$, $[l_1, l_2]$, ..., $[l_{2m}, l_{2m+1}(= X_d)]$ $(l_i \le l_{i+1})$, where m is the number of feasible location intervals of all location jobs, and each l_i is either a release location or a deadline location. Then we transform each location interval to a time interval using $s_i = l_i/v_0$, and obtain $(2m+1)$ time intervals $[s_0(= 0), s_1]$, $[s_1, s_2]$, ..., $[s_{2m}, s_{2m+1}]$. Note each s_i is a variable, since v_0 is a variable. In the same way, we convert each location jobs in $\mathcal{J}_L$ to a job by transforming each feasible location interval of the job to a feasible time interval, and obtain a new set of jobs $\mathcal{J}$.

For each $\tau \in \mathcal{J}$, we consider variables $p_0(\tau), ..., p_{2m}(\tau)$, in which $p_i(\tau)$ represents the time allocated to job τ during the time interval $[s_i, s_{i+1}]$. Equivalently, $p_i(\tau)$ represents the time allocated to job τ_L within the location interval $[l_i, l_{i+1}]$.

We construct a linear programming problem as follows:

Variables:

- v_0: speed of data mule
- $p_i(\tau)$ $(0 \le i \le 2m)$: time allocated to job τ in interval $[s_i, s_{i+1}]$ (or equivalently, time allocated to location job τ_L in location interval $[l_i, l_{i+1}]$)

Objective: Maximize v_0

Constraints:

- (Positiveness) $p_i(\tau) \ge 0$
- (Feasible intervals) For all $\tau \in \mathcal{J}$, if $[l_i, l_{i+1}] \notin \mathcal{I}(\tau_L)$,

$$p_i(\tau) \quad = \quad 0 \qquad (1)$$

where $\tau_L \in \mathcal{J}_L$ is converted to $\tau \in \mathcal{J}$

- (Job completion) For all $\tau \in \mathcal{J}$,

$$\sum_{i=0}^{2m} p_i(\tau) \quad = \quad e(\tau) \quad (= e(\tau_L)) \qquad (2)$$

- (Processor demand) For all $0 \le i \le 2m$,

$$\begin{aligned} \sum_{\tau \in \mathcal{J}} p_i(\tau) \quad &\le \quad s_{i+1} - s_i \\ &= \quad \frac{l_{i+1} - l_i}{v_0} \end{aligned} \qquad (3)$$

The processor demand constraint becomes a linear constraint by introducing a new variable $u_0 = \frac{1}{v_0}$ instead of v_0.

B. Variable speed

In variable speed case, the data mule can change its speed anytime. To make the problem realistic[2], we enforce constraints on speed for this case and the data mule can choose its speed within the range $[v_{min}, v_{max}]$.

1) Simple location jobs: When $v_{min} = 0$, the following EDF-based online algorithm is optimal for this case:

- Move at v_{max} while executing a job with the earliest deadline
- When reached at a job's deadline location and that job is not finished yet, the data mule stops and finish it.

When $v_{min} > 0$, there is no optimal online algorithm. We omit the proofs due to the space constraint.

2) General location jobs: Similarly to the constant speed case, no optimal online scheduling algorithm exists for this problem and we design an offline algorithm by linear programming formulation.

When $v_{min} > 0$, we can construct a formulation as follows:

Variables: For each location interval $[l_i, l_{i+1}]$ $(0 \leq i \leq 2m)$,

- v_i: speed of data mule
- $p_i(\tau)$: time allocated to job τ

Objective: Minimize the total travel time

$$\sum_{i=0}^{2m} \frac{l_{i+1} - l_i}{v_i} \tag{4}$$

Constraints:

- (Speed)

$$v_{min} \leq v_i \leq v_{max} \tag{5}$$

- (Processor demand) For all $0 \leq i \leq 2m$,

$$\sum_{\tau \in \mathcal{J}} p_i(\tau) \quad \leq \quad \frac{l_{i+1} - l_i}{v_i} \tag{6}$$

- (Positiveness), (Feasible intervals), and (Job completion) are same as the formulation for constant speed case.

We can eliminate $\frac{1}{v_i}$ terms by introducing new variables $u_i = \frac{1}{v_i}$ instead of v_i. From (5), the range of u_i is $\frac{1}{v_{max}} \leq u_i \leq \frac{1}{v_{min}}$. Now the objective and all the constraints are linear to the variables.

When $v_{min} = 0$, we can use a slightly different formulation by substituting new variables d_i for $\frac{l_{i+1} - l_i}{v_i}$, in the same way we replaced variables above. Notice that, when $v_{min} = 0$, even a job only with zero-length feasible location intervals (i.e., $r(I) = d(I)$ for all $I \in \mathcal{I}(\tau_L)$) can be scheduled by making the data mule stop at the location to execute the job. We can handle this situation as well by changing the formulation as above.

[2]Without speed constraints, the data mule can always minimize the total travel time simply by moving at infinite speed and stopping to execute a job (and repeat this for each job).

C. Similarities with speed scaling problem

In data mule scheduling, we map each location to a time point by determining the speed of the data mule and obtain corresponding real-time scheduling problems. Conversely, we can think of mapping time points to locations: release and deadline locations are unchanged and execution time changes according to the speed of the data mule. The resulting problem is analogous to speed scaling problem.

For constant speed case, there is an exact correspondence between data mule scheduling and static speed scaling (SSS) problem, in which a processor can choose its speed but cannot change once it starts to run. The processor speed is minimized in SSS problem to minimize the energy consumption, whereas we maximize the speed of data mule to minimize the total travel time. We can see an inverse relation between the speed of data mule and the processor speed.

The inverse relation is same for variable speed case, but there are some differences. In this case the corresponding problem is dynamic speed scaling (DSS) problem such as DVS. The main differences are due to allowed ranges of speed and the objective functions, and we are still analyzing them.

V. Summary

In this work-in-progress paper, we defined the data mule scheduling problem for data collection in sensor networks. After discussing how the problem relates to real-time scheduling problems, we presented efficient algorithms and linear programming formulations for some simple cases.

References

[1] S. K. Baruah, R. R. Howell, and L. E. Rosier. Feasibility problems for recurring tasks on one processor. *Theoretical Computer Science*, 118(1):3–20, 1993.

[2] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *IPSN '03*, pages 129–145, 2003.

[3] J.-J. Chen, J. Wu, C. Shih, and T.-W. Kuo. Approximation algorithms for scheduling multiple feasible interval jobs. In *RTCSA '05*, pages 11–16, 2005.

[4] D. Jea, A. A. Somasundara, and M. B. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *DCOSS '05*, pages 244–257, 2005.

[5] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *MobiSys '04*, pages 111–124, 2004.

[6] J. W. Liu. *Real-time systems*. Prentice Hall, 2000.

[7] M. Ma and Y. Yang. Sencar: An energy efficient data gathering mechanism for large scale multihop sensor networks. In *DCOSS '06*, pages 498–513, 2006.

[8] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 30–41, 2003.

[9] C. Shih, J. W. Liu, and I. K. Cheong. Scheduling jobs with multiple feasible intervals. In *Real-Time and Embedded Computing Systems and Applications (LNCS 2968)*, pages 53–71, 2003.

[10] B. Simons and M. Sipser. On scheduling unit-length jobs with multiple release time/deadline intervals. *Operations Research*, 32(1):80–88, 1984.

[11] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *RTSS '04*, pages 296–305, 2004.

[12] R. Sugihara and R. K. Gupta. Data mule scheduling in sensor networks: Scheduling under location and time constraints. *UCSD Technical Report*, CS2007-0911, 2007.