Scheduling in data gathering networks with variable communication speed and a processing stage

Joanna Berlińska[1] Baruch Mor[2]

[1]Adam Mickiewicz University, Poznań, Poland

[2]Ariel University, Ariel, Israel

Outline

Motivation

Data gathering applications:

- wireless sensor networks
- collecting results of distributed computations

Changing communication speed:

- other users/applications sharing the network
- maintenance activities

Data gathering network

- A set of m worker nodes $P_1, \dots, P_m$ are connected to a single base station.
- Node P_i holds dataset D_i of size α_i that has to be transferred to the base station for processing.
- Sending data of size α over a free link between P_i and the base station takes time $c_i\alpha$.
- Processing dataset D_i requires time $a\alpha_i$.
- At most one node can communicate with the base station at a time and at most one dataset can be processed at a time.
- Preemptions are allowed.

Communication speed changes

- Background communications may degrade the link performance.
- If the link between P_i and the base station is loaded, data of size α is transferred in time $\delta c_i \alpha$, for some given $\delta > 1$.
- Hence, the maximum time that may be necessary to gather all data is $\overline{T_c} = \delta \sum_{i=1}^{m} c_i \alpha_i$
- The link between P_i and the base station is loaded in n_i given disjoint time intervals $[t'_{ij}, t''_{ij}]$, where $j = 1, \ldots, n_i$, $t'_{ij} < \overline{T_c}$.

Scheduling problem

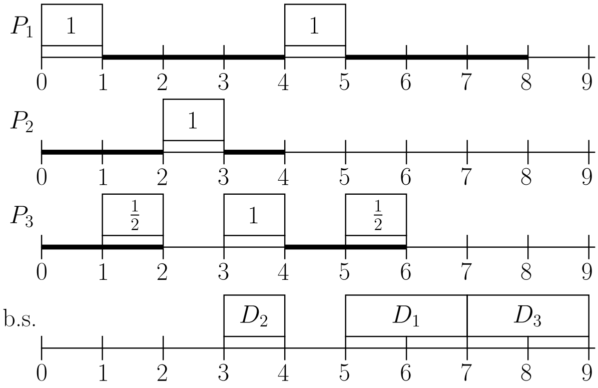
Given:

- number of workers m,
- link parameters c_i, for $1 \leq i \leq m$,
- link degradation parameter δ,
- $n = n_1 + \cdots + n_m$ loaded intervals $[t'_{ij}, t''_{ij}]$, where $i = 1, \ldots, m$, $j = 1, \ldots, n_i$,
- base station parameter a,

minimize the total time T required to gather and process all data.

Example

$m = 3, \quad \delta = 2, \quad c_1 = c_2 = c_3 = a = 1$
$\alpha_1 = 2, \quad \alpha_2 = 1, \quad \alpha_3 = 2$



Properties

- It can be assumed without loss of generality that:
  - the communication network is never idle before transferring all data,
  - datasets are processed in the order in which they arrive at the base station.
- If $a = 0$, the problem can be solved in polynomial time using linear programming (Berlińska, 2016).
- If $a = 0$ and communication preemptions are not allowed, the problem is strongly NP-hard (Berlińska, 2018).

Outline

Complexity: general case

Proposition 1

The analyzed scheduling problem is strongly NP-hard, even if $a = 1$ and $c_i = 1$ for $i = 1, \ldots, m$.

Proof: reduction from the 3-Partition problem.

3-Partition

Given a positive integer K and a set of $3s$ positive integers $\{x_1, \ldots, x_{3s}\}$ such that $K/4 < x_j < K/2$ for each $j \in \mathcal{S} = \{1, 2, \ldots, 3s\}$, and $\sum_{j=1}^{3s} x_j = sK$, is it possible to partition the index set $\mathcal{S}$ into s disjoint sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_s$ such that for each $1 \leq i \leq s$, $\sum_{j \in \mathcal{S}_i} x_j = K$?

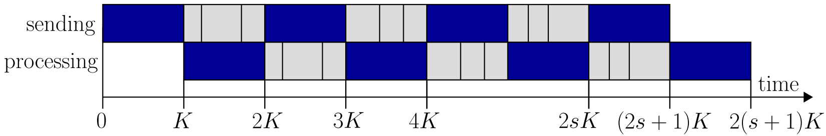
Complexity: general case

Instance construction:

- $m = 4s + 1$
- $c_i = 1$ for all i, $a = 1$
- $3s$ "regular" datasets with $\alpha_i = x_i$ for $i = 1, \ldots, 3s$
- $s + 1$ "blocker" datasets with $\alpha_i = K$ for $i = 3s + 1, \ldots, 4s + 1$
- the links used by nodes $P_1, \ldots, P_{3s}$ are always free
- the links used by nodes $P_{3s+1}, \ldots, P_{4s+1}$ are loaded in time intervals $[(2j - 1)K, 2jK)$ for $j = 1, \ldots, s + 1$
- $T = (2s + 2)K$

Complexity: general case

3-Partition $\Rightarrow$ schedule:

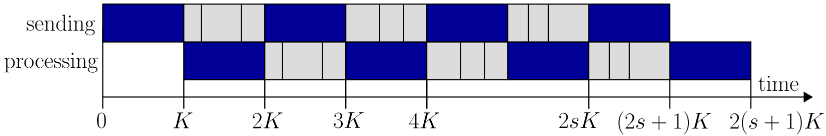


no 3-Partition $\Rightarrow$ no schedule:

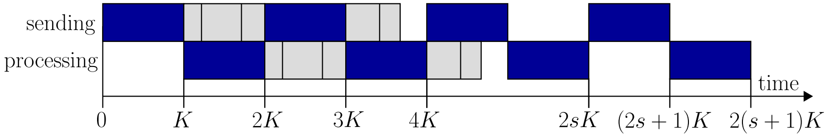
- Minimum communication idle time in interval $[0, T)$ is $K \Rightarrow$ all transfers have to use free links only.
- No data processed in interval $[0, K) \Rightarrow$ there can be no processing idle times after time K.

Complexity: general case

3-Partition ⇒ schedule:

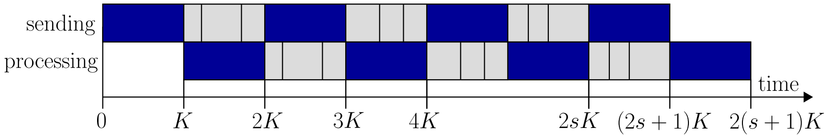


no 3-Partition ⇒ no schedule:

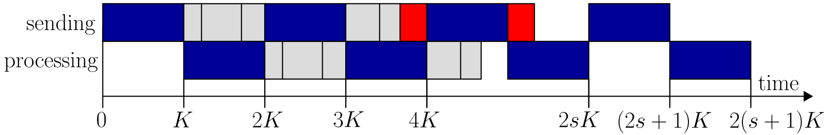


Complexity: general case

3-Partition ⇒ schedule:

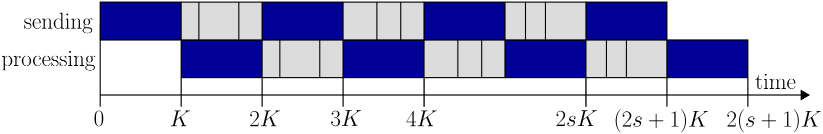


no 3-Partition ⇒ no schedule:

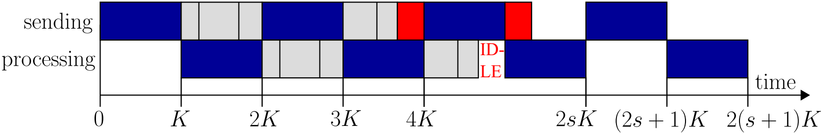


Complexity: general case

3-Partition ⇒ schedule:



no 3-Partition ⇒ no schedule:



Polynomial cases

The following special cases can be solved in polynomial time

1. $n = 0 \Rightarrow F2|pmtn|C_{max}$, Johnson's rule (Johnson, 1954)
2. $a = 0 \Rightarrow$ linear programming (Berlińska, 2016)
3. $a \leq c_i$ for all i, all links loaded in the same intervals (this work)
4. $a \geq \delta c_i$ for all i, all links loaded in the same intervals (this work)

Towards polynomial cases (3) and (4)

Lemma 1

If all links are loaded in the same intervals, then the time required to transfer a given set of datasets $\{D_{i_1}, \ldots, D_{i_k}\}$, starting at time 0, does not depend on the order of communications.

Proof:

- Sending data of size α at unit communication time c is equivalent to sending data of size $c\alpha$ at unit comunication time 1.
- If all links are loaded in the same intervals, sending datasets $D_{i_1}, \ldots, D_{i_k}$ over the respective links is equivalent to sending a single dataset of size $\sum_{j=1}^{k} c_{i_j}\alpha_{i_j}$ over a link with communication speed 1 in the free intervals and $1/\delta$ in the loaded intervals.

Towards polynomial cases (3) and (4)

Lemma 2

If all links are loaded in the same intervals, there exists an optimum non-preemptive schedule.

Proof:

- Suppose dataset D_i is transferred in several pieces in an optimum schedule Σ.
- Construct a new schedule Σ' by moving all messages containing parts of D_i just before its last piece. The other communications preceding the transfer of D_i are moved to the left.
- By Lemma 1, Σ' is not longer than Σ.
- Repeat until there are no preemptions.

Polynomial case: $a \leq c_i$, common intervals

Proposition 2

If $a \leq c_i$ for all i, and all links are loaded in the same intervals, then the optimum schedule can be constructed in $O(m \log m + n)$ time by sending the datasets in the order of non-increasing sizes α_i.

Proof:

- By Lemma 2, it is enough to consider non-preemptive schedules.
- Start with an optimum schedule Σ such that D_i is sent just before D_j, and $\alpha_i < \alpha_j$.
- Show that swapping D_i and D_j does not increase schedule length (Lemma 1 + calculations).
- Repeat.

Polynomial case: $a \geq \delta c_i$, common intervals

Proposition 3

If $a \geq \delta c_i$ for all i, and all links are loaded in the same intervals, then the optimum schedule can be constructed in $O(m \log m + n)$ time by sending the datasets in the order of non-decreasing $c_i \alpha_i$.

Proof:

- By Lemma 2, it is enough to consider non-preemptive schedules.
- Start with an optimum schedule Σ such that D_i is sent just before D_j, and $c_i \alpha_i > c_j \alpha_j$.
- Show that swapping D_i and D_j does not increase schedule length (Lemma 1 + calculations).
- Repeat.

Outline

Property

Proposition 4

Constructing an optimum schedule for the analyzed problem may require preempting a dataset transfer at a time when no link speed changes.

Proof: Let $m = 2$, $c_1 = c_2 = 1$, $a = 0$, $\alpha_1 = \alpha_2 = 2$. Suppose the first link is loaded in interval $[2, 3)$, and the second link is loaded in interval $[3, 4)$. The optimum schedule length 4 can be achieved only if all data are transferred over free links. This is not possible if no preemption takes place before time 2.

Heuristic algorithms

Greedy heuristics running in $O((m+n)^2)$ time: every time a dataset transfer completes or the speed of some link changes, the dataset to be transferred is selected according to a given rule.

- *gTime* chooses the dataset whose transfer will complete in the shortest time;
- *gRate* selects the dataset which will be sent at the best average communication rate (under the assumption that there will be no preemption);
- *gJohnson* associates with each available dataset a job consisting of two operations: sending the remaining part of this dataset, and processing this dataset, then chooses the best job using Johnson's rule.

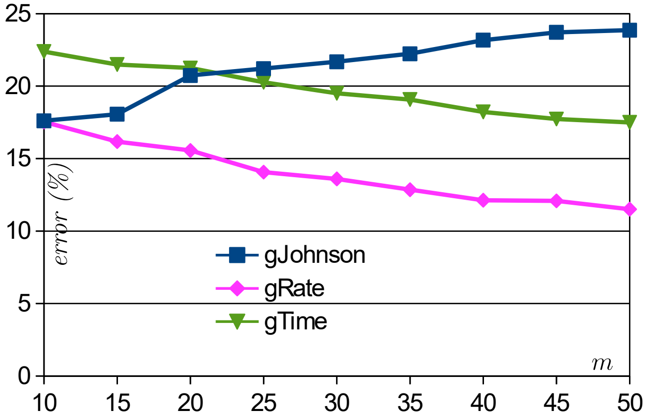
Test instances

- $m \in \{10, 15, \ldots, 50\}$
- α_i chosen randomly from the range $[1, 20]$
- $\delta = 2$
- $c_i = c$ for $i = 1, \ldots, m$, $\quad c \in \{0.5, 0.75, 1\}$, $\;a = 1$, representing the three cases of $\delta c \leq a$, $c < a < \delta c$ and $a \leq c$
- for given $F, L \in \{1, 5\}$, the lengths of consecutive free and loaded intervals chosen randomly from the ranges $[0, \frac{F}{m}\sum_{i=1}^{m} c_i\alpha_i]$ and $[0, \frac{L}{m}\sum_{i=1}^{m} c_i\alpha_i]$ correspondingly, independently for each link, until reaching the communication time horizon $\overline{T_c}$

Measure of schedule quality

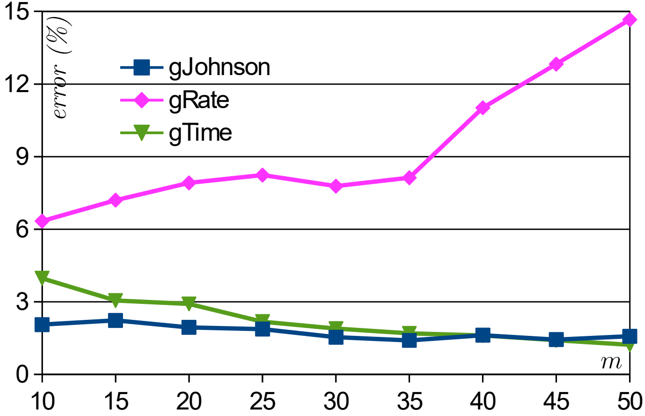
- Let $T_c^{(i)}$ be the time required to transfer dataset D_i (starting at time 0), and let T_c be the minimum time necessary for transferring all data to the base station.
- Let T_J be the minimum time required for transferring and processing all datasets under the assumption that the communication links are always free.
- Schedule quality was measured by the average percentage error with respect to the lower bound $LB = \max\{LB_1, LB_2, T_J\}$, where
  - $LB_1 = \max_{i=1}^{m}\{T_c^{(i)} + a\alpha_i\}$
  - $LB_2 = T_c + \min_{i=1}^{m}\{a\alpha_i\}$

Results vs. m for $c = 1$, $F = 1$, $L = 1$



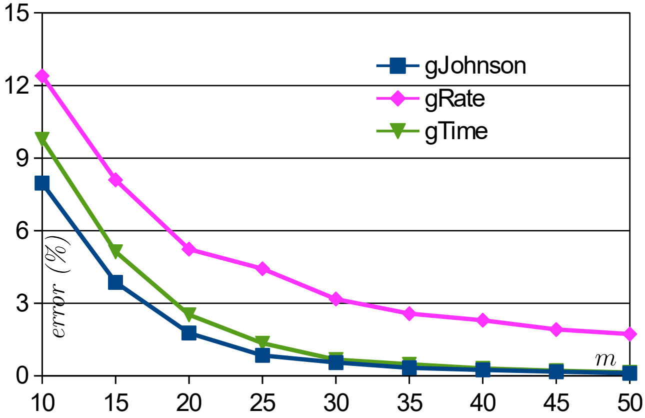
- Best results obtained by *gRate*.
- Results of *gRate* and *gTime* improve with growing m, *gJohnson* performs worse for larger m.

Results vs. m for $c = 1$, $F = 5$, $L = 1$



- Best results obtained by *gJohnson* for $m \leq 45$ and *gTime* for $m = 50$.
- Results of *gJohnson* and *gTime* improve with growing m, *gRate* performs worse for larger m.

Results vs. m for $c = 0.75$, $F = 1$, $L = 1$



- Best results obtained by *gJohnson*.
- Results of all algorithms improve with growing m.

Outline

Conclusions

- We analyzed minimizing the time of data gathering in networks with variable communication speed and a processing stage.
- The problem is strongly NP-hard.
- Several polynomially-solvable cases were identified.
- Three greedy heuristics were proposed.
- Computational experiments showed that *gRate* is the best choice when communication is slow and the links are rarely free, but in other settings *gJohnson* and *gTime* are better.
- Future research: analysis of further special cases.