The Third International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University, Poznań, July 5th – 6th, 2021
DOI: 10.14708/isbn.978-83-951298-6-5p27-32

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

Joanna Berlińska*
Adam Mickiewicz University, Poznań, Poland

Baruch Mor
Ariel University, Ariel, Israel

Keywords: scheduling, data gathering networks, variable communication speed

1 Introduction

Data gathering is an important step of many applications running in distributed systems. Computation results scattered on a large number of workers have to be collected for merging and analysis. Algorithms minimizing the total time of data gathering were proposed by Berlińska for networks with limited base station memory in [1], and for networks with dataset release times in [3]. Luo et al. [7] and Luo et al. [8] studied minimizing the data gathering time in networks with data compression. Berlińska and Przybylski [4] analyzed the problem for networks with local computations. All the mentioned articles share the assumption that the communication parameters of the network are constant. However, in reality, the communication speed may change with time, due to sharing the links with other users and applications, maintenance activities, etc. Scheduling in data gathering networks with variable communication speed was studied by Berlińska [2]. The analyzed problem was to transfer the data from the workers to a single base station in the shortest possible time. In this work, we generalize this problem to the case when each dataset has to be processed after being received by the base station. We show that minimizing the total time of data gathering and processing is strongly $\mathcal{NP}$-hard. Polynomial-time algorithms are proposed for several special cases of the problem. For the general case, heuristic algorithms are designed and compared by means of computational experiments.

2 Problem formulation

We study a star data gathering network consisting of m workers $P_1, \ldots, P_m$ and a single base station P_0. Each worker P_i holds dataset D_i of size α_i, which has to be

*Speaker, e-mail: Joanna.Berlinska@amu.edu.pl

transferred to the base station for processing. At most one node can communicate with the base station at a time. The communication rate of node P_i depends on the corresponding link being used by other applications. We will be calling a link *loaded* if it is used by background communications at a given time, and *free* in the opposite case. Transferring one unit of data from P_i to P_0 over a free link takes time c_i, for $i = 1, \ldots, m$. A loaded link becomes δ times slower, for some fixed rational $\delta > 1$. Thus, sending a unit of data over a loaded link between P_i and P_0 takes time δc_i. After being received by the base station, dataset D_i has to be processed, which takes time $a\alpha_i$. At most one dataset can be processed at a time. Preemptions are allowed both in communication and computations.

The maximum time that may be necessary to gather data from all worker nodes is $\overline{T_c} = \delta \sum_{i=1}^{m} c_i\alpha_i$. The communication speed changes are described in the following way. For each node P_i, we are given a set of n_i disjoint time intervals $[t'_{i,j}, t''_{i,j})$ (where $j = 1, \ldots, n_i$, $t''_{i,j} < t'_{i,j+1}$ for $j < n_i$, and $t'_{i,n_i} < \overline{T_c}$), in which the corresponding communication link is loaded. The total number of such intervals is $n_1 + \cdots + n_m = n$.

The scheduling problem is to minimize the total time T needed to gather and process all data. It is obvious that nothing can be gained by introducing idle times in communication. Moreover, for a fixed communication schedule, the order of processing the datasets and possible processing preemptions do not affect T, as long as no unnecessary idle times appear. Therefore, we assume without loss of generality that the communication network is never idle before transferring all data and that the datasets are processed in the order in which they arrive at the base station.

3 Computational complexity

In this section, we analyze the computational complexity of our problem and its special cases. Due to limited space, longer proofs are omitted.

The following complexity result is achieved by a pseudopolynomial reduction from the strongly $\mathcal{NP}$-complete 3-Partition problem (Garey and Johnson [5]).

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

In the next proposition, we show the main difficulty in constructing exact algorithms for our problem.

Proposition 2. *Constructing an optimum schedule for the analyzed problem may require preempting a dataset transfer at a time when no link speed changes.*

Proof. Let $m = 2$, $c_1 = c_2 = 1$, $a = 0$, $\alpha_1 = \alpha_2 = 2$, and let δ be an arbitrary number greater than 1. 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. For example, dataset D_1 can be sent in intervals $[0, 1)$ and $[3, 4)$, and D_2 in interval $[1, 3)$. It is easy to check that if no preemption takes place before time 2, then a part of one of the datasets has to be sent over a loaded link. □

According to Proposition 2, it is not known which moments should be taken into account as possible communication preemption points. Thus, constructing a full search or branch-and bound algorithm for our problem is a challenge.

Note that if the communication links are never loaded, our problem reduces to $F2|pmtn|C_{\max}$, and hence, it can be solved in $O(m \log m)$ time using Johnson's algorithm (Johnson [6]). If $a = 0$, which means there is no processing stage, then our problem is also solvable in polynomial time, using the algorithm by Berlińska [2]. In order to present other polynomial special cases, we prove a few structural properties.

Proposition 3. *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. Hence, 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. □

Proposition 4. *If all links are loaded in the same intervals, there exists an optimum non-preemptive schedule.*

Proof. Suppose that dataset D_i is transferred in several pieces in an optimum schedule Σ. We 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. Although the transfer times of individual datasets may change during this process, by Proposition 3 the transfer of each dataset finishes in Σ' not later than in Σ. Hence, dataset processing also finishes in Σ' not later than in Σ, and consequently, Σ' is an optimum schedule. By repeating this procedure for all datasets sent in several messages, we arrive at a non-preemptive optimum schedule. □

Using Propositions 3 and 4, and the interchange argument, we prove that the following two special cases of our problem are polynomially solvable.

Proposition 5. *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*.*

Proposition 6. *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 $c_i \alpha_i$.*

4 Heuristics and computational experiments

In this section, we propose greedy heuristics running in $O((m+n)^2)$ time. In each of these algorithms, 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. Algorithm *gTime* chooses the dataset whose transfer will complete in the shortest time. Heuristic *gRate* selects the dataset which will be sent at the best average communication rate (under the assumption that there will be no preemption). Algorithm *gJohnson* associates with each available dataset a job consisting of two operations: sending the remaining part of this dataset, and processing this dataset. A job is selected using Johnson's rule (Johnson [6]), and the corresponding dataset is transferred.

The makespan obtained by any algorithm that does not introduce communication idle times is at most $\delta \sum_{i=1}^{m} c_i \alpha_i + a \sum_{i=1}^{m} \alpha_i$, while the optimum makespan is not smaller than $\max\{\sum_{i=1}^{m} c_i \alpha_i, a \sum_{i=1}^{m} \alpha_i\}$. Hence, each of our algorithms delivers a $(\delta + 1)$-approximation of the optimum solution.

The quality of the results delivered by the proposed heuristics was analyzed by means of computational experiments. The number of datasets in the test instances was $m \in \{10, 15, \ldots, 50\}$. Dataset sizes α_i were chosen randomly from the range $[1, 20]$. Parameter δ was set to 2. The basic communication costs of all links were equal, $c_i = c$ for $i = 1, \ldots, m$. We used $c \in \{0.5, 0.75, 1\}$ and $a = 1$, thus representing the three cases of $\delta c \leq a$, $c < a < \delta c$ and $a \leq c$. The communication speed changes were generated similarly as those in Berlińska [2]. Namely, for given values $F, L \in \{1, 5\}$, the length of the first free interval of link i was selected randomly from the range $[0, \frac{F}{m} \sum_{i=1}^{m} c_i \alpha_i]$. Then, the length of the first loaded interval was chosen randomly from the range $[0, \frac{L}{m} \sum_{i=1}^{m} c_i \alpha_i]$. The lengths of consecutive free and loaded intervals were being selected until reaching the communication time horizon $\overline{T_c}$. For each analyzed setting, 100 instances were generated and solved.

As the optimum solutions for the generated instances were not known, schedule quality was measured by the average percentage error with respect to the lower bound LB computed as follows. 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. Note that T_c can be computed using linear programming as described by Berlińska [2]. Finally, let T_J be the minimum time required for transferring and processing all datasets under the assumption that the communication links are always

free, obtained by Johnson's algorithm. We set

$$LB = \max\{LB_1, LB_2, T_J\}, \tag{1}$$

where

$$LB_1 = \max_{i=1}^{m}\{T_c^{(i)} + a\alpha_i\} \tag{2}$$

and

$$LB_2 = T_c + \min_{i=1}^{m}\{a\alpha_i\}. \tag{3}$$

The results of our experiments lead us to the following conclusions:

1. Naturally, the obtained makespans are closest to LB when c and L are small, and F is large.
2. In general, better results are obtained for large instances than for the smaller ones. Indeed, a larger number of datasets gives more opportunities to avoid loaded links, which may result both in finding better solutions and in LB being closer to the actual optimum.
3. When communication is slow and the links are rarely free ($c = 1$ and $F = 1$), the best results are delivered by algorithm *gRate*. The reported errors are below 18% for $L = 1$, and below 40% for $L = 5$.
4. In the remaining settings, the best results are usually obtained either by *gJohnson* or *gTime* (the only exception are the tests with $c = 0.75$, $F = 1$, $L = 5$ and $m \geq 30$, for which *gRate* is the winner). The makespans delivered by the best of algorithms *gJohnson* and *gTime* are, on average, at most 23% from LB for the most difficult instances in this group (i.e., when $c = 0.75$, $F = 1$, $L = 5$, $m = 10$), but less than 3% from LB for all tests with $c = 0.5$.

5 Future research

Future research should include the complexity analysis of the special case when all links are loaded in the same intervals, but the relations between c_i and a are arbitrary, and the case when $a \geq \delta c_i$ for all i, but different links are loaded in different time intervals.

Acknowledgements

The research of the first author was partially supported by the National Science Centre, Poland, grant 2016/23/D/ST6/00410.

References

[1] J. Berlińska, Heuristics for scheduling data gathering with limited base station memory, *Annals of Operations Research*, **285** (2020), 149–159, doi: 10.1007/s10479-019-03185-3.

[2] J. Berlińska, Scheduling in data gathering networks with background communications, *Journal of Scheduling*, **23** (2020), 681–691, doi: 10.1007/s10951-020-00648-5.

[3] J. Berlińska, Makespan minimization in data gathering networks with dataset release times, *Lecture Notes in Computer Science*, **12044** (2020), 230–241, doi: 10.1007/978-3-030-43222-5_20.

[4] J. Berlińska, B. Przybylski, Scheduling for gathering multitype data with local computations, *European Journal of Operational Research*, 2021, doi: 10.1016/j.ejor.2021.01.043.

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

[6] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, **1** (1954), 61–68, doi: 10.1002/nav.3800010110.

[7] W. Luo, Y. Xu, B. Gu, W. Tong, R. Goebel, G. Lin, Algorithms for communication scheduling in data gathering network with data compression, *Algorithmica*, **80** (2018), 3158–3176, doi: 10.1007/s00453-017-0373-6.

[8] W. Luo, B. Gu, G. Lin, Communication scheduling in data gathering networks of heterogeneous sensors with data compression: Algorithms and empirical experiments, *European Journal of Operational Research*, **271** (2018), 462–473, doi: 10.1016/j.ejor.2018.05.047.