Scheduling to minimize maximum lateness in tree data gathering networks

Joanna Berlińska

Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland
Joanna.Berlinska@amu.edu.pl

Keywords: scheduling, data gathering network, maximum lateness, hybrid flow shop.

1 Introduction

Scheduling for data gathering has been attracting increasing attention in recent years. Choi and Robertazzi (2008) and Moges and Robertazzi (2006) constructed algorithms for partitioning the total amount of measurements between the nodes of a wireless sensor network in order to gather the data in the shortest possible time. Algorithms minimizing schedule length were also proposed for networks with data compression (Berlińska 2015, Luo et al. 2018), and with limited memory (Berlińska 2020). Scheduling with maximum lateness criterion in star networks was studied by Berlińska (2018), Berlińska (2019).

In this paper, we analyze minimizing maximum lateness in 2-level tree networks. The data gathering application consists of three partially overlapping stages. First, each of the leaf nodes of the network has to transfer acquired data to an appropriate intermediate node. In the second stage, datasets are preprocessed by the intermediate nodes. Finally, they are transferred from the intermediate nodes to a single base station. It is assumed that an intermediate node can receive at most one dataset at a time, and it can process at most one dataset at a time. Therefore, a subnetwork consisting of an intermediate node and all leaves that communicate with it, can be seen as a sequence of two machines working in a flow shop mode. The base station can also receive at most one dataset at a time, and hence, the third stage consists in executing a sequence of jobs on a single machine. Thus, our network is a three-machine hybrid flow shop with m dedicated machines in the first stage and the second stage, and one machine in the last stage. Hybrid flow shops with dedicated machines were studied mostly in two-machine setting (see the survey Hwang and Lin (2018)). Three-machine hybrid flow shop with one machine in the first and the third stage, and two dedicated machines in the second stage, was studied by Riane et al. (1998). The case of two dedicated machines in the last stage, and one machine in the first and the second stage was analyzed by Bedhief and Dridi (2019). To our best knowledge, three-machine hybrid flow shops with multiple dedicated machines in more than one stage were not studied in the earlier literature.

2 Problem formulation and complexity

The data gathering network consists of n leaf nodes, m intermediate nodes and a single base station. Intermediate node P_j $(1 \leq j \leq m)$ collects data from n_j leaf nodes P_{jk}, where $k = 1, \ldots, n_j$. Thus, $n_1 + \ldots + n_m = n$. Leaf node P_{jk} acquires dataset D_{jk} of size α_{jk} at time r_{jk}. The due date for receiving this dataset at the base station is denoted by d_{jk}. The time necessary to transfer one unit of data is denoted by C. Thus, dataset D_{jk} is sent from P_{jk} to P_j in time $C\alpha_{jk}$. After receiving the whole dataset, node P_j has to preprocess it, which takes time $A\alpha_{jk}$. During this process, the dataset size changes to $\gamma\alpha_{jk}$, where γ is a given application parameter. Afterwards, the dataset has to be sent to the base station, which takes time $C\gamma\alpha_{jk}$. Each node can receive at most one dataset at

a time. An intermediate node can simultaneously receive one dataset, preprocess another dataset, and send yet another dataset to the base station. Preemptions are allowed both in communication and computation.

Let T_{jk} be the time when dataset D_{jk} arrives at the base station. The lateness of D_{jk} is $L_{jk} = T_{jk} - d_{jk}$. Our goal is to organize dataset transfer and processing so that the maximum lateness $L_{max} = \max_{j=1}^{m} \max_{k=1}^{n_j} \{L_{jk}\}$ is minimized.

When $m = 1$ and $\gamma = 0$, our scheduling problem becomes equivalent to minimizing the maximum dataset lateness in a star network with datasets processed at the base station, which was proved to be strongly NP-hard by Berlińska (2019). Thus, the problem analyzed in this work is also strongly NP-hard.

3 Algorithms

In this section, we propose heuristic algorithms for solving our scheduling problem. Note that if a schedule for the first two stages of the application is fixed, an optimum schedule for the last stage can be easily found. Indeed, for each dataset D_{jk}, the time r'_{jk} when it becomes available for transfer to the base station is known, and it remains to solve an instance of problem $1|r_j, pmtn|L_{max}$, which can be done using the preemptive earliest due date first rule (Horn 1974). Therefore, our algorithms concentrate on building a good schedule for transferring datasets to intermediate nodes and preprocessing them, separately for each of the m subnetworks.

Firstly, we implement an exponential algorithm BB, which uses the branch-and-bound technique to obtain a schedule which minimizes the maximum dataset lateness after two stages of the application. A detailed description of this algorithm can be found in Berlińska (2019). Note that a partial schedule that minimizes the maximum lateness after the first two stages, does not necessarily result in the optimum schedule for the whole application. Thus, although algorithm BB uses an enumerative approach, it does not guarantee finding optimum solutions.

Secondly, we propose algorithms that build a communication schedule first, and after fixing it, construct a schedule for dataset preprocessing. For each of these two stages, we use one of the following rules:

- FIFO: choose datasets in the order in which they are released (no preemptions);
- EDD: select an available dataset with the smallest due date (possible preemptions);
- SRT: choose an available dataset with the shortest remaining transfer/preprocessing time (possible preemptions).

An algorithm which uses Rule1 for dataset transfer, and Rule2 for dataset preprocessing, will be called Rule1-Rule2. We study all possible combinations of the above rules, resulting in 9 different algorithms. Each of these algorithms finds a schedule for the j-th subnetwork in $O(n_j \log n_j)$ time, for $j = 1, \ldots, m$. After fixing subnetwork schedules, a schedule for sending datasets to the base station is computed in $O(n \log n)$ time. Since $n = n_1 + \ldots + n_m$, the total algorithm running time is also $O(n \log n)$.

4 Experimental results

In this section, we compare the performance of the proposed heuristics. The algorithms were implemented in C++ and run an Intel Core i5-3570K CPU @ 3.40 GHz with 8GB RAM. Due to limited space, we report here only on a subset of the obtained results. In the experiments presented here, the network consisted of $m = 5$ subnetworks, containing $n_i = 10$ leaf nodes each. Note that if $m > 1$, A is small in comparison to C, and γ is large, then

the third stage of the application dominates the whole schedule, i.e. it takes a significantly longer time than each of the first two stages, and hence, it has the largest impact on the obtained maximum lateness. Since the last stage is always scheduled optimally, building good solutions is easy in this case. Therefore, in order to construct demanding instances, we used $C = 1$, $A \in \{1, 2, 3\}$ and $\gamma \in \{0.01, 0.1, 0.5\}$. Dataset release times r_{jk} were generated separately for each subnetwork j as follows. The release time of the first dataset was $r_{j1} = 0$. The remaining release times were computed from the formula $r_{jk} = r_{j,k-1} + \delta_{jk}$, with δ_{jk} chosen randomly from interval $[1, 10]$, for each j and k independently. Due dates d_{jk} were selected randomly from interval $[0, 100]$, and dataset sizes α_{jk} from interval $[1, 15]$. For each analyzed combination of parameters A and γ, 30 instances were generated and solved.

Since the optimum solutions for the test instances were not known, in order to assess the schedule quality we computed a lower bound

$$LB = \max_{j=1,\ldots,m} \max_{k=1,\ldots,n_j} \{r_{jk} + (C(1+\gamma) + A)\alpha_{jk} - d_{jk}\}. \quad (1)$$

Measuring schedule quality for the L_{max} criterion may be problematic, as the optimum value can be positive, zero or negative, which precludes using relative measures. Therefore, we measure solution quality by the difference between the maximum lateness delivered by a given algorithm and the lower bound LB.

Table 1. Average distance of the solutions from the lower bound.

	$\gamma = 0.01$			$\gamma = 0.1$			$\gamma = 0.5$		
Algorithm	$A = 1$	$A = 2$	$A = 3$	$A = 1$	$A = 2$	$A = 3$	$A = 1$	$A = 2$	$A = 3$
BB	2.201	42.858	119.080	1.313	39.883	124.100	49.900	59.550	124.400
FIFO-FIFO	32.784	101.835	171.999	34.127	99.437	180.550	50.450	97.783	179.500
FIFO-EDD	28.299	44.493	120.244	29.027	43.243	125.040	50.350	59.550	125.017
FIFO-SRT	31.901	98.147	173.014	31.463	95.050	172.930	51.150	95.533	177.200
EDD-FIFO	5.731	52.349	124.176	6.183	48.993	129.527	51.000	54.050	128.483
EDD-EDD	3.602	49.125	123.044	3.197	46.710	129.527	51.217	57.633	127.617
EDD-SRT	18.915	95.982	172.949	19.417	91.010	173.167	52.917	95.183	176.383
SRT-FIFO	30.856	97.945	172.479	28.157	94.933	177.153	50.883	97.250	178.517
SRT-EDD	27.891	48.163	120.943	25.143	45.720	127.367	50.633	58.750	126.633
SRT-SRT	31.324	100.948	173.217	30.500	95.207	175.857	51.467	94.933	175.633

The average quality of the obtained solutions is presented in Table 1. The distances from LB obtained by all algorithms grow with A. The main reason for this is that the distance between LB and the actual optimum increases with A. Algorithm BB delivers the best results for all settings except $A = 2$, $\gamma = 0.05$. However, BB has high computational costs. Its average running time ranged from about 7 seconds for $A = 1$, $\gamma = 0.5$ to approximately 2075 seconds for $A = 3$, $\gamma = 0.5$, while the remaining heuristics needed about 0.004 seconds in all analyzed settings. All algorithms return similar results when $A = 1$ and $\gamma = 0.5$. This illustrates the mentioned above fact that the combination of small A and big γ leads to easy instances.

Let us now compare the performance of the fast heuristics in the remaining settings. When $A = 1$ and $\gamma \in \{0.01, 0.1\}$, algorithm EDD-EDD is the winner. When $A = 3$, the best results are obtained by FIFO-EDD, for all values of γ. For tests with $A = 2$, the best results are returned by algorithm FIFO-EDD when $\gamma \in \{0.01, 0.1\}$, and by EDD-FIFO when $\gamma = 0.5$. It seems that the choice between the EDD and FIFO rules should

be based on the expected durations of the three stages of our application. When $A = 2$ and $\gamma \in \{0.01, 0.1\}$, or when $A = 3$, the second stage dominates, and the best strategy is to use FIFO in the first stage. For $A = 2$ and $\gamma = 0.5$, the third stage is the longest (because $m = 5$), and FIFO should be applied in the second stage. In the remaining cases, the best strategy is to use only EDD rule. We infer that if stage i dominates the schedule ($i = 2, 3$), then the FIFO rule should be applied in stage $i - 1$ in order to pass some data to stage i as soon as possible, and EDD should be used in the remaining stages. If there is no dominating stage, algorithm EDD-EDD seems the best choice.

5 Conclusions

In this work, we analyzed minimizing maximum lateness in tree data gathering networks. As the problem is computationally hard, we proposed several heuristic algorithms. Computational experiments showed that algorithm BB usually delivers the best results, but at a high computational cost. Good schedules can be obtained in polynomial time using an adequate combination of EDD and FIFO rules. Future research may include investigating theoretical performance guarantees of the proposed algorithms.

Acknowledgements

This research was partially supported by the National Science Centre, Poland, grant 2016/23/D/ST6/00410.

References

Bedhief, A., N. Dridi, 2019, "Minimizing makespan in a three-stage hybrid flow shop with dedicated machines", *International Journal of Industrial Engineering Computations*, Vol. 10, pp. 161-176.

Berlińska J., 2015, "Scheduling for data gathering networks with data compression", *European Journal of Operational Research*, Vol. 246, pp. 744-749.

Berlińska J., 2018, "Scheduling Data Gathering with Maximum Lateness Objective", In: R. Wyrzykowski et al., *Parallel Processing and Applied Mathematics: 12th International Conference PPAM 2017, Part II*, LNCS 10778, pp. 135-144, Springer, Cham.

Berlińska J., 2019, "Scheduling in a data gathering network to minimize maximum lateness", In: B. Fortz, M. Labbé, *Operations Research Proceedings 2018*, pp. 453-458, Springer, Cham.

Berlińska J., 2020, "Heuristics for scheduling data gathering with limited base station memory", *Annals of Operations Research*, Vol. 285, pp. 149-159.

Choi K., T.G. Robertazzi, 2008, "Divisible Load Scheduling in Wireless Sensor Networks with Information Utility", In: *IEEE International Performance Computing and Communications Conference 2008: IPCCC 2008*, pp. 9-17.

Horn, W.A., 1974, "Some simple scheduling algorithms", *Naval Research Logistics Quarterly*, Vol. 21, pp. 177-185.

Hwang, F.J., B.M.T. Lin, 2018, "Survey and extensions of manufacturing models in two-stage flexible flow shops with de dicated machines", *Computers and Operations Research*, Vol. 98, pp. 103-112.

Luo, W., Y. Xu, B. Gu, W. Tong, R. Goebel, G. Lin, 2018, "Algorithms for Communication Scheduling in Data Gathering Network with Data Compression", *Algorithmica*, Vol. 80, pp. 3158-3176.

Moges M., T.G. Robertazzi, 2006, "Wireless Sensor Networks: Scheduling for Measurement and Data Reporting", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 42, pp. 327-340.

Riane, F., A. Artiba, S.E. Elmaghraby, 1998, "A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan", *European Journal of Operational Research*, Vol. 109, pp. 321-329.