# Scheduling data gathering with limited base station memory

Joanna Berlińska

Faculty of Mathematics and Computer Science, Adam Mickiewicz University in Poznań,
Umultowska 87, 61-614 Poznań, Poland
Joanna.Berlinska@amu.edu.pl

**Keywords:** scheduling, data gathering network, limited memory, flow shop.

## 1 Introduction

Data gathering networks are widely used in many types of contemporary applications. Distributed computing introduces the need for collecting the results obtained by remote workers. Data gathering wireless sensor networks find environmental, military, health and home applications (Akyildiz *et al.* 2002). Scheduling algorithms for data gathering networks were proposed, e.g., by Moges and Robertazzi (2006), Choi and Robertazzi (2008), Berlińska (2014) and Berlińska (2015).

In this work, we analyze gathering data in a network with limited base station memory. A dataset being received or processed by the base station occupies a block of memory of given size. The total size of coexisting memory blocks cannot exceed the base station buffer capacity. Our goal is to gather and process all data within the minimum possible time.

## 2 Problem formulation and complexity

We study a data gathering network consisting of $m$ identical worker nodes $P_1, \ldots, P_m$ and a single base station. Node $P_i$ has to transfer dataset $D_i$ of size $\alpha_i$ directly to the base station. When dataset $D_i$ starts being sent, a memory block of size $\alpha_i$ is allocated at the base station. The base station has limited memory of size $B \geq \max_{i=1}^{m}\{\alpha_i\}$. The transfer of dataset $D_i$ may start only if the amount of available memory is at least $\alpha_i$. Sending dataset $D_i$ takes time $C\alpha_i$, where $C$ is the network communication rate (inverse of speed). After dataset $D_i$ is transferred, it has to be processed by the base station. This takes time $A\alpha_i$, where $A$ is the base station computation rate. Datasets are processed in the order in which they were received. As soon as processing a dataset finishes, the corresponding memory block is released. It is assumed that both communication and computation on a dataset are non-preemptive. The base station can communicate with at most one node at a time and it can process at most one dataset at a time. The scheduling problem is to organize dataset transfers so that the total data gathering and processing time is minimized.

We prove that the above problem is strongly NP-hard even if $A = C = 1$, using a pseudopolynomial reduction from the bin packing problem (Garey and Johnson 1979).

## 3 Related work

As only one node can communicate with the base station at a time, our data gathering network can be seen as a two-machine flow shop, where the communication network is the first machine, and the base station is the second machine. Job $i$ consists of two operations: sending and processing dataset $D_i$, and requires $\alpha_i$ units of base station memory resource. Thus, we solve a resource-constrained flow shop scheduling problem (Błażewicz *et al.* 1983). It may seem similar to two-machine flow shop scheduling with limited buffer storage (see,

e.g., Leisten (1990)), but there are substantial differences between them. In a flow shop with limited buffer storage, the buffer can hold a fixed number of jobs, and a job is stored in the buffer when it has already been processed on the first machine but not yet started on the second machine. In our problem, the buffer can hold a fixed amount of data (for example, only one big dataset, but up to three small datasets), and the buffer is occupied by a dataset not only between, but also during its transfer and processing.

Lin and Huang (2006) proposed a relocation problem with a second working crew for resource recycling. Each job was executed on two machines in a flow shop style. The $i$-th job required $\alpha_i$ units of a resource, and returned $\beta_i$ units of this resource on completion. The goal was to minimize the makespan while not exceeding the available amount of the resource. This problem, denoted by $F2|rp|C_{max}$, was shown to be strongly NP-hard, and heuristic algorithms for solving it were proposed. The problem was further analyzed by Cheng *et al.* (2012), who formulated it as an integer linear program. Complexity results for a number of special cases of the problem were also presented.

Our data gathering scheduling problem is equivalent to yet another special case of problem $F2|rp|C_{max}$, which can be denoted by $F2|rp, p_i = C\alpha_i, q_i = A\alpha_i, \beta_i = \alpha_i|C_{max}$, and was not studied in the earlier literature.

## 4 Algorithms

In our problem, a schedule is determined by the order in which the datasets are transferred to the base station. Each dataset is sent without unnecessary delay, as soon as sufficient amount of memory is available.

We first observe the following symmetry property. Suppose that $A = kC$, where $k \geq 1$, and $\Sigma$ is a schedule of length $T$ for given values of $B$ and $(\alpha_i)_{i=1}^m$. Then, by reversing schedule $\Sigma$, we obtain a schedule of length $T$ for the same values of $B$ and $(\alpha_i)_{i=1}^m$, communication rate $C' = kC$ and computation rate $A' = C$. In consequence, we can assume without loss of generality that $A \geq C$.

We propose three "simple" heuristics. Algorithm **Inc** sorts the datasets in the order of increasing sizes. Since $A \geq C$, this is the order that would be returned by the Johnson's algorithm for problem $F2||C_{max}$ (Johnson 1954), and hence, algorithm Inc delivers optimum solutions if the memory limit $B$ is big enough. Algorithm **Alter** starts with sending the smallest dataset, then the greatest one, the second smallest, the second greatest, etc., thus alternating big and small datasets. Finally, algorithm **Rnd** transfers the datasets in a random order. This algorithm will be used to verify the quality of the results delivered by the remaining heuristics.

The second group of algorithms are "advanced" heuristics **IncLocal**, **AlterLocal** and **RndLocal**. Each of them starts with generating a schedule using the corresponding simple heuristic, and then applies the following local search procedure. For each pair of datasets, we check if swapping their positions in the current schedule leads to decreasing the makespan. The swap that results in the shortest schedule is executed, and the search is continued until no further improvement is possible.

Note that our algorithms cover the three heuristics $H_1, H_2, H_3$ proposed by Lin and Huang (2006) for solving problem $F2|rp|C_{max}$. In our special case, both $H_1$ and $H_2$ return the same results as IncLocal, and $H_3$ is equivalent to RndLocal.

To finish this section, let us observe that the length of a schedule obtained for an arbitrary dataset sequence does not exceed $(A + C)\sum_{i=1}^m \alpha_i$, and $A\sum_{i=1}^m \alpha_i$ is a lower bound on a schedule length. Thus, the approximation ratio of any algorithm for solving our problem is at most $1 + C/A$. Hence, we can say that our problem becomes easier to solve when $A$ gets large in comparison to $C$.

## 5 Experimental results

In this section, we compare the quality of the solutions and the computational costs of the proposed heuristics. The algorithms were implemented in C++ and run on an Intel Core i5-2500K CPU @ 3.30 GHz with 6GB RAM. The test instances were constructed as follows. The communication rate was $C = 1$ and the computation rate was $A \in \{1, 2, 5, 10\}$. We generated "small" tests with $m = 10$ and "big" tests with $m = 100$. The dataset sizes $\alpha_i$ were chosen randomly from the interval $[1, 2]$. For a given set of $\alpha_i$, we computed the minimum amount of memory that allows to hold more than one dataset in the buffer, $B_{min} = \min_{i \neq j}\{\alpha_i + \alpha_j\}$. Then, the memory limit was set to $B = \delta_B B_{min}$, where $\delta_B = 1 + i/10$, for $i = 1, 2, \ldots, 7$. For each triple of $m$, $A$ and $\delta_B$ values, 30 instances were generated. Due to limited space, we report here only on a small subset of the obtained results.

The makespans returned by the heuristics for the small tests were compared to the optimum values computed using the ILP formulation from Cheng *et al.* (2012). It turns out that the local search procedure is very effective, as for each tested setting, the average relative errors of all the advanced heuristics were below 0.5%. The average relative errors of the simple algorithms were between 3% and 20% for the most difficult tests (with $A = 1$).

Constructing optimum solutions for instances with $m = 100$ was not possible because of the exponential complexity of the exact algorithm. Therefore, the obtained makespans were compared to the lower bound computed by disregarding the memory limit and solving problem $F2||C_{max}$ for given $A$ and $(\alpha_i)_{i=1}^m$. The results are summarized in Table 1.

The solutions obtained by the advanced algorithms are much better than those of the simple algorithms. The differences between algorithms IncLocal and AlterLocal are very small, while RndLocal performs slightly worse. However, it is clear that for $\delta_B = 1.2$ the best choice among the simple heuristics is the Inc algorithm, and the results of algorithm Alter are even worse than those of the random algorithm. For $\delta_B = 1.5$ we have the reverse situation: algorithm Alter is the winner, and Inc is even worse than Rnd if $A > 1$.

**Table 1.** Average relative distance of the solutions from the lower bound, for $m = 100$.

| $A$ | $\delta_B$ | **Inc** | **Alter** | **Rnd** | **IncLocal** | **AlterLocal** | **RndLocal** |
|---|---|---|---|---|---|---|---|
| 1 | 1.2 | 0.814 | 0.974 | 0.907 | 0.702 | 0.711 | 0.727 |
| | 1.5 | 0.557 | 0.463 | 0.586 | 0.216 | 0.211 | 0.245 |
| 2 | 1.2 | 0.411 | 0.495 | 0.456 | 0.340 | 0.347 | 0.361 |
| | 1.5 | 0.262 | 0.082 | 0.231 | 0.015 | 0.015 | 0.039 |
| 5 | 1.2 | 0.166 | 0.198 | 0.183 | 0.135 | 0.137 | 0.143 |
| | 1.5 | 0.109 | 0.047 | 0.097 | 0.009 | 0.010 | 0.017 |
| 10 | 1.2 | 0.084 | 0.099 | 0.093 | 0.070 | 0.072 | 0.075 |
| | 1.5 | 0.055 | 0.020 | 0.049 | 0.005 | 0.004 | 0.009 |

We report on the execution times of the algorithms in Table 2. Here we group the results for all tested values of $A$ together. The running time of all simple algorithms is very short, and the advanced algorithms are five orders of magnitude slower. The slowest heuristic is RndLocal, and the relation between IncLocal and AlterLocal depends on $\delta_B$. For $\delta_B = 1.2$, algorithm IncLocal is much faster than AlterLocal, and for $\delta_B = 1.5$ we have the opposite situation. Thus, conforming the (initial) dataset sequence to $\delta_B$ value leads to obtaining better schedules in the case of the simple heuristics, and to shorter execution time in the case of the advanced heuristics.

**Table 2.** Average algorithm running time (in seconds), for $m = 100$.

| $\delta_B$ | **Inc** | **Alter** | **Rnd** | **IncLocal** | **AlterLocal** | **RndLocal** |
|---|---|---|---|---|---|---|
| 1.2 | 3.18E−3 | 2.90E−3 | 3.33E−3 | 2.08E+2 | 3.11E+2 | 3.89E+2 |
| 1.5 | 2.68E−3 | 2.58E−3 | 2.40E−3 | 3.11E+2 | 2.06E+2 | 6.30E+2 |

## 6 Conclusions

In this work, we analyzed scheduling data gathering with limited base station memory. As we showed that this problem is strongly NP-hard, groups of simple and advanced heuristics were proposed. Their performance was tested in computational experiments. The simple algorithms are very fast, but the results they obtain are not very good in most cases. The advanced heuristics produce high quality schedules, but their execution times are long. We showed that sorting datasets according to their sizes is a good idea if the base station memory limit is rather small. If the base station buffer is big enough to hold the smallest and the biggest dataset together, then alternating small and big datasets allows to obtain better results.

## Acknowledgements

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

## References

Akyildiz I.F., W. Su, Y. Sankarasubramaniam and E. Cayirci, 2002, "Wireless sensor networks: a survey", *Computer Networks*, Vol. 38, pp. 393-422.

Berlińska J., 2014, "Communication scheduling in data gathering networks with limited memory", *Applied Mathematics and Computation*, Vol. 235, pp. 530-537.

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

Błażewicz J., J.K. Lenstra and A.H.G. Rinnooy Kan, 1983, "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11-24.

Cheng T.C.E., B.M.T. Lin and H.L. Huang, 2012, "Resource-constrained flowshop scheduling with separate resource recycling operations", *Computers & Operations Research*, Vol. 39, pp. 1206-1212.

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.

Garey M.R., D.S. Johnson, 1979. "Computers and intractability: A guide to the theory of NP-completeness", Freeman, San Francisco.

Johnson S.M., 1954, "Optimal two- and three-stage production schedules with setup times included", *Naval Research Logistics Quarterly*, Vol. 1, pp. 61-68.

Leisten R., 1990, "Flowshop sequencing problems with limited buffer storage", *International Journal of Production Research*, Vol. 28, pp. 2085-2100.

Lin B.M.T., H.L. Huang, 2006, "On the relocation problem with a second working crew for resource recycling", *International Journal of Systems Science*, Vol. 37, pp. 27-34.

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.