Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 431–440
DOI: 10.15439/2023F5436
ISSN 2300-5963 ACSIS, Vol. 35

Optimum Large Sensor Data Filtering, Networking and Computing

Joanna Berlińska
0000-0003-2120-2595
Adam Mickiewicz University,
Poznan, Poland
Joanna.Berlinska@amu.edu.pl

Maciej Drozdowski
0000-0001-9314-529X
Poznan University of Technology
Poznan, Poland
Maciej.Drozdowski@put.poznan.pl

Thomas Robertazzi
0000-0002-2382-1843
Stony Brook University
Stony Brook NY. USA
Thomas.Robertazzi@stonybrook.edu

***Abstract*—In this paper we consider filtering and processing large data streams in intelligent data acquisition systems. It is assumed that raw data arrives in discrete events from a single expensive sensor. Not all raw data, however, comprises records of interesting events and hence some part of the input must be filtered out. The intensity of filtering is an important design choice because it determines the complexity of filtering hardware and software and the amount of data that must be transferred to the following processing stages for further analysis. This, in turn, dictates needs for the following stages communication and computational capacity. In this paper we analyze the optimum intensity of filtering and its relationship with the capacity of the following processing stages. A set of generic filtering intensity, data transfer, and processing archetypes are modeled and evaluated.**

I. Introduction

SENSOR technology has developed rapidly over the past decades. Numerous surveys can be found, each limited to a specific sensor technology type. An important research topic is the computational and communication integration of multiple sensors in a network [1], [2], [3], [4], [5]. There is an implicit assumption here that the economics of constructing, deploying and operating such a system are such that the use of multiple sensors is plausible within project budget constraints.

In this paper we envision a problem with a different emphasis where there is a single, very expensive "sensor". Examples of such a device include detectors for particle accelerator experiments such as the CERN Large Hadron Collider [6], or the recently announced Electron Ion Collider [7] to be built at Brookhaven National Laboratory. Another example is the use of a large radar equipped drone for monitoring ocean traffic. A final example is the use of imaging satellites for ocean, weather, environmental monitoring and earth resources sensing [8]. Note that what we generically consider a "sensor", may consist of a large number of individual sensing elements (as in a particle detector) but we refer to the ensemble collection as a sensor.

The commonality in all these examples is that the sensor increasingly can generate raw data at rates that are much faster than the data can be downloaded over a communication channel(s) onto servers for processing. Moreover, not all data collected by the sensor is valuable enough to be retained for further processing. The generally proposed solution for this problem is to have onboard pre-processing of the data at the detector, the drone or the satellite. This processing is not classical data compression but more so the use of data analysis, machine learning (ML) or other type of broadly understood artificial intelligence techniques to pre-process the raw data for a much smaller processed summary that is more amenable to transmit. Thus, only interesting particle tracks, ship tracks and weather patterns may be transmitted, but the majority of the raw data isn't. For simplicity of exposition we will say that the ML algorithm serves as a filter on the raw data, independently of the field of origin of the actually applied filtering technique. Clearly this situation has similarities with edge computing where sensors/actuators at the network edge do local processing in order to reduce the amount of network traffic to distant cloud facilities. Again, the specification of a single sensor here makes our problem have a unique character.

In this paper we analyze the intensity of data filtering in the first stages of data processing necessary for the shortest time to obtain results. The intensity of filtering is important for the following reasons: (i) filtering algorithms are often hardware-implemented, and consequently, have costly realization, their changes are less flexible than in software (especially in remote posts like satellites); (ii) the higher intensity of filtering, the more complex algorithm is applied which results in longer filtering time and/or more extensive hardware system; (iii) the size of data emerging from the filtering stage determines needs for capacity in the further stages of data processing pipeline where more sophisticated algorithms are executed; (iv) and vice versa the speed of communication between the stages of data pipeline and processing in the stages following the initial filtering have impact on the required intensity of raw data filtering. For the purpose of analyzing systems of the above nature, an extensive set of the single expensive sensor (SES) models for filtering and processing problem will be examined. These cases are meant to illustrate the modeling and solution possibilities and be representative in a generic way. A common sense expectation is that high intensity of initial data filtering reduces amounts of the data analyzed in the pipeline and thus reduces the time to obtaining the results. However, more intensive filtering is also more time-consuming. Thus, due to interaction between nonlinear speed of filtering and complexity of processing the data in

TABLE I
SUMMARY OF NOTATIONS

α_i	size of load part assigned to stage 2 processor i [byte]
A_0	reciprocal of the first stage processing speed [e.g. s/byte]
A_i	reciprocal of the heterogeneous second stage processing speed on processor i [e.g. s/byte]
A	reciprocal of the second stage processing speed for identical processors [e.g. s/byte]
C_i	reciprocal of the communication speed between stage 1 and heterogeneous stage 2 processor i
C	reciprocal of the communication speed between stage 1 and stage 2 for identical processors [e.g. s/byte]
F	fraction of retained data
m	number of machines (a.k.a. servers, processors) in the second stage
T	execution time of the whole filtering and processing workflow
V	size of input data (at the front-end of the system)

the later stages, the processing time may have a minimum at a certain filtering intensity. We investigate such minima in this paper. Furthermore, options for combining advanced processing algorithms of various complexity classes in the data processing workflow are analyzed.

Our models are largely tractable. Parts of the evaluation of the models use concepts from the theory of divisible (i.e. partitionable) loads, a well established concept, that provides elegant solutions particularly for linear models [9], [10], [11], [12], [13]. The divisibility of the loads means that big volumes of data are processed and the discrete units of data are small in relation to the whole data size. It is also assumed that the loads can be divided into parts processed independently in parallel.

Further organization of this paper is the following. In the next section related literature is outlined. The filtering and parallel processing problem is formally defined in Section III. Section IV is dedicated to analytical derivation of the formulas guiding selection of the optimum filtering intensity. Results of numerical modeling of filtering and processing systems are provided in Section V. The last section is dedicated to conclusions. The notations are summarized in Table I.

II. RELATED WORK

In the literature on sensor networks, particularly wireless sensor networks, there are general surveys [14] and surveys on specific technological aspects of sensor networks such as transport and routing [15], fault detection [16], security solutions [17], optimizing sensor-source geometries and minimizing the number of sensors [18], the use of swarm intelligence for performance optimization [19] and numerous applications.

There has been some work on analytical models of sensor data generation, communication and computation. For instance, an early work is [1] which examined scheduling for measurement and data reporting in wireless sensor networks. Data gathering networks have been the subject of research by Berlińska and recently by Luo et. al. Data gathering networks have been studied in connection with background communication [3], limited base station memory [4], data compression [5], [20], [21], energy minimization [22] and in the case of tree data gathering networks [2]. The case when the load is processed in a pipeline fashion has been studied in [23], [24].

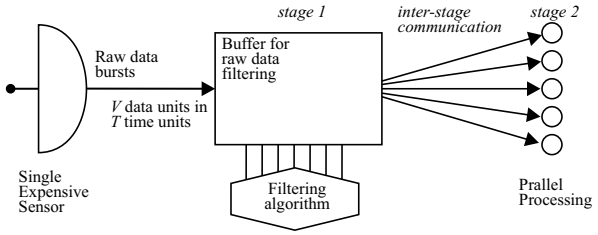

Fig. 1. Data filtering and processing system architecture.

Most work to date has involved multiple sensors, unlike the single expensive sensor paradigm of this paper. An LHC data acquisition system [6] is a good example of a single expensive sensor with data filtering and parallel processing. In LHC protons circulate in bunches and opposing beams that cross each other resulting in collisions with 40MHz frequency. Only data from particle collisions with sufficient energy and momentum are allowed to proceed from the so-called level-1 trigger to the second stage of processing (so called high-level trigger) for further reconstruction of particle trajectories and analysis.

III. PROBLEM FORMULATION

It is assumed that there is a two-stage workflow: (1) the first stage is related to sensor data filtering, (2) the second stage conducts further data processing. An overview of the system architecture is shown in Fig.1. The data from the sensor arrive in discrete events each delivering V units of data. The arriving chunk of data is intercepted in the input buffer, and all filtering algorithms use this buffer. The same buffer is used to send the filtered data to the second stage of processing. The use of a single buffer in this model is an aggregate representation of specialized buffer architectures that may be needed in practice to handle the influx, staging and filtering of massive amounts of data. The data chunks may arrive repetitively, then it is assumed that at most V units of data arrive in a chunk once in T time units. The first stage filtering is done in linear time, but the intensity of filtering, i.e. fraction F of the initial data transferred to the second stage, is related to the speed of filtering.

In the second stage more intricate, than filtering, data processing is conducted requiring machines with substantial computing power, memory and storage. Hence, these can be dedicated data centers or cloud systems. The machines running in the second stage will be referred to as servers or processors. Many different algorithms may be executed in parallel in the second stage. For example, different algorithms discovering

unrelated artifacts may be run in parallel. In such a case it is assumed that each specific data-processing algorithm receives the same data set, has its own set of processors, and all the many algorithms are executed independently of each other. The longest path in the data-processing would always go to the processor(s) running the most time-consuming algorithm. Consequently, in the following we analyze only one of the possibly many parallel paths in data-processing. Namely, the longest one.

A. First Stage Filtering Intensity and Complexity

The intensity of filtering is expressed by fraction $F \in (0,1]$ controlling the amount of produced results. The amount of results delivered from stage 1 to stage 2 is FV, where V is the size of data injected into the first stage from the sensor. This volume of data is filtered in time A_0V. It is assumed that the intensity of filtering F and speed of filtering are interdependent. Precisely, the inverse of filtering speed is some function $A_0(F)$. Depending on the application, $A_0(F)$ may assume various forms. Here we list a few possibilities:

Case 1: $\mathbf{A_0(F) = 1/F^c}$, where $c > 0$ is some constant. This kind of relationship may emerge as a consequence of iterative filtering. Let i be the number of iterations that are executed on each data unit, then A_0 and F can be expressed as

$$F = f^i \quad (1)$$

$$A_0 = a^i, \quad (2)$$

where $f \in (0,1)$ is the fraction of data remaining after each iteration of filtering, and f and $a > 1$ are some given constants determined by the filtering algorithm. This means that an iterative filtering algorithm is executed on *each* data unit, and extending the algorithm by each new iteration takes exponentially longer to process a data unit. This can be the case when each data unit (e.g. a picture) is rectified with increasing resolution. From equation (1), we get $i = \frac{\ln F}{\ln f}$. From (2), $\ln A_0 = i \ln a$, and hence, $\ln A_0 \ln f = \ln F \ln a$. Equivalently, we have $\ln A_0 = \ln F \ln a / \ln f$, and hence, $A_0 = F^{\frac{\ln a}{\ln f}}$. Since $f \in (0,1), a > 1$, we have $\frac{\ln a}{\ln f} < 0$ and $A_0(F) = \frac{1}{F^c}$, where $c = -\frac{\ln a}{\ln f} > 0$.

Case 2: $\mathbf{A_0(F) = c \ln(1/F)}$. Again all filtering iterations are executed on each data unit, each iteration takes the same time and reduces output data size $f \in (0,1)$ times. Then as in the previous case $F = f^i$, $i = \frac{\ln F}{\ln f}$, but since each iteration takes the same time, we have $A_0 = ai$, and hence we obtain $A_0(F) = c \ln(1/F)$, where $c = -\frac{a}{\ln f} > 0$.

Case 3: $\mathbf{A_0(F) = c(1 - \ln F)}$. Suppose the filtering is a *sieve*, i.e., the filtering algorithm sifts data in some buffer and with each iteration part of the data is dropped. Suppose $\frac{1}{j}$-th of the initial data is removed in iteration j. Thus, after i iterations $V \times \frac{1}{2} \times \frac{2}{3} \times \cdots \times \frac{i-1}{i} = V/i = FV$ data units remain. Hence, $F = 1/i$. The filtering time T_1 is proportional to the diminishing data sizes: $T_1 = aV(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{i}) = aH_iV = A_0V$, where H_i is the ith harmonic number. Since $H_i \le 1 + \ln i$, $A_0(F) = aH_i \le a(1 + \ln 1/F) = c(1 - \ln F)$, where $a = c$ is some constant.

Case 4: $\mathbf{A_0(F) = c(1 - F)}$. The data filtering is a sieve again, and after i iterations the remaining data size is $FV = f^iV$ where $f \in (0,1)$. Filtering time is $T_1 = A_0V = aV \sum_{j=1}^{i} f^i = aV \frac{1-f^i}{1-f}$. Hence, $F = f^i$, $A_0 = a\frac{1-f^i}{1-f}$. Consequently, $A_0(F) = \frac{a(1-F)}{1-f} = c(1-F)$ where $\frac{a}{1-f} = c$ is constant.

Let us observe that in all the above cases $A_0(F)$ decreases with F which means that more intensive filtering (F decreases) requires longer computation ($A_0(F)$ increases).

B. Inter-Stage Communication

The flow of results from stage 1 to stage 2 can be organized in a number of ways. Here we assume two alternatives:

A. Sequential communication – time to transfer x_i bytes to server i and x_j bytes to server j is $C_ix_i + C_jx_j$.

B. Parallel communication – stage 1 to server connections are mutually independent. Time to transfer x_i bytes from stage 1 to server i and x_j bytes to server j is equal to $\max\{C_ix_i, C_jx_j\}$.

C. Computational Complexity of the Second Stage

Computational complexity of the second stage depends on the executed algorithms, necessary result integration and storage. We adopt divisible load theory assumption [10], [12], [13] that the data processed in the second stage is arbitrarily divisible and can be processed in parallel. Potential ways of parallelization are determined by a particular algorithm. Since the variety of possible second stage algorithms and ways of parallelizing them seems unlimited, we will consider a limited set of archetype algorithms as examples of typical computational complexity functions:

Linear – The time to process x units of data is A_ix on processor i. Typical examples are searching for patterns, compression, message digest (e.g. MD5) calculation or scoring data units (like in the LHC example). Here we assume that result collection time is negligible because the size of output data is small or the results are left on the servers and storing is included in the algorithm run time. Since result return is neglected, it can be shown that in the optimum schedule all servers must finish computations at the same moment [10], [12], [13].

Loglinear – Sorting is a typical example of an algorithm with loglinear complexity. Classic sequential sorting algorithms like heapsort, quicksort have complexity $O(n \log n)$, where n is the number of sorted items. We will assume that a parallel version of these algorithms consists in splitting the volume of data into parts, sorting the parts in parallel, and then sequentially merging the results. Thus, on m identical processors the complexity of this method would be $O((n/m)\log(n/m) + n \log m)$.

Quadratic - Computing a similarity matrix can serve as an example of an algorithm with quadratic computational complexity. Its sequential complexity is $O(n^2)$. In the case of parallel processing, the square area of work, e.g. a similarity matrix, may be partitioned into $\ell \times \ell$ squares distributed to processors, where integer ℓ is a tunable parameter of the

algorithm. There are ℓ^2 tiles each of of size n^2/ℓ^2. Sending and processing one square takes $O(n^2/\ell^2)$ time. If equal numbers of ℓ^2/m tiles are assigned to each of m processors, then receiving and processing them can be executed in time $O(n^2/\ell^2 \times \ell^2/m)$ which is $O(n^2/m)$. Note that tile distribution may be different, that is, it may depend on the communication and computing speeds of the processors.

IV. Optimum Filtering Intensity

Our goal in this section is to derive close-form solutions (i.e. formulas) linking optimum filtering intensity F with other system parameters to minimize the time required to transmit and process all data. In many cases the obtained formulations are not amenable to analytic solutions. In such a situation further study is delegated to numerical modeling described in the next section.

A. Parallel Communication, Linear Second Stage

In the optimum schedule, all servers communicate and process in parallel, finishing at the same time T. Hence, we have

$$\alpha_1(A_1 + C_1) = \alpha_i(A_i + C_i) \qquad i = 2, \ldots, m \quad (3)$$

$$FV = \sum_{i=1}^{m} \alpha_i \quad (4)$$

In the above equation system all processors communicate and compute in the same interval by (3), and all work is done by (4). From (3) we get $\alpha_i = (A_1 + C_1)/(A_i + C_i)\alpha_1$ and

$$FV = (A_1 + C_1) \sum_{i=1}^{m} \frac{\alpha_1}{A_i + C_i} = K(A_1 + C_1)\alpha_1, \quad (5)$$

where

$$K = \sum_{i=1}^{m} \frac{1}{A_i + C_i} \quad (6)$$

is a constant. The schedule length is a sum of filtering, communication and processing times:

$$T = A_0(F)V + (A_1 + C_1)\alpha_1 = A_0(F)V + FV/K. \quad (7)$$

Thus, in order to minimize T, we have to minimize the function

$$t(F) = A_0(F) + F/K. \quad (8)$$

We will now compute the optimum value of F for the considered functions $A_0(F)$. Note that practical values of F belong to some interval $[F_{min}, F_{max}] \subset (0, 1]$. Thus, if the computed optimum value F^* is larger than F_{max}, we should set $F = F_{max}$. Similarly, if $F^* < F_{min}$, then the smallest possible value of F should be chosen.

1) $\mathbf{A_0(F) = 1/F^c}$, where $c > 0$ is a constant. Then,

$$t(F) = F^{-c} + F/K, \quad (9)$$

$$t'(F) = -cF^{-(c+1)} + 1/K \quad (10)$$

and

$$t''(F) = c(c+1)F^{-(c+2)} > 0. \quad (11)$$

Thus, $t(F)$ is minimized when $t'(F) = 0$, i.e., for

$$cF^{-(c+1)} = 1/K, \quad (12)$$

$$F = (Kc)^{1/(c+1)}. \quad (13)$$

2) $\mathbf{A_0(F) = c \ln(1/F)}$, where $c > 0$ is a constant. We have

$$t(F) = -c \ln F + F/K, \quad (14)$$

$$t'(F) = -c/F + 1/K \quad (15)$$

and

$$t''(F) = c/F^2 > 0. \quad (16)$$

Hence, $t(F)$ is minimized when

$$F = cK. \quad (17)$$

3) $\mathbf{A_0(F) = c(1 - \ln F)}$, where $c > 0$ is a constant. Then,

$$t(F) = c(1 - \ln F) + F/K, \quad (18)$$

$$t'(F) = -c/F + 1/K \quad (19)$$

and

$$t''(F) = c/F^2 > 0. \quad (20)$$

The minimum value of $t(F)$ is obtained for

$$F = cK. \quad (21)$$

4) $\mathbf{A_0(F) = c(1 - F)}$, where $c > 0$ is a constant. Now we have

$$t(F) = c(1 - F) + F/K, \quad (22)$$

$$t'(F) = -c + 1/K. \quad (23)$$

Thus, $t'(F)$ does not depend on F. If $c > 1/K$, then $t(F)$ is decreasing, and the maximum possible value of F should be chosen. If $c < 1/K$, then $t(F)$ is increasing and the smallest possible F should be selected.

B. Sequential Communication, Linear Second Stage

We assume that the sensor communicates with the processors in the order of their identifiers. Hence, we have

$$\alpha_i(A_i + C_i) = \alpha_{i-1}A_{i-1} \qquad i = 2, \ldots, m \quad (24)$$

$$FV = \sum_{i=1}^{m} \alpha_i \quad (25)$$

Equations (24) mean that communication to and computation on processor i is preformed in parallel with computation on processor $i - i$. It follows implicitly that processor i is started after activating processor $i - 1$. Hence, we obtain

$$\alpha_i = \frac{\alpha_1 \prod_{j=1}^{i-1} A_j}{\prod_{j=2}^{i}(A_j + C_j)} \qquad i = 2, \ldots, m, \quad (26)$$

$$FV = \alpha_1 \sum_{i=1}^{m} \frac{\prod_{j=1}^{i-1} A_j}{\prod_{j=2}^{i}(A_j + C_j)} \quad (27)$$

and

$$T = A_0(F)V + (A_1 + C_1)\alpha_1 = A_0(F)V + FV/L, \quad (28)$$

where

$$L = \sum_{i=1}^{m} \frac{\prod_{j=1}^{i-1} A_j}{\prod_{j=1}^{i}(A_j + C_j)}. \tag{29}$$

Equation (28) has the same form as (7), and hence, the considerations from Section IV-A can be also applied in the case of sequential communication.

C. Parallel Communication, Loglinear Second Stage

In the case of loglinear complexity of the second stage, it is assumed that processing consists of three steps: parallel communication, parallel processing chunks of data, and sequential merging of the results. The latter can be executed in time $FVC_M \log m$, where FV is the amount of data that has to be collected, C_M is reciprocal of merging speed (e.g. in sec/byte) which is taking into account the speed of communication between the servers providing data to merge and the merging server, $\log m$ is a factor representing time to elect the smallest value among m servers in the merging step. The former two steps (parallel communication and processing) take the same time T_1 on all servers. Hence we have:

$$T_1 = C_i\alpha_i + A_i\alpha_i \ln \alpha_i \qquad i = 1, \ldots, m. \tag{30}$$

Let us define

$$y_i = \frac{C_i}{A_i} + \ln \alpha_i. \tag{31}$$

We have

$$\begin{aligned} y_i e^{y_i} &= \left(\frac{C_i}{A_i} + \ln \alpha_i\right) e^{\frac{C_i}{A_i}} \alpha_i = \\ &= \frac{1}{A_i}\left(C_i\alpha_i + A_i\alpha_i \ln \alpha_i\right) e^{\frac{C_i}{A_i}} = \frac{T_1}{A_i} e^{\frac{C_i}{A_i}}. \end{aligned} \tag{32}$$

Recall that if $ye^y = x$ then $y = W(x)$, where W is the Lambert function [25]. Thus, we have

$$y_i = W\left(\frac{T_1}{A_i} e^{\frac{C_i}{A_i}}\right), \tag{33}$$

and (30) can be written as

$$T_1 = A_i\alpha_i W\left(\frac{T_1}{A_i} e^{\frac{C_i}{A_i}}\right). \tag{34}$$

Hence, the load chunk sizes are:

$$\alpha_i = \frac{T_1}{A_i W\left(\frac{T_1}{A_i} e^{\frac{C_i}{A_i}}\right)}. \tag{35}$$

The Lambert function cannot be expressed in terms of elementary functions, but T_1 can be found numerically by solving

$$FV = \sum_{i=1}^{m} \frac{T_1}{A_i W\left(\frac{T_1}{A_i} e^{\frac{C_i}{A_i}}\right)}. \tag{36}$$

The above equation is easier to solve in homogeneous systems because all processors have the same parameters and load to process is split equally. Then, each processor receives load of size $\alpha_i = \frac{FV}{m}$ and equation (36) becomes:

$$FV = \frac{mT_1}{AW\left(\frac{T_1}{A} e^{\frac{C}{A}}\right)}. \tag{37}$$

Moreover, we get from (33)

$$\begin{aligned} W\left(\frac{T_1}{A} e^{\frac{C}{A}}\right) &= \frac{C}{A} + \ln \alpha_i = \frac{C}{A} + \ln\left(\frac{FV}{m}\right) = \\ &= \ln\left(\frac{FV e^{\frac{C}{A}}}{m}\right) \end{aligned} \tag{38}$$

and hence,

$$T_1 = \frac{AFV}{m} \ln\left(\frac{FV e^{\frac{C}{A}}}{m}\right). \tag{39}$$

The schedule length including filtering, communication and processing is

$$T(F) = A_0(F)V + \frac{AFV}{m} \ln\left(\frac{FV e^{\frac{C}{A}}}{m}\right) + FVC_M \ln m \tag{40}$$

We will now compute the optimum value of F for which T is minimum.

1) $\mathbf{A_0}(\mathbf{F}) = \mathbf{1}/\mathbf{F^c}$, where $c > 0$ is a constant. Then,

$$\begin{aligned} T'(F) = -cVF^{-(c+1)} + \frac{AV}{m}\left[\ln\left(\frac{FV e^{\frac{C}{A}}}{m}\right) + 1\right] + \\ + VC_M \ln m \end{aligned} \tag{41}$$

$$T''(F) = c(c+1)VF^{-(c+2)} + \frac{AV}{mF} > 0 \tag{42}$$

Hence, $T(F)$ is a minimum when $T'(F) = 0$.

2) $\mathbf{A_0}(\mathbf{F}) = \mathbf{c}\ln(\mathbf{1}/\mathbf{F})$, where $c > 0$ is a constant. We have

$$\begin{aligned} T'(F) = -cV/F + + \frac{AV}{m}\left[\ln\left(\frac{FV e^{\frac{C}{A}}}{m}\right) + 1\right] + \\ + VC_M \ln m \end{aligned} \tag{43}$$

and

$$T''(F) = cV/F^2 + \frac{AV}{mF} > 0 \tag{44}$$

Again, for F satisfying $T'(F) = 0$, $T(F)$ is minimum.

3) $\mathbf{A_0}(\mathbf{F}) = \mathbf{c}(\mathbf{1} - \ln \mathbf{F})$, where $c > 0$ is a constant, is dealt in the same way as the previous case because $[c\ln(1/F)]' = c(1 - \ln F)' = -c/F$.

4) $\mathbf{A_0}(\mathbf{F}) = \mathbf{c}(\mathbf{1} - \mathbf{F})$, where $c > 0$ is a constant.

$$T'(F) = -cV + \frac{AV}{m}\left[\ln\left(\frac{FV e^{\frac{C}{A}}}{m}\right) + 1\right] + VC_M \ln m \tag{45}$$

and

$$T''(F) = \frac{AV}{mF} > 0 \tag{46}$$

and $T(F)$ is a minimum when $T'(F) = 0$.

D. Sequential Communication, Loglinear Second Stage

In this case communications and parts of processed load are linked by the system of equations:

$$C_{i+1}\alpha_{i+1} + A_{i+1}\alpha_{i+1}\ln\alpha_{i+1} = A_i\alpha_i\ln\alpha_i \quad i = 1,\ldots,m-1 \tag{47}$$

$$FV = \sum_{i=1}^{m}\alpha_i \tag{48}$$

Equations (47) ensure that work on processor $i-1$ is processed in parallel with communication to and computation on processor i. This set of nonlinear equations does not seem to have an easy analytical solution. Therefore, we will recourse to numerical methods to solve (47)-(48) and find F for which the processing time is minimum.

E. Parallel Communication, Quadratic Second Stage

As mentioned in Section III we assume that the quadratic amount of work is shared between the m processors. This amount of work can be split into ℓ^2 work units, each of size $(FV/\ell)^2$. We will assume that ℓ is large and hence work can be sufficiently flexibly divided as in the linear case. Yet, mind that the amount of work, i.e. data to be processed, grows proportionately to $(FV)^2$. Furthermore, a homogeneous system is considered. Similarly to the linear case (Section IV-A), results are not explicitly merged. We have

$$T_1 = \alpha_i\left(A\left(\frac{FV}{\ell}\right)^2 + 2C\left(\frac{FV}{\ell}\right)\right) \quad i = 2,\ldots,m \tag{49}$$

$$\ell^2 = \sum_{i=1}^{m}\alpha_i \tag{50}$$

Equations (49) mean that communication and processing is performed in the same interval on all processors. Since the system is homogeneous, $\alpha_i = \ell^2/m$, for $i = 1,\ldots,m$, the whole schedule length is

$$T(F) = A_0(F)V + \frac{\ell^2}{m}\left(A\left(\frac{FV}{\ell}\right)^2 + 2C\left(\frac{FV}{\ell}\right)\right) \tag{51}$$

We will now compute the optimum value of F for which T is minimum.

1) $\mathbf{A_0}(\mathbf{F}) = \mathbf{1}/\mathbf{F^c}$, where $c > 0$ is a constant. Then,

$$T'(F) = -cF^{-(c+1)}V + \frac{2FAV^2}{m} + \frac{2CV\ell}{m} \tag{52}$$

$$T''(F) = c(c+1)VF^{-(c+2)} + \frac{2AV^2}{m} > 0 \tag{53}$$

Hence, $T(F)$ is minimum when $T'(F) = 0$. Unfortunately, equation (52) does not seem to have an easy analytical solution for $T'(F) = 0$ and has to be solved numerically.

2) $\mathbf{A_0}(\mathbf{F}) = \mathbf{c}\ln(\mathbf{1}/\mathbf{F})$, where $c > 0$ is a constant. We have

$$T'(F) = -cV/F + \frac{2FAV^2}{m} + \frac{2CV\ell}{m} \tag{54}$$

$$T''(F) = cV/F^2 + \frac{2AV^2}{m} > 0 \tag{55}$$

Again, $T(F)$ is minimum when $T'(F) = 0$ and

$$F = \frac{\sqrt{4C^2V^2\ell^2/m^2 + 8cAV^3/m} - 2CV\ell/m}{4AV^2/m} \tag{56}$$

3) $\mathbf{A_0}(\mathbf{F}) = \mathbf{c}(\mathbf{1} - \ln\mathbf{F})$, where $c > 0$ is a constant, is dealt in the same way as the previous case because $[c\ln(1/F)]' = c(1-\ln F)' = -c/F$.
4) $\mathbf{A_0}(\mathbf{F}) = \mathbf{c}(\mathbf{1} - \mathbf{F})$, where $c > 0$ is a constant.

$$T'(F) = -cV + \frac{2FAV^2}{m} + \frac{2CV\ell}{m} \tag{57}$$

and

$$T''(F) = \frac{2AV^2}{m} > 0. \tag{58}$$

Hence, $T(F)$ is minimum when $F = \frac{mc-2C\ell}{2AV}$.

F. Sequential Communication, Quadratic Second Stage

We have a set of equations determining work distribution:

$$2C\alpha_{i+1}\left(\frac{FV}{\ell}\right) + A\alpha_{i+1}\left(\frac{FV}{\ell}\right)^2 = A\alpha_i\left(\frac{FV}{\ell}\right)^2 \quad i = 1,\ldots,m-1 \tag{59}$$

$$\ell^2 = \sum_{i=1}^{m}\alpha_i \tag{60}$$

Unfortunately, this set of nonlinear equations does not seem to have an easy analytical solution for F minimizing the schedule length. Therefore, we will recourse to numerical methods to solve (59)-(60) and find F for which the processing time is minimum.

V. Numerical Modeling

This section is dedicated to showing tendencies in the system parameters when the filtering intensity is optimum with respect to the minimum total processing time. In cases not amenable to representation with a closed-form formula, the optimum value of F was found by use of Python method `scipy.optimize.fsolve`. We assume that the amount of input data is $V = 1\text{E}6$. We will analyze recurring patterns in performance rather than particular numbers. Therefore, only representative examples of the cases introduced in Section III-A are extensively discussed. For simplicity, only homogeneous systems are analyzed.

A. Parallel Communication, Linear Second Stage

Fig. 2 presents the relationship between the retained data fraction F and the schedule length T in case 2, i.e. $A_0(F) = c\ln(1/F)$, for $m = 100$ and several combinations of A, C and c values. The smallest value of F for which T is shown in Fig. 2 is 0.01, because F must be greater than 0. When $A = C = 5$ and $c = 0.001$, filtering is fast, while data transfer and processing in the second stage are rather slow. Hence, the smaller amount of data is retained, the shorter schedule is obtained. Contrarily, when $A = C = 1$ and $c = 0.3$, filtering

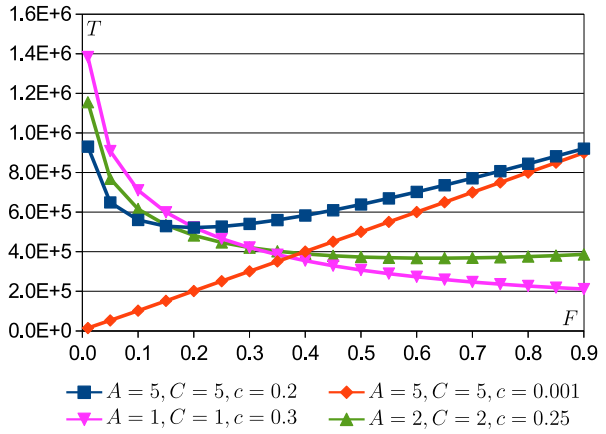

Fig. 2. T vs. F for parallel communication, linear second stage, case 2, $m = 100$.

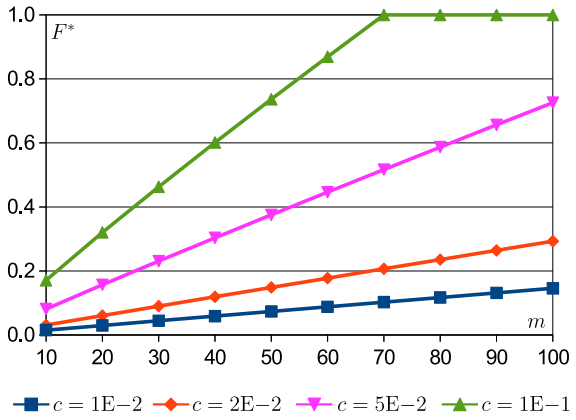

Fig. 3. F^* vs. m for parallel communication, linear second stage, case 1, $A = 5$, $C = 2$.

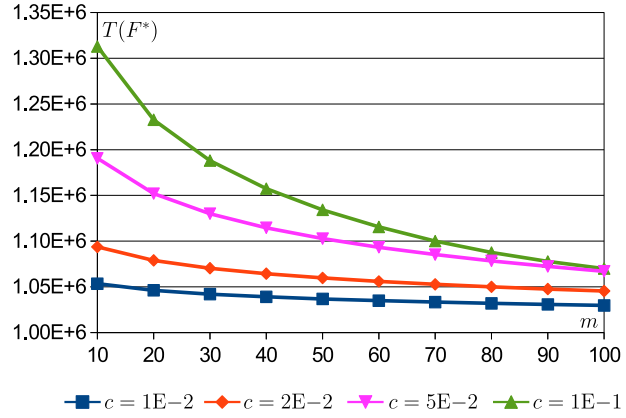

Fig. 4. $T(F^*)$ vs. m for parallel communication, linear second stage, case 1, $A = 5$, $C = 2$.

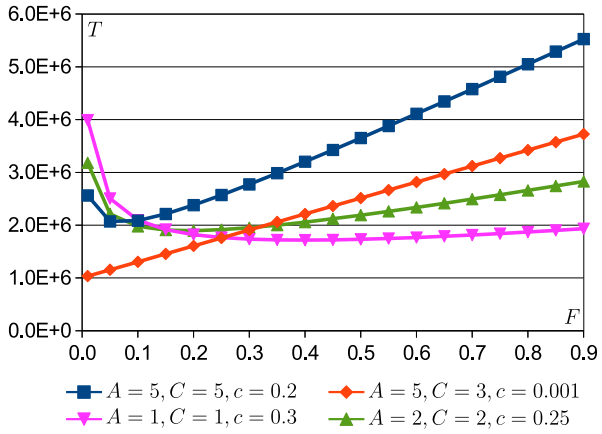

Fig. 5. T vs. F for sequential communication, linear second stage, case 1, $m = 10$.

is slow, while data transfer and processing are rather fast. In consequence, larger F (i.e. lower filtering intensity) results in a smaller schedule length T. In the remaining two presented cases, the optimum value of F is neither the minimum possible (close to 0) nor the maximum possible (1). For $A = C = 5$ and $c = 0.2$, the best value of F is 0.2, and for $A = C = 2$, $c = 0.25$ it is 0.65.

Fig. 3 shows how the optimum value of retained data fraction F^* depends on the number m of second stage processors, for case 1 ($A_0(F) = 1/F^c$) with $A = 5$ and $C = 2$. When m grows, parallel data transfer and processing take less time in comparison to the filtering stage. Therefore, a larger fraction of data should be retained, in order to decrease the filtering time. Naturally, the optimal filtering intensity decreases when filtering is slow, i.e., for large c. In particular, when $c = 1\text{E}{-1}$ and $m \geq 70$, no filtering should take place.

The total processing time resulting from filtering the optimum size of data for different values of c and m is depicted in Fig. 4. Naturally, the schedule length decreases when more processors are used. This effect is stronger when c is large. Indeed, in this case, decreasing filtering intensity (F increases, $A_0(F)$ decreases), which is possible because of using a larger number of processors, has a large impact on the filtering time.

B. Sequential Communication, Linear Second Stage

When communication is sequential, a smaller number of second stage processors can be effectively used than in the case of parallel communication. Therefore, in Fig. 5, we present the schedule lengths obtained for different values of F and network parameters, $m = 10$, and for filtering case 1. In general, the visible tendencies are similar to the ones in Fig. 2. However, even when $A = C = 1$ and $c = 0.3$, the optimum value of F is much smaller than 1, i.e. filtering must be more intensive than for parallel communication. The best among values analyzed here is 0.4. Indeed, sequential communication is a bottleneck, and even very costly filtering can be beneficial, because it decreases the communication time.

The optimum data fractions F^* for $A = 5$ and $C = 2$ are presented in Fig. 6. The values are much smaller than in the case of parallel communication (see Fig. 3). As we already explained, intensive filtering decreases the communication time, which dominates in the schedule length. The fraction

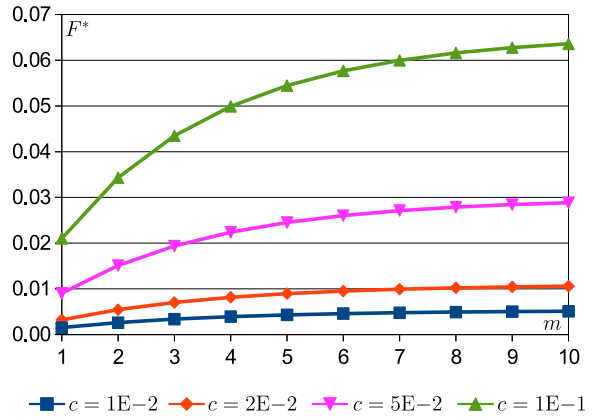

Fig. 6. F^* vs. m for sequential communication, linear second stage, case 1, $A = 5$, $C = 2$.

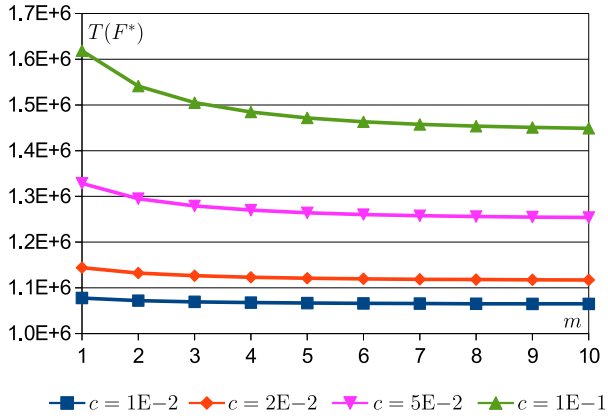

Fig. 7. $T(F^*)$ vs. m for sequential communication, linear second stage, case 1, $A = 5$, $C = 2$.

of retained data grows with increasing m at a slower pace than in the case of parallel communication.

The optimum schedule lengths are shown in Fig. 7. Using a larger number of servers results in a shorter processing time, but the impact of changing m is smaller than in Fig. 4, because using more processors m does not decrease the communication time.

C. Parallel Communication, Loglinear Second Stage

Fig. 8 presents the schedule lengths T obtained for different values of F and network parameters, for parallel communication, loglinear second stage case 4. i.e. $A_0(F) = c(1 - F)$. Recall that in the case of linear second stage and case 4, the function $T(F)$ was always monotonous (Section IV-A, equations (22, 23)). It can be seen in Fig. 8 that when the second stage complexity is loglinear, $T(F)$ is also monotonous for many choices of network parameters, but not for all of them. In particular, when $A = 10$, $C = 1$, $c = 1$ and $C_M = 0.01$, the best among analyzed values of F is 0.45.

In cases 1 and 2 of data filtering complexity, the impact of increasing the number of processors m on the optimum value of F and schedule length is similar as for linear processing

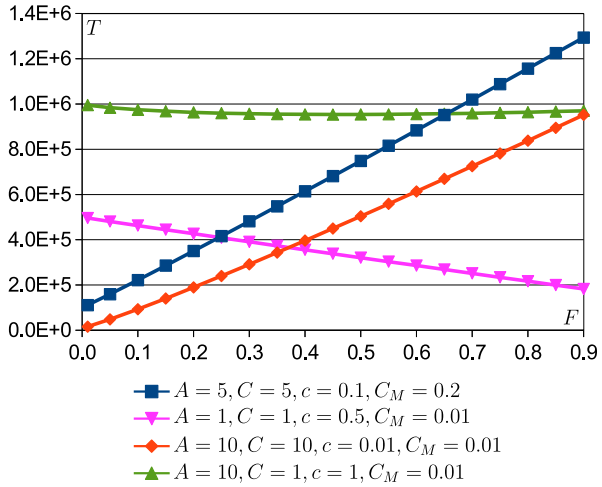

Fig. 8. T vs. F for parallel communication, loglinear second stage, case 4, $m = 100$.

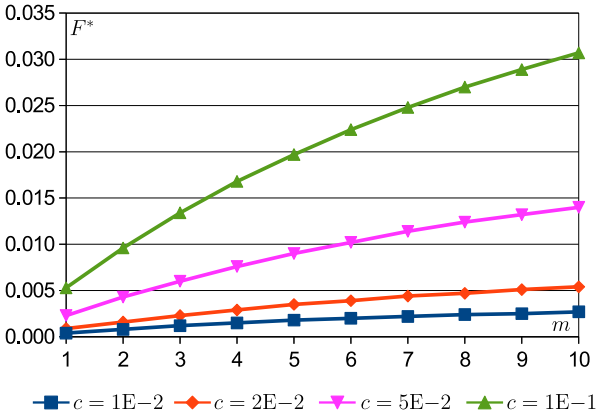

Fig. 9. F^* vs. m for sequential communication, loglinear second stage, case 1, $A = C = 3$, $C_M = 1\text{E}{-2}$.

complexity. The main difference is that the time of merging the results also influences the results. When C_M is big, the merging stage becomes a bottleneck. Hence, more intensive filtering is required to reduce its duration and thus to obtain an optimum schedule.

D. Sequential Communication, Loglinear Second Stage

The differences between systems with sequential and parallel communication in the case of loglinear second stage are similar to those present when the second stage is linear. Since sequential communication is a bottleneck, optimum schedules are obtained by more intensive data filtering. Fig. 9 shows that even if m is large and filtering is slow, the fraction of retained load should be at most several percent in case 1 of filtering complexity. The optimum fractions obtained for cases 2, 3 and 4 are even smaller.

E. Parallel Communication, Quadratic Second Stage

When the second stage complexity is quadratic, intensive data filtering is required to obtain a short schedule by de-

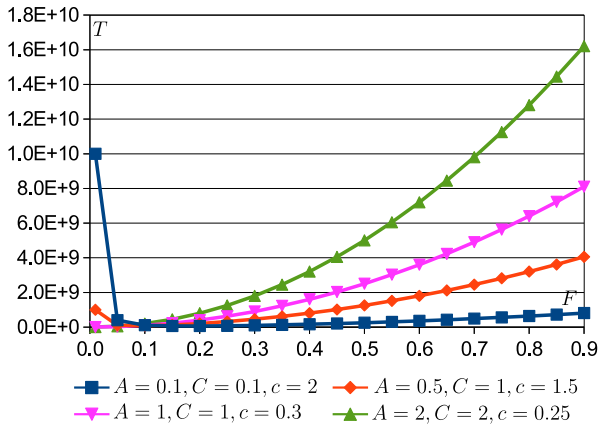

Fig. 10. T vs. F for parallel communication, quadratic second stage, case 1, $m = 100$.

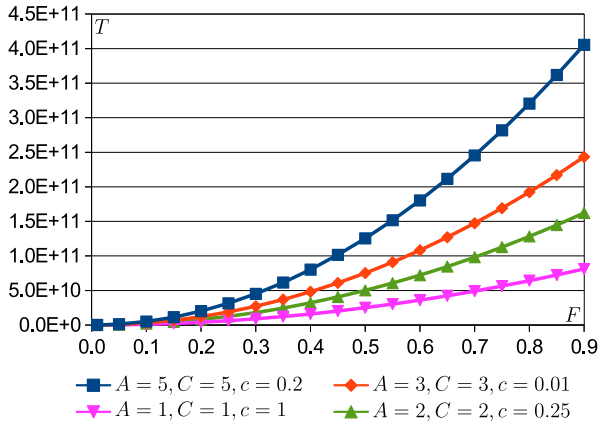

Fig. 11. T vs. F for sequential communication, quadratic second stage, case 1, $m = 10$.

creasing the duration of processing. It can be seen in Fig. 10, representing case 1 of the filtering complexity, that only when c is really large (i.e. $c > 1$), it may not be beneficial to decrease the data size as much as possible. In cases 2, 3 and 4 the fraction of retained data should be practically always as small as possible.

F. Sequential Communication, Quadratic Second Stage

When communication is sequential and the second stage complexity is quadratic, very intensive filtering should be used, even if it is costly. For all combinations of parameter values we studied, $T(F)$ is an increasing function (see Fig. 11). Although the optimum fraction F^* increases slightly with growing m, it stays below 0.01 for all settings we tested. Taking into account the practical limitations on F, this means that the smallest possible amount of data should be retained.

VI. Conclusions

In this paper we analyzed impact of data filtering intensity on the performance of systems with single expensive sensors. The analysis covered two-stage systems with generic representations of filtering algorithms, communication patterns and data processing methods. It appears that due to the interaction of nonlinear complexity of filtering, transmission time and further data processing stages, there exists filtering intensity which is optimum for the overall processing performance. These optima were investigated both analytically and computationally. It appeared that the communication subsystem and the second stage algorithm complexity have a large impact on the first stage filtering intensity. The ability of certain combinations of the system designs to scale is very limited. In systems with sequential communication gains from using parallel processors in the second stage are quickly diminishing because data transfer easily becomes a bottleneck. Processing with algorithms of high complexity (loglinear, quadratic) should be delegated to even further stages of data processing workflows because they incur needs for filtering intensities which may be hard to realize. Thus, by exposing scalability issues we demonstrated in this paper that designers of workflows with data filtering and distributed processing should strive for parallel data transfers and linear processing algorithms when handling large volumes of data from the sensors.

References

[1] M. Moges and T. Robertazzi, "Wireless sensor networks: Scheduling for measurement and data reporting," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 327–340, 2006. doi: 10.1109/TAES.2006.1603426

[2] J. Berlińska, "A comparison of priority rules for minimizing the maximum lateness in tree data gathering networks," *Engineering Optimization*, vol. 54, pp. 218–231, 2022. doi: 10.1080/0305215X.2020.1861263

[3] ——, "Scheduling in data gathering networks with background communication," *Journal of Scheduling*, vol. 23, pp. 681–691, 2020. doi: 10.1007/s10951-020-00648-5

[4] ——, "Heuristics for scheduling data gathering with limited base station memory," *Annals of Operations Research*, vol. 285, pp. 149–159, 2020. doi: 10.1007/s10479-019-03185-3

[5] ——, "Scheduling for data gathering networks with data compression," *European Journal of Operations Research*, vol. 246, pp. 744–749, 2015. doi: 10.1016/j.ejor.2015.05.026

[6] T. Colombo, "Trigger & DAQ at the LHC, filtering data from 50 TB/s to 1 GB/s," https://indico.cern.ch/event/825688/attachments/1872900/3082664/trigger_daq_at_lhc.pdf, CERN EP/LBC, July 2019, accessed 31/3/2022.

[7] Committee on U.S.-Based Electron-Ion Collider Science Assessment, *An Assessment of U.S.-Based Electron-Ion Collider Science*. Washington D.C.: The National Academy of Science, Engineering and Medicine, The National Academy Press, 2018.

[8] C. Toth and C. Jóźków, "Remote sensing platforms and sensors: A survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 22–36, 2016. doi: 10.1016/j.isprsjprs.2015.10.004

[9] Y.-C. Cheng and T. Robertazzi, "Distributed computation with communication delay," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, no. 6, pp. 700–712, 1988. doi: 10.1109/7.18637

[10] V. Bharadwaj, D. Ghose, V. Mani, and T. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[11] T. Robertazzi, "Ten reasons to use divisible load theory," *IEEE Computer*, vol. 36, no. 5, pp. 63–68, 2003. doi: 10.1109/MC.2003.1198238

[12] M. Drozdowski, *Scheduling for Parallel Processing*. London: Springer, 2009.

[13] H. Casanova, A. Legrand, and Y. Robert, *Parallel Algorithms*. London, UK: CRC Press, Taylor and Francis, 2009.

[14] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, pp. 102–114, 2002. doi: 10.1109/MCOM.2002.1024422

[15] P. Pereira, A. Grilo, and F. Rocha, *End-to-End Reliability in Sensor Networks: Survey and Research Challenges in P. Pereira (ed), EuroFGI Workshop in IP Qos and Traffic Control*. Academia, 2007.

[16] T. Muhammed and A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, pp. 267–287, 2017. doi: 10.1016/j.jnca.2016.10.019

[17] M. Dener, "Security analysis in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 10, p. 303501, 2014. doi: 10.1155/2014/303501

[18] W. Meng, L. Xie, and W. Xiao, "Optimality analysis of sensor-source geometries in heterogeneous sensor networks," *IEEE Transactions on Wireless Communication*, vol. 12, pp. 1958–1967, 2013. doi: 10.1109/twc.2013.021213.121269

[19] L. Cao, Y. Cai, and Y. Yue, "Swarm intelligence-based performance optimization for mobile wireless sensor networks: Survey, challenges, and future directions," *IEEE Access*, vol. 7, pp. 161 524–161 553, 2019. doi: 10.1109/access.2019.2951370

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

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

[22] C. Li and W. Luo, "Exact and approximation algorithms for minimizing energy in wireless sensor data gathering network with data compression," *American Journal of Mathematical and Management Sciences*, vol. 41, no. 4, pp. 305–315, 2022. doi: 10.1080/01966324.2021.1960226

[23] J. Berlińska and M. Drozdowski, "Scheduling divisible mapreduce computations," *Journal of Parallel and Distributed Computing*, vol. 71, no. 3, pp. 450–459, 2011. doi: 10.1016/j.jpdc.2010.12.004

[24] ——, "Comparing load-balancing algorithms for mapreduce under zipfian data skews," *Parallel Computing*, vol. 72, pp. 14–28, 2018. doi: 10.1016/j.parco.2017.12.003

[25] Wikipedia contributors, "Lambert W function," https://en.wikipedia.org/wiki/Lambert_W_function, [Online; accessed 5-August-2022].