Research Article

Multivariate Multiple Regression Models for a Big Data-Empowered SON Framework in Mobile Wireless Networks

Yoonsu Shin, Chan-Byoung Chae, and Songkuk Kim

School of Integrated Technology, Yonsei Institute of Convergence Technology, Yonsei University, Incheon, Republic of Korea

Correspondence should be addressed to Songkuk Kim; songkuk@yonsei.ac.kr

Received 22 April 2016; Accepted 26 July 2016

Academic Editor: Yeong M. Jang

Copyright © 2016 Yoonsu Shin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the 5G era, the operational cost of mobile wireless networks will significantly increase. Further, massive network capacity and zero latency will be needed because everything will be connected to mobile networks. Thus, self-organizing networks (SON) are needed, which expedite automatic operation of mobile wireless networks, but have challenges to satisfy the 5G requirements. Therefore, researchers have proposed a framework to empower SON using big data. The recent framework of a big data-empowered SON analyzes the relationship between key performance indicators (KPIs) and related network parameters (NPs) using machine-learning tools, and it develops regression models using a Gaussian process with those parameters. The problem, however, is that the methods of finding the NPs related to the KPIs differ individually. Moreover, the Gaussian process regression model cannot determine the relationship between a KPI and its various related NPs. In this paper, to solve these problems, we proposed multivariate multiple regression models to determine the relationship between various KPIs and NPs. If we assume one KPI and multiple NPs as one set, the proposed models help us process multiple sets at one time. Also, we can find out whether some KPIs are conflicting or not. We implement the proposed models using MapReduce.

1. Introduction

The technology of self-organizing networks (SON) has been developed to more economically manage wireless communication and mobile networks in increasingly complex environments [1, 2]. SON, however, do not fully handle data from all sources in mobile wireless networks such as mobile app-based data (mobile data) and channel baseband power (wireless communication information) [3, 4]. Thus, SON encounter challenges that hinder the current self-organizing networking paradigm from meeting the 5G requirements because 5G networks are more complex [4].

Engineers have thus come up with a big data-empowered SON (BSON), which develops a SON with big data in mobile wireless networks. BSON, currently a necessary technology for 5G [3–6], is still in its initial stage. Indeed, in its current iteration, it is insufficient for practical use. The BSON framework was proposed in [4] and includes the concrete concept of using big data in mobile wireless networks and applied them to SON. It ranks the key performance indicators (KPIs), selects network parameters (NPs) related to each KPI,

and creates a Gaussian process regression model in which the KPI is the dependent variable and each NP related to this KPI is the independent variable. The Gaussian process regression models are then applied to the SON engine for management optimization. In this context, the KPIs include capacity, quality of service (QoS), capital expenditure (CAPEX), and operational expenditure (OPEX) from the perspective of a wireless communication operator. In addition, from a user perspective, the KPIs include seamless connectivity, spatiotemporal uniformity of service, demand for almost infinite capacity or zero latency, and cost of service. For instance, because 5G technology aims to connect everything such as automobile, wearable devices, and home network and to help human escape emergency situations, massive network capacity and zero latency are needed in a wireless ecosystem.

This BSON framework [4], however, has some aspects that need be improved. For example, the individual selection of NPs related to a KPI is considerably intricate because a typical 5G node is expected to have more than 2000 parameters. Moreover, in a single Gaussian process regression model, computing an exact KPI value according to the

NP values is difficult [7]. To address these problems, we proposed multiple regression models [8], which allow us easily distinguish the NPs related to each KPI from those unrelated ones. Simultaneously, we can generate models that can be immediately applied to the SON engine. Because of the many available NPs with massive values, we need to solve the issues concerning the multiplication of two large-sized matrices and the inverse of a large-sized matrix for multiple regressions using MapReduce. We describe and implement method that calculates matrices consisting of a KPI and NPs for multiple regression models using MapReduce [8].

These multiple regression models, however, suffer from weaknesses. We can calculate the relationship between only one KPI and the NPs at a single time. However, recognizing the relationship between various KPIs and all the NPs at a single time is important because some KPIs are conflicting, such as the relationship between QoS and CAPEX. If we want to know the relation between the KPIs and NPs, we need to individually calculate the multiple regressions of each KPI. Therefore, in this paper, we proposed improved models, namely, multivariate multiple regression models, that help us determine the relationship between the KPIs and NPs at a single time. We explain these models in the next section.

The remainder of this paper is organized as follows. Section 2 provides the background of big data in 5G and BSON framework, multiple regression models, MapReduce, and LU decomposition. Section 3 explores the proposed multivariate multiple regression models for the BSON framework and describes the implementation of these models using MapReduce. Section 4 presents the theoretical time complexity of the algorithms of these models. Section 5 presents the implementation of MapReduce and the execution time for executing these models in a cloud as a result. Finally, we conclude this paper in Section 6.

2. Background and Related Work

SON facilitates automatic operation of mobile wireless networks. It initially exploits big data in mobile wireless networks to improve the networks. This current research is devoted to BSON [3]. The researcher in [4] proposed the BSON framework.

2.1. SON. Operating mobile wireless networks is a challenging task, especially in cellular mobile communication systems due to their latent complexity. This complexity arises from the number of network elements and interconnections among their configurations. In heterogeneous networks, handling various technologies and their precise operational paradigms is difficult. Today, planning and optimization tools are typically semiautomated and the management tasks need to be closely supervised by human operators. This manual effort by a human operator is time-consuming, expensive, and error prone and requires a high degree of expertise. SON can be used to reduce the operating costs by reducing the tasks at hand and enhancing profit by minimizing human error. The next subsection details the SON taxonomies.

2.1.1. Self-Configuration. Configuration of base stations (eNBs), relay stations, and femtocells is required during deployment, extension, and upgrade of network terminals.

Configurations may also be needed when a change in the system is required, such as failure of a node, drop in network performance, or change in service type. In future systems, the conventional process of manual configuration must be replaced with self-configuration. We can foresee that nodes in future cellular networks should be able to self-configure all their initial parameters including the IP addresses, neighbor lists, and radio-access parameters.

2.1.2. Self-Optimization. After the initial self-configuration phase, we need to continuously optimize the system parameters to ensure efficient performance of the system to maintain all optimization objectives. Optimization in the legacy systems can be done through periodic drive tests or analysis from log reports generated from network operating center. Self-optimization includes load balancing, interference control, coverage extension, and capacity optimization.

2.1.3. Self-Healing. Wireless cellular systems are prone to faults and failures due to component malfunctions or natural disasters. In traditional systems, failures are mainly detected by the centralized operation and maintenance (O&M) software. Events are recorded and necessary alarms are set off. When alarms cannot be remotely cleared, radio network engineers are usually mobilized and sent to cell sites. This process could take days or even weeks before the system returns to normal operation. In future self-organized cellular systems, this process needs to be improved by consolidating the self-healing functionality. Self-healing is a process that consolidates remote detection, diagnosis, and triggering of compensation or recovery actions to minimize the effect of faults in the mobile wireless network equipment.

2.2. Big Data in 5G and BSON. The massive amount of information comes from various elements in the mobile wireless networks, such as base stations, mobile terminals, gateways, and management entities, as shown in Figure 1 [3]. The authors in [4] classified the big data in cellular networks as follows.

2.2.1. Subscriber Level Data. This classification contains control data, contextual data, and voice data, which not only can be used to optimize, configure, and plan network-centric operations, but also are equally meaningful to support key business processes such as customer experience and retention enhancement.

2.2.2. Cell Level Data. This classification contains physical layer measurements that are reported by a base station and all user equipment within the coverage of this base station to the O&M center. The utilities of the cell level data can complement the subscriber level data. For example, minimization of drive test measurements, which contains the reference signal received power and reference signal received quality values of the serving and adjacent cells, are particularly useful for autonomous coverage estimation and optimization [9].

2.2.3. Core Network Level Data. This classification can be exploited to fully automate fault detection and troubleshoot

FIGURE 1: Big data gathering path in mobile wireless network architecture.

network level problems. The complexity of identifying problems in a core network is increased many times, particularly if the equipment used is supplied by different vendors that provide their own proprietary solutions for different network performance.

2.2.4. Additional Sources of Data. This classification contains the structured information already stored in the separate databases, including customer relationship management as well as billing data. This also includes unstructured information such as social media feeds, specific application usage patterns, and data from smart phone built-in sensors and applications.

As discussed in the Introduction, SON technology uses this aforementioned big data to improve itself. This process is facilitated using BSON. The three main features that make BSON distinct from the state-of-the-art SON are the following:

- (i) full intelligence of the current network status,
- (ii) capability in predicting user behavior,
- (iii) capability in dynamically associating the network response to the NPs.

These three capabilities can go a long way in designing a SON that can meet the 5G requirements. The BSON framework shown in Figure 2 involves the following steps.

Step 1 (data gathering). This includes gathering of data from all sources of information into an aggregate data set.

Step 2 (transforming). This includes transforming the big data into right data.

The steps in this transformation are explained below. The underlying machine learning and data analytics are subsequently explained.

- (1) *Classifying.* This means classifying the data with respect to key operational and business objectives

(OBOs) in which accessibility, retainability, integrity, mobility, and business intelligence are present.

- (2) *Unifying/Diffusing.* This means unifying multiple PIs into more significant KPIs.
- (3) *Ranking.* This means ranking KPIs within each OBO with respect to their effect on that OBO.
- (4) *Filtering.* This means filtering out KPIs that affect the OBO below a predefined threshold.
- (5) *Relating.* This means, for each KPI, finding the NP that affects that KPI.
- (6) *Ordering.* This means, for each KPI, ordering the associated NP with respect to the strength of their association.
- (7) *Cross-Correlation.* This means, for each NP, determining a vector that quantifies its association with each KPI.

Step 3 (modeling). This includes developing a network behavior model by learning from the right data obtained in Step 2 using the Gaussian process regression and Kolmogorov-Wiener prediction.

Step 4 (running the SON engine). This includes using the SON engine on the model to determine a new NP and expected new KPIs.

Step 5 (validating). If the simulated behavior tallies with the expected behavior (KPIs), proceed with the new NPs.

Step 6 (relearning/improving). If the validation in Step 5 fails, make feedback to the concept drift block, which updates in turn the behavior model.

2.3. Multiple Regression Models [8, 10]. Step 2 (transforming) and Step 3 (modeling) presented in Section 2.2 (BSON framework) are replaced with the multiple regression models. The key factors in Step 2 (transforming) and Step 3 (modeling) are finding the associated NPs for each KPI and creating the model using a KPI and the associated NPs. They should, however, separately determine the associated NPs using machine-learning tools [11]. Moreover, calculating the accurate value of a KPI according to the change in the NP values is difficult. In other words, the model presented in Section 2.2 allows us to determine the value of a KPI according to only one NP because the model is merely a single regression model.

The single regression model shown in Figure 2 identifies the relationship between a KPI and only one NP. Of course, many single regression models exist according to the NPs, but calculating a KPI value when the NP values simultaneously change is difficult. In contrast, the multiple regression models shown in Figure 3 enable easy identification of the relationship between a KPI and the NPs.

We proposed the multiple regression models to enhance the previous BSON framework [8]. The multiple regression model is written in [10] as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m + \varepsilon, \quad (1)$$

FIGURE 2: Big data-empowered SON framework [4].

FIGURE 3: Multiple regression models.

and it can be expressed as

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E}, \quad (2)$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & \text{NP}_1 & \text{NP}_2 & \dots \\ 1 & \cdot & & \\ 1 & \cdot & & \\ 1 & \cdot & & \end{bmatrix}, \quad (3)$$

$$\mathbf{Y} = \begin{bmatrix} \text{KPI}_k \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}.$$

The elements in $\mathbf{X}$ and $\mathbf{Y}$ are the values of the NPs and KPI, and the parameter is estimated as

$$\hat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1} (\mathbf{X}'\mathbf{Y}). \quad (4)$$

We can create multiple regression models ($\hat{\mathbf{B}}$) by calculating the multiplication of $(\mathbf{X}'\mathbf{X})^{-1}$ and $(\mathbf{X}'\mathbf{Y})$. Figure 3 shows four steps to compute $\hat{\mathbf{B}}$ using MapReduce, and we provided the detail of each step in [8].

2.4. Matrix Multiplication Using MapReduce [12, 13]. MapReduce is a computation method that has been implemented in several systems, including Google internal implementation and the popular open-source implementation Hadoop. (Hadoop can be obtained, along with Hadoop Distributed

FIGURE 4: Block method for LU decomposition.

File System from the Apache Foundation.) We can use an implementation of MapReduce to manage many large-scale computations in a manner that is tolerant of hardware faults. Only two functions need to be written—Map and Reduce—while the system manages the parallel execution, coordinates tasks that execute Map or Reduce, and deals with the possibility that one of these tasks will fail to execute.

Matrix Multiplication with One MapReduce Step. If $\mathbf{M}$ is a matrix with element m_{ij} in row i and column j and $\mathbf{N}$ is a matrix with element n_{jk} in row j and column k , then the product, $\mathbf{P} = \mathbf{MN}$, is matrix $\mathbf{P}$ with element p_{ik} in row i and column k , where

$$p_{ik} = \sum_j m_{ij} n_{jk}. \quad (5)$$

We can possibly use only a single MapReduce pass to perform the matrix multiplication, $\mathbf{P} = \mathbf{MN}$. Here, we present an abstract of the Map and Reduce functions.

- (1) *Map Function.* For each element m_{ij} of $\mathbf{M}$, we produce all key-value pairs $((i, k), (M, j, m_{ij}))$ for $k = 1, 2, \dots$ up to the number of columns of $\mathbf{N}$. Similarly, for each element n_{ij} of $\mathbf{N}$, we produce all key-value pairs $((i, k), (N, j, n_{ij}))$ for $i = 1, 2, \dots$ up to the number of columns of $\mathbf{M}$.
- (2) *Reduce Function.* Each key (i, k) will have an associated list with all values (M, j, m_{ij}) and (N, j, n_{ij}) , for all possible values of j . The j th values on each list must have their third components, namely, m_{ij} and n_{jk} , extracted and multiplied. Then, these products are added, and the result is paired with (i, k) in the output of the Reduce function.

2.5. Matrix Inversion Using MapReduce [14]. The LU algorithm splits the matrix into square submatrices and individually updates these submatrices. The block method splits the input matrix, as shown in Figure 4.

In this method, the lower triangular matrix $\mathbf{L}$ and the upper triangular matrix $\mathbf{U}$ are both split into three submatrices, whereas the original matrix $\mathbf{A}$ is split into four

submatrices. These smaller matrices satisfy the following equations:

$$\begin{aligned} \mathbf{L}_1 \mathbf{U}_1 &= \mathbf{P}_1 \mathbf{A}_1, \\ \mathbf{L}_1 \mathbf{U}_2 &= \mathbf{P}_1 \mathbf{A}_2, \\ \mathbf{L}_2' \mathbf{U}_1 &= \mathbf{A}_3, \\ \mathbf{L}_3 \mathbf{U}_3 &= \mathbf{P}_2 (\mathbf{A}_4 - \mathbf{L}_2' \mathbf{U}_2), \\ \mathbf{L}_2 &= \mathbf{P}_2 \mathbf{L}_2', \end{aligned} \quad (6)$$

where both $\mathbf{P}_1$ and $\mathbf{P}_2$ are permutations of the rows. The entire LU decomposition can be represented as

$$\mathbf{LU} = \begin{pmatrix} \mathbf{P}_1 & 0 \\ 0 & \mathbf{P}_2 \end{pmatrix} \mathbf{A} = \mathbf{PA}, \quad (7)$$

where $\mathbf{P}$ is also a permutation of the rows obtained by augmenting $\mathbf{P}_1$ and $\mathbf{P}_2$.

If submatrix $\mathbf{A}_1$ is sufficiently small (e.g., on the order of 10^3 or less), it can be very efficiently decomposed into $\mathbf{L}_1$ and $\mathbf{U}_1$ on a single node. If submatrix $\mathbf{A}_1$ is not small enough, we can recursively partition it into smaller submatrices, as shown in Figure 4. After obtaining $\mathbf{L}_1$ and $\mathbf{U}_1$, the elements of $\mathbf{L}_2'$ and $\mathbf{U}_2$ can be computed using the following two equations:

$$\begin{aligned} [\mathbf{L}_2']_{ij} &= \frac{1}{[\mathbf{U}_1]_{jj}} \left([\mathbf{A}_3]_{ij} - \sum_{k=1}^{j-1} [\mathbf{L}_2']_{ik} [\mathbf{U}_1]_{kj} \right), \\ [\mathbf{U}_2]_{ij} &= \frac{1}{[\mathbf{L}_1]_{ii}} \left([\mathbf{A}_2]_{ij} - \sum_{k=1}^{j-1} [\mathbf{L}_1]_{ik} [\mathbf{U}_2]_{kj} \right). \end{aligned} \quad (8)$$

We can compute $\mathbf{A}_4 - \mathbf{L}_2' \mathbf{U}_2$ using the $\mathbf{L}_2'$ and $\mathbf{U}_2$ matrices mentioned above. Subsequently, we can decompose it into $\mathbf{L}_3$ and $\mathbf{U}_3$.

3. Multivariate Multiple Regression Models for BSON Framework

The multiple regression models presented in Section 2.3 suffer from a shortcoming—they can calculate the relationship between only one KPI and NPs. Many KPIs exist, however, such as those from the operator perspective that include OPEX, CAPEX, QoS, and capacity and from the user perspective that include seamless connectivity, cost of service, capacity, and latency [4]. These are high-level KPIs; however, many precise technical KPIs also exist, such as the cell power and cell coverage. To reveal the relationship between the KPIs and NPs, we must calculate the multiple regression models several times for each KPI in the previous multiple regression models. This process is inconvenient and requires a long time.

Meanwhile, finding the conflicting or concordant relationship among KPIs is not easy when the NP values simultaneously change. As we mentioned earlier, we should perform multiple regressions several times for each KPI to finally learn the conflicting or concordant relationship among KPIs.

FIGURE 5: Proposed multivariate multiple regression models.

In contrast, the proposed multivariate multiple regression models shown in Figure 5 allow simultaneous determination of the relationship between the KPIs and NPs.

To enhance the multiple regression models for BSON, we propose the multivariate multiple regression models. The multivariate multiple regression is expressed as follows [15, 16]:

$$Y_j = \beta_0 + \beta_1 z_{j1} + \beta_2 z_{j2} + \dots + \beta_r z_{jr} + \varepsilon_j, \quad (9)$$

and it can also be expressed as

$$\mathbf{Y}_{n \times p} = \mathbf{Z}_{n \times (r+1)} \mathbf{B}_{(r+1) \times p} + \varepsilon_{n \times p}, \quad (10)$$

where

$$\mathbf{Z} = \begin{bmatrix} 1 & \text{NP}_1 & \text{NP}_2 & \dots & \text{NP}_r \\ 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad (11)$$

$$\mathbf{Y} = \begin{bmatrix} \text{KPI}_1 & \text{KPI}_2 & \dots & \text{KPI}_p \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

The elements in $\mathbf{Z}$ and $\mathbf{Y}$ are values of NPs and KPIs, and the parameter is estimated as

$$\hat{\mathbf{B}}_{(r+1) \times p} = (\mathbf{Z}'\mathbf{Z})^{-1} (\mathbf{Z}'\mathbf{Y}). \quad (12)$$

We can create multivariate multiple regression models ($\hat{\mathbf{B}}$) by calculating the multiplication of $(\mathbf{Z}'\mathbf{Z})^{-1}$ and $(\mathbf{Z}'\mathbf{Y})$. Figure 5 shows four steps to compute $\hat{\mathbf{B}}$ using MapReduce, and we specifically describe each step below.

Step 1 (integrating). Each message has limited information such as the location, time, reception sensitivity, cell power,

mobile power, data traffic, and mobility status. Hence, we simultaneously integrate the whole messages to determine the values of the KPIs according to the NPs in the Map function. Then, we extract the values of the KPI and all the NPs in the Reduce function. The MapReduce key-value pair in Step 1 is presented in Algorithm 1.

In the Map function, the key is time, and the value is the name and value of each NP and KPI. When the Map tasks are all completed, the key-value pairs are grouped in terms of time. Thus, the input of the Reduce task contains the corresponding information and the key-value pairs are grouped according to each KPI (i.e., KPI_k) in the Reduce tasks. Therefore, we can simultaneously obtain the value of each KPI and NP as the output of the Reduce tasks.

For example, if we take one sample per minute for 1 hour, we can obtain 60 samples. Assuming that the numbers of NPs and KPIs are 30 and 10, respectively, then the orders of $\mathbf{Z}$ and $\mathbf{Y}$ are 60×30 and 60×10 , respectively. Therefore, we can convert key (i.e., time_ℓ), NP_m elements and KPI_n elements in the Reduce function into the ℓ th row of $\mathbf{Z}$ and $\mathbf{Y}$ and the m th column of $\mathbf{Z}$ and n th column of $\mathbf{Y}$, respectively.

Algorithm 1 (the MapReduce key-value pair of Step 1).

The Map Function

{time, (NP₁, NP₁ value, NP₂, NP₂ value, ..., KPI₁, KPI₁ value, KPI₂, KPI₂ value, ...)}

The Reduce Function

{time, (NP₁, NP₁ value, NP₂, NP₂ value, ..., KPI₁, KPI₁ value, KPI₂, KPI₂ value, ...)}

Step 2 (computing $\mathbf{Z}'\mathbf{Z}$ and $\mathbf{Z}'\mathbf{Y}$). We compute $\mathbf{Z}'\mathbf{Z}$ and $\mathbf{Z}'\mathbf{Y}$ using the result in Step 1. Because the result in Step 1 includes the $\mathbf{Z}$ and $\mathbf{Y}$ matrices, we can easily compute $\mathbf{Z}'\mathbf{Z}$ and $\mathbf{Z}'\mathbf{Y}$ using MapReduce. As noted in Section 2.4, we can obtain matrix multiplication with one MapReduce step [12]. For instance, if we calculate matrix multiplication, $\mathbf{P} = \mathbf{M}\mathbf{N}$, m_{ik} is used to obtain $p_{i1}, p_{i2}, \dots, p_{ij}$ (j is the number of columns

in $\mathbf{N}$). Therefore, through m_{ik} forking off the j th elements in the Map function, we can calculate the element of $\mathbf{P}_{ij}$ in the Reduce function at the same time.

The MapReduce key-value pair in Step 2 is presented in Algorithm 2. Note that $Z'Z, Z'Y, Z', Z$, or Y are the names of these matrices and not of the entire matrix. Note also that k reaches up to the number of samples (i.e., time), i reaches up to the number of NPs plus one, and ℓ reaches up to the number of KPIs.

Algorithm 2 (the MapReduce key-value pair of Step 2).

The Map Function

$\{(Z'Z, i, j), (Z', k, z'_{ik})\}$ for $j = 1, 2, \dots$ up to the number of columns of $\mathbf{Z}$

$\{(Z'Z, i, j), (Z, k, z_{kj})\}$ for $i = 1, 2, \dots$ up to the number of rows of $\mathbf{Z}'$

or

$\{(Z'Y, i, \ell), (Z', k, z'_{ik})\}$ for $\ell = 1, 2, \dots$ up to the number of columns of $\mathbf{Y}$

$\{(Z'Y, i, \ell), (Y, k, y_{i\ell})\}$ for $i = 1, 2, \dots$ up to the number of rows of $\mathbf{Z}'$

The Reduce Function

$\{(Z'Z, i, j), (Z'\mathbf{Z}_{ij} \text{ value})\}$ or

$\{(Z'Y, i, \ell), (Z'\mathbf{Y}_{i\ell} \text{ value})\}$

Step 3 (computing $(\mathbf{Z}'\mathbf{Z})^{-1}$). To calculate the multivariate multiple regression, we compute $(\mathbf{Z}'\mathbf{Z})^{-1}$ using the result in Step 2. However, computing the inverse of a matrix using MapReduce is difficult when the order of the matrix is large. Fortunately, the authors in [14] proposed a method of matrix inversion using MapReduce. They proposed a block method for scalable matrix inversion using MapReduce. The block method enables parallel calculation of the LU decomposition. If the order of the matrices is not large ($\leq 10^3$), the matrix can be very efficiently decomposed into $\mathbf{L}$ and $\mathbf{U}$ on a single node. If the order of the matrices is not large, sequentially calculating the inverse of a matrix using $\mathbf{LU}$ decomposition in one node becomes easy. We can compute the $\mathbf{L}$ and $\mathbf{U}$ matrices using the following equations for the $\mathbf{LU}$ decomposition algorithm [14, 17]:

$$\begin{aligned} u_{ij} &= a_{ij} - \sum_{k=1}^{j-1} \ell_{jk} u_{kj}, \\ \ell_{ij} &= \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj} \right). \end{aligned} \quad (13)$$

We can then easily compute $\mathbf{L}^{-1}$ using the following equations [14], and the inverse of the upper triangular matrix ($\mathbf{U}^{-1}$) can be equivalently computed. We invert upper triangular

matrix, $\mathbf{U}$, by calculating the inverse of $\mathbf{U}^T$, which is a lower triangular matrix ($\mathbf{L}$):

$$[\mathbf{L}^{-1}]_{ij} = \begin{cases} 0 & \text{for } i < j \\ \frac{1}{[\mathbf{L}_{ii}]} & \text{for } i = j \\ -\frac{1}{[\mathbf{L}_{ii}]} \sum_{k=j}^{i-1} [\mathbf{L}]_{ik} [\mathbf{L}^{-1}]_{kj} & \text{for } i > j. \end{cases} \quad (14)$$

The output key-value pair in Step 3 is presented in Algorithm 3. Note that $(\mathbf{Z}'\mathbf{Z})^{-1}$ is the name of this matrix, and not of the entire matrix.

Algorithm 3 (the output key-value pair of Step 3).

$$\{((\mathbf{Z}'\mathbf{Z})^{-1}, i, j), ((\mathbf{Z}'\mathbf{Z})^{-1} \text{ value})\}$$

Step 4 (computing $\hat{\mathbf{B}}$). We compute $\hat{\mathbf{B}} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{Y}$ using the results in Steps 2 and 3. We perform the multiplication of two matrices (i.e., $(\mathbf{Z}'\mathbf{Z})^{-1}$ and $\mathbf{Z}'\mathbf{Y}$) using MapReduce. We can also perform matrix multiplication using one MapReduce step such as in Step 2 [12].

The MapReduce key-value pair in Step 4 is presented in Algorithm 4. Note that $(\mathbf{Z}'\mathbf{Z})^{-1}$ and $(\mathbf{Z}'\mathbf{Y})$ are the names of these matrices and not of the entire matrix in the Map function. In the Reduce function, the j th element of $(\mathbf{Z}'\mathbf{Z})^{-1}$ multiplies the j th element of $\mathbf{Z}'\mathbf{Y}$ in same (i, k) key; then all the results are added. The result is the (i, k) element of $\hat{\mathbf{B}}$. In the Reduce function, note that i reaches up to the number of NPs plus one, and k reaches up to the number of KPIs.

Algorithm 4 (the MapReduce key-value pair of Step 4).

The Map Function

$\{(i, k), ((\mathbf{Z}'\mathbf{Z})^{-1}, j, (\mathbf{Z}'\mathbf{Z})_{ij}^{-1})\}$ for $k = 1, 2, \dots$ up to number of rows of $(\mathbf{Z}'\mathbf{Y})$

or

$\{(i, k), ((\mathbf{Z}'\mathbf{Y}), j, (\mathbf{Z}'\mathbf{Y})_{jk})\}$ for $i = 1, 2, \dots$ up to number of rows of $(\mathbf{Z}'\mathbf{Z})^{-1}$

The Reduce Function

$$\{(i, k), (\beta_{(i-1)k})\}$$

We can recognize that estimated parameters (i.e., $\beta_{(i-1)k}$) separate the NPs from the NPs unrelated to a KPI. If $\beta_{(i-1)k}$ is close to zero at KPI_k , then NP_{i-1} is unrelated to KPI_k . In addition, we can identify whether a conflicting or concordant relationship among KPIs exists. For example, if the sign of all row elements of β_{ip} and β_{iq} for KPI_p and KPI_q are totally different, these KPIs are conflicting. Otherwise, they are concordant.

FIGURE 6: MapReduce pipeline for estimate parameter (β_{ik}).

TABLE 1: Time complexity of matrix multiplication.

Matrix	Input order	Output order	Time complexity
$Z' \times Z$	$(r+1) \times n$ $n \times (r+1)$	$(r+1) \times (r+1)$	$O(r^2n)$
$(Z'Z)^{-1}$	$(r+1) \times (r+1)$	$(r+1) \times (r+1)$	$O(r^3)$
$Z' \times Y$	$(r+1) \times n$ $n \times p$	$(r+1) \times p$	$O(nrp)$
$(Z'Z)^{-1} \times (Z'Y)$	$(r+1) \times (r+1)$ $(r+1) \times p$	$(r+1) \times p$	$O(r^2p)$

4. Time Complexity of Multiple Regression Models

We calculate the time complexity of the multivariate multiple regression models. The result of the multivariate multiple regression models can be obtained as the product of $(Z'Z)^{-1}$ and $Z'Y$. The time complexity of $Z'Z$ is $O(r^2n)$ because the order of Z is $n \times (r+1)$. The time complexities of $Z'Y$, $(Z'Z)^{-1}$ and $(Z'Z)^{-1}Z'Y$ are $O(n \times r \times p)$, $O(r^3)$, and $O(r^2 \times p)$, respectively, as listed in Table 1 [18, 19]. Thus, the entire time complexity of the multivariate multiple regression models is $O(r^3)$ when $n < r$.

We can reduce this time complexity using distributed programming such as MapReduce. Let $T(L)$ be the time complexity of N tasks. $T(L)$ can then be presented as follows, assuming an ideal case without consideration of a network bottleneck:

$$T(L) = \frac{T(1)}{L}. \quad (15)$$

Thus, the time complexity of the N tasks is $O(r^3/L)$, and if L is sufficiently large, we can obtain almost constant or linear time complexity, which shows that the time complexity of the proposed models is equal to that of the multiple regression models [8].

5. Implementation in MapReduce

We implemented our models using Hadoop 2.7.1 [20, 21]. All experiments were performed in our laboratory cluster, which has 32 machines. Each machine has four CPU cores and 24 GB of memory, where each CPU is an Intel® Xeon® CPU X5650 at 2.67 GHz.

FIGURE 7: Execution time for calculating each phase (MR_i) according to the number of tasks.

For implementation in MapReduce, several phases were required. Thus, we had a pipeline of MapReduce jobs as shown in Figure 6. MR_i is one MapReduce job. Three phases are required to calculate $(Z'Z)^{-1}$.

In MR_0 , we computed the product of Z' and Z . In MR_1 , we computed the L and U matrices using (13). In addition, in MR_1 , we can easily compute L^{-1} using (14), and the inverse of the upper triangular matrix (U^{-1}) can be equivalently computed. We inverted upper triangular matrix, U , by calculating the inverse of U^T , which is a lower triangular matrix (L). Finally, in MR_2 , we computed $(Z'Z)^{-1}$ as the product of U^{-1} and L^{-1} .

Meanwhile, MR_3 is required to calculate $Z'Y$. From the output of MR_2 and MR_3 , we can calculate estimated parameters (i.e., β_{ik}) as the product of $(Z'Z)^{-1}$ and $Z'Y$. In reference to Section 3, Step 1 phase creates Z and Y . Step 2 presents MR_0 and MR_3 . Step 3 presents MR_1 and MR_2 . Finally, Step 4 presents MR_4 .

In this implementation, we compared the execution time according to the number of MapReduce jobs as shown in Figure 7. We used 600×400 matrix as input Z and 600×100

FIGURE 8: Execution time for calculating estimated parameter (β_{ik}).

matrix as input Y . Thus, the order of estimated parameter (i.e., β_{ik}) was 400×100 . In a practical experiment, we need to calculate a large order of matrices. Much time, however, is needed to calculate matrix multiplication in our cloud when the matrices are in a large order. Hence, we reduced the order of matrices and simply compared the execution times according to the number of tasks.

Figure 7 shows the execution time for calculating each phase (i.e., MR_i). In Figure 7, the execution times of MR_0 , MR_2 , MR_3 , and MR_4 are linearly reduced when the number of Reduce tasks increases from 10 to 20. They, however, later gradually decreased when the number of Reduce tasks increases from 20 to 50 because network bottleneck, communication cost, or additional management time exists [22, 23].

To the left of the three bars in MR_1 in Figure 7, we can see the execution time for calculating MR_1 on a single node. No reduction in the execution time can be observed by increasing the Map tasks. Thus, if we want to reduce the execution time in MR_1 , we need to use parallel LU decomposition.

The last bar in MR_1 in Figure 7 shows the execution time for calculating MR_1 using parallel LU decomposition as presented in Section 2.5. On a single node (i.e., one Reduce), this process takes approximately 110 s to calculate the LU decomposition of a 400×400 matrix and to obtain the inverse of L and U matrices, whereas, on parallel LU, we split the 400×400 matrix into four submatrices, from A_1 to A_4 (the order of each matrix is 200×200), and then obtain L_1 , L_2 , L_3 , U_1 , U_2 , and U_3 as presented in Section 2.5. We need two MapReduce phases and require 89 s to calculate the results to be the same as those in a single node.

Figure 8 shows the total execution time to obtain estimated parameter (i.e., β_{ik}). By increasing the number of tasks, the execution time is reduced. If we can increase the task capacity by building additional machines in a cluster, we may be able to calculate matrix operations faster than we can currently perform. In addition, we can easily perform numerous matrix operations using MapReduce.

Figure 9 shows the comparison of the execution times to calculate MR_3 and MR_4 when we use multiple regression and multivariate multiple regression models. The reason why we compare these two models using only MR_3 and MR_4 is that

FIGURE 9: Execution time for calculating MR_3 and MR_4 according to the number of tasks when the order of Y is 600×100 or 600×1 .

MR_0 , MR_1 , and MR_2 of the two models are the same. We consider only one KPI at a time in the multiple regression models. Thus, the order of the Y matrix is 600×1 . In the multivariate multiple regression models, we consider 100 KPIs; thus, the order of the Y matrix is 600×100 . Given the complexity of matrix multiplication, it is likely that the execution time for MR_3 and MR_4 in the multiple regression models is 100 times faster than that in the multivariate multiple regression models.

In Figure 9, however, the execution time for MR_3 and MR_4 when the order of Y matrix is 600×1 is about 1.4 times faster than that when the order of Y matrix is 600×100 . This happens because a minimum amount of time is needed for MapReduce execution, which includes time for forking Map, sorting, and merging Reduce. Therefore, in this case, multivariate multiple regression models are more efficient than multiple regression models.

6. Conclusion

In BSON, recent research has indicated that a framework using machine-learning tools and the Gaussian process regression model facilitates a more automatic operation of SON. This approach suffers from some limitations. However, although it determines NPs individually related to a KPI, it cannot inform us of the exact value of the KPI according to the change in the NP values. Therefore, we have proposed the multiple regression models to easily determine the relationship between a KPI and the NPs [8]. These multiple regression models, however, were found to have their own shortcomings. If we want to identify the relationship between various KPIs and NPs, we must calculate the multiple regression models several times.

To eliminate these limitations, we have proposed in this paper multivariate multiple regression models. These models separate the NPs unrelated to a KPI from those that are related and allow us to determine at once the relationship between various KPIs and NPs. If $\beta_{(i-1)k}$ is close to zero at KPI_k , then NP_{i-1} is unrelated to KPI_k . Further, we can identify if two KPIs (e.g., KPI_p and KPI_q) are conflicting if the signs of all the row elements of β_{ip} and β_{iq} are entirely different.

We implemented these proposed models using MapReduce. By increasing the number of tasks, the execution time was reduced. We have also shown through experiments that the proposed multivariate multiple regression models are more efficient than the multiple regression models, as shown in Figure 9. Naturally, this approach suffers from limitations, such as communication cost. However, using distributed programming such as MapReduce, we can easily simultaneously calculate numerous matrix operations. We can also possibly achieve faster and more frequent calculations by introducing additional machines in a cluster. In our future work, we will analyze the proposed models using real big data in mobile wireless networks.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Republic of Korea, under the IT Consilience Creative Program (IITP-2015-R0346-15-1008) supervised by the IITP (Institute for Information & Communications Technology Promotion) and the ICT R&D Program of MSIP/IITP (B0126-15-1017).

References

- [1] O. G. Aliu, A. Imran, M. A. Imran, and B. Evans, "A survey of self organisation in future cellular networks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 336–361, 2013.
- [2] S. Hamalainen, H. Sanneck, and C. Sartori, *LTE Self-Organizing Networks (SON)*, John & Wiley Sons, Ltd, New York, NY, USA, 2012.
- [3] N. Baldo, L. Giupponi, and J. Mangues-Bafalluy, "Big data empowered self organized networks," in *Proceedings of the 20th European Wireless Conference (EW '14)*, pp. 181–188, May 2014.
- [4] A. Imran, A. Zoha, and A. Abu-Dayya, "Challenges in 5G: how to empower SON with big data for enabling 5G," *IEEE Network*, vol. 28, no. 6, pp. 27–33, 2014.
- [5] E. J. Khatib, R. Barco, P. Munoz, I. D. La Bandera, and I. Serrano, "Self-healing in mobile networks with big data," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 114–120, 2016.
- [6] E. J. Khatib, R. Barco, A. Gómez-Andrades, P. Muñoz, and I. Serrano, "Data mining for fuzzy diagnosis systems in LTE networks," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7549–7559, 2015.
- [7] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Regression*, MIT Press, Cambridge, Mass, USA, 1996.
- [8] Y. Shin, C.-B. Chae, and S. Kim, "Multiple regression models for a big data empowered SON framework," in *Proceedings of the 7th International Conference on Ubiquitous and Future Networks (ICUFN '15)*, pp. 982–984, IEEE, Sapporo, Japan, July 2015.
- [9] Ö. F. Çelebi, E. Zeydan, Ö. F. Kurt et al., "On use of big data for enhancing network coverage analysis," in *Proceedings of the 20th International Conference on Telecommunications (ICT '13)*, pp. 1–5, Casablanca, Morocco, May 2013.
- [10] R. J. Freund, D. Mohr, and W. J. Wilson, *Statistical Methods*, Elsevier/Academic Press, Amsterdam, Netherlands, 3rd edition, 2010.
- [11] I. Witten and F. Eibe, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.
- [12] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*, Cambridge University Press, Cambridge, UK, 2011.
- [13] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [14] J. Xiang, H. Meng, and A. Aboulmaga, "Scalable matrix inversion using mapreduce," in *Proceedings of the 23rd ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC '14)*, pp. 177–190, ACM, Vancouver, Canada, June 2014.
- [15] J. P. Stevens, *Applied Multivariate Statistics for the Social Sciences*, Routledge, 2012.
- [16] M. Bilodeau and D. Brenner, *Theory of Multivariate Statistics*, Springer Science & Business Media, 2008.
- [17] J. Kiusalaas, *Numerical Methods in Engineering with MATLAB*, Cambridge University Press, 2010.
- [18] D. Serre, *Matrices*, vol. 216 of *Graduate Texts in Mathematics*, Springer, New York, NY, USA, 2nd edition, 2010.
- [19] S. Skiena, *The Algorithm Design Manual*, Springer Science+Business Media, Berlin, Germany, 1998.
- [20] Hadoop: Open source implementation of MapReduce, <http://hadoop.apache.org>.
- [21] V. K. Vavilapalli, A. C. Murthy, C. Douglas et al., "Apache hadoop YARN: yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing (SoCC '13)*, ACM, October 2013.
- [22] A. D. Sarma, F. Afrati, S. Salihoglu, and J. Ullman, "Upper and lower bounds on the cost of a map-reduce computation," *Proceedings of the VLDB Endowment*, vol. 6, no. 4, pp. 277–288, 2013.
- [23] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on MapReduce," *Neuro-computing*, vol. 102, pp. 52–58, 2013.