IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2018, accepted August 24, 2018, date of publication August 29, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2867633

QoS-Aware Rule-Based Traffic-Efficient Multiobjective Service Selection in Big Data Space

T. H. AKILA S. SIRIWEERA AND INCHEON PAIK (Member, IEEE)
School of Computer Science and Engineering, University of Aizu, Fukushima 965-8580, Japan
Corresponding author: Incheon Paik (paikic@u-aizu.ac.jp)

ABSTRACT The number of Web services has increased dramatically during the last few years. This has resulted in an increase in the volume of candidate services for tasks in composition systems. This has led to growth in the variety of nonfunctional properties in service selection, resulting in uncertainty (veracity issues) among such properties, which has severely affected the NP-hard aspects of service selection. Despite this, consumers in many areas would like access to a variety of selection methods such as linear programming and dynamic programming techniques. An additional problem is that the composition length (the number of tasks) of the workflow has increased, with the incorporation of research domains such as data science. These trending composition issues are challenging the computational power of existing methods. Such concerns have opened the door to research involving Big Data space. We propose a flexible, distributed selection algorithm that facilitates heterogeneous-selection methods to satisfy multiobjective composition requirements rather than rigid, specific composition requirements. However, service-selection processes in a Big Data space will inevitably increase traffic congestion caused by the increased volume of internal communication, particularly external traffic, such as Zipf and Pareto phenomena, and internal traffic during shuffling. To address these concerns, we propose solutions for each case. Our experiments demonstrate that the proposed traffic-efficient multiobjective method is well behaved when selecting services in Big Data space.

INDEX TERMS Big Data space, multiobjective service selection, QoS preferences, rule-based, traffic-efficient.

I. INTRODUCTION

The expansion in services has led to increased opportunities. Studies show that the growth in revenue has more than doubled in bi-annually, with an increase more than 220% in service-based data-science-related activities in the Amazon Web Services Platform in 2015, 2016 and 2017 compared to the respective 2013, 2014 and 2015.[1] In addition, studies show a doubling of the volume of services in the ProgrammableWeb.com store each year. Moreover, ProgrammableWeb is becoming a popular platform for well-known providers such as IBM, Google, and Microsoft [1]. This confirms that growth in the consumer market and the availability of services is extensive.

There has been a dramatic surge in the availability of candidate services, which, in turn, has led to ''resource starvation'' in selection processes. The variety and uncertainty (veracity) of quality of services (QoS) among the high volume of candidate services increases the NP-hardness in the search for solutions [1].

Introducing new research areas such as data science (DS) also increases the complexity of the composition system. Lengthy processes such as data analysis (DA) for business intelligence in a DS field typically comprise four different phases: preparation, analysis, reflection, and dissemination [2]. In the preparation stage, data must be collected, formatted, and cleaned. In the analysis stage, the analysis, debugging, and inspection of substages must be cleared. In the reflection stage, there are comparisons and calls to re-execute the substages of the analysis stage. In the dissemination stage, steps such as displaying, deploying, and retrieving the statistics of the detailed process must be at least minimally performed. Each of these substages can contain multiple tasks and modern composition systems have lengthier processes than do their conventional counterparts. We recognize this as a challenge of modern service selection. In response, new DA

[1] www.statista.com/statistics/233725/development-of-amazon-web-services-revenue

research is emerging in DS, involving approaches such as Big Data analytics (BDA) [3], which is lengthier than general DA methods.

It is evident that the native service selection process is a well-known NP-hard problem [35], [36]. Moreover, conventional standalone processing platforms are reaching their limits in dealing with NP-hardness [1]. Thus, we move toward Big Data-related research topics. Hadoop and Spark are currently the most widely used Big Data processing platforms. However, the Spark platform is generally regarded as involving expensive in-memory processing and lacking its own file-management system. This hinders the study of native traffic congestion occurring during the selection process. Hadoop is the most popular native Big Data processing platform. It includes its own file-management system and facilitates a diverse range of techniques for file handling. Therefore, it allows studying the occurring-traffic during the selection in a more effective manner. It also provides a gateway to addressing a diverse range of problems associated with research and industry domains in a cost-effective manner. MapReduce (MR) is a well-known fundamental programming technique in the Hadoop space. Therefore, we devised our selection method based on MR in the Hadoop space.

Service composition is a key component of the highly diverse DS [1], [37]–[39], including the fields of BDA and deep learning. While facilitating these highly diverse composition requirements, a peer user should have the freedom to select and switch between composition requirements for the given problem using the given algorithm without affecting the fundamental data structure, thereby avoiding complex steps during decision-making or changing preferences. However, there are no existing methods that facilitate such dynamically changing composition requirements and most have rigidly specific composition patterns, such as optimizations that are focused only on linear [4], [5], [40] combinatorial [6], [7], [41], [42], or multivariate patterns [1], [8], [43], [44]. Therefore, it is very important to facilitate multiobjective selection methods, which are flexible with respect to different selection requirements without affecting the overall flow of the algorithm. Accordingly, our aim is to propose a multiobjective selection algorithm that facilitates user-driven flexibility about selection methods.

Given this aim, we propose three types of selection approaches by considering the three main types of composition requirements, the first being based on linear programming and the other two involving dynamic programming.

In some cases, users need to find the global optimal QoS requirements from a given set of candidate services. Here, to satisfy the linear programming requirements, we propose a Dijkstra algorithm-based method as a novel technique for selection domain.

In other cases, users need to satisfy a combinatorial optimization of their QoS requirements with respective to a maximum upper bound for the overall negatively affected values, while maximizing the overall positively affected QoS of the services. Here, we propose a 0-1 Multi-Constraint Knapsack Problem (0-1 MCKP) [6], [7], [9] to satisfy these requirements.

Finally, peer users may seek multivariate optimization of their QoS parameters for their composition system. To satisfy this requirement, we propose using selection criteria based on the Artificial Bee Colony (ABC) algorithm. These three methods are integrated into the multiobjective selection method referred to in this paper.

Service-selection processes in Big Data spaces can involve excessive traffic congestion because of both internal and external factors in the MR process [10], [11]. Considering the MR algorithm as the base, shuffling traffic called the internal traffic occurs during the MR process, and ZipF, Pareto traffic occurs due to the hotness of the data and outside of the MR algorithm. Then they are called as the external traffic perspective to the MR algorithm. We refer to these two issues as ''internal traffic'' and ''external traffic,'' respectively, throughout this paper.

Two commonly identified types of external traffic congestions are the Zipf and Pareto phenomena's [11]. They affect negatively to the overall performance of an MR job. Both phenomena's occur naturally in any environment, including local machines, local area networks, and clouds. However, they show particularly adverse behavior in the Hadoop space.

ZipF [45]–[47] and Pareto [46], [48], [49] are native and intrinsic traffic congestions occurring network data processing. However, these two factors show adverse behavior in the Big Data environment [11], [12].

The Zipf phenomenon in a Hadoop distributed file system (HDFS) refers to an access pattern distribution of replicas in a given file according to Zipf's law. This results in a ''hot'' replica (higher access rate) among replicas available in a given file. In the service-selection process, the aim is to deal with QoS service preferences during this process, but the hotness phenomena tend to favor services with lower QoS preferences. This leads to a reduction in the accuracy of the selection process. In addition, because of the limited number of replica preferences, it generates heavy traffic congestion and causes a reduction in overall performance. To address these concerns, we propose a QoS-aware service distribution method.

The Pareto phenomenon in general environments refers to 80% of the data usage being represented by 20% of the data. This is known as the ''80/20 rule'' [12]. However, the Hadoop process demonstrates the further adverse effect in the Hadoop environment. This phenomenon causes to increase the hot files. To address this traffic congestion phenomenon, we propose a traffic-aware service-replica distribution method.

In addition to that, service selection is a natively NP-hard problem for finding the optimal utility (optimized for linear, combinatorial or multivariate) values of QoS in composition plan among services available. To address these issues, we employed the heuristic methods to find the optimal selection composition plan. Then it is obvious, the given selection process always tries to achieve the global/local optimal utility QoS plan among the given services. Then service

selection also generates the 'hot' services and composition plans during the selection process. Therefore, we assume, it has more tendency to affect the intrinsic traffic congestions which are caused by ZipF and Pareto phenomena's than these two traffic congestions appear in the native traffic in the Hadoop environment.

Moreover, one of the most common reasons for internal traffic congestion is data communication between the map and reduce phases [10]. Usually, this shuffles a large chunk of the map results in the reduce phase based on the key value of the MR job, and thereby on the internal traffic of the MR job. Service selection generates an excessive amount of intermediate data and this tends to increase the internal traffic. Therefore, we propose a combiner-based intermediate MR agent to address this issue. We called this method the intermediate MR agent. This agent procedure results in two major benefits, firstly, this reduces traffic congestion between mapper and reducer by reducing the intermediate data, and secondly reduces the rest job of reducer phase. This results in curtailing the processing time of the reducer phase. These two are the main reasons behind the reflexing the traffic in the shuffling stage and workload of the reducer phase.

Experimental results show that our solution is well adapted to Big Data spaces. A shortened form of this paper has been presented previously in conference papers [5], [12]. Here, we expand on the theoretical and evaluation aspects of the optimization model and propose a multiobjective service-selection algorithm for Big Data spaces. We are among the first to propose bidirectional (internal and external) traffic optimization based on QoS awareness for multiobjective service-selection algorithms in a distributed environment. Our main contributions are:

- The Multiobjective service-selection algorithm in a distributed environment.
- QoS-aware rule-based traffic solutions to optimize traffic caused by the ZipF and the Pareto.
- Combiner based traffic solution on optimizing the traffic in shuffling stage.

The remainder of the paper is structured as follows. In Section II, we introduce some preliminaries and formulate the problem statement. In Section III, we present our proposed solutions for multiobjective selection requirements and bidirectional traffic concerns. Section IV discusses the proposed traffic-efficient multiobjective selection algorithm. In Section V, we discuss evaluation, and Section VI considers related work. Section VII concludes the paper.

II. PRELIMINARIES AND PROBLEM STATEMENT

In Section A, we introduce some preliminaries to selection-requirement issues. In Section B, we elaborate on the selection data flow in the MR process. Section C formulates the selection traffic problem for Big Data space.

A. PRELIMINARIES

In this subsection, we describe preliminary studies in the problem domain. First, we define the types of selection requirements and traffic congestion. Next, we define a problem statement for selection in the Big Data domain and model the selection problem under the external and internal traffic concerns.

We define service-selection requirements that are identified as three of the most common types. We use these three requirements to show the adaptability of the proposed method. One of these three requirements will be used in the relevant selection scenario during the MR process in the Big Data space.

- QoS values for web services (WS) that are affected positively by the performance of the service are denoted by QoS^P, and there will be x QoS constraints for the given WS WS_{ij}. Here, $x, i, j > 0$, i represents the i^{th} task in a composition planner, and the j represents the j^{th} candidate service for the given i^{th} task.
- QoS values of WS that are affected negatively by the performance of the service are denoted by QoS^N, and there will be y QoS constraints for the given WS WS_{ij}. Here, $y, i, j > 0$ and $x + y$ represents the total number of QoS values for the given WS_{ij}.
- The weight values of QoS^P and QoS^N are represented by w_α and w_β, respectively, such that $\sum_{\alpha=1}^{x} w_\alpha + \sum_{\beta=1}^{y} w_\beta = 1$ and $0 < w_\alpha . w_\beta < 1$.
- T is the number of tasks in the workflow, such that $0 < i \leq T$. v_α^{avg} is the average value of the particular QoS attribute among the given candidate services and v_α^{max} and v_α^{min} are the maximum and minimum values of the respective attributes.

$Q_{All}^{Constraints}$ is the collective value of negatively affected QoS attributes set by the user, with $Q_\beta^{Constraints}$ being an individual constraint.

Based on these notations, we define the heterogeneous-selection problem using Definitions (and Scenarios) 1, 2, and 3 below.

Definition 1 (Linear Optimal Service Selection): Select the composition plan that represents the global optimal solution with the most profitable overall QoS criteria.

Scenario 1: Users need to find the global optimal service composition sequence that maximizes the overall profit of QoS criteria as shown in Eq. 1.

$$Max \sum_{i=1}^{T} \left(\sum_{\alpha=1}^{x} \left(\frac{V(ws_{i\alpha}) - v_\alpha^{avg}}{v_\alpha^{max} - v_\alpha^{min}} \right) w_\alpha + \sum_{\beta=1}^{y} \left(1 - \frac{V(ws_{i\beta}) - v_\beta^{avg}}{v_\beta^{max} - v_\beta^{min}} \right) w_\beta \right) \quad (1)$$

We propose a solution to the linear optimal service-selection requirement based on the Dijkstra algorithm described in Section III-A.

Definition 2 (Combinatorial Service Selection): Select the composition plan that satisfies the QoS requirements, namely the upper limit of negatively affected QoS criteria and maximum normalized QoS criteria.

Scenario 2: Users need to find the global optimal service composition sequence that maximizes the overall profit and sets the maximum upper bound for negatively affected QoS criteria as shown in Eq. 2 and Eq. 3.

$$Max \sum_{i=1}^{T}\left(\sum_{\propto=1}^{x}\left(\frac{V(ws_{i\propto})-v_{\propto}^{avg}}{v_{\propto}^{max}-v_{\propto}^{min}}\right)w_{\propto} + \sum_{\beta=1}^{y}\left(1-\frac{V(ws_{i\beta})-v_{\beta}^{avg}}{v_{\beta}^{max}-v_{\beta}^{min}}\right)w_{\beta}\right) \quad (2)$$

$$Subject\ to \sum_{i=1}^{T}\left(\sum_{\beta=1}^{y}\left(\frac{V(ws_{i\beta})-v_{\beta}^{avg}}{v_{\beta}^{max}-v_{\beta}^{min}}\right)w_{\beta}\right) \leq Q_{All}^{Constraints} \quad (3)$$

We propose a solution to the combinatorial service-selection requirement based on the 0-1 MCKP algorithm described in Section III-B.

Definition 3 (Multivariate Optimal Service Selection): Select the composition plan that satisfies the combinatorial QoS-constrained set of each service, which is the resultant service in the composition plan that satisfies the respective minimum upper limit for negatively affected QoS criteria with respect to the maximum normalized QoS criteria.

Scenario 3: Users need to find the global optimal service composition sequence that maximizes each profit as shown in Eq. 4 subject to the maximum upper limits for each of the negatively affected QoS criteria as shown in Eq. 5. Here, the user sets a number β of constraints forβ QoS attributes for candidate WSs for the jth task. Here $1 \leq k \leq \beta$.

$$Max \sum_{\propto=1}^{x}\left(\frac{V(ws_{i\propto})-v_{\propto}^{avg}}{v_{\propto}^{max}-v_{\propto}^{min}}\right)w_{\propto} + \sum_{\beta=1}^{y}\left(1-\frac{V(ws_{i\beta})-v_{\beta}^{avg}}{v_{\beta}^{max}-v_{\beta}^{min}}\right)w_{\beta} \quad (4)$$

$$Subject\ to \left(\frac{V(ws_{ik})-v_{k}^{avg}}{v_{k}^{max}-v_{k}^{min}}\right)w_{j} \leq Q_{k}^{Constraint} \quad \text{Here } 1 \leq k \leq \beta \quad (5)$$

We propose a solution to the multivariate service-selection requirement based on the ABC algorithm described in Section III-C.

The main constraint behind a successful selection process is optimal resource utilization. However, traffic congestion is one of the most constraining factors behind optimal resource utilization in a Big Data environment. This accounts for considerable inefficiencies in the overall process. According to our literature review, we identified two key types of traffic congestion that affect the MR process internally and externally. We, therefore, define key traffic congestion in these two categories as follows.

Definition 4 (External Traffic Congestion—The Zipf Problem): This refers to the fact that the relative probability of access or processing requests for the ith most popular data block is proportional to $1/1^{\theta}$. When $\theta = 1$, the data block-request distribution strictly follows Zipf's law. Otherwise, it follows a more general Zipf-like distribution. Generally, it follows a Zipf-like distribution for Hadoop in the corresponding data blocks [13], [14].

Chen *et al.* [11] described the Zipf distribution of an MR job for both the input and output files of the MR job. They experimentally demonstrated and discussed the resulting unusual phenomena in terms of Zipf's law. Zipf generates *hot files* (higher access rate) and *cold files* (lower access rate) during the MR job. This causes an increase in the internal traffic and considerably reduces the performance of the overall MR job. Henceforth, we will use the term *hot* for *more popular files* and the term *cold* for *not so popular files*. We propose a traffic solution to the Zipf problem in Definition 7 of Section III-D-1.

Definition 5 (External Traffic Congestion—The Pareto Problem): The Pareto problem refers to the 80/20 rule. From a Hadoop perspective, the Pareto problem says that 80% of access or processing happens in 20% of the most popular data blocks. However, in actual cases, this 80/20 rule varies somewhat, becoming an ''80/10'' rule. It can then severely affect internal data communication and traffic in the MR job, increasing the ratio of hot vs cold files. Zipf and Pareto problems severely imbalance the data population in HDFS. This leads to increased traffic to hot data and dramatically affects the overall traffic of the MR process. We propose a traffic solution to the Pareto problem in Definition 8 of Section III-D-1.

Definition 6 (Internal Traffic Congestion-The Shuffling Problem): The mapper generates large chunks of intermediate data that are passed on to the reducer for further processing, which leads to massive network congestion. Shuffling data account for 58.6% of the cross-pod traffic and amounts to over 200 petabytes of data in an analysis of SCOPE jobs [10]. For shuffle-heavy MR tasks, this high traffic can incur a considerable performance overhead of up to 30–40%, as described in [15]. Therefore, we propose a traffic solution to internal traffic congestion in Section III-D-2.

B. SERVICE-SELECTION DATA FLOW IN THE MR PROCESS

Fig. 1 shows the typical anatomy of an MR process. According to the figure, part of the selection operation is scheduled in the given MR job. The input split constrains information about the candidate services for the first tasks in the given composition requirement, combined with mapper, then continues a portion of the overall selection operation. Respective files are then split, merged, copied, and sorted before being fed to the reducer phase. A different set of intermediate keys are assigned to each respective reducer node. These sets are the input to the reduce tasks. Reduce tasks are responsible for reducing the value associated with intermediate keys. Therefore, a set of intermediate keys are sorted on a single node before being fed to the reducer.

The respective outputs of the reducers represent an optimal composition planner for the given selection criteria.

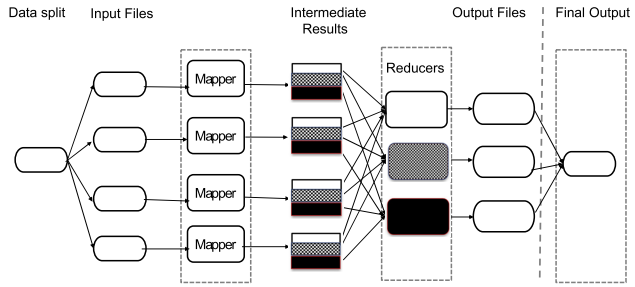

FIGURE 1. Anatomy of a typical MR process.

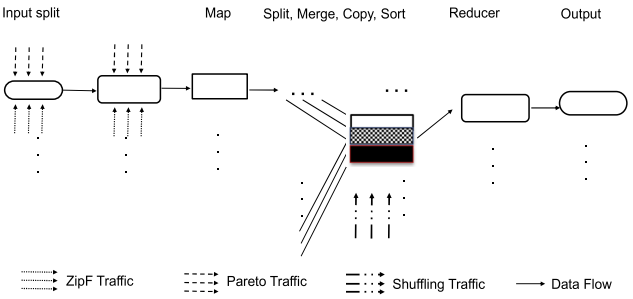

FIGURE 2. MapReduce job vs main traffic observed during the process.

This executes the respective key-related results from the reducers and outputs the respective optimal planner associated with each key (servicing the first tasks in the composition requirement).

C. TRAFFIC PROBLEM STATEMENT

According to the network traffic and the data flow of the selection process in the MR job, we can state the problem as follows, in terms of an effective-selection job. Fig. 2 shows the anatomy of an MR job. It has to pass rigorous steps in the data flow while facing severe constraints, namely, internal and external traffic. C_{MR_i} is the cost of the i^{th} job of the MR process for the selection. C_{Joint_i} is the traffic cost that affects the i^{th} job of the MR process. In addition, C_{Sel_i} is the cost of the selection process during the i^{th} job of the MR process. We can then describe the cost of the i^{th} job of the MR process as shown in Eq. 6.

$$C_{MR_i} = C_{Joint_i} + C_{Sel_i} + a_1 \tag{6}$$

Here, $C_{Sel_i} = C_{Map_i} + C_{Red_i}$ involves the map and reduce phases during selection. C_{Joint_i} is mainly affected via internal and external traffic during the process. Therefore, we can give Eq. 7 as the traffic cost of the i^{th} job of the process.

$$C_{Joint_i} = C_{Int_i} + C_{Ext_i} + a_2 \tag{7}$$

Here, C_{Int_i} is the internal traffic cost of the i^{th} job of the process and C_{Ext_i} is the external traffic cost of the i^{th} job of the process. These external and internal costs can then be further divided and expressed by Eq. 8 and Eq. 9 as follows.

$$C_{Int_i} = C_{S_i} + a_3 \tag{8}$$

$$C_{Ext_i} = C_{Z_i} + C_{P_i} + a_4 \tag{9}$$

C_{S_i}, C_{Z_i} *and* C_{P_i} are the costs of the shuffling, Zipf, and Pareto, respectively, of the ith job of the MR process, respectively, and a_i:$i = 1, 2, 3, 4, 5$ is constant. We can then represent C_{Joint_i} as shown in Eq. 10. Here, c is a constant.

$$C_{Joint_i} = C_{S_i} + C_{Z_i} + C_{P_i} + a_5 \tag{10}$$

Therefore, to have an effective and efficient selection process, we have to address the concerns described in Eq. 10. This means that the respective C_{Map_i}, C_{S_i}, C_{Red_i}, C_{Z_i} and C_{P_i} values need to be reduced to reduce the respective C_{Joint_i}. This directly affects C_{MR_i}, as shown in Eq. 6.

We can then deduce the Eq. 11 to show the effects of traffic congestion in reducing the overall traffic. This implies that we can reduce the overall execution cost of the MR job of the selection process when we minimize the collective internal and external traffic congestions, as shown below. Here γ is the number of jobs in the MR process.

$$\min_{i\in\gamma}\left(\sum\left(C_{S_i} + C_{Z_i} + C_{P_i}\right)\right) \quad leads\ to \quad \min_{i\in\gamma} \sum C_{MR_i} \tag{11}$$

We prove this as a joint optimization of the traffic problems in Section III-D-3.

III. PROPOSED SOLUTION

In this section, we present our proposed solutions to the issues described in Section II. In Sections III-A, III-B, and III-C, we present the proposed solutions for multiobjective selection requirements. In Section III-D, we develop the proposed solution for bidirectional traffic concerns.

A. PROPOSED SOLUTION FOR A LINEAR OPTIMAL SERVICE-SELECTION REQUIREMENT

We propose using a graph-theory-based linear programming technique to address the scenario described in Definition 1 in Section II. This is a novel approach to the service-selection domain. First, we prepare a directed acrylic graph (DAG) network from the candidate services, as shown in Fig. 3, and then use our proposed algorithm to calculate the longest path between the first and last vertex, as described by Zeng *et al.* [4]. Dijkstra employed a longest-path-finding algorithm to calculate the optimal selection composition plan for such a DAG network. Table 1 shows the respective QoS utility representations for the Dijkstra method.

Plans are made by the given candidate services. We use Eq. 12 to calculate the distance of vertices (services) between the i^{th} and the $(i+1)^{th}$ tasks containing any two services, which is called the $L(S_i, S_{i+1})$. According to the Fig. 3 shown DAG graph, it does not allow to create edges in between services in the same task. Fig. 3 DAG allows edges between adjacent tasks only, where c is a constant. For service S_{i+1}, its utility value $U_{S_{i+1}}$ is given by Eq. 13.

$$L\left(S_i, S_{i+1}\right) = c + U_{S_{i+1}} \tag{12}$$

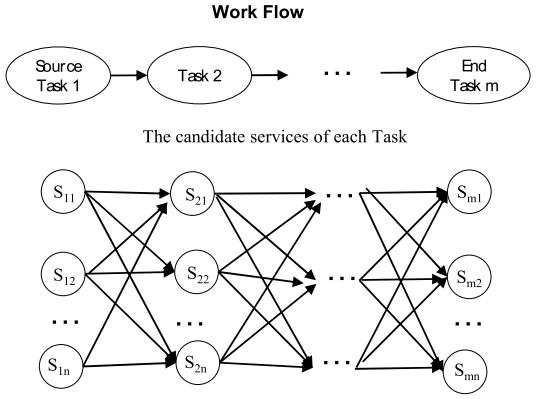

FIGURE 3. DAG network created by candidate services of each task.

TABLE 1. Tasks vs utility values of candidate services used in Dijkstra. *Here, n is the number of candidate WSs and m is the number of tasks in the work flow.*

Source Task	Task 2	…	End Task
S_{11}	S_{21}	…	S_{m1}
U.QoS: $U_{S_{11}}$	U.QoS: $U_{S_{21}}$		U.QoS: $U_{S_{m1}}$
S_{12}	S_{22}	…	S_{m2}
U.QoS: $U_{S_{12}}$	U.QoS: $U_{S_{22}}$		U.QoS: $U_{S_{m2}}$
…	…	…	…
S_{1n}	S_{2n}	…	S_{mn}
U.QoS: $U_{S_{1n}}$	U.QoS: $U_{S_{2n}}$		U.QoS: $U_{S_{mn}}$

$$U_{S_{i+1}} = \sum_{\propto=1}^{x} \left(\frac{V(ws_{i\propto}) - v_{\propto}^{avg}}{v_{\propto}^{max} - v_{\propto}^{min}} \right) w_{\propto} + \sum_{\beta=1}^{y} \left(1 - \frac{V(ws_{i\beta}) - v_{\beta}^{avg}}{v_{\beta}^{max} - v_{\beta}^{min}} \right) w_{\beta} \quad (13)$$

Here, we have taken c = 0. A DAG network is defined as a source task (Task 1) to the last task, which Task m services. This graph connects exactly m services. Each service represents a vertex and the distance between two services is calculated from Eq. 12. It does not create edges between services in the same Task itself. We define the graph $G(V, E)$, in which V is created from S_i and E is calculated by $L(S_i, S_{i+1})$. Source nodes are represented by Task-1 services and target nodes are represented by end-task services. If the cost of the $(ij)^{th}$ service is represented as E_{ij}, we have to maximize $\sum_{ij \in E} E_{ij}.S_{ij}$, subject to $S \geq 0$, for all i.

B. PROPOSED SOLUTION FOR A COMBINATORIAL SERVICE-SELECTION REQUIREMENT

We propose a dynamic programming technique to address the combinatorial selection requirement. We use 0-1 MCKP to simulate the scenario described in Definition 2 in Section II. The utility-value calculations that represent the profit and weight used in 0-1 MCKP are given by Eq. 14 and Eq. 15. Table 2 shows the respective weights and profit consumption in the selection scenario.

TABLE 2. Task vs utility values of candidate services used in 0-1 MCKP. *Here, n is the number of candidate WSs and m is the number of tasks in the work flow.*

Source Task	Task 2	…	End Task
S_{11}	S_{21}	…	S_{m1}
Profit:$U_{1,1:P}$	Profit:$U_{2,1:P}$		Profit:$U_{m,1:P}$
Weight: $U_{1,1:N}$	Weight: $U_{2,1:N}$		Weight: $U_{m,1:N}$
S_{12}	S_{22}		S_{m2}
Profit:$U_{1,2:P}$	Profit:$U_{2,2:P}$		Profit:$U_{m,2:P}$
Weight: $U_{1,2:N}$	Weight: $U_{2,2:N}$		Weight: $U_{m,2:N}$
…	…	…	…
S_{1n}	S_{2n}		S_{mn}
Profit:$U_{1,n:P}$	Profit:$U_{2,n:P}$		Profit:$U_{m,n:P}$
Weight: $U_{1,n:N}$	Weight: $U_{2,n:N}$		Weight: $U_{m,n:N}$

Table 2 shows the number m of Tj (i ≤ m) tasks in the workflow. Each Tj contains n candidate services Sij (i ≤ n, j ≤ m). The profit that can be gained from Sij services is represented as $U_{ij}^{P\aleph}$ and the weight as $U_{ij}^{N\aleph}$. The custom QoS requirement is called the capacity, denoted as C.

Our 0-1 MCKP requirement is then expressed as follows. To maximize the normalized utility QoS P_u, $P_u = \sum_{i=1}^{m} \sum_{j=1}^{n} U_{i,j:P}$, subject to $\sum_{j=1}^{n} U_{i,j:N}.S_{ij} \leq C$, where $i \in M\{1, 2, \ldots, m\}$. Moreover, $\sum_{i=1}^{m} S_{ij} = 1, j \in N\{1, 2, \ldots, n\}$, and $S_{ij} = 0$ *or* 1, where $i \in M, j \in N$. $U_{i,j:P}$ positively affects the utility QoS of the j^{th} candidate WS of the i^{th} task. $U_{i,j:,N}$ negatively affects the utility QoS of the j^{th} candidate WS of the i^{th} task.

$$U_{i,j:P} = \sum_{i=1}^{T} \left(\sum_{\propto=1}^{x} \left(\frac{V(ws_{i\propto}) - v_{\propto}^{avg}}{v_{\propto}^{max} - v_{\propto}^{min}} \right) w_{\propto} + \sum_{\beta=1}^{y} \left(1 - \frac{V(ws_{i\beta}) - v_{\beta}^{avg}}{v_{\beta}^{max} - v_{\beta}^{min}} \right) w_{\beta} \right) \quad (14)$$

$$U_{i,j:,N} = \sum_{i=1}^{T} \left(\sum_{\beta=1}^{y} \left(\frac{V(ws_{i\beta}) - v_{\beta}^{avg}}{v_{\beta}^{max} - v_{\beta}^{min}} \right) w_{\beta} \right) \quad (15)$$

C. PROPOSED SOLUTION FOR A MULTIVARIATE SERVICE-SELECTION REQUIREMENT

The WS selection requirement defined by Definition 3 in Section III is simulated using the ABC algorithm [16]. The utility functions for the positively affected QoS (profit) and negatively affected attributes are represented by Eq. 16 and Eq. 17, respectively. Table 3 gives the respective value distributions across the candidate services for the given task. The utility QoS of the jth candidate WS of the ith task is positively affected by $U_{i,j:P}$, whereas $U_{i,j,\beta:,N}$ is the βth negatively affected utility QoS of the jth candidate WS of the ith task. The user sets β constraints for the β QoS attributes for candidate WS of the jth task.

Initially, the ABC algorithm initializes the generated ''food source'' randomly. Here, it uses Eq. 16 and Eq. 17 to set the respective utility values for the food sources.

Next, it sends the employed ''bees'' to the food sources (identified plans) and determines the amount of ''nectar''

TABLE 3. Task vs utility values of candidate services used in ABC. *Here, n is the number of candidate WSs and m is the number of tasks in the work flow.*

Source Task	Task 2	…	End Task
S_{11} Profit:$U_{1,1:P}$ Neg. 1st : $U_{1,1,1:N}$ Neg. 2nd : $U_{1,1,2:N}$ .. Neg. β^{th}: $U_{1,1}\beta_{:N}$	S_{21} Profit:$U_{2,1:P}$ Neg. 1st : $U_{2,1,1:N}$ Neg. 2nd : $U_{2,1,2:N}$ .. Neg. β^{th}: $U_{2,1}\beta_{:N}$	…	S_{m1} Profit:$U_{m,1:P}$ Neg. 1st : $U_{m,1,1:N}$ Neg. 2nd : $U_{m,1,2:N}$ .. Neg. β^{th}: $U_{m,1}\beta_{:N}$
S_{12} Profit:$U_{1,2:P}$ Neg. 1st : $U_{1,2,1:N}$ Neg. 2nd : $U_{1,2,2:N}$ .. Neg. β^{th}: $U_{1,2}\beta_{:N}$	S_{22} Profit:$U_{2,2:P}$ Neg. 1st : $U_{2,2,1:N}$ Neg. 2nd : $U_{2,2,2:N}$ .. Neg. β^{th}: $U_{2,2}\beta_{:N}$	…	S_{m2} Profit:$U_{m,2:P}$ Neg. 1st : $U_{m,2,1:N}$ Neg. 2nd : $U_{m,2,2:N}$ .. Neg. β^{th}: $U_{m,2}\beta_{:N}$
…	…	…	…
S_{1n} Profit:$U_{1,2:P}$ Neg. 1st : $U_{1,n,1:N}$ Neg. 2nd : $U_{1,n,2:N}$ .. Neg. β^{th}: $U_{1,n}\beta_{:N}$	S_{2n} Profit:$U_{2,2:P}$ Neg. 1st : $U_{2,n,1:N}$ Neg. 2nd : $U_{2,n,2:N}$ .. Neg. β^{th}: $U_{2,n}\beta_{:N}$	…	S_{mn} Profit:$U_{m,2:P}$ Neg. 1st : $U_{m,n,1:N}$ Neg. 2nd : $U_{m,n,2:N}$ .. Neg. β^{th}: $U_{m,n}\beta_{:N}$

(overall profit). It then calculates the fitness values for each food source. Greedy selection is applied to the current solution and its mutant. If the mutant solution is an improvement, it replaces the previous solution and the trial counter of the solution is reset. If a better solution cannot be found, the trial counter is incremented.

The algorithm then calculates the probability of the fitness value. Here, it evaluates the nectar information from all employed bees and chooses a food source (identified solution) with a probability related to the amount of nectar (overall profit). If it cannot find a better solution, then it enjoins a ''scout bee'' to find a new food source in the same way as does the employee bee. It continues this process until the termination condition is reached.

$$U_{i,j:P} = \sum_{\propto=1}^{x} \left(\frac{V\left(ws_{i\propto}\right) - v_{\propto}^{avg}}{v_{\propto}^{max} - v_{\propto}^{min}} \right) w_{\propto} + \sum_{\beta=1}^{y} \left(1 - \frac{V(ws_{i\beta}) - v_{\beta}^{avg}}{v_{\beta}^{max} - v_{\beta}^{min}} \right) w_{\beta} \quad (16)$$

$$U_{i,j,k:N} = \left(\frac{V(ws_{ik}) - v_{k}^{avg}}{v_{k}^{max} - v_{k}^{min}} \right) w_k \quad \text{Here } 1 \leq \text{k} \leq \beta \quad (17)$$

D. PROPOSED SOLUTION FOR BIDIRECTIONAL TRAFFIC CONCERNS

In this subsection, we present the proposed solutions for the bidirectional traffic concerns. We first describe a solution for external traffic and then for internal traffic. Finally, in Section III-D-3, we present a joint optimization of bidirectional traffic concerns.

1) EXTERNAL TRAFFIC SOLUTION

We now describe our proposed solutions for the two external traffic concerns, namely the Zipf and Pareto problems.

a: ZIPF PROBLEM: QoS-AWARE SERVICE DISTRIBUTION (QSD RULE)

We propose an efficient, rule-based traffic technique to address the concerns that occur during the selection process in an MR job. From our observations, Zipf raised the most concerns, especially in the split and map stages of the MR process shown in Fig. 2. These areas are highly correlated with specific files and their replica access for specific needs in the selection process, incurring Zipf phenomena and hot files. To address this concern, we propose the QSD rule.

Definition 7 (QSD Rule): Service distribution according to the proportional values of the normalized utility QoS are inversely proportional to the hotness caused by the Zipf phenomena during the selection process.

Let f be the functional representation of the proportional distribution of services and φ the functional representation of the hotness that occurs from Zipf during service selection. The rules are formulated in Eq. 18 and Eq. 19.

$$f^*\left(S_i, S_{i+1}\right) = \frac{QoS\left(S_{i+1}\right)}{QoS\left(S_i\right)} \quad (18)$$

$$f^*\left(S_i, S_{i+1}\right) = \frac{\varphi\left(S_{i+1}\right)}{\varphi\left(S_i\right)} \quad (19)$$

Here, $f^*(S_i, S_{i+1})$ is the optimal proportional replica distribution of S_i and S_{i+1} services according to their normalized QoS proportions, as shown in Eq. 18, and $\frac{\varphi(S_{i+1})}{\varphi(S_i)}$ is the proportional value of the hotness caused by the Zipf of S_i and S_{i+1} services, as shown in Eq. 19.

We prove the QSD rule in terms of Theorem 1, Lemma 1, and Lemma 2, as below.

Theorem 1: Traffic caused by Zipf is proportional to the normalized QoS criteria of the services.

Proof: Consider that there are services called $S_1, S_2, \ldots, S_n$, and their respective normalized QoS are $QoS_{S_1}, QoS_{S_2}, \ldots, QoS_{S_n}$. The functional representation of the QoS distribution is given as $\emptyset\left(U_{S_1}, U_{S_2}, \ldots, U_{S_n}\right)$. Traffic occurring by Zipf for respective $S_1, S_2, \ldots, S_n$ services is denoted as $T_{S_1}^{Z}, T_{2}^{Z}, \ldots, T_{n}^{Z}$, and the functional representation of the hotness distribution is given as $\mathrm{T}\left(S_1, S_2, \ldots, S_n\right)$. Zipf occurs at the splitting and map stages. At the split stage, it is trying to find an accessible replica of a given service from among the available replicas.

Here, it is most actively and exhaustively used for the highest normalized QoS replicas. This means that the highest traffic occurs for the highest QoS replicas, and the lowest traffic for the lowest QoS replicas received.

In turn, this implies that the splitting stage traffic for the i^{th} job of the MR, $T_{S_{j,split}}^{MR_i,Z}$, is directly proportional to its U_{S_j}, where k_1 and c_1 are constants.

Therefore,

$$T_{S_{j,split}}^{MR_i,Z} = k_1 * U_{S_j} + c_1, \quad (20)$$

In addition, the mapper part is working hard to achieve the optimal composition plan among the given list of plans, as described in Sections III-A, III-B, and III-C. Here, exponential traffic occurs to the highest QoS because all three

native techniques are designed to follow the optimal QoS. Therefore, the mapper stage traffic for the i^{th} job of the MR, $T^{MR_i,Z}_{S_{j,Map}}$, is directly proportional to its U_{S_j}. That is,

$$T^{MR_i,Z}_{S_{j,Map}} = k_2 U_{S_j} + c_2, \tag{21}$$

Where k_2 and c_2 are constants. We now consider both Eq. 19 and Eq. 20:

$$T^{MR_i,Z}_{S_{j,Map}} + T^{MR_i,Z}_{S_{j,split}} = k_3 U_{S_j} + c_3,$$

Where $k_3 = k_1 + k_2$ and$d$ $c_3 = c_1 + c_2$ are constants.

This means that traffic caused by j^{th} services for the i^{th} job is given by Eq. 22:

$$T^{MR_i,Z}_{S_j} = k_3 U_{S_j} + c_3 \tag{22}$$

Moreover, based on Eq. 22, traffic caused by all services of the i^{th} job can be expressed as shown in Eq. 23. Here, N is the number of services and γ is the number of jobs in the MR process.

$$\sum_{j=1}^{N} T^{MR_i,Z}_{S_j} = k_3 \sum_{j=1}^{N} (U_{S_j} + c_4) \tag{23}$$

Finally, based on (22), traffic caused by all services of all jobs contained in the MR process is given by Eq. 24:

$$\sum_{i=1}^{\gamma}\sum_{j=1}^{N} T^{MR_i,Z}_{S_j} = k_3 \sum_{i=1}^{\gamma}\sum_{j=1}^{N} (U_{S_j} + c_5) \tag{24}$$

According to Eq. 24, therefore, the overall traffic caused by Zipf is proportional to the normalized utility QoS of services.

According to Eq. 22, assuming services S_j and S_{j+1} and their $U_{S_j} > U_{S_j+1}$, the traffic imposed by Zipf for the i^{th} MR job is shown by Eq. 25:

$$T^{MR_i,Z}_{S_j} > T^{MR_i,Z}_{S_j+1} \tag{25}$$

According to Eq. 25, traffic to S_j and S_{j+1} caused by Zipf for the entire MR process is:

$$\sum_{i=1}^{\gamma} T^{MR_i,Z}_{S_j} > \sum_{i=1}^{\gamma} T^{MR_i,Z}_{S_{j+1}} \tag{26}$$

Services that have higher normalized QoS have more traffic during the MR selection process. Then, from Eq. 24 and Eq. 26, traffic increments of S_j and S_{j+1} are proportional to their normalized QoS. This is represented by Eq. 27.

$$\frac{T^{MR_i,Z}_{S_{j+1}}}{T^{MR_i,Z}_{S_j}} = \frac{U_{S_{j+1}}}{U_{S_j}} \tag{27}$$

Lemma 1: Traffic caused by Zipf is inversely proportional to the service distribution in the normalized QoS.

Proof: According to the Eq. 27, traffic in the given service increases with increasing QoS. It then seeks more file instances of that service to overcome the demand. Let $f(S_i) = f(S_{i+1})$ be the initial service distribution of S_i and S_{i+1} services, with their QoS being $U_{S_i} > U_{S_i+1}$. Then, according to Eq. 25, Zipf traffic for these services is $T^1_{S_i} > T^1_{S_i+1}$. This implies that S_i suffers more demand than S_{i+1}. These demands are proportional to their utility QoS because Zipf traffic is proportional to their QoS. The new file distribution is then f^*, with $f^*(S_i)/f^*(S_{i+1}) = U_{S_i}/U_{S_i+1}$. New traffic on S_i and S_{i+1} is redeemed in the same proportional manner: $T^2_{S_i} < T^1_{S_i}$, $T^2_{S_i+1} < T^1_{S_i+1}$ and $T^2_{S_i}/T^2_{S_i+1} = T^1_{S_i}/T^1_{S_i+1} = U_{S_i+1}/U_{S_i}$. This means that Zipf traffic is inversely proportional to the f^*. We can express this as Eq. 28:

$$T^2_{S_i}/T^2_{S_i+1} = T^1_{S_i}/T^1_{S_i+1} = U_{S_i+1}/U_i \tag{28}$$

Lemma 2: Traffic occurring with Zipf is proportional to the hotness caused by Zipf for a given service.

Proof: According to Definition 4, Zipf hotness refers to the popularity generated by Zipf. Nevertheless, ''very hot'' implies more traffic, ''average hot'' implies average traffic, and ''not hot'' implies no traffic. This means that the hotness (caused by Zipf) is equally proportional to the traffic (caused by Zipf) on that service. $H^1_{S_j}$ is the hotness caused by the Zipf.

Then, according to Eq. 28 and Eq. 29 below, it proves Definition 7 and can be expressed as Eq. 30.

$$T^1_{S_j}/T^1_{S_j+1} = H^1_{S_j}/H^1_{S_j+1} \tag{29}$$

$$H^1_{S_j}/H^1_{S_j+1} = U_{S_j+1}/U_{S_j} \tag{30}$$

□

b: PARETO PROBLEM: TRAFFIC-AWARE REPLICA DISTRIBUTION (TRD RULE)

The Pareto raised most of its concerns in the split and map stages of the MR process. These areas are highly correlated with specific replica accesses for specific needs when Pareto phenomena and hot replicas occur. To address this concern, we propose the TRD rule.

Definition 8 (TRD Rule): Hotness caused by the Pareto is inversely proportional to the traffic-aware replica distribution.

We prove the TRD rule based on Theorem 1 above and Lemmas 3 and 4 below.

Lemma 3: Availability vs traffic: increasing the availability of densely hot services (caused by the Pareto replicas) is inversely proportional to the overall traffic.

Proof: We can apply Theorem 1 to the Pareto traffic in services, which is proportional to the normalized QoS criteria of the services. However, replicas in the HDFS represent particular services. Therefore, we can extend Theorem 1 to the replica level of the S_j service, assuming S_j is replicated by r_1, r_2, .., r_n, with n being the default replication factor in the HDFS. According to the Pareto rule, it makes hot replicas from available replicas among the given services. In addition, according to Theorem 1, we can address this traffic by increasing the number of instances of a particular file. This implies that an increment of hot replicas according to their popularity (hotness) is inversely proportional to the traffic.

Lemma 4: Traffic occurring with Pareto is proportional to the hotness caused by Pareto for the given replicas.

Proof: According to Lemma 2, the hotness caused by a particular replica will be directly proportional to the traffic. This can be applied to Pareto, where the hotness caused by the Pareto is directly proportional to the traffic.

According to Lemmas 3 and 4, we can conclude that the hotness caused by Pareto is inversely proportional to the traffic-aware replica distribution. □

2) INTERNAL TRAFFIC SOLUTION

According to our observations, the shuffling stage of the selection process faces heavy traffic congestion because of a large number of intermediate results. To address this concern, we propose a middle agent for the MR process that uses a combiner. This job-wise combiner method allows us to sort and short-list the shuffling results of each job for both the mapping and reducing sides. The result is a considerably reduced overall job in the reducer phase. We call this the intermediate MR (IMR) agent.

IMR Agent: The mapper phase generates large chunks of intermediate data that are passed on to the reducer phase for further processing. At the beginning of the reducer phase, large chunks of intermediate data are shuffled to facilitate reducer processing. This leads to massive network congestion. To address this, we propose to introduce an IMR agent that reduces network congestion and relieves the workload of the reducer phase. The IMR agent mainly handles the task of shuffling the intermediate chunks, which is separated from the reducer phase and plays a crucial role in reducing network congestion. It is important to note that the primary job of the IMR Agent is to process the output data from the mapper before being passed to the reducer phase to reduce the workload of the reducer phase and minimized the shuffling data chunk. Therefore, IMR results in two major benefits, firstly, it accomplishes the part of the workload of the reducer phase, and secondly, it reduces the shuffling data traffic. We have employed a combiner to implement the IMR Agent class.

3) JOINT OPTIMIZATION FOR THE TRAFFIC PROBLEM STATEMEN

Section II-C contains the problem statement for overall traffic congestion. According to Eq. 8, we have to minimize the traffic concerns occurring during the selection stage of the MR process. These are expressed as $\sum C_{MR_i}$, where i is the i^{th} job of the MR process. In terms of the proposed techniques explained earlier in Section III-D, we minimize the overall traffic concerns as follows.

According to the Definition 7, by distributing services according to the QRD rule, we will obtain:

$$\min_{i\in\gamma} \sum C_{Z_i}$$

Then, according to the Definition 8, by distributing the service replicas according to the TRD rule, we will obtain:

$$\min_{i\in\gamma} \sum C_{P_i}$$

Finally, by considering the internal traffic congestion by using the IMR agent, we will obtain:

$$\min_{i\in\gamma} \sum C_{S_i}$$

By aggregating these three techniques for overall external and internal traffic, we minimize the overall traffic caused by the selection process. That is, we obtain:

$$\min_{i\in\gamma} \left(\sum \left(C_{S_i} + C_{Z_i} + C_{P_i}\right)\right)$$

This represents a solution to the problem expressed by Eq. 11 in Section II-C.

IV. QoS-AWARE TRAFFIC–EFFICIENT ALGORITHM

This section describes the proposed algorithm for service selection in Big Data space. Finding the optimal service composition sequence is a cumbersome task, mainly because of the complex composition requirements sought within a large search space. It requires high-performance infrastructure to complete such a resource-intensive heavy-duty job. We propose an MR algorithm that aims to meet heterogeneous-selection requirements, achieve global optima for linear and combinatorial selection requirements, and near-optimal multivariate-selection requirements.

According to the definition 1, 2 and 3, the proposed solutions for the multiobjective selection requirements are described in previous section III-A, B, and C. The selection solutions are running behind the distributed environment. In addition to that, traffic solutions are applied. Section A describes the initial setup and overall flow, which is called as driver procedure; Section B describes the algorithm of the mapper; Section C describes the algorithm of the IMR agent; Section D describes the algorithm of the reducer and retrieve the final output.

A. SELECTION PROCEDURE: FLOW OF THE DRIVER

Selection procedure represents the driver of the MR process. Let's define the scenario for the better understanding the process.

Scenario 4: As shown in Fig. 3; Consider a case comprising four tasks ($m = 4$) and n candidate services for each task. Assume p batches are processed by each mapper. Let U_i be the normalized utility QoS. Next, assume that we have services S_1, S_2, and S_3, with respective U_i values: $U_1 = 0.5$, $U_2 = 0.75$, and $U_3 = 0.25$. The proportional values of these three utilities will then be $U_1{:}U_2{:}U_3 = 2{:}3{:}1$.

Initial Setup: First, we prepared the availability of the service as follows. The HDFS needs to contain the respective service information according to Definitions 7 and 8 in

Section III-D. According to the QRD rule, we follow the service distribution of each batch of services as follows. S_1 has a $C \times 2$ replications, S_2 has $C \times 3$ services, S_3, and S4 have $C \times 1$ replications. Here, C is a constant and represents the replication factor for the TRD rule. This is the preparation of the environment to address the external traffic.

Next, it decides the types of QoS-awareness (linear optimal, combinatorial and multivariate) need for the composition. If he needs the linear optimal then he should use the Dijkstra method and doesn't have to do anything, just need to start the selection process. If he needs combinatorial, then he should use the 0-1 MCKP and define the minimum lower bound for the collective value of negatively affected criteria's as shown in Eq. 3. If he needs multivariate, then he should use the ABC and define the minimum lower bound for each of negatively affected criteria's as shown in Eq. 5. Finally, the user sets the above parameters in the selection Procedure 1 line 3. Procedure 1 shows the flow of selection procedure and arrangement of the driver of the MR process. Lines 2 to 9 contains the driver with three sections: n number of the mappers, an IMR Agent, and the reducer. At the beginning of the Selection (Line 3), value for the Threshold_Plans and the Type of Selection are assigned. Next, n number of mappers are specified (Line 5). Line 6 includes the IMR Agent in the procedure, which is responsible for reducing the traffic in the shuffling stage and thereby reducing the burden on the reducer stage. Line 7 includes the reducer class, which processes intermediate results given by the IMR Agent and outputs the optimal (or near-optimal) composition service sequences.

B. MAPPER ALGORITHM

Algorithm 1 represents the proposed mapper algorithm for the selection procedure. Lines 1 - 15 represents the k^{th} mapper of the Procedure 1 of the n number of mappers. In Line 2, it generates possible plans for the given search space. Next, Lines 3 to 14 show looping through the set of plans according to the arranged number of threshold plans. During this process, the procedure executes the selection *Selection_Requirement* that was initially set by the user. This generates a set of possible optimal planners for respective batches of plans. Lines 4 to 6 involve the linear optimal selection requirement and the Dijkstra algorithm is invoked. Lines 7 to 9 involve the combinatorial selection requirement and the 0-1 MCKP algorithm is invoked. Lines 10 to 12 involve the multivariate-selection requirement and the ABC algorithm is invoked. The inputs and outputs of all three algorithms are synchronized to the same data structure, thereby facilitating smooth operation throughout the selection process without compromising the overall selection criteria. Each job of the mapper results in a chunk of composition results, initiated from the given candidate service of the first task. Line 13 writes the set of resulting plans to the context.

For the above example, the results are in the form of Level id vs optimal service selection sequence#profits, for $1 \leq k \leq n$, as follows.

1: $S_{11}@S_{21}@S_{31}@S_{41}$#2.964
<Results of all possible combinations from S_{11} to 4th Task of n mapper of p batches>
2: $S_{12}@S_{2n}@S_{3n}@S_{4n}$#1.460
<Results of all possible combinations from S_{12} to 4th Task of n mapper of p batches>
...
k: $S_{1k}@S_{23}@S_{34}@S_{4k}$#2.960
<Results of all possible combinations from S_{1k} to 4th Task of n mapper of p batches>
...
n: $S_{1n}@S_{2n}@S_{3k}@S_{4n}$#1.360
<Results of all possible combinations from S_{n1} to 4th Task of n mapper classes of p batches>

C. IMR ALGORITHM

Algorithm 2 represents the algorithm for IMR Agent. This procedure receives batches of intermediate results of the selection. Two main objectives are involved in the IMR. Reduce the shuffling traffic congestion, and reduce the workload of the reducer phase. This leads to a dramatic reduction in internal traffic and the multiple reducers. In Line 2, the procedure initializes the optimal profit and optimal plan sequence values. Lines 3 to 8 loop to find optimal profit under the given key value and find the optimal plan sequence under the key. Line 9 writes the key with the concatenated optimal plan and profit to the MR context.

During the IMR Agent class, the context is processed as an intermediate process, below is sample output from the IMR Agent. The information is KEY string value vs optimal plan sequence with the profit value for the given sequence, namely KEY vs optimal service selection sequence#profits, for $1 \leq k \leq n$.

KEY: $S_{11}@S_{21}@S_{31}@S_{41}$#2.964 // This is the optimal sequence among all possible combinations from S_{11} to 4th Task of the 1^{st} mapper.

<Results of all possible combinations from S_{11} for n mapper>

KEY: $S_{12}@S_{2n}@S_{3n}@S_{4n}$#1.460 // This is the optimal sequence among all possible combinations from S_{12} to 4th Task of the 2^{nd} mapper.

<Results of all possible combinations from S_{12} for n mapper>

...

KEY: $S_{1k}@S_{23}@S_{34}@S_{4k}$#2.960 // This is the optimal sequence among all possible combinations from S_{1k} to 4th Task of the k^{th} mapper.

<Results of all possible combinations from S_{1k} for n mapper> ...

KEY: $S_{1n}@S_{2n}@S_{3k}@S_{4n}$#1.360 // This is the optimal sequence among all possible combinations from S_{1n} to 4th Task of the n^{th} mapper.

<Results of all possible combinations from S_{1n} for n mapper>

D. REDUCER ALGORITHM

Algorithm 3 represents the flow of the reducer. This finds an optimal plan for the given key by splitting the optimal plan and profit string resulting from the IMR Agent. Part of the reducer job already completed by the IMR agent. Lines 3 to 8, procedure find the optimal plan and profit value for the particular key. Line 9 writes the plan and profit to the MR context. Below is sample output from the reducer. The information is an optimal (or near $P =$ optimal) plan sequence vs profit value for the given sequence, where $1 \leq k \leq n$.

$S_{1k}@S_{21}@S_{31}@S_{41}$ vs 2.960

Procedure 1 Selection Procedure

1: Setup initial service data according to the
   - QSD Rule to address the Zip
   - TRD Rule to address the Pareto
2: **Selection Class {**
3: // Set initial parameters
   Threshold_Plans ← Number of Plans per Each Batch
   Selection_Requirement ← Type of Selection & QoS constraints
4: **Driver Class {**
5: // Set n number of Mapper Classes under the MultipleInputs
   MultipleInputs.addInput(job, Input_file_1, Mapper_1.class)
   MultipleInputs.addInput(job, Input_file_2, Mapper_2.class)
   ...
   MultipleInputs.addInput(job, Input_file_n, Mapper_n.class)
6: // Set Combiner Class
   job.setCombiner(IMR_Agent.class)
7: // Set Reducer Class
   job.setReducer(Reducer.class)
8: }**end Driver Class**
9:}**end of Selection Class**

V. EVALUATION

We conducted experiments to evaluate the proposed method. We considered two key research metrics, namely the efficiency of the proposed method and the effectiveness of the heterogeneous-selection approaches in the Big Data space. We, therefore, performed experiments in these two main areas.

Efficiency: To evaluate the efficiency, we conducted experiments that observed the internal, external, and jointly optimized traffic efficiencies of multiobjective selection methods while increasing the number of plans and data nodes in the Hadoop cluster.

Effectiveness: To evaluate the effectiveness, we conducted experiments that observed, in terms of computational complexity, the internal, external, and jointly optimized traffic effectiveness of multiobjective selection methods while increasing the number of plans and data nodes. In particular, we observed the precision of the proposed methods.

The experiments were conducted in CentOs 7, with Hadoop 2.2 and Java 1.8 installed on a four-node Hadoop

Algorithm 1 Mapper_k

1:**procedure** Map (key, service Information)
2: setup () ← Create all possible plans from initial tasks to end task
3: **for all** plan_num < Threshold_Plans **do**
4: **if** Selection_Requirement = Linear Optimal **then**
5: Mapper_Result ← Execute Dijkstra Algorithm
6: **end if**
7: **else if** Selection_Requirement = Combinatorial **then**
8: Mapper_Result ← Execute 0-1 MCKP Algorithm
9: **end else if**
10: **else then** //Selection_Requirement = Multivariate
11: Mapper_Result ← Execute ABC Algorithm
12: **end else**
13: context.write (Mapper_Result ← Task_1 service, optimal_plan#profit)
14: **end for**
15:**end of Map() procedure**

Algorithm 2 IMR Agent

// Designed to avoid multi cross shuffling, and continue same Key of Mapper
// used in as the Key of Reducer to address shuffling stage traffic concerns
1:**procedure** IMR Agent (key, values, context)
2: initialize optimal_profit and optimal_plan
3: **for all** values **do**
4: **if** existing_profit < new_profit **then**
5: optimal_profit = new_profit
6: optimal_plan = new_plan
7: **end if**
8: **end for**
9: context.write (''KEY'', optimal_plan#profit)
10:**end of IMR Agent procedure**

Algorithm 3 Reducer

1:**procedure** reducer (key, values, context)
2: initialize optimal_profit and optimal_plan
3: **for all** values **do**
4: **if** existing_profit < new_profit **then**
5: optimal_profit = new_profit
6: optimal_plan = new_plan
7: **end if**
8: **end for**
9: context.write (optimal_plan, profit)
10:**end of reducer procedure**

cluster. The master node contained an Intel Core i7 3.4-GHz 8-core processor and 8 GB RAM. Each data node contained an Intel Core i5 3.0-GHz 4-core processor and 8 GB RAM.

Dataset: We conducted our experiments with a real-world dataset provided by Al-Masri and Mahmoud [17] called the QWS dataset, which contained 2500 items of real service information for 10 types of QoS data. For our testing purposes, we use two negatively affecting QoS criteria (response time, and latency) and three positively affecting QoS criteria (availability, throughput, and reliability). We repeated each test case 30 times to obtain the average values for that test case. We prepared 2,000,000 to 10,000,000 planners in a Big Data environment from the QWS data.

Evaluation Metrics: Our aim was to evaluate the efficiency and effectiveness of the internal, external, and joint traffic optimizations of multiobjective selection methods in a Hadoop environment. To achieve this, we defined four modes (Mode 1, Mode 2, Mode 3, and Mode 4) and their respective computational complexities (*P, Q, R*, and *S*) as shown in Eq. 31, Eq. 32, Eq. 33, and Eq. 34.

Next, based on the computational complexities of Eq. 31, Eq. 32, Eq. 33, and Eq. 34, we defined three traffic metrics C_{Int}, C_{Ext}, and C_{Joint} in Eq. 35, Eq. 36, and Eq. 37.

$$P = \textit{Computational complexity of Mode } 1 \textit{ (Default)}: \textit{ No external and internal traffic solutions are applied} \quad (31)$$

$$Q = \textit{Computational complexity of Mode } 2: \textit{ Internal solution is applied but external traffic solutions are not applied} \quad (32)$$

$$R = \textit{Computational complexity of Mode } 3: \textit{ External solutions are applied but internal traffic solution is not applied.} \quad (33)$$

$$S = \textit{Computational complexity of Mode } 4: \textit{ External solutions and internal traffic solutions are applied.} \quad (34)$$

$$C_{Int} = R - S \quad (35)$$

$$C_{Ext} = Q - S \quad (36)$$

$$C_{Joint} = P - S \quad (37)$$

To calculate these measures, we executed multiobjective selection methods for the various modes by increasing the number of plans while increasing the number of nodes of the Hadoop environment. We maintained a default mode (Mode 1) with minimal replica and block distribution and no IMR agent. Modes 2, 3, and 4 were devised to satisfy particular traffic solutions as follows.

In Section III-D-1, we proposed two approaches to address the key external traffic congestions occurring during selection. We integrated these two methods to conduct experiments that evaluated the efficiency of the proposed method under multiobjective selection approaches. First, we sorted the WSs based on their utility QoS values and partitioned the dataset into *n* sets. We then took the average QoS of each batch and found their proportional values. We set the default mode as one and the remaining batches according to their proportional values. This is simulated in the *QSD rule* defined in Section III-D. Next, we multiplied the respective proportional values by *m* to simulate the *TRD rule*, also defined Section III-D. We then observed the traffic-aware replica distribution across the network (here, *m* and *n* are positive integers). According to these results, we distributed services and their replicas across the HDFS. We conducted experiments in Mode 3 and 4 with external traffic solutions to obtain the respective metrics.

In Section III-D-2, we proposed an *IMR agent* approach to address the key internal traffic congestion occurring during selection. We conducted experiments in Modes 2 and 3 with an *IMR agent* to obtain the respective metrics.

A. EFFICIENCY

We calculated the efficiency of the proposed internal, external. and jointly optimized methods as $E(T_{Internal})$, $E(T_{External})$, and $E(T_{joint})$, respectively, based on the computational complexities of four modes (Eq. 31, Eq. 32, Eq. 33, and Eq. 34) and three basic traffic metrics (Eq. 35, Eq. 36, and Eq. 37).

Based on Eq. 38, Eq. 39, and Eq. 40, we can evaluate the traffic efficiencies of the three multiobjective selection methods, as follows.

$$E(C_{Int}) = C_{Int}/R \quad (38)$$

$$E(C_{Ext}) = C_{Ext}/Q \quad (39)$$

$$E(C_{Joint}) = C_{Joint}/P \quad (40)$$

1) INTERNAL TRAFFIC EFFICIENCY

We considered the internal traffic efficiency of the three selection methods, namely Dijkstra, 0-1 MCKP, and ABC. We calculated the internal traffic efficiencies using Eq. 38 above. Tables 4, 5, and 6 give results for the internal traffic efficiencies for each method. Here, M represents the number of plans, in millions. Fig. 4 shows the processing costs for Mode 1. This can be used as a reference for the processing costs for the various methods in the Hadoop environment.

For all three methods, the efficiency is suddenly reduced as the number of plans changes from 2M to 4M. The number of plans is increased by changing the number of mappers. For 2M, a single mapper is used. For 4M, two mappers are used. If a single mapper is used, the internal efficiency will be higher than when two mappers are used because using two mappers will generate more internal traffic. That is, increasing the number of plans will increase the number of mappers, thereby increasing the internal traffic. In our experiments, we gradually increased the number of mappers and the number of data nodes in the selection process. The internal traffic should increase when increasing the number of mappers because of the increased cross shuffling in the selection process. However, by using an IMR agent,

TABLE 4. Internal traffic efficiencies for the 0-1 MCKP method.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	21%	19%	9%	14%	15%
3	25%	20%	14%	12%	19%
4	27%	20%	18%	25%	28%

TABLE 5. Internal traffic efficiencies for the ABC method.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	10%	8%	13%	14%	16%
3	13%	11%	7%	8%	15%
4	17%	9%	13%	17%	21%

TABLE 6. Internal traffic efficiencies for the Dijkstra method.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	12%	10%	8%	12%	14%
3	13%	11%	7%	8%	15%
4	11%	9%	8%	11%	17%

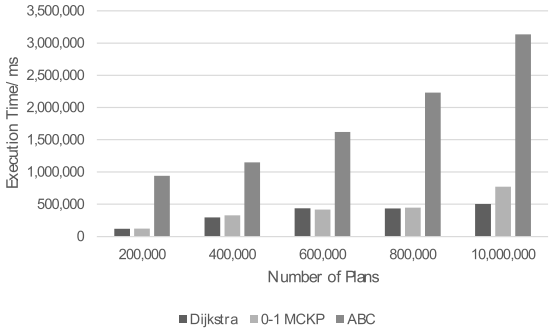

FIGURE 4. 'P' values for two data nodes of the Hadoop cluster.

the internal traffic efficiency is increased, from two mappers upwards. That is, as the internal traffic of the selection process increases, the IMR agent causes an increase in the internal traffic efficiency of the selection process. We found that, across the two-node, three-node, and four-node modes, the 0-1 MCKP method improved from an average 16% internal traffic efficiency to 23%. For Dijkstra, it was from 10.8% to 11.2%. For ABC, it was from 10.8% to 15.6%.

The 0-1 MCKP method demonstrated a higher traffic efficiency than the Dijkstra and ABC methods. According to our investigation of the internal data used in the shuffling stage, 0-1 MCKP generated more internal data than the other two methods. This implies that the IMR agent works more effectively when increasing the internal data of the process. More generally, the proposed IMR agent works efficiently when increasing the number of data nodes, the number of mappers, and the internal data in the selection process.

2) EXTERNAL TRAFFIC EFFICIENCY

We now consider the external traffic efficiency for the three selection methods. We calculated the external traffic efficiencies using Eq. 39 above. Tables 7, 8, and 9 give results for the external traffic efficiencies for the Dijkstra, 0-1 MCKP, and ABC methods, respectively. Again, M represents the number of plans, in millions.

As for internal traffic efficiency, a sudden reduction in efficiency occurs when the second mapper is introduced, as the number of plans increases from 2M to 4M. However, efficiency is increased as the number of mappers is further increased. In our experiments, we gradually increased the number of mappers and the number of data nodes in the selection process. This means that the popularity of the service data of the previous user case and the test case will directly affect the following test case. According to our hypothesis, popularity should be proportional to the hotness of the service data, which implies that increasing the popularity of service data should reduce external traffic. We observed that the proposed rules for external traffic successfully reduced the overall traffic for the selection process. We found that, across the two-node, three-node, and four-node modes, the ABC method improved from an average 21.2% external traffic efficiency to 30.4%. For 0-1 MCKP, it was from 12.2% to 22.6%. For Dijkstra, it was from 10.4% to 21.6%.

The ABC method demonstrated a higher traffic efficiency than the other two methods. According to Fig. 4, ABC also has the highest processing cost. This implies that ABC should have the highest external traffic. Therefore, the proposed method for external traffic efficiency works best for increased external traffic in the selection process.

TABLE 7. External traffic efficiencies for the ABC method.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	22%	18%	17%	21%	28%
3	27%	25%	21%	26%	34%
4	33%	27%	21%	32%	39%

TABLE 8. External traffic efficiencies for the 0-1 MCKP method.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	12%	10%	9%	8%	22%
3	23%	21%	17%	15%	27%
4	26%	18%	16%	22%	31%

TABLE 9. External traffic efficiencies for the Dijkstra method.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	8%	7%	6%	14%	17%
3	21%	14%	12%	21%	26%
4	19%	18%	18%	26%	28%

3) JOINT OPTIMIZATION OF TRAFFIC EFFICIENCY

Finally, we consider the efficiency of joint optimization of the internal and external traffic for the three selection methods. We calculated the jointly optimized traffic efficiencies using Eq. 40 above. Tables 10, 11, and 12 give results for the jointly optimized efficiencies for the three methods while increasing the number of plans and the data nodes in the Hadoop environment. Again, M represents the number of plans, in millions.

We found that, across the two-node, three-node, and four-node modes, the 0-1 MCKP method improved from an average 18.2% jointly optimized traffic efficiency to 36.2%. For ABC, it was from 20.4% to 31.6%. For Dijkstra, it was from 15.8% to 19.6%.

The 0-1 MCKP method demonstrated the highest joint optimization among the methods, with ABC being next best. The 0-1 MCKP method could achieve 49% efficiency by joint optimization when the number of data nodes was increased to four. For ABC, 40% could be achieved and Dijkstra could achieve a maximum of 28%. The selection process usually reduces the efficiency of joint optimization when the process introduces a second mapper. However, the process increases efficiency as the number of plans in the process increases. This implies that the proposed method for jointly optimized traffic-aware selection can work efficiently.

TABLE 10. Jointly optimized traffic efficiencies for 0-1 MCKP.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	23%	19%	8%	14%	27%
3	38%	31%	21%	17%	36%
4	43%	28%	24%	37%	49%

TABLE 11. Jointly optimized traffic efficiencies for ABC.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6M	8M	10M
2	19%	13%	17%	22%	31%
3	27%	23%	15%	21%	36%
4	37%	23%	22%	36%	40%

TABLE 12. Jointly optimized traffic efficiencies for Dijkstra.

Number of Nodes	Traffic efficiency with respect to the number of plans				
	2M	4M	6NM	8M	10M
2	11%	14%	10%	22%	24%
3	14%	13%	12%	18%	22%
4	17%	16%	16%	21%	28%

B. EFFECTIVENESS

For this metric, we mainly considered the computational complexities and the precision of the three methods in the Hadoop space.

1) EFFECTIVENESS OF THE TRAFFIC SOLUTION

We calculated the effectiveness of the respective internal (C_{Int}), external (C_{Ext}), and jointly optimized (C_{Joint}) methods by taking the average values for the various numbers of plans, (2 M, 4 M, 6 M, 8 M, and 10 M) in all three test cases (2, 3, and 4 data nodes). For example, we consider 2 million plans for all three data-node numbers in calculating the respective C_{Int} as $C_{Int,a}$, $C_{Int,b}$, and $C_{Int,c}$ values for ABC. We then specify the average traffic effectiveness for ABC in Hadoop as $(C_{Int,a} + C_{Int,b} + C_{Int,c})/3$. Likewise, we calculate the average values for all numbers of plans for the three selection methods.

The results are shown in Figs. 5, 6, and 7, with respect to their internal, external, and jointly optimized traffic costs. All three graphs maintain the same pattern with only slight deviations. The average line graph for ABC is concave in shape, whereas 0-1 MCKP and Dijkstra maintained a nearly linear relationship with execution time. This means that the derivative of the ABC line graph increases with an increasing

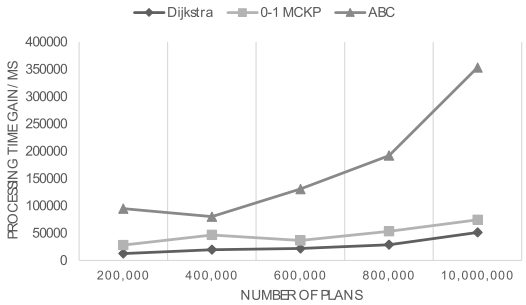

FIGURE 5. Average processing time gain by internal traffic solution for multiobjective selections in a multinode Hadoop cluster.

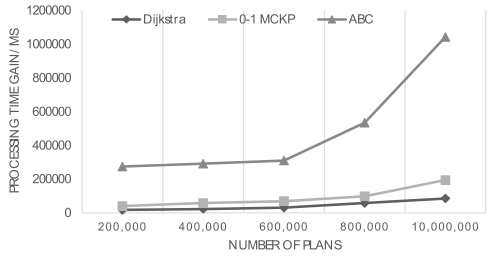

FIGURE 6. Average processing time gain by external traffic solution for multi-objective selections in a multi-node Hadoop cluster.

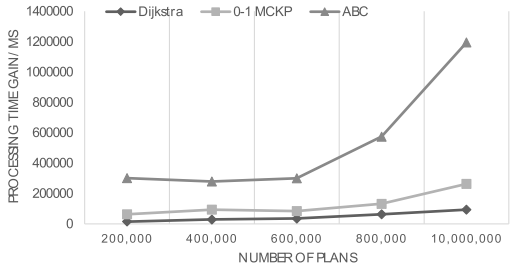

FIGURE 7. Average processing time gain by joint traffic solution in multiobjective selections in the multinode Hadoop cluster.

number of plans, but the other two methods do not show such an exponential increment. That is, internal, external, and jointly optimized ABC traffic effectiveness for the ABC method increases exponentially, whereas the increase is linear for the other two methods. This is because the ABC method has a multiobjective selection requirement, in contrast to combinatorial and linear selection requirements. Numbers of service plans are directly affected to increase traffic in all three methods.

The 0-1 MCKP method maintained relatively high traffic costs for the various traffic types. This is because 0-1 MCKP has to solve combinatorial selection requirements, unlike the Dijkstra method. This implies that the linear optimal selection requirement solved by the Dijkstra method causes it to generate the lowest possible traffic cost among the three methods.

These three graphs plotted the caused and solved the average traffic for the various selection methods, implying that the proposed traffic solutions work effectively with respect to the various traffic concerns occurring with the various methods.

2) EFFECTIVENESS IN PRECISION

In Section IV, we proposed a *threshold plan for each batch*, which is an approach that addresses the precision of the proposed method by reducing the search space and segmenting it in terms of batchwise content. To evaluate the proposed method from the perspective of the efficiency of the results, we conducted two different tests with two data sets.

The Dijkstra and 0-1 MCKP methods always result in a globally optimal solution. Therefore, we needed to consider only the ABC method to measure the precision of the composition plan for multivariate optimization. We conducted experiments using the QWS dataset under various types of candidate services to find the average precision of the given result results for the ABC algorithm. We found the respective error deviations, as shown in Eq. 41 and calculated the average precision, as shown in Eq. 42. Here, D_{sum} is the deviation from the ascending ordered result and n is the number of attempts.

$$D_{sum} = \sum_{i}^{n} Deviated\ Result \tag{41}$$

$$Average\ Precision = 100 - D_{sum}/n \tag{42}$$

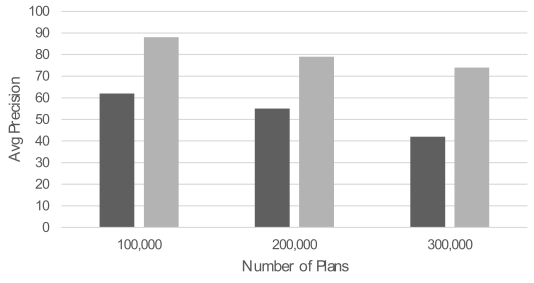

FIGURE 8. Comparison of the precision with and without the batchwise operator.

Fig. 8 shows the results with and without the batchwise operator for the QWS data set. It shows that the proposed method consistently maintained a relatively high precision, whereas the alternative would have a drastic reduction in precision. This indicates that the batchwise operator is the main reason for maintaining the effectiveness of the results in a consistent manner.

Summaries of the respective methods are discussed in the following. To find the summarized average values of the respective methods, first we calculate the average values of three rows separately and then calculate the average of the respective three average values of Table 4 to 12.

Internal traffic efficiencies of each method are 19% by 0-1 MCKP, 13% by ABC and 11% by Dijkstra. The highest internal traffic efficiency is earned by the 0-1 MCKP and lowest by the ABC. That means, for combinatorial requirement (0-1 MCKP) result in the highest number of intermediate results while multivariate requirement (ABC) has minimal intermediate results during the process. All three methods show relatively low increment value while increasing number of nodes for a particular number of plans and increasing the number of plans for a particular number of nodes.

External traffic efficiencies of each method are 26% by ABC, 18% by 0-1 MCKP and 17% by Dijkstra. The highest external traffic efficiency is earned by the ABC and lowest by the Dijkstra. That means, during the execution of the multivariate algorithm (ABC), process earned the highest external traffic efficiency compared to the other two methods. Relatively ABC has an exponential increase of processing cost and this is caused by the computation complexity of multivariate composition requirement. However, ABC results in more hot data due to the highest computation complexity than the other two methods. In return, ABC results in the highest efficiency as well. This implies, proposed external traffic solution works better while the presence of more hot data in Big Data space. Meantime, Dijkstra has minimal computation complexity compared to the other two methods and it results in relatively lowest efficiency. This means the computation complexity of the objective function shows a roughly proportional relationship to the external traffic.

Joint traffic efficiencies of each method are 28% by 0-1 MCKP, 25% by ABC and 17% by Dijkstra. The highest joint traffic efficiency is earned by the 0-1 MCKP and minimal by the Dijkstra method. Dijkstra has minimal computation complexity, therefore linear optimal algorithm earned minimal joint traffic benefits. However, joint traffic efficiency doesn't show the proportional relationship to the computation complexity of the objective function.

ABC shows the highest traffic effectiveness compared to the other two methods. It maintains a considerable gap between the other two methods. This caused by the computation complexity of the ABC method. Respectively 0-1 MCKP and Dijkstra show relatively low and increment in all three methods (internal, external and joint) while increasing the number of plans in the environment as shown in Fig. 5, 6 and 7.

According to the Fig 4, the highest computation cost is shown by the ABC and lowest by the Dijkstra. However, ABC shows exponential growth and the considerable gap in the processing time compared to the other two methods.

VI. RELATED WORK

This section provides an overview of the corresponding contributions related to this work. Service selection based on Big Data space is scarce. Wang *et al.* [1] proposed the BigData Space service selection, a method based on heterogeneous QoS-aware and trust values. It focused on accumulating both qualitative and quantitative values into a single trade-off model. It can be considered a comprehensive qualitative and quantitative approach. However, it does not consider heterogeneous-selection requirements, which is the key difference in our approach. It considers a multi-CPU IBM high-end server for a Big Data space, but not traffic issues. In contrast, we have used one of the most populated Big Data space, the Hadoop multicluster approach, as our Big Data environment. In addition, their method does not have the ability to deal with dynamic selection requirements. In our case, we have analyzed our solution for three distinct selection requirements. Moreover, they proposed qualitative and quantitative QoS-aware static service selection, whereas we propose a quantitative QoS-aware dynamic service selection.

A traffic-efficient Big Data processing model has been proposed by Xia *et al.* [18]. They proposed an architecture that could deal with both the front-end and back-end traffic in a Big Data space. Lin *et al.* [19] discussed the issues that occur because of the Zipf phenomena in a case study. It focused on the limitations to the parallelization of the MR job. The Pareto phenomenon is the naturally observable concern in most cases.

Kim and Doh [20] proposed a solution that can select services based on QoS and trust. A two-layered preference service-selection solution was proposed. It had to pass two distinct phases because of the two-layered approach. The overall process was layered, and this is also a qualitative and quantitative QoS-aware service-selection solution, but it is not a framework. A further advanced approach has been proposed by Wahab *et al.* [21], which focuses on providing a precise reputation-assessment mechanism in an open environment. This reputation-evaluation process is interesting, and it offers a means of manipulating the reputation for overall service selection. Wan *et al.* [22] proposed a cloud-based service-selection method. It also discussed trending concerns in the service domain and proposed an architecture for discussing how to deal with cloud services. Huang [23] proposed a QoS estimation method through online communication. This might be very useful during the selection process because of the QoS preferential estimation that reduces the overall selection time. It is based on both qualitative and quantitative concerns. However, none of these methods are designed to cope with Big Data space.

Kang *et al.* [24] proposed a method that considered globally optimal service selection for multiple competitive peer users. They also discussed an interesting topic, namely resolving conflicting requests and allowing them to find their optimal selection within the range of service distributions. They proposed an agent as a solution, which also features in our proposal. It is the key element in avoiding conflicts in the selection scenario. El Hadad *et al.* [25] proposed a QoS broker architecture to find the optimum WS, which can be a provided service, based on user queries. They proposed a selection solution that can offer automatic service composition (ASC), which is also one of the trending requirements in the industry. It focused on the transactional constraints of the ASC process and aimed to satisfy the selection requirements while considering the functional requirements, transactional properties, and QoS characteristics.

Gao *et al.* [26] proposed a QoS-aware service selection based on a genetic algorithm that was mainly oriented toward trust in the QoS. They designed a trust-oriented genetic algorithm called TOGA, and their aim was to find a near-optimal plan for the composition system. Zhang *et al.* [27] proposed an ant-colony-based service-selection algorithm for large-scale QoS preference selection. They proposed a clustering-guided method that uses a skyline-guided process to filter the candidate service classes and cluster them by shrinking. These methods are based on intelligent agent-based service selection. In our approach, we proposed a middle agent (not an AI agent) to address traffic congestion during the selection process. Moreover, we proposed ABC for multivariate QoS optimization in the service selection.

An interactive analytical process was proposed for Hadoop space by Chen *et al.* [11]. They focused on addressing the traffic congestion caused by Zipf and Pareto phenomena. They proposed a solution based on a cross-industry study of the efficient management of MR workloads. Ke *et al.* [28] proposed a traffic-aware partition-based Big Data method that focused on reducing internal traffic congestion during an MR job. They proposed an intermediate data-partition solution to address these concerns. Their solution used a decomposition-based distributed algorithm to deal with large-scale optimization. However, none of these solutions were related to service selection but were discussed as the internal and external traffic solutions to MR jobs. We are proposing a solution that can cope with both internal and external traffic congestion in an efficient manner. In addition, we have applied this solution to practical application areas such as the service selection domain.

Traffic occurrence and optimization beyond the data center has been studied by Ersoz *et al.* [29]. They considered cluster-based network traffic characterization for multi-tier data centers. Their focus was the characteristics of network behavior within a clustered, multi-tiered data center with respect to the interarrival-time distribution of requests to individual server nodes and tiers. This approach gave insights about low-level traffic handling and communication between tiers.

Traffic and communication optimization in Big Data infrastructures has been studied by various scholars.

Zhang *et al.* [10] proposed a method for optimizing data shuffling in parallel computation by user-defined functions in an MR process. In addition, they proposed a framework called SUDO, which is an optimization framework that reasons about data-partition properties, functional properties, and data shuffling. This differed from our work, where we proposed shuffling and data utilization in the parallel processing of the MR process. Aledhari *et al.* [30] proposed a deep-learning-based data-minimization algorithm for the fast and secure transfer of big genomic data. They focused on maximizing the channel utilization by decreasing the bits needed for transmission of the dataset. They proposed a novel deep-learning CNN-based algorithm that minimizes the dataset during transfer and protects the data from middleman attacks and other types of attack, such as changing the binary representation of the dataset. Zhao *et al.* [31] proposed a distributed graph-parallel computing system with lightweight communications. Their system, called Ligraph, processes large-scale graph data in a distributed mode with optimal communication overhead. Yan *et al.* [32] proposed heterogeneous data-storage management with deduplication for cloud computing. They focused on encrypted data storage, management, and the deduplication process across the cloud environment. Comparing our method with this method, we selected a Big Data environment and the MR process because of their lightweight data and extensive resource-starvation processing with minimal overhead traffic for solving heterogeneous-selection optimization problems.

For threshold-based policies, service selection and scheduling are both NP-hard problems. Scheduling problems are applicable to many domains including services, communication, and planning. Chen *et al.* [33] proposed buffer-aware scheduling with the adaptive transmission, which focused on obtaining the optimal trade-off between the average delay and power consumption. We focused on the QoS when aiming for optimal traffic efficiency. X. Chen et al. modeled the problem based on a Markov decision process and proposed an algorithm to obtain the optimal solution. Asadi *et al.* [34] surveyed scheduling in wireless communication, describing opportunistic scheduling from various perspectives such as capacity, QoS, fairness, and distributed scheduling. Our proposal also considered if the processing-resource capacity for Big Data, with linear, combinatorial and multi-objective QoS, and with distributed computing would be satisfied with respect to resource starvation.

VII. CONCLUSION

We have proposed a multiobjective serviceselection solution that considers external and internal traffic congestions in Big Data space. Proposed method addresses the concerns arising from the flooding of services and then the selection processes can be handled more efficiently in Big Data space. Data traffic caused by the ZipF and Pareto are called as the external traffic congestions. Data traffic caused by the shuffling stage of the MR process is identified as the internal traffic. We have proposed a complete model for traffic control while dealing with the selection process in MR jobs. Novel QoS-aware traffic-efficient methods have been proposed for external traffic congestion. In addition, we have introduced a middle agent to address the internal traffic congestion, which eases the reducer workload in an efficient manner. We adopted three selection criteria's for the multiobjective selection methods, which are linear optimal, combinatorial and multivariate QoS optimizations. These methods are based on both linear programming and dynamic programming. The proposed distributed algorithm can adapt easily to dynamic selection requirements. Our experimental results demonstrate that the proposed method handles traffic congestions efficiently and effectively in producing composition plans for multiobjective selection requirements. Internal traffic efficiency shows relatively low compared to the external traffic efficiency. Only external traffic efficiency shows a nearly proportional relationship to the computation complexity of the objective functions. The Proposed method is a well-behaved QoS-aware rule-based traffic-efficient service-selection method for Big Data space. In future work, we aim to investigate qualitative QoS criteria and uncertainty in the QoS values for the selection method.

REFERENCES

[1] H. Wang, C. Yu, L. Wang, and Q. Yu, "Effective bigdata-space service selection over trust and heterogeneous QoS preferences," *IEEE Trans. Serv. Comput.*, vol. 11, no. 4, pp. 644–657, Jul./Aug. 2018.

[2] P. J. Guo, "Software tools to facilitate research programming," Stanford Univ., Stanford, CA, USA, 2012. [Online]. Available: http://www.pgbovine.net/publications/Philip-Guo_PhD-dissertation_software-tools-for-research-programming.pdf

[3] T. H. S. A. Siriweera, I. Paik, B. T. G. S. Kumara, and C. K. Koswatta, "Architecture for intelligent big data analysis based on automatic service composition," *Int. J. Big Data*, vol. 2, no. 2, pp. 1–16, 2015.

[4] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[5] T. H. S. A. Siriweera, I. Paik, and B. T. G. S. Kumara, "QoS-aware traffic-efficient Web service selection over bigdata space," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.*, Dec. 2016, pp. 197–203.

[6] T. Yu and K.-J. Lin, "Service selection algorithms for Web services with end-to-end QoS constraints," *Inf. Syst. E-Bus. Manage.*, vol. 3, no. 2, pp. 103–126, 2005.

[7] M. Alrifai, T. Risse, P. Dolog, and W. Nejdl, "A scalable approach for QoS-based Web service selection," in *Proc. ICSOC Workshops*, 2008, pp. 190–199.

[8] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for QoS-based Web service composition," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 11–20.

[9] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 881–890.

[10] J. Zhang *et al.*, "Optimizing data shuffling in data-parallel computation by understanding user-defined functions," in *Proc. NSDI*, vol. 12, 2012, p. 22.

[11] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads," *VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.

[12] T. H. S. A. Siriweera and I. Paik, "Service selection on bigdata-space based on heterogeneous QoS preferences," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Oct. 2016, pp. 1–4.

[13] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. INFOCOM 18th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 1, Mar. 1999, pp. 126–134.

[14] D. G. Dolgikh and A. M. Sukhov, "Parameters of cache systems based on a Zipf-like distribution," *Comput. Netw.*, vol. 37, no. 6, pp. 711–716, 2001.

[15] F. Ahmad, S. Lee, M. Thottethodi, and T. N. Vijaykumar, "MapReduce with communication overlap (MaRCO)," *J. Parallel Distrib. Comput.*, vol. 73, no. 5, pp. 608–620, 2013.
[16] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Apr. 2007.
[17] E. Al-Masri and Q. H. Mahmoud, "Discovering the best Web service," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 1257–1258.
[18] Y. Xia, J. Chen, X. Lu, C. Wang, and C. Xu, "Big traffic data processing framework for intelligent monitoring and recording systems," *Neurocomputing*, vol. 181, pp. 139–146, Mar. 2016.
[19] J. Lin, "The curse of Zipf and limits to parallelization: A look at the stragglers problem in mapreduce," in *Proc. 7th Workshop Large-Scale Distrib. Syst. Inf. Retr.*, vol. 1, 2009, pp. 57–62.
[20] Y. Kim and K.-G. Doh, "A trust management model for QoS-based service selection," in *Proc. WISA*, 2012, pp. 345–357.
[21] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for Web services: Single, composite, and communities," *Decision Support Syst.*, vol. 74, pp. 121–134, Jun. 2015.
[22] Z. Wan, F. J. Meng, J. M. Xu, and P. Wang, "Service composition pattern generation for cloud migration: A graph similarity analysis approach," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun./Jul. 2014, pp. 321–328.
[23] X. Huang, "UsageQoS: Estimating the QoS of Web services through online user communities," *ACM Trans. Web*, vol. 8, no. 1, 2013, Art. no. 1.
[24] G. Kang, J. Liu, M. Tang, X. Liu, and K. K. Fletcher, "Web service selection for resolving conflicting service requests," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2011, pp. 387–394.
[25] J. El Hadad, M. Manouvrier, and M. Rukoz, "TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition," *IEEE Trans. Serv. Comput.*, vol. 3, no. 1, pp. 73–85, Jan./Mar. 2010.
[26] H. Gao, J. Yan, and Y. Mu, "Trust-oriented QoS-aware composite service selection based on genetic algorithms," *Concurrency Comput. Pract. Express*, vol. 26, no. 2, pp. 500–515, 2014.
[27] C. Zhang, H. Yin, and B. Zhang, "A novel ant colony optimization algorithm for large scale QoS-based service selection problem," *Discrete Dyn. Nature Soc.*, vol. 2013, Jun. 2013, Art. no. 815193. [Online]. Available: https://www.hindawi.com/journals/ddns/2013/815193/
[28] H. Ke, P. Li, S. Guo, and M. Guo, "On traffic-aware partition and aggregation in mapreduce for big data applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 3, pp. 818–828, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7079380/
[29] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *Proc. 27th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2007, p. 59.
[30] M. Aledhari, M. Di Pierro, M. Hefeida, and F. Saeed, "A deep learning-based data minimization algorithm for fast and secure transfer of big genomic datasets," *IEEE Trans. Big Data*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8290833/
[31] Y. Zhao, K. Yoshigoe, J. Bian, M. Xie, Z. Xue, and Y. Feng, "A distributed graph-parallel computing system with lightweight communication overhead," *IEEE Trans. Big Data*, vol. 2, no. 3, pp. 204–218, Sep. 2016.
[32] Z. Yan, L. Zhang, W. Ding, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," *IEEE Trans. Big Data*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/7919246/
[33] X. Chen, W. Chen, J. Lee, and N. B. Shroff, "Delay-optimal buffer-aware scheduling with adaptive transmission," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2917–2930, Jul. 2017.
[34] A. Asadi and V. Mancuso, "A survey on opportunistic scheduling in wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1671–1688, 4th Quart., 2013.
[35] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1478–1490, Jul. 2017.
[36] J.-H. Cho, H.-G. Ko, and I.-Y. Ko, "Adaptive service selection according to the service density in multiple QoS aspects," *IEEE Trans. Serv. Comput.* vol. 9, no. 6, pp. 883–894, Nov./Dec. 2016.
[37] W. Song and H.-A. Jacobsen, "Static and dynamic process change," *IEEE Trans. Serv. Comput.*, vol. 11, no. 1, pp. 215–231, Jan./Feb. 2016.
[38] S. Hu, V. Muthusamy, G. Li, and H.-A. Jacobsen, "Distributed automatic service composition in large-scale systems," in *Proc. Distrib. Event-Based Syst. Conf.*, Jul. 2008, pp. 233–244.
[39] G. Li, V. Muthusamy, and H.-A. Jacobsen, "A distributed service-oriented architecture for business process execution," *ACM Trans. Web*, vol. 4, no. 1, Jan. 2010, Art. no. 2.
[40] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.
[41] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction," *IEEE Trans. Softw. Eng.*, vol. 40, no. 2, pp. 192–215, Feb. 2014.
[42] L. Wang, S. Guo, X. Li, B. Du, and W. Xu, "Distributed manufacturing resource selection strategy in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 94, nos. 9–12, pp. 3375–3388, 2018.
[43] J. Zhou and X. Yao, "A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition," *Int. J. Adv. Manuf. Technol.*, vol. 88, nos. 9–12, pp. 3371–3387, 2017.
[44] X. Xue, Y.-M. Kou, S.-F. Wang, and Z.-Z. Liu, "Computational experiment research on the equalization-oriented service strategy in collaborative manufacturing," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 369–383, Mar./Apr. 2018.
[45] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things," *IEEE Trans. Ind. Inform.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8377998/
[46] S. Kanrar and N. K. Mandal, "Traffic analysis and control at proxy server," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2017, pp. 164–167.
[47] M. Furqan, C. Zhang, W. Yan, A. Shahid, M. Wasim, and Y. Huang, "A collaborative hotspot caching design for 5G cellular network," *IEEE Access*, vol. 6, pp. 38161–38170, Jul. 2018.
[48] H. Beyranvand, M. Lévesque, M. Maier, J. A. Salehi, C. Verikoukis, and D. Tipper, "Toward 5G: FiWi enhanced LTE-A HetNets with reliable low-latency fiber backhaul sharing and WiFi offloading," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 690–707, Apr. 2017.
[49] K. Wang, X. Li, H. Ji, and X. Du, "Modeling and optimizing the LTE discontinuous reception mechanism under self-similar traffic," *IEEE Trans. Veh. Technol.*, vol. 65, no. 7, pp. 5595–5610, Jul. 2016.

T. H. AKILA S. SIRIWEERA received the B.Sc. degree in mathematics and computer science from the University of Peradeniya, Sri Lanka, in 2005, and the M.Sc. degree in computer science and engineering from the University of Aizu, Japan, in 2016, where he is currently pursuing the Ph.D. degree in computer and science in engineering under the supervision of Prof. I. Paik.

Since 2016, he has been a Research Assistant with the University of Aizu. His current research interests include service computing, big data analytics, artificial intelligent planning, and deep learning.

Mr. Siriweera received a special merit award as the Best eHealth Producer in 2011 by the ICT Agency, the government of Sri Lanka; achieved outstanding achievements in scientific research in the Aizu region in 2017 by the Aizu-Wakamatsu Foundation for the promotion of education and science; and was honored as an Outstanding Research Assistant for the years 2016 and 2017 by the University of Aizu.

INCHEON PAIK received the master's and Ph.D. degrees in electronics engineering from Korea University in 1987 and 1992, respectively. He is currently a Full Professor with the University of Aizu, Japan. His research interests include semantic Web, Web services and their composition, Web data mining, big data analytics, deep learning, awareness computing, and agents on semantic Web. He served in several conferences as the chair. He also serves as an Editor for the journals of JIPS and IEICE.