Journal of the Operational Research Society

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tjor20

Gaussian relevance vector MapReduce-based annealed Glowworm optimization for big medical data scheduling

Rizwan Patan, Suresh Kallam, Amir H. Gandomi, Thomas Hanne & Manikandan Ramachandran

To cite this article: Rizwan Patan, Suresh Kallam, Amir H. Gandomi, Thomas Hanne & Manikandan Ramachandran (2022) Gaussian relevance vector MapReduce-based annealed Glowworm optimization for big medical data scheduling, Journal of the Operational Research Society, 73:10, 2204-2215, DOI: 10.1080/01605682.2021.1960908

To link to this article: https://doi.org/10.1080/01605682.2021.1960908

Published online: 23 Aug 2021.

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
2022, VOL. 73, NO. 10, 2204–2215
https://doi.org/10.1080/01605682.2021.1960908

ORIGINAL ARTICLE

OPEN ACCESS Check for updates

Gaussian relevance vector MapReduce-based annealed Glowworm optimization for big medical data scheduling

Rizwan Patan[a], Suresh Kallam[b], Amir H. Gandomi[c], Thomas Hanne[d] and Manikandan Ramachandran[e]

[a]Department of Computer Science and Engineering, Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada, India; [b]Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, Tirupati, India; [c]Faculty of Engineering and Information Technology, University of Technology Sydney, Australia; [d]Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland; [e]School of Computing, SASTRA Deemed University, Thanjavur, India

ABSTRACT

Various big-data analytics tools and techniques have been developed for handling massive amounts of data in the healthcare sector. However, scheduling is a significant problem to be solved in smart healthcare applications to provide better quality healthcare services and improve the efficiency of related processes when considering large medical files. For this purpose, a new hybrid model called Gaussian Relevance Vector MapReduce-based Annealed Glowworm Optimization Scheduling (GRVM-AGS) was designed to improve the balancing of large medical data files between different physicians with higher scheduling efficiency and minimal time. First, a GRVM model was developed for the predictive analysis of input medical data. This model reduces the storage complexity of large medical data analysis by means of eliminating unwanted patient information and predicts the disease class with help of a Gaussian kernel function. Afterwards, GRVM performs AGS to schedule the efficient workloads among multiple datacenters based on the luciferin value in the smart healthcare environment with reduced scheduling time. Through computational experiments, we demonstrate that GRVM-AGS increases the scheduling efficiency and reduces the scheduling time of large medical data analysis compared to state-of-the-art approaches.

ARTICLE HISTORY
Received 25 March 2020
Accepted 16 July 2021

KEYWORDS
Annealed Glowworm optimization scheduling; big data analytics; datacenters; Gaussian kernel function; medical files; workload balancing

1. Introduction

Big Data analytics has received enhanced recognition in many fields, including healthcare systems as one of the most promising sectors. Smart healthcare applications that use Internet of Things (IoT) technologies have recently gained greater significance because they allow a patient's health conditions to be monitored remotely. The healthcare domain is comprised of hierarchical terminologies of different diseases, their syndromes and diagnosis information, laboratory results, and patient details, such as admission histories, drug, and billing information. Thus, such databases for the availed clinical services are huge in size and complex to analyze. Therefore, in our proposed methodology, a scheduling process is introduced to achieve fast and efficient action in the healthcare system. While scheduling systems prioritize the most important projects first, the cloud and IoT assist healthcare teams, such as doctors, nurses, and specialists, in monitoring a patient's conditions. In recent years, scheduling has become one of the most critical issues in large medical data analytics regarding the distribution of workload among physicians and other resources in a smart healthcare environment. Numerous research works have been conducted to design scheduling systems for medical data. However, the scheduling performance of existing methods is not effective when considering large medical files. Therefore, a novel metaheuristic, called Gaussian Relevance Vector MapReduce-based Annealed Glowworm Optimization Scheduling (GRVM-AGS), is introduced in this study to enhance the scheduling performance by minimizing storage complexity and the time to conduct large medical data analytics.

This article is organized as follows. Section 2 explains the background and reviews related works. In Section 3, the proposed GRVM-AGS technique is described in further details. In Section 4, experimental settings are presented, and the analysis of results is explained in Section 5. Section 6 concludes the article.

CONTACT Thomas Hanne thomas.hanne@fhnw.ch Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland

2. Related work

In the study by Elhoseny et al. (2018), a hybrid model was developed to handle big data in healthcare service applications. However, the performance of the scheduling was poor when examining the multimedia data of patients. A task-level adaptive MapReduce framework was presented in Zhang et al. (2015) to decrease the storage space of big healthcare data, but workload scheduling remained an open issue.

A novel technique was developed by Pagán et al. (2018) to optimize energy utilization through workload scheduling in datacenters in an eHealth scenario. However, the problem of storage optimization was not solved. A survey of different storage optimization techniques in a big data cluster environment was analyzed by Chakravarthy et al. (2019), but effective storage optimization was not achieved.

An early big data reduction framework was introduced by ur Rehman et al. (2016) to reduce the cost of cloud service consumption when performing big data analysis, yet the performance of scheduling was poor. Real-time awareness scheduling was presented by Xu et al. (2018) to address the issue of real-time multimedia big data computing in IoT and to achieve better workload distribution with less latency. However, the minimization of scheduling time was not sufficient.

A resource-based data accessing method (UDA-IoT) was introduced by Xu et al. (2014) to increase access to IoT data resources, but the study failed to consider storage complexity. An adaptive streaming technique was designed by Mohammadi et al. (2018) to transmit and share the enormous media healthcare data timely with assured quality of service (QoS) support at remote terminals. However, the scheduling performance was not improved by the adaptive streaming technique.

A CPU-intensive job scheduling algorithm was reported by Sahoo and Dehury (2018) using a healthcare cloud. Nevertheless, similar to other applications, also an assigned virtual computing node may take a longer time to execute jobs. Mora et al. (2017) introduced a distributed framework using the IoT paradigm to study human biomedical signals, but the time complexity involved during big medical data scheduling was not reduced by the distributed framework.

A review of different deep learning techniques designed for IoT big data and streaming analytics was presented by Mohammadi et al. (2018). However, the overall runtime reduction of such selection was not significant. Although a real-time and energy-efficient resource scheduling and an optimization framework were introduced by Sun et al. (2015) for a big data stream, a low response time was not achieved.

Big Data Streaming Applications Scheduling was designed by Kanoun et al. (2016) employing staged multi-armed bandits to maximize the scheduling performance. Similarly, a fast future feature-aware online scheduling approach was developed by Sun and Tang (2017). However, both proposed methods failed to achieve sufficient scheduling performance.

Another study related to the analysis of large electronic health records was conducted by Wu et al. (2017) to achieve improved care efficiency. Moreover, an extensible big data architecture was designed by El Aboudi and Benhlima (2018) using stream computing and batch computing to increase the trustworthiness of healthcare systems. Yet, both of these studies did not consider storage complexity.

A big data-driven model was introduced by Koufi et al. (2015) for the optimization of healthcare processes. An optimization strategy was presented by He et al. (2016) to decrease the storage surplus and diminish the load to achieve highly efficient data processing operations. However, these two existing methods showed to be unable to achieve efficient scheduling performance.

The impact of big data in healthcare was resolved by Palanisamy and Thirunavukarasu (2019) for the handling of stream-based data for patient monitoring. However, the performance of storage complexity was not considered. While a policy enforcement framework was introduced by Sicari et al. (2017) to solve the safety and quality threats in heterogeneous smart healthcare environments, the scheduling process was not achieved at the required level.

In another application, predictive approaches in healthcare are used for the purpose of predicting patient no-shows (Harris et al., 2016). Although a model was proposed for big data applications, based on regression-like modelling in connection with function approximation using the sum of exponential functions to predict no-show probabilities, the storage complexity was not minimized.

Another approach for scheduling patients to hospitals, also considering no-shows, is discussed by Samorani and LaGanga (2015). The main goal of this approach is to maximize the number of patients while minimizing wait time and overtime. Unfortunately, the scheduling time was not reduced.

An example of a more complex healthcare scheduling problem is patient admission scheduling, as discussed by Bastos et al. (2019), whereby patients are scheduled to scarce resources ("beds") over a period of 28 to 91 days. While this issue is more complex than our problem regarding the resource scheduling aspect, it does not include any aspects of forecasting (such as disease prediction).

A general overview of using operational research techniques in healthcare (within Europe) is provided by Brailsford and Vissers (2011), which covers all nine identified phases of the healthcare services lifecycles. Almost 50% (204 of 411) of the considered

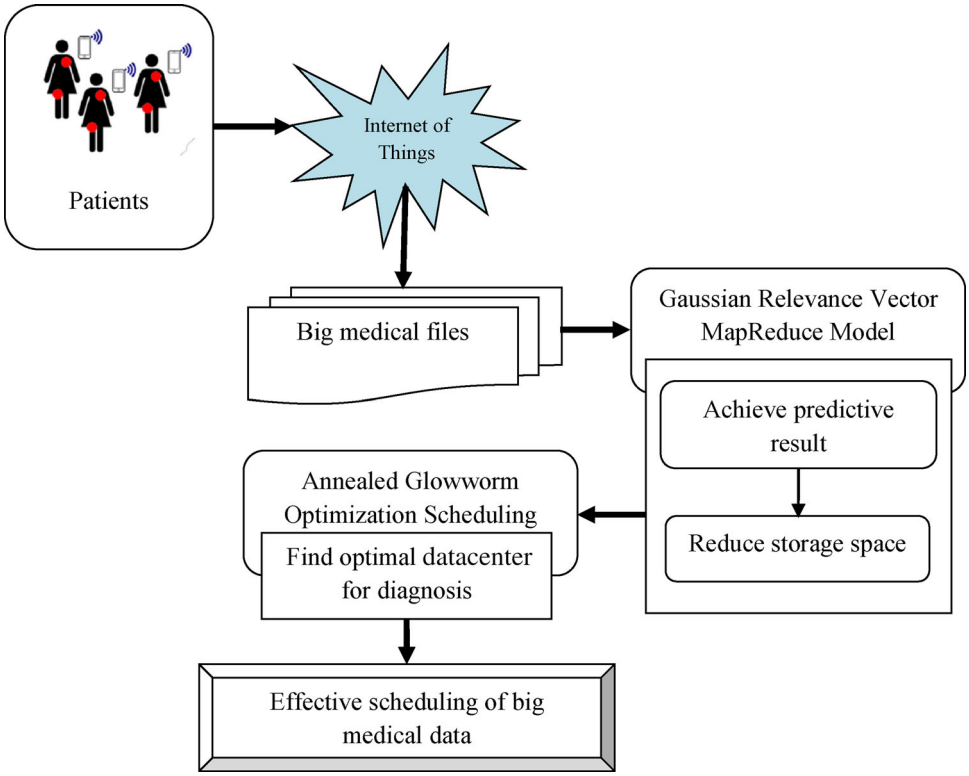

Figure 1. Architecture diagram of the GRVM-AGS technique for big medical data scheduling in a smart healthcare application using IoT.

research is related to units and hospitals and approaches related to performance management in service delivery dominate, which is also the focus of our research. However, only 7 of the 411 publications related to the unit/hospital level specifically focused on forecasting the demand of services (another focus of our study).

In addition, the paper by Ahmadi-Javid et al. (2017) discusses the increasing importance of outpatient appointment scheduling and related information systems. The work also provides a comprehensive survey of analytical and numerical optimization studies that present decision-support tools for designing and planning outpatient appointment systems (OASs), considering strategic, tactical, and operational levels. Yet, an effective patient medical data scheduling performance was not achieved.

To solve the aforementioned issues, the GRVM-AGS technique is proposed. The key contributions of GRVM-AGS are described below:

- The Gaussian Relevance Vector MapReduce (GRVM model) was developed using Relevance Vector Machine learning, the MapReduce framework, and a Gaussian kernel function, which is an extension of existing works. This model is introduced in the proposed GRVM-AGS to significantly minimize the storage space in processes of large medical data analytics compared to state-of-the-art works.
- The GRVM model is combined with Annealed Glowworm Optimization Scheduling (AGS) to enhance the scheduling performance of big medical data, as compared to state-of-the-art works.
- The AGO-MDS algorithm was designed using Glowworm optimization and annealed selection to identify the optimal datacenter for the diagnosis of patient medical data. The algorithm is applied in the GRVM-AGS technique to efficiently compensate for the workloads between multiple datacenters (i.e. physicians) in a smart healthcare environment (with a lower planning time) compared to conventional approaches.

3. Gaussian relevance vector MapReduce-based annealed Glowworm optimization scheduling

The GRVM-AGS technique is introduced with the aim of enhancing the scheduling performance of big medical data in a smart healthcare application. To diagnose and evaluate a patient, physicians need access to the patient's electronic medical files, which include huge multimedia data from ECGs, EEGs, X-rays, ultrasounds, CT scans, and MRI reports. Scheduling the patient's medical data files is required to balance the workloads of physicians in a smart healthcare environment. Therefore, we combined the Gaussian Relevance Vector MapReduce model and Annealed Glowworm Optimization Scheduling (GRVM-AGS) to improve this scheduling process.

Unlike existing techniques, the GRVM model is introduced in GRVM-AGS to obtain the predictive

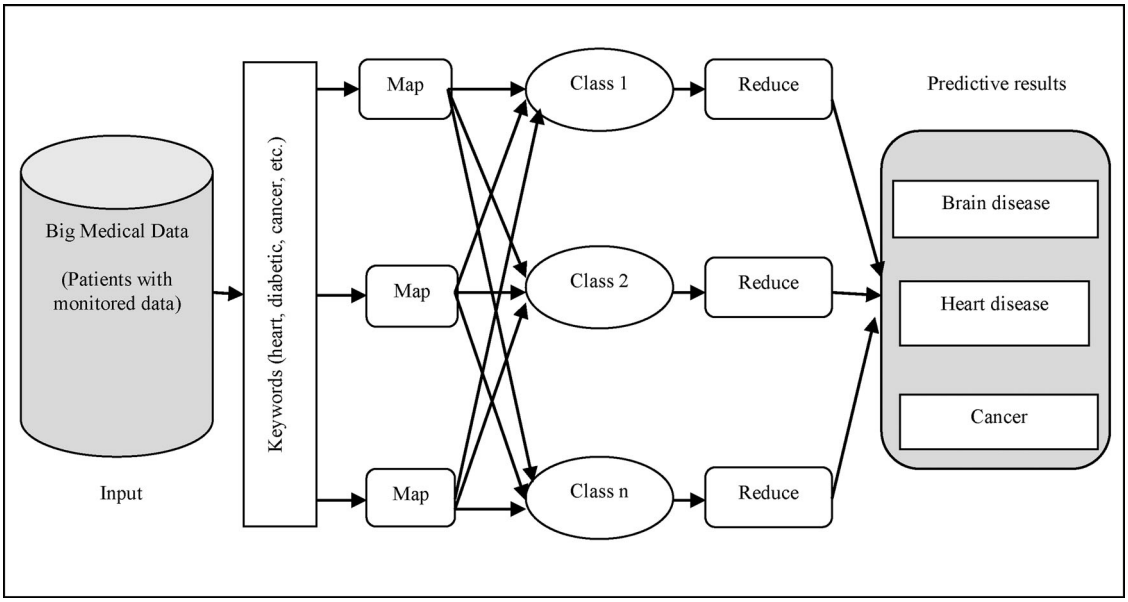

Figure 2. Processes of the GRVM model for predictive analytics and storage optimization.

analytic results of each patient's medical file while optimizing storage. During the predictive analytics process, GRVM-AGS seeks the disease class from the patient's medical files to schedule a task to a datacenter. Compared to existing state-of-the-art works, the number of relevance vectors in a GRVM model is smaller, thereby GRVM-AGS can perform the predictive analysis faster. In addition, a Gaussian kernel function applied in the GRVM model (within GRVM-AGS) maps only the related patient medical data to a corresponding class and eradicates irrelevant data. By removing unrelated information and selecting only significant medical features, GRVM-AGS uses a minimal amount of storage space for effective predictive analytics and, thus, reduces the storage complexity compared to state-of-the-art works. The overall architecture diagram of the GRVM-AGS technique is shown in Figure 1.

The Annealed Glowworm Optimization Scheduling (AGS) is employed in GRVM-AGS to find an optimized schedule. Unlike existing scheduling methods, AGS provides an optimal solution based on the medical files of patients with a lower scheduling time, allowing physicians to better assess the medical records and identify the best treatment and medication for the patient.

Figure 1 displays the overall processes of GRVM-AGS to achieve higher scheduling performance for large medical files. As shown in the figure, the GRVM model first predicts the disease class of the input medical files and optimizes the storage space during big medical data analytics. Then, Annealed Glowworm Optimization Scheduling (AGS) is applied to find and assign the optimal physicians for the identified diagnosis based on the medical file. The detailed process of the GRVM-AGS technique is described in the next subsections.

3.1. Gaussian relevance vector MapReduce for predictive analytics

The Gaussian Relevance Vector MapReduce (GRVM) model advantageously combines relevance vector machine learning (Fei, 2017), a MapReduce framework, and a Gaussian radial basis function. This combination can reduce the size of big medical data to effectively perform predictive analytics and achieve storage optimization. The GRVM model contains two main processes, namely Map and Reduce. First, a big medical dataset is processed by splitting the input dataset into independent chunks in the form of key/value pairs, which are then processed by the map tasks in a parallel manner. The biggest advantage of MapReduce (Khezr & Navimipour, 2017) is this parallel processing based on the division of work among multiple processor nodes. During the mapping process, the GRVM model uses a map function for each input key/value pair and generates output key/value pairs as intermediate results. During the reduction process, the GRVM model combines all interim results, groups the medical data of each patient according to keys and finally provides the predictive class result. Then, a Gaussian radial basis kernel function is employed to remove unrelated data by mapping the patient medical data into different disease classes, such as brain tumor, diabetes, and heart diseases. The MapReduce process in the GRVM model is depicted below.

Figure 2 provides the block diagram of the GRVM model for predictive analytics of patient medical data for workload scheduling. The relationship between the medical data of patients is categorized with patients' medical files associated with a disease. Let us consider a number of patient medical files

"$M_i = M_1, M_2, ..., M_n$" and a different number of disease classes, such as brain tumors, diabetes, heart disease, and cancer, represented as "$C_i = C_1, C_2, ..., C_n$" where each class contains a similar medical data related to a particular disease, denoted as "$C_i = \{M_1, M_2, .., M_n\}$." Here, "$n$" indicates the given number of patient medical data. In GRVM-AGS, big multimedia medical data (i.e. MRI cardiac images and MRI brain images) are considered as a medical file "M_i." The GRVM model captures and utilizes big multimedia medical data as input then extracts features and vital signs from these images. Using the features and vital signs, key "α_i"/value "β_i" pairs are created for the patients in relation to the disease values from various device inputs. The GRVM model then creates MapReduce classes for predicting the results. As an example, this process was applied for COVID-19 diagnosis data using CT (computed tomography) scans of patients' lungs as the input.

CT scans were obtained to detect COVID-19 and, if suspected, its severity and condition level. To evaluate the scans, the COVID-19 Reporting and Data System (CORADS) and CT severity index were used, which are described in detail below.

1) *CORADS*: This provides an assessment scheme to evaluate patients who are suspected of having COVID-19. Specifically, the chance of having the disease increases from Stages 1 to 6; this does not mean that the severity of the disease increases as the stage increases. The key/value pairs per the Stages 1–6 are as follows:

$\alpha_1, \beta_1 =$ CORADS 1: There is no suspicion.
$\alpha_2, \beta_2 =$ CORADS 2: There is slight suspicion, where the detected abnormalities in the CT scan are most likely not due to COVID-19.
$\alpha_3, \beta_3 =$ CORADS 3: There is mild suspicion, where abnormalities in the scan are consistent with COVID-19 and another disease.
$\alpha_4, \beta_4 =$ CORADS 4: There is high suspicion, where abnormalities are consistent with COVID-19.
$\alpha_5, \beta_5 =$ CORADS 5: There is a definite presence of COVID-19, where findings are consistent with other confirmed cases.
$\alpha_6, \beta_6 =$ CORADS 6: This stage is identified if a COVID-19 RT-PCR or rapid antigen test result is positive.

2) *CT Severity Index:* In the human body, the right lung has three lobes and the left has two, thus, totaling 5 lobes. The CT severity index is used to identify the extent of damage of the lungs. The score ranges from 0 to 25; that is, each lobe is assigned 0–5 points depending on the involvement of that lobe in a disease. Herein, we used the following key values associated with the severity index score:

$\alpha_1 =$ Score 1: Less than 5% involvement
$\alpha_2 =$ Score 2: 5–25% involvement
$\alpha_3 =$ Score 3: 26–49% involvement
$\alpha_4 =$ Score 4: 50–75% involvement
$\alpha_5 =$ Score 5: Greater than 75% involvement
After totalling the scores of each lobe, the severity of the disease, in this case COVID-19, can be determined:

$\beta_1 =$ Less than 8: Mild severity
$\beta_2 =$ 8–15: Moderate severity
$\beta_3 =$ More than 15: High severity

It is critical to acknowledge that these scores are consistent with condition of the disease on the day the CT scans were obtained, and that the severity is likely to change over time. Therefore, scans should be repeated from time to time to evaluate if the disease has progressed and to which stage.

Another important factor in diagnosing COVD-19 is SpO2, or the saturation of peripheral oxygen, which is measured by a pulse oximeter. An SpO2 less than 93% means that our lungs are not working properly. Therefore, the data obtained from CT images used as input in this study included the CORADS stage, CT severity index score, and SpO2. The features of lungs observed in the CT scan images also give insight to the severity and progression of the disease, including:

Ground glass appearance: common radiographic finding in COVID-19 lungs, indicates presence of the virus within 5 days of imaging.

Crazy-paving pattern: multiple areas of ground glass opacities, indicates progression of the virus between 5 and 10 days.

Consolidation: more extensive lung involvement in 10 to 13 days.

Gradual resolution: clearance of abnormalities in lungs after 14 days, indicates regression of the virus.

Using the above COVID-19 CT scan data as input, the GRVM model can be applied to create a mapping phase by considering disease features (keys/values) in the images or to predict the severity level based on the parameters (1). Specifically, the relevance vector machine learning creates patterns with the above values for diagnosing COVID-19 and the MapReduce framework with a Gaussian radial basis function to calculate CORADS and CT scan severity index. The Gaussian radial basis function is able to produce sparse solutions by linearly weighting a small number of features instead of using all of them, thus, requiring less time.

In the GRVM model, the mapping phase takes one pair of features and constructs a list of (key "α_i," value "β_i") pairs in different domains using the mapping below:

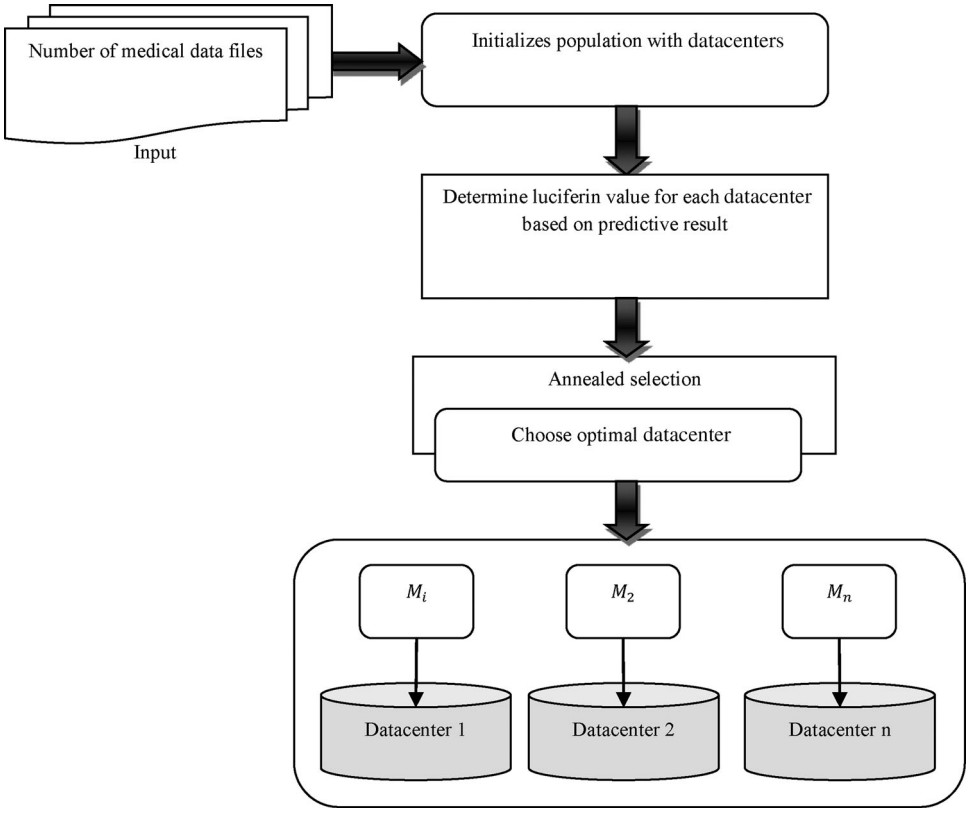

Figure 3. Process of AGO-MDS for medical data scheduling.

$$\mathrm{Map}(\alpha_1, \beta_1) \rightarrow List(\alpha_2, \beta_2) \quad (1)$$

Next, the GRVM model determines the probability of the desired output class for each patient medical data with the aid of a probability function using the following equation:

$$P\left(\frac{C_i}{d_i}\right) = \prod_{i=1}^{n} \sigma\{x(M_i;\delta)\}\left[1 - \sigma\{x(M_i;\delta)\}\right] \quad (2)$$

where "δ" refers to a weight vector; "$\sigma\{x(M_i;\delta)\}$" denotes a logistic sigmoid function; and "x" represents the different features of the medical data file "M_i." The logistic sigmoid function is mathematically specified as:

$$\sigma\{x(M_i;\delta)\} = \frac{1}{1 + e^{-x(M_i;\delta)}} \quad (3)$$

As mentioned, the GRVM model uses a Gaussian radial basis function as a kernel function, which maps significant medical features "M_i" of patients to the corresponding disease-related class "C_i." The mapping of patient medical data "M_i" using a Gaussian radial basis kernel function is mathematically described as:

$$K(C_i,\ M_i) = \sum_{i=0}^{n} \frac{1}{\delta_i} \varphi(C_i,\ M_i) \quad (4)$$

where "φ" represents the Gaussian kernel function; and "δ_i" denotes a weight value. The Gaussian kernel function "φ" in the GRVM model maps every medical feature of a patient in a corresponding class "C_i" for predictive analytics to schedule the workload among different physicians. Subsequently, the GRVM model gathers all pairs with similar keys from all lists, groups them together, and creates a class for each group. This process helps the GRVM model to remove unwanted data from large healthcare datasets and, thus, to achieve storage optimization during workload scheduling processes.

Algorithm 1 explains the step-by-step process of the GRVM model to attain the predictive analytic results of medical files through storage optimization. As demonstrated in the algorithm, the (key/value) pairs for a patient's medical features are first created. Then, the GRVM model estimates the probability of the desired output class (disease) for the medical features of each patient with the help of a probability function. Next, the GRVM model applies a Gaussian kernel function for mapping patient medical features into the appropriate class and, consequently, removes irrelevant features. The mapping of medical features into a corresponding disease class helps to obtain the predictive analytic result for balancing the workload of physicians in smart healthcare monitoring. Furthermore, the removal of unwanted features supports the GRVM model to optimize the storage complexity during big medical data analysis.

Algorithm 1. Gaussian Relevance Vector MapReduce Model

// **Gaussian Relevance Vector MapReduce Model**
Input: Patient medical files '$M_i = M_1, M_2, ., M_n$'
Output: Obtain predictive analytic results

Step 1: Begin
Step 2: **For** each medical data of patient 'M_i'
Step 3: Extract features
Step 4: Construct (key/value) pairs using (1)
Step 5: Compute the probability of the desired output class using (2)
Step 6: Apply the Gaussian kernel function
Step 7: Patient medical features are mapped into an appropriate class using (4)
Step 8: Reduce phase: remove the irrelevant medical features
Step 9: Generate predictive analytic result
Step 10: **End for**
Step 11: End

3.2. Annealed Glowworm optimization-based medical data scheduling

After the predicted results are obtained, the patients' medical files are scheduled to the corresponding datacenters (i.e. physicians) for diagnosis and treatment. In our proposed technique, the Annealed Glowworm Optimization-based Medical Data Scheduling (AGO-MDS) algorithm is combined with GRVM to balance the workloads among multiple datacenters in the smart healthcare environment. The AGO-MDS algorithm is based on combining Glowworm optimization and annealed selection to select the optimal datacenter and to increase the scheduling performance of workloads. As proposed by Krishnanand and Ghose (2009), AGO-MDS was motivated by the behavior of Glowworms (i.e. fireflies or lightning bugs), whereby a Glowworm with less lighting behavior is attracted to a brighter Glowworm. In other words, the AGO-MDS algorithm classifies the working set into the smallest possible two-element sets to evaluate the movement probabilities of each datacenter (or Glowworm). Subsequently, it updates the location of each datacenter to find the optimal datacenter (or brightest Glowworm) for scheduling a patient's medical data, according to Bentayeb et al. (2019).

As presented in Figure 3, AGO-MDS first initializes a population of datacenters (i.e. Glowworms), then each datacenter is associated with a luciferin value to be used by the objective function (i.e. predicted analysis result). Here, the objective is to select the datacenters that are best suited to a patient's file based on the results of predictive analytics. A datacenter (i.e. the physician) with a higher luciferin value indicates that it is more related to the predictive result of a considered patient. Therefore, AGO-MDS identifies the datacenter with a higher luciferin value in several datacenters in the smart healthcare environment and then applies annealed selection to schedule the patient's medical file to the most that datacenter.

Let us consider a number of datacenters "$DC_1, DC_2, ..., DC_n$" in the smart healthcare environment. Here, datacenters are considered to consist of various physicians, such as cardiologists, diabetologists, orthopaedists, and neurologists. In the AGO-MDS algorithm, the luciferin value is measured for each datacenter depending on the objective function (OF) using Equation (5):

$$\gamma_{DC_i} = \text{PAR} \tag{5}$$

where the luciferin value "γ_{DC_i}" of all datacenters in the smart healthcare environment is determined according to the obtained predictive analytics result (PAR) of a patient. The datacenter (i.e. Glowworm) with a small luciferin value is attracted to the datacenter with a higher luciferin value. When a Glowworm (datacenter) "DC_2" with a lower luciferin value moves toward another brighter Glowworm "DC_1" with a higher luciferin value, the moving probability is mathematically obtained as:

$$P = \frac{\gamma_{DC_2}(t) - \gamma_{DC_1}(t)}{\sum \gamma_{DC_{n_t}}(t) - \gamma_{DC_1}(t)} \tag{6}$$

where "P" is the movement probability of a Glowworm; "$\gamma_{DC_2}(t)$" is the luciferin value of the Glowworm "DC_2" at the time "t"; where "$\gamma_{DC_1}(t)$" is a luciferin value of the Glowworm "DC_1" at the time "t"; and "$\gamma_{DC_{n_t}}(t)$" represents the number of desired neighbors at time "t." The location of a Glowworm changes based on the movement probability. From this, the location of a Glowworm is updated as:

$$l_{DC_1\ (t+1)} = l_{DC_1(t)} + \ S\left(\frac{l_{DC_2}(t) - l_{DC_1}(t)}{d_{DC_2DC_1}}\right) \tag{7}$$

where "$l_{DC_1\ (t+1)}$" is an updated location of the Glowworm "DC_1" at time "$t+1$"; "$l_{DC_1(t)}$" is a location of Glowworm "DC_1" at time "t"; "$l_{DC_2}(t)$" denotes the location of Glowworm "DC_2" at time "t"; "$d_{DC_2DC_1}$" indicates the distance between the two Glowworms "DC_1" and "DC_2"; and "S" refers to step-size. As the luciferin value of a Glowworm changes according to its location, the value is updated using Equation (8):

$$\gamma_{(t+1)} = (1-\rho)*\gamma_{(t-1)} + \ \beta*OF \tag{8}$$

where "$\gamma_{(t+1)}$" refers to the updated luciferin value; "ρ" is a constant luciferin decay rate with $(0 < \rho < 1)$; "OF" represents the objective function; "$\gamma_{(t-1)}$" is a previous luciferin value of the

datacenter at a time "$t-1$"; and "β" is the change rate of the neighborhood range. After updating the luciferin value, the AGO-MDS algorithm applies annealed selection to identify a datacenter with a higher luciferin value for a better diagnosis of medical files, according to Equation (9):

$$Y = \arg\max \gamma_{DC_i} \tag{9}$$

where "arg max" refers to a datacenter with a maximum luciferin value among the considered datacenters; and "Y" denotes the final selection result of the AGO-MDS algorithm to schedule a medical file.

Algorithm 2 presents the step-by-step process of AGO-MDS to select the optimal datacenters based on the results of the predictive analytic disease for each input medical file "M_i." By applying a set of patient data for scheduling, AGO-MDS categorizes the data before distributing the task among the computation nodes, which are then updated according to the patient medical key/value pairs. The luciferin value of datacenters is then updated to reduce the time required for scheduling and improve scheduling performance. This allows AGO-MDS to schedule the workload for different physicians with minimal time for diagnosis and to provide treatment for patients who connect remotely in smart healthcare applications using IoT. As a result, the GRVM-AGS technique achieves an improved scheduling performance and reduces scheduling time for big medical data analysis compared to existing methods.

Algorithm 2. Annealed Glowworm Optimization-based Medical Data Scheduling

// Annealed Glowworm Optimization-based Medical Data Scheduling Algorithm
Input: Patient medical files "$M_i = M_1, M_2, ., M_n$," datacenters "$DC_1, DC_2, ..., DC_n$," luciferin value γ_{DC_i}, luciferin update value $\gamma_{(t+1)} l_{DC_1(t+1)}$, movement probability P, luciferin decay coefficient ρ, change rate of the neighborhood range β, step size of movement S.
Output: Schedule with improved efficiency

Step 1: Begin
Step 2: **For** each medical file "M_i"with predictive analytics result "PAR"
Step 3: Initialize the population using a number of datacenters "DC_i"
Step 4: Measure "γ_{DC_i}" using (5)
Step 5: Compute "P" using (6)
Step 6: Update "$l_{DC_1\ (t+1)}$" using (7)
Step 7: Update "$\gamma_{(t+1)}$"using (8)
Step 8: Apply annealed selection to find an optimal datacenter using (9)
Step 9: Schedule the medical file to the diagnosis
Step 10: **End For**
Step 11: End

AGO-MDS provides an efficient self-scheduling strategy of the patient data. Figure 4 shows the detailed self-scheduling process flow before and after applying the AGO-MDS. In Steps 2–5 of Algorithm 2, the computation node of the datacenter is first read, then each node is allocated and computed for further process allocation. In Steps 6–8, the computation node is updated based on the medical file category, either in the same workspace or in different workspaces, to optimize the datacenter medical file allocation. The process runs until the self-scheduling of the patient data with the correct category of medical files is optimized. The secondary MapReduce model helps to create patterns within the patient data frames to create the relationship between the medical data of patients and the associated disease. In the self-scheduling process model, similar categories of the medical files are moved into one workspace with less computation nodes, unlike other scheduling strategies that handle large data files. Figure 4 demonstrates the categorization of the medical files in several workspaces and computation nodes. Before applying AGO-MDS, the three medical files that do not follow any pattern are randomly allocated for computing, taking up more space in computing node 2. After applying AGO-MDS, it is apparent that the allocated of medical files is better balanced and requires less effort for computation. This demonstrates that the proposed GRVM-AGS technique can achieve efficient scheduling using less space than other models.

4. Experimental settings

The GRVM-AGS technique was implemented in Java to evaluate its performance. In the experiment, GRVM-AGS collected medical multimedia data (i.e. MRI images and vital signs) from two datasets, namely the Cardiac MRI dataset (Andreopoulos & Tsotsos, 2008) and the Atlas brain database (Johnson & Becker, 2019). The Cardiac MRI dataset was constructed from 33 subjects, where each subject's sequence includes 20 frames and 8–15 slices along the long axis to make a total of 7980 images. The Atlas brain database is comprised of the MRI images of remotely monitored patients affected by cerebrovascular diseases, neoplastic diseases, and degenerative diseases. The GRVM-AGS technique considers the various medical files to accomplish the experimental process. During the scheduling process, GRVM-AGS distributed medical files to the optimal datacenter (i.e. physicians) to balance the workload in the smart healthcare application. For example, heart disease medical information was scheduled to a cardiologist and brain tumor disease medical files to a neurologist for diagnosis. The

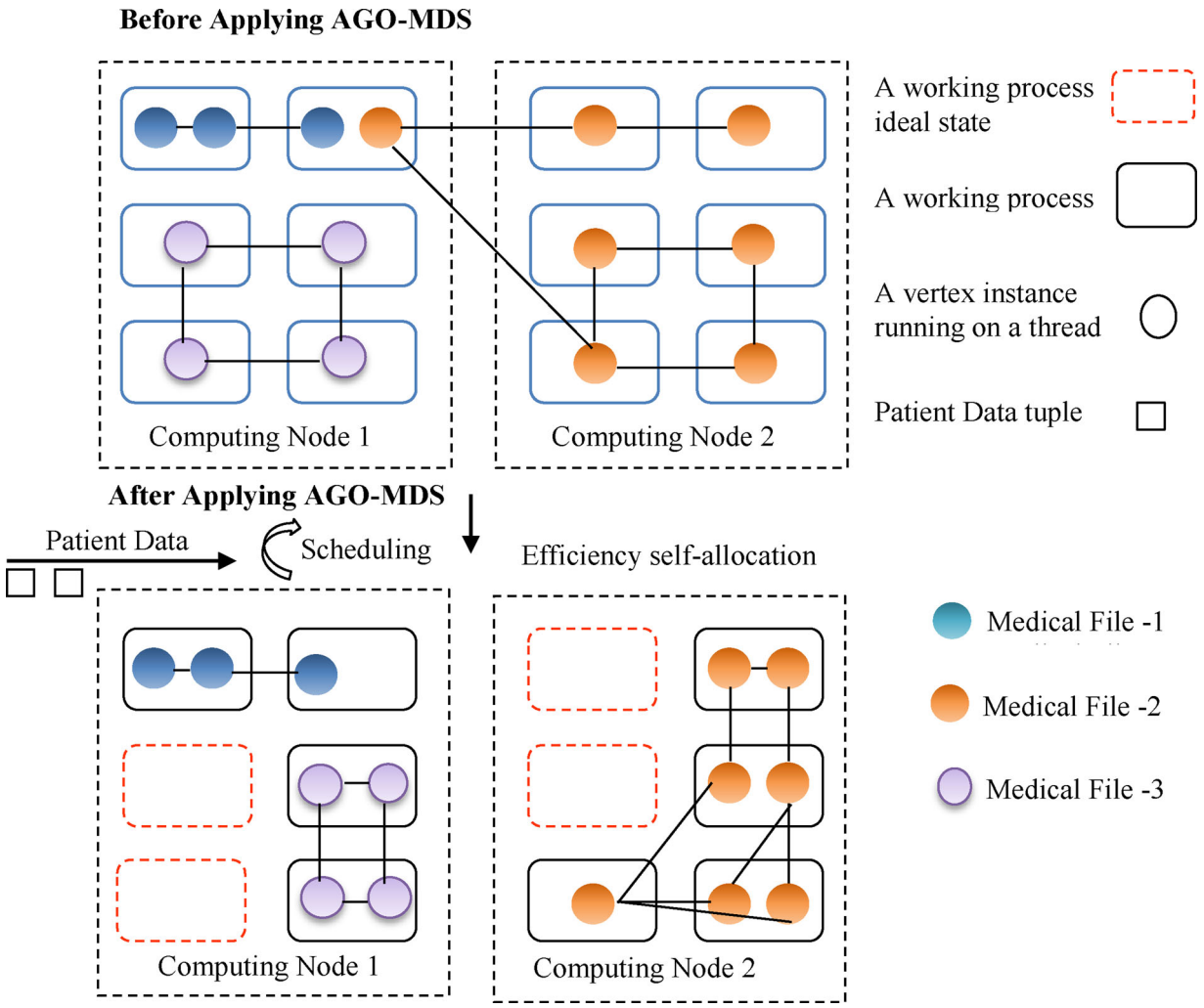

Figure 4. GRVM-AGS efficient self-scheduling strategy process flow.

Table 1. Parameter values of the GRVM-AGS technique in the experiment.

Parameter	ρ	γ_{DC_i}	β	n_t	S
Value	0.4	0.6	0.08	5	0.03

number of datacenters for the experiments was set to 10. (This number was not provided in the mentioned datasets but was assumed to provide a realistic setting.)

The performance of GRVM-AGS technique was compared to that of two existing approaches, namely a hybrid model (Elhoseny et al., 2018) and a task-level adaptive MapReduce framework (Zhang et al., 2015), using storage complexity, scheduling performance, and scheduling time as indicators.

In Table 1, the simulation settings of the parameters in the AGO-MDS algorithm, such as luciferin decay coefficient ρ, luciferin valueγ_{DC_i}, change rate of the neighborhood range β, number of desired neighbors n_t, and step size S were all fixed. The specified parameter values have been chosen based on some experiments with different values as these were the settings leading to the best results.

5. Results and discussions

In this section, we present the results of GRVM-AGS for scheduling workloads efficiently among multiple datacenters based on the luciferin value in the smart healthcare environment. The performance results of GRVM-AGS are compared with those of the hybrid model (Elhoseny et al., 2018) and the task-level adaptive MapReduce framework (Zhang et al., 2015) based on the metrics discussed below. The analysis results of GRVM-AGS are presented in tables and graphs.

5.1. Storage complexity

Storage complexity "SC" measures the amount of memory space the datacenter needs to store the medical files of patients. The storage complexity is computed as follows:

$$\mathrm{SC} = n * \mathrm{Space}\ (\mathrm{SSP}) \tag{10}$$

where "n" represents the number of patients' medical files being considered for the experiments; and "Space (SSP)" denotes the memory being utilized to store an average patient's medical file. The storage

complexity is measured in terms of megabytes (MB).

A sample calculation for storage complexity using the proposed GRVM-AGS technique and comparative methods is presented below. For each method, assume that there are 25 medical files ($n = 25$).

Hybrid Model: Requires 1.44 (MB) to store a patient's medical file. Thus, the storage complexity (SC) is measured as:

$$\mathrm{SC} = 25 * 1.44 = 36 \ (\mathrm{MB})$$

Task-level Adaptive MapReduce Framework: Requires 1.32 (MB) to store a patient's medical file. Thus, SC is measured as:

$$\mathrm{SC} = 25 * 1.32 = 33 \ (\mathrm{MB})$$

GRVM-AGS Technique: Requires 1.12 (MB) to store a patient's medical file. Thus, SC is measured as:

$$\mathrm{SC} = 25 * 1.12 = 28 \ (\mathrm{MB})$$

To estimate the space complexity of big medical data scheduling and analysis, GRVM-AGS was implemented in Java using two medical datasets with a range of 25–250 medical files. With 150 medical files, GRVM-AGS achieved a storage complexity of 65 MB, whereas the hybrid model presented by Elhoseny et al. (2018) and the task-level adaptive MapReduce framework by Zhang et al. (2015) achieved storage capacities of 89 and 80 MB, respectively. From the results, it is obvious that GRVM-AGS can realize a lower storage complexity for the effective analysis of big medical data compared to previous methods. The complete performance results of storage complexity are provided in Table 2.

The superior performance of GRVM-AGS can be attributed to the implemented GRVM model, which measures the probability of the desired output class for all medical data via the likelihood function. Afterward, GRVM-AGS maps the medical data into a respective class and then removes unrelated data. The elimination of unnecessary information supports GRVM-AGS to achieve a lower storage complexity in big medical data analysis. Therefore, GRVM-AGS allows for a higher reduction of the amount of memory space the datacenter needs to store the patients' medical files compared to the other methods. Specifically, GRVM-AGS reduced the storage complexity by 19% and 12% compared to the existing hybrid model (Elhoseny et al., 2018) and the task-level adaptive MapReduce framework (Zhang et al., 2015), respectively.

5.2. Scheduling performance

Scheduling performance "(SP)" is the ratio of the number of medical files that are correctly scheduled to the optimal datacenter to the total number of medical data. The formula for scheduling performance is depicted as follows:

$$\mathrm{SP} = \frac{\mathrm{NCS}}{n} * 100 \qquad (11)$$

where "n" denotes the number of medical files; and "NCS" represents the number of medical files that are accurately scheduled. The scheduling performance is measured in percentage (%).

To determine the scheduling performance of big medical data, GRVM-AGS was implemented in Java with a varying number of medical data files (25–250) from two medical datasets. With 175 medical files, GRVM-AGS achieved a scheduling performance of 94%, while the hybrid model (Elhoseny et al., 2018) and the task-level adaptive MapReduce framework (Zhang et al., 2015) achieved performances of 81% and 83%, respectively. The results clearly demonstrate that GRVM-AGS can realize better scheduling performance for balancing workloads in smart healthcare applications compared to other existing works. The complete results of scheduling performance are illustrated in Table 3.

Due to the application of annealed Glowworm optimization scheduling, GRVM-AGS was able to identify the optimal physicians to analyze the medical files. Specifically, GRVM-AGS achieved a higher ratio of correctly scheduled medical files and enhanced the efficiency of medical data scheduling by 16% and 11% compared to the hybrid model (Elhoseny et al., 2018) and the task-level adaptive MapReduce framework (Zhang et al., 2015), respectively.

5.3. Scheduling time

Scheduling time "(ST)" determines the amount of time required to schedule medical files to the optimal datacenter. The scheduling time is calculated as follows:

$$\mathrm{ST} = n * \mathrm{time} \ (\mathrm{SSM}) \qquad (12)$$

where "n" is the number of medical files; and "time (SSM)" denotes the average time utilized for scheduling a single medical file to an optimal datacenter. The scheduling time is measured in milliseconds (ms).

The GRVM-AGS technique was implemented in Java with 25–250 medical data files from two medical datasets to measure the amount of time required to schedule medical data. With 200 medical files, GRVM-AGS required 52 ms scheduling time, while the conventional hybrid model and the task-level adaptive MapReduce framework required 62 and 56 ms, respectively. This finding confirms the superior performance of GRVM-AGS for the effective analysis of big medical data. All results for scheduling times are shown in Table 4.

Table 2. Results of storage complexity.

	Storage complexity (MB)		
Number of medical files (*n*)	Hybrid model	Task-level adaptive MapReduce framework	GRVM-AGS technique
25	36	33	28
50	46	40	35
75	59	56	49
100	74	70	60
125	86	81	69
150	89	80	65
175	98	88	79
200	94	90	82
225	99	95	86
250	105	100	93

Table 3. Results of scheduling performance.

	Scheduling performance (%)		
Number of medical files	Hybrid model	Task-level adaptive MapReduce framework	GRVM-AGS technique
25	68	76	92
50	74	80	90
75	80	83	93
100	82	85	94
125	82	84	91
150	85	87	95
175	81	83	94
200	83	85	92
225	84	88	93
250	85	89	95

Table 4. Results of scheduling times.

	Scheduling time (ms)		
Number of medical files (*n*)	Hybrid model	Task-level adaptive MapReduce framework	GRVM-AGS technique
25	33	25	20
50	40	33	30
75	45	41	38
100	55	48	45
125	56	49	44
150	59	53	48
175	58	54	49
200	62	56	52
225	63	59	54
250	70	65	58

With the help of annealed Glowworm optimization scheduling (AGS), GRVM-AGS initializes a population with a number of datacenters, and then, estimates the luciferin values depending on the objective function. Using these computed luciferin values and the objective function of AGS, GRVM-AGS can identify datacenters that fit well with the predictive results of each medical file. After that, GRVM-AGS schedules the medical files to the identified optimal datacenter with minimal time for health status analyses. In this process, GRVM-AGS requires 20% and 10% less time to schedule the medical data to the optimal datacenter compared to the hybrid model (Elhoseny et al., 2018) and the task-level adaptive MapReduce framework (Zhang et al., 2015), respectively.

6. Conclusion

An effective technique combining the Gaussian Relevance Vector MapReduce (GRVM) model and Annealed Glowworm Optimization Scheduling (AGS) has been designed with the aim of increasing the scheduling performance of large medical data files. Results show that the proposed GRVM-AGS technique significantly improves the scheduling performances of big medical data analytics, effectively balancing the workloads among diverse datacenters in a smart healthcare environment with reduced time and complexity. In scheduling the big medical data, a file is assigned to optimal datacenters, where a physician then analyses the medical information and determines the correct treatments and medication for patients remotely connected through IoT. The effectiveness of GRVM-AGS was evaluated in terms of storage complexity, scheduling efficiency, and scheduling time. Compared with two state-of-the-art methods, namely a hybrid model and task-level adaptive MapReduce framework, the results show that GRVM-AGS leads to better performances for big medical data diagnostics with enhanced scheduling performance and a reduced storage complexity. In addition, the application of the GRVM model is presented for COVID-19 diagnosis as an example, utilizing CT scans as the data input and MapReduce and Gaussian radial basis function to calculate features of the evaluation methods (i.e. CORADS and CT scan severity index).

For future research, it would be insightful to compare the suggested technique with other approaches. In addition, the proposed GRVM-AGS could be used for different scheduling problems, such as scheduling tasks in other service industries.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Amir H. Gandomi http://orcid.org/0000-0002-2798-0104
Thomas Hanne http://orcid.org/0000-0002-5636-1660

References

Ahmadi-Javid, A., Jalali, Z., & Klassen, K. J. (2017). Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research*, *258*(1), 3–34. https://doi.org/10.1016/j.ejor.2016.06.064

Andreopoulos, A., & Tsotsos, J. K. (2008). Efficient and generalizable statistical models of shape and appearance for analysis of cardiac MRI. *Medical Image Analysis*, *12*(3), 335–357. https://doi.org/10.1016/j.media.2007.12.003

Bastos, L. S., Marchesi, J. F., Hamacher, S., & Fleck, J. L. (2019). A mixed integer programming approach to the patient admission scheduling problem. *European Journal of Operational Research*, *273*(3), 831–840. https://doi.org/10.1016/j.ejor.2018.09.003

Bentayeb, D., Lahrichi, N., & Rousseau, L.-M. (2019). Patient scheduling based on a service-time prediction model: a data-driven study for a radiotherapy center. *Health Care Management Science 22*(4), 768–782.

Brailsford, S., & Vissers, J. (2011). OR in healthcare: A European perspective. *European Journal of Operational Research*, *212*(2), 223–234. https://doi.org/10.1016/j.ejor.2010.10.026

Chakravarthy, S. K., Sudhakar, N., Reddy, E. S., Subramanian, D. V., & Shankar, P. (2019). Dimension reduction and storage optimization techniques for distributed and big data cluster environment. In *Soft computing and medical bioinformatics* (pp. 47–54). Springer. https://doi.org/10.1007/978-981-13-0059-2_6

El Aboudi, N., & Benhlima, L. (2018). Big data management for healthcare systems: Architecture, requirements, and implementation. *Advances in Bioinformatics*, *2018*, 4059010–4059018. https://doi.org/10.1155/2018/4059018

Elhoseny, M., Abdelaziz, A., Salama, A. S., Riad, A. M., Muhammad, K., & Sangaiah, A. K. (2018). A hybrid model of internet of things and cloud computing to manage big data in health services applications. *Future Generation Computer Systems*, *86*, 1383–1394. https://doi.org/10.1016/j.future.2018.03.005

Fei, S. W. (2017). Fault diagnosis of bearing based on relevance vector machine classifier with improved binary bat algorithm for feature selection and parameter optimization. *Advances in Mechanical Engineering*, *9*(1), 168781401668528–168781401668529. https://doi.org/10.1177/1687814016685294

Harris, S. L., May, J. H., & Vargas, L. G. (2016). Predictive analytics model for healthcare planning and scheduling. *European Journal of Operational Research*, *253*(1), 121–131. https://doi.org/10.1016/j.ejor.2016.02.017

He, H., Du, Z., Zhang, W., & Chen, A. (2016). Optimization strategy of Hadoop small file storage for big data in healthcare. *The Journal of Supercomputing*, *72*(10), 3696–3707. https://doi.org/10.1007/s11227-015-1462-4

Johnson, K. A., Becker, J. A. (2019). Atlas brain database. http://www.med.harvard.edu/aanlib/home.html

Kanoun, K., Tekin, C., Atienza, D., & Van Der Schaar, M. (2016). Big-data streaming applications scheduling based on staged multi-armed bandits. *IEEE Transactions on Computers*, *65*(12), 3591–3605. https://doi.org/10.1109/TC.2016.2550454

Khezr, S. N., & Navimipour, N. J. (2017). MapReduce and its applications, challenges, and architecture: A comprehensive review and directions for future research. *Journal of Grid Computing*, *15*(3), 295–321. https://doi.org/10.1007/s10723-017-9408-0

Koufi, V., Malamateniou, F., & Vassilacopoulos, G., (2015). A big data-driven model for the optimization of healthcare processes. In R. Cornet (Ed.), *Digital healthcare empowering Europeans* (pp. 697–701). IOS Press. https://doi.org/10.3233/978-1-61499-512-8-697

Krishnanand, K. N., & Ghose, D. (2009). Glowworm swarm optimisation: A new method for optimising multi-modal functions. *International Journal of Computational Intelligence Studies*, *1*(1), 93–119. https://doi.org/10.1504/IJCISTUDIES.2009.515637 https://doi.org/10.1504/IJCISTUDIES.2009.025340

Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, *20*(4), 2923–2960. https://doi.org/10.1109/COMST.2018.2844341

Mora, H., Gil, D., Terol, R. M., Azorín, J., & Szymanski, J. (2017). An IoT-based computational framework for healthcare monitoring in mobile environments. *Sensors*, *17*(10), 2302–2327. https://doi.org/10.3390/s17102302

Pagán, J., Zapater, M., & Ayala, J. L. (2018). Power transmission and workload balancing policies in eHealth mobile cloud computing scenarios. *Future Generation Computer Systems*, *78*, 587–601. https://doi.org/10.1016/j.future.2017.02.015

Palanisamy, V., & Thirunavukarasu, R. (2019). Implications of big data analytics in developing healthcare frameworks–A review. *Journal of King Saud University - Computer and Information Sciences*, *31*(4), 411–415. https://doi.org/10.1016/j.jksuci.2017.12.007

Sahoo, P. K., & Dehury, C. K. (2018). Efficient data and CPU-intensive job scheduling algorithms for healthcare cloud. *Computers & Electrical Engineering*, *68*, 119–139. https://doi.org/10.1016/j.compeleceng.2018.04.001

Samorani, M., & LaGanga, L. R. (2015). Outpatient appointment scheduling given individual day-dependent no-show predictions. *European Journal of Operational Research*, *240*(1), 245–257. https://doi.org/10.1016/j.ejor.2014.06.034

Sicari, S., Rizzardi, A., Grieco, L. A., Piro, G., & Coen-Porisini, A. (2017). A policy enforcement framework for Internet of Things applications in the smart health. *Smart Health*, *3–4*, 39–74. https://doi.org/10.1016/j.smhl.2017.06.001

Sun, D., & Tang, H. (2017). Fast-FFA: A fast online scheduling approach for big data stream computing with future features-aware. *International Journal of Bio-Inspired Computation*, *10*(3), 205–217. https://doi.org/10.1504/IJBIC.2017.086717

Sun, D., Zhang, G., Yang, S., Zheng, W., Khan, S. U., & Li, K. (2015). Re-Stream: Real-time and energy-efficient resource scheduling in big data stream computing environments. *Information Sciences*, *319*, 92–112. https://doi.org/10.1016/j.ins.2015.03.027

ur Rehman, M. H., Chang, V., Batool, A., & Wah, T. Y. (2016). Big data reduction framework for value creation in sustainable enterprises. *International Journal of Information Management*, *36*(6), 917–928. https://doi.org/10.1016/j.ijinfomgt.2016.05.013

Wu, P. Y., Cheng, C. W., Kaddi, C. D., Venugopalan, J., Hoffman, R., & Wang, M. D. (2017). Omic and electronic health record big data analytics for precision medicine. *IEEE Transactions on Bio-Medical Engineering*, *64*(2), 263–273. https://doi.org/10.1109/TBME.2016.2573285

Xu, B., Da Xu, L., Cai, H., Xie, C., Hu, J., & Bu, F. (2014). Ubiquitous data accessing method in IoT-based information system for emergency medical services. *IEEE Transactions on Industrial Informatics*, *10*(2), 1578–1586. https://doi.org/10.1109/TII.2014.2306382

Xu, J., Ota, K., & Dong, M. (2018). Real-time awareness scheduling for multimedia big data oriented in-memory computing. *IEEE Internet of Things Journal*, *5*(5), 3464–3473. https://doi.org/10.1109/JIOT.2018.2802913

Zhang, F., Cao, J., Khan, S. U., Li, K., & Hwang, K. (2015). A task-level adaptive MapReduce framework for real-time streaming data in healthcare applications. *Future Generation Computer Systems*, *43–44*, 149–160. https://doi.org/10.1016/j.future.2014.06.009