IEEE Open Journal of the Computer Society

Received 3 December 2020; accepted 1 January 2021. Date of publication 12 January 2021;
date of current version 11 February 2021. The review of this article was arranged by Associate Editor Peng Li.

Digital Object Identifier 10.1109/OJCS.2021.3050998

Real-Time Cost Minimization of Fog Computing in Mobile-Base-Station Networked Disaster Areas

MICHAEL CONRAD MEYER (Member, IEEE), YU WANG (Member, IEEE), AND TAKAHIRO WATANABE (Life Member, IEEE)

Waseda University Graduate School of Information, Production, and Systems, Kitakyushu 808-0135, Japan

CORRESPONDING AUTHOR: MICHAEL MEYER (e-mail: meyer@aoni.waseda.jp)

This work was supported in part by the Waseda University Special Research Project Grant 2019C-582, and in part by the JSPS KAKENHI Grant Number 18K11226.

ABSTRACT The use of big data has led to many technologies that were previously thought to be impossible. We are now able to analyze the spread of a disaster automatically through the use of social networking analysis, which is effectively served by internet or cloud services. One problem with using such algorithms in these cases is that internet services and connections to the cloud can often be damaged. In order to combat this issue, mobile base stations can be deployed, allowing for an emergency internet network to be used until the landlines can be repaired. These emergency networks have limitations in speed and cost, but seem to be the most promising technology for the future. Forwarding all of the data through the network results in the lowest cost but yields a large amount of data overflow, forcing the system to cache data, thus increasing the delay. Fully processing data in the edge resources results in a higher cost. A genetic algorithm was used to find the ideal balance between processing and sending the data, which allowed for the most data to be transmitted without causing data overflow. Results show that the proposed algorithm closely matched the results of the genetic algorithm, while being executable with minimal clock cycles.

INDEX TERMS Cost function, data flow computing, edge computing, emergency services, mobile nodes, networks, optimization, real-time systems.

I. INTRODUCTION

With over 3 billion smartphone users existing in the world today, and the number is increasing rapidly. Users often upload posts to social media, which is used for updating other people about their lives. This action being performed by this many users creates an ocean of data, which can be properly processed and filtered by an algorithm.

The data collected from mobile phones can be images, videos, text (i.e. Twitter messages), etc. Data can be and is used in many ways, such as estimating the demand for a product, which is useful for estimating pricing, but it can also be used to help people. One way to use this data is spatial big data analytics, which is the category of big data pertaining to locations and mapping. Many authors have proposed different uses for spatial big data analytics [1], including: spatial prediction for understanding the situation in the areas that are not covered by the network; spatial outlier detection to find abnormal situations; spatial co-location feature detection and spatial clustering. In today's world of common global disasters, spatial big data analysis, which can benefit victims, increases in need and importance. Kwan *et al.* [2] explored an emergency response service that could detect obstructions caused by a major disaster, such as hurricanes, by using LiDAR data.

One way that big data is commonly processed is via cloud computing. Cloud computing was originally presented by Rochwerger *et al.* [3], when they deployed private and hybrid clouds, and created the federation of clouds. The federation was a joint venture between NASA and IBM, which became an on-demand processing service. For a predetermined

amount of processing, there was a predetermined fee. This led to the first cost-based processing model. Eventually, cloud computing became more popular and was adopted by many for their processing needs, but for some consumers lower delays were required.

More recently, AI-based applications have moved processing to the cloud to reduce hardware requirements [4]. By moving the processing to the cloud, smaller low-power devices are capable of features that would normally only be afforded to more powerful computers. One feature that many manufacturers take use in is artificial intelligence and machine learning. Devices such as small cellular phones can offload their AI processing to the cloud, and receive a result from the AI algorithm. This reduces the performance requirement of the phone, which can improve battery life, all while still managing to perform complex AI calculations.

To improve the performance of cloud computing systems, devices were placed closer to users, which had computational and communication resources. These devices acted as support for the cloud for tasks which required lower latency. The new system, called fog computing, was introduced by Cisco [5]. One reason not everyone utilizes these resources is because their cost can be greater than processing in the cloud.

Original models for these systems accounted for blocks of data, because the original cost model was based on a fixed amount of data to process, but as their popularity grew, the resources never stopped being used, and so the model of data through the system was more like a stream, sometimes referred to as data flow [6]. This is because it is rare for users to collectively tweet at one moment, and then have no users tweet immediately after. One problem that arises with a data flow model is that the amount of data flowing into one part may be greater than the amount of data that can flow out from that point. Storing this data in a cache and creating a queue can curb the maximum flow times, and offload the overflow to times where the flow in is lower. This data overflow has several negative effects on the system, such as energy costs and delay. The systems perform much better if we can avoid overflow as much as possible.

There are many kinds of natural disasters that can affect human lives. The two most famous ones are earthquakes and hurricanes because of the damages they can cause to both infrastructure and humans. Other natural disasters such as the spread of pandemics focus mostly on the human cost, and are optimal targets for tracking via the use of big data, but ones that damage the infrastructure greatly mitigate the ability for data to be collected in that area. This lack of data coverage in the disaster area decreases the efficacy of disaster-response big data algorithms.

In order to address this issue, a few companies have devised methods of distributing an emergency communication network (ECN). These ECNs are typically created by deploying several Mobile Base Stations (MBSs) which can connect to each other wirelessly, to give mobile coverage in the damaged areas. One example of an MBS for ECNs is the Movable and Deployable Resource Unit (MDRU) by NTT [7]. These typically have a smaller coverage area per unit, lower transmission capabilities and less performance overall, but are readily deployable within hours, when a typical base station can take several days to repair.

In this paper, we study cost minimization for big-data processing in wirelessly networked disaster areas. We target a scenario where MDRUs were deployed after a disaster to create an emergency communication network. The MDRUs collect data from nearby smartphones, and then transmit the data via an MDRU chain until they reach an MDRU which is connected with an undamaged communication line. The data coming from the smartphones to the MDRUs as well as the data from one MDRU to another is modeled as a data stream. Each MBS can decide what percentage of data to process locally. The data resulting from the processing as well as the data from the unprocessed portion is transmitted to the next MBS in the chain. Any data that remains unprocessed when it reaches the MBS connected to the hardline will be processed in the cloud. The rest of the paper is organized as follows: Section II discusses works related to this paper; Section III goes over the system models; Section IV details our proposed real-time solution; section V evaluates the proposal; Section VI wraps up the paper with some conclusions and possible future works to extend the paper.

II. RELATED WORK

Other authors, such as the ones in [8] have proposed the use of mobile ad-hoc networks (MANETs) in emergencies. These do not utilize any central base station to handle the data. These would require a complete overhaul of the network, and would need to interface with the existing hard-wire connections for server access. These may be suitable in the future when every car has been manufactured with this technology, but would not suffice if they needed to be deployed immediately for a disaster, where a fleet of MDRUs could be deployed, and the end users may not even notice a difference in use other than performance. These systems lead to stochastic networks which continuously change their connections, while MBS-based networks generally keep their structure at least for several minutes.

Cloudlets [9] provided a possible solution for offloading tasks from the cloud to the devices near users by deploying servers near the users. These behave similarly to MDRUs, but cloudlets have stronger processing and communication capabilities due to being permanent infrastructure. This means that our work can be considered to be an extension of Cloudlet-system optimization. Authors in [10] optimized the delay of a cloudlet system via VM migration with Transmission Power Control.

Task offloading and scheduling have been researched to great extents for typical fog computing networks [11]–[13]. One problem with using these researches for Wirelessly Networked Disaster-areas (WNDAs) is that a large technological gap exists between the capabilities of typical fog systems and ECNs which rely on the wireless antennas and mobile computing powers of MBSs. The authors in [11] proposed a

fog-computing-based medical system, and formalized the task distribution and VM placement problems as mixed-integer non-linear problems (MINLP). Once the problem was linearized, it was solved in a simpler manner. The authors in [14] worked on a task offloading algorithm for wearable devices, mobile terminals, and a cloud center. This paper is parallels the aforementioned one because of the inherent similarities of mobile cloud computing, and MBS-based fog networks. Other works attempted to solve the offloading problem using game theory. One example of such works is [15].

The authors in [16] minimized the cost of a fog computing network. They commenced by formulating the resource allocation and computation offloading for load reduction on the cloud center for fog networks. This is then solved using a deep neural network. The results were promising, but could also lead to intense computing costs for the algorithm itself.

The authors in [17] minimized the cost of a fog system by determining the ideal deployment. They targeted a heterogeneous network of both user-based networks and cloudlets. The problem was formulated in an integer linear programming form, and a low-complexity heuristic algorithm was applied to find a result.

The above-referenced works primarily focus on binary decisions, which result in maximum processing or minimum processing in the fog nodes. A more realistic solution should consider partial processing in the fog and cloud. These types of solutions are used as comparisons in the evaluation section. The previous works also model the data as blocks, even though social media data is constantly flowing with new posts every minute.

The authors of [18] presented a genetic algorithm (GA) to minimize the delay of a Wirelessly Networked Disaster Area, by using varying processing levels at each node. We differentiate our work in three ways: (1) this paper's system model is based on streaming data; (2) this paper's algorithm attempts to minimize the cost of the system while preventing overflow, which can cause large amounts of delay; and (3) this paper's algorithm is targeted at a real-time solution, which can be run on MDRUs in the real-world.

In our previous work [19], we targeted a similar problem and solved it through the use of a GA. The GA yielded very good results, but took several minutes for a small system with very few MDRU nodes, and multiple hours for a large system consisting of hundreds of nodes. The data flow rate often changes after several minutes, so even for the small system, the GA would yield non-ideal results due to the delay from calculating. This work expands on our previous work by creating an algorithm that can run in real-time, which allows for it to be implemented in the real world.

III. SYSTEM MODEL AND PROBLEM DEFINITION

A. SYSTEM MODEL

Fig. 1 represents a system model for an Wirelessly Networked Disaster Area(WNDA). After a disaster, movable base stations (MBSs) can be deployed to the disaster-stricken areas, to reconstruct a communication network. Users can upload data through the MBSs.

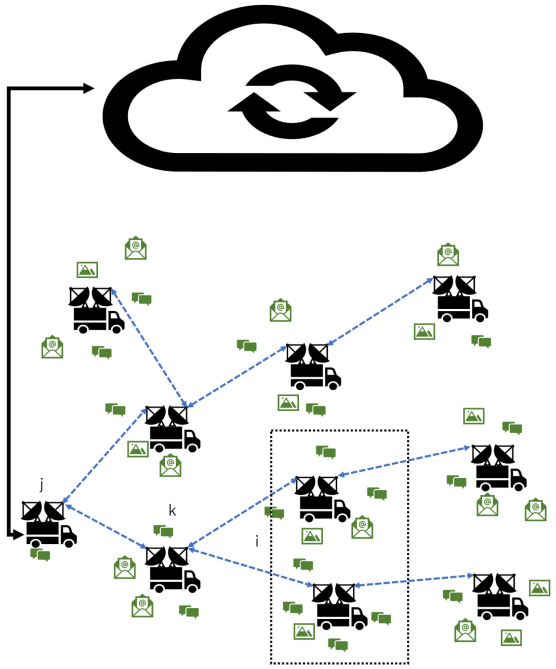


FIGURE 1. System Model. k is the current node, with child nodes in i, and the final destination node j that has a hard wire connection to the cloud.

We used k to denote an MBS, which integrates both wireless communication and computation functions. First, smartphone users connect with k and upload data with spatial information, where the data stream is represented as S_k. Then we assumed a part of the Real-time Continuous Data Cost-minimization Algorithm has been deployed in the MBS beforehand, and that the data size compression ratio after processing in node k is β, where $0 < \beta \leq 1$. The processing rate in each MBS k is represented as μ_k.

We also assume that the MBSs are connected via a wireless medium, and that the communication rate is represented by $\mathcal{R}$. We assume all MBSs send data into a special node j, which is located at the edge of the disaster area and can transmit data to the cloud quickly with a wired connection. To simplify the discussions, we consider a single j node case, but a multiple j node case can be modeled and solved similarly. Generally speaking, data analysis in cloud centers will not start until all the data is collected from the whole area. To enable quick decision-making in the disaster scenario, in this paper, we investigate the minimal overall delay between mobile phones and the node j. Meanwhile, we suppose that for each end node i, there is a designated path from i to j, denoted as $\mathbb{P}_i$. We also group all child nodes of k in the set $\mathbb{C}_k$, and all nodes in the set $\mathbb{N}$.

B. PROBLEM FORMULATION

Based on the system model in Fig. 1, we studied the minimal cost from each end node to j for transmission to the cloud

center. We adjust the data processing ratios, denoted as $\mathbb{X}_k$, in each MBS k to find the system cost and overflow. $\mathbb{X}$ is the set that consists of the processing ratios for all nodes. The two nodes shown int he dotted box represent the child nodes of node k.

The cost of each node, Π_k, can be calculated based on the amount of data that it transmits, and the amount of data that it processes.

1) COST OF PROCESSING DATA IN NODE *k*

In order to model the cost (Π) of the data in each node, we first have to know how much data is being transmitted and how much is being processed.

The amount of the data flowing out of the node after processing is denoted as Ψ_k for node k, and can be represented by two parts. One is the size of processed data and the other part is the size of the unprocessed data. They are different since the data size will be compressed by β after data processing in the MBSs.

$$\Psi_k = \Psi_k^p + \Psi_k^u \tag{1}$$

where Ψ_k^p represents the processed data stream and Ψ_k^u represents the unprocessed data stream of node k. The combined data stream cannot exceed the capabilities of the wireless antenna, which is to say that:

$$\Psi_k \leq R_{ik} \tag{2}$$

In situations where $\Psi_k > R_{ik}$, the excess data must be buffered in the node. We refer to this buffered data as the data overflow (OF).

Accounting for the size of the data stream collected at node k (Φ_k) and the amount of unprocessed data streamed from k's child nodes $\mathbb{C}_k$, then Ψ_k^u can be formalized as follows:

$$\Psi_k^u = \left(\Phi_k + \sum_{i\in\mathbb{C}_k} \Psi_i^u\right) * (1 - \mathbb{X}_k) \tag{3}$$

But, we have to limit the amount of data that can be processed based on the hardware available, which is to say:

$$\left(\Phi_k + \sum_{i\in\mathbb{C}_k} \Psi_i^u\right) * \mathbb{X}_k \leq \mu_k \tag{4}$$

Then, Ψ_k^p is calculated as follows:

$$\Psi_k^p = \left(\Phi_k + \sum_{i\in\mathbb{C}_k} \Psi_i^u\right) * \beta * \mathbb{X}_k + \sum_{i\in\mathbb{C}_k} \Psi_i^p \tag{5}$$

where already considers the data streams from k's child nodes $\mathbb{C}_k$.

In general, the data processing usually starts collecting data from all data resources, so the processing cost in node k is calculated as follows:

$$\Pi_k^p = \left(\sum_{i\in\mathbb{C}_k} \Psi_i^u + \Phi_k\right) * \mathbb{X}_k * C^P \tag{6}$$

Where C^P is the cost of processing one bit of data. One beneficial aspect of this model is that cost can represent many different things: energy, time, etc. All that needs to be changed is the C^P and C^R values. In this paper, we mainly consider energy costs, but the normalized results will be similar so long as the ratio of $\frac{C^P}{C^R}$ is the same, which is why we focus on the normalized results.

2) COST FOR RECEIVING DATA IN NODE *k*

The second source of cost in node k comes from receiving data from k's child nodes, $\mathbb{C}_k$. This cost is commonly referred to transmission cost, but because what is transmitted must also be received, we have grouped the costs together, and calculated it on the receiving end. Here we assumed the data collection and transmission in the network are well scheduled, so that the data reception cost in node k can be represented as:

$$\Pi_k^t = C^R * \sum_{i\in\mathbb{C}_k} \Psi_i \tag{7}$$

Where C^R is the cost for receiving a single bit of data.

3) OVERALL COST AND PROBLEM FORMULATION

Then the overall cost in node k can be calculated as:

$$\Pi_k = \Pi_k^p + \Pi_k^t \tag{8}$$

Then overall cost of the system can be formalized as:

$$\Pi^o = \sum_{k\in\mathbb{N}} \Pi_k \tag{9}$$

Then, the cost minimization problem (CMP) for big-data processing in wirelessly networked disaster areas can be formalized as follows.

CMP-WNDA Problem: In wirelessly connected disaster areas, the goal is to minimize the cost and have zero data overflow while sending all data outside under the system model in section III-A, by adjusting the processing ratio of each MBS.

The mathematic representation is:

Find the set $\mathbb{X}$ to $\min(\Pi^o) \wedge OF = 0$

IV. PROPOSED SOLUTION (REAL-TIME ALGORITHM)

The goal of our system is to realize the ideal set $\mathbb{X}$ for yielding the minimal overall system cost while also causing no data to overflow at any node. In order to solve this in real-time, we start by trying to estimate the overflow and the cost that each node can put on the system. In order to estimate the overflow (OF) of the network, we use the equation below.

$$E_{OF}^k = (N_T)^{H_k} * \Psi_k - R_{ik} \tag{10}$$

Where H_k is the number of remaining hops in the path for the data in node k to reach the cloud, and N_T is the average number of child nodes for the nodes in this topology. This equation yields negative numbers, but actual overflow can not be less than 0. This is irrelevant as the next step is to check where the estimated overflow (E_{OF}^k) is greater than 0. If the E_{OF}^k is in fact greater than 0, then the algorithm allocates that node

Algorithm 1: Real-time Continuous Data Cost-Minimization Algorithm (RTCMA).

Require: E^k_{OF}: objective function 1; $\Delta\mathcal{C}'_k$: objective function 2;
Ensure: x_k, the processing ratio for the current node
1: *Calculate* E^k_{OF} using equation (10)
2: **if** $E^k_{OF} > 0$ **then**
3: $x_k \leftarrow 1$;
4: **else** *Calculate* $\Delta\mathcal{C}'_k$ using equation (13)
5: **if** $\Delta\mathcal{C}'_k > 0$ **then**
6: $x_k \leftarrow 0$;
7: **else** $x_k \leftarrow 1$;
8: **end if**
9: **end if**
10: **return** X_k for the current node

to do the maximum possible processing by setting the x_k to 1. If E^k_{OF} is less than or equal to 0, then the algorithm proceeds with calculating the solution which will yield the least cost to the system, which starts with estimating the change in cost of the network based on the processing in node k ($\Delta\mathcal{C}_k$).

$$\Delta\mathcal{C}_k = \Pi^t_{k+1} * H_k + \Pi^p_k \tag{11}$$

$$= H_k * \left(C^R * \sum_{i\in\mathbb{C}_k} \Psi_k\right) + \left(\left(\sum_{i\in\mathbb{C}_k} \Psi^u_i + \Phi_k\right) * x_k * C^P\right) \tag{12}$$

We then take the derivative with respect to x_k and get $\Delta\mathcal{C}'_k$.

$$\Delta\mathcal{C}'_k = \left(C^P + C^R * \rho * H_k - H_k * C^R\right) * \left(\sum_{i\in\mathbb{C}_k} \Psi^u_i + \Phi_k\right) \tag{13}$$

This is useful for letting us know if the cost of the system will increase or decrease as we increase x_k. The entire algorithm can be seen in Algorithm 1.

Solving two simple mathematical models should take less than a handful of ALU cycles. Therefore, this algorithm can be applied to a system that is changing in real-time, for example if the transmission is being blocked by a tree branch that is moving in the wind, and the transmission rate is varying. Or if the cost of processing or cost of transmitting changes throughout the day. Additionally, this algorithm will be more suitable for systems where base stations are currently moving, because even assuming highway speeds, which is not a use case for MDRUs [7], a base station is not likely to move a significant distance in a few milliseconds.

V. EVALUATION

A. EVALUATION METHODOLOGY

The modeling and calculations are all coded in MATLAB. We first determine the capabilities of each algorithm across different network sizes. We do this by first generating 3 random networks with 3 random sizes. The small network is an 11 node system. The medium-sized system has 35 nodes. The large system has 154 nodes. The large system is approximately equivalent to the island of Okinawa, which according to the 2016 report by the Japanese Ministry of Internal Affairs and Communications has stated to have 105 LTE base stations [20]. In order to let the evaluation match with real cases, we surveyed some realistic values for $\mathcal{R}$, μ_k, and β. Most 5 G antennas currently in development suggest that the wireless speeds will reach 20Gbps [21]. Hinitt *et al.* [22] estimated that a GPU-based processor for wireless networks is capable of processing speeds of at least 4 Gbps. The amount of processing that a standard computer can handle is still scaling very rapidly, so we expected this number to increase rapidly as well. Guo *et al.* [23] suggest that LTE compression is capable of a one-third ratio, so we adopted this value for evaluation. In order to simulate a city-setting, the Φ_k values for nodes near the geographic center have and increased probability of a higher value to simulate the city-centre. Furthermore, for a variety of applications, β can be significantly different, from a very small value (e.g. feature extraction from data) to a bigger value (e.g. data compression). We considered the above issue and have performed evaluation accordingly to different β in Fig. 8. A summary of symbol definitions can be seen in Table 1.

All of these values, and other ones used for testing the scalability of the algorithms can be seen in Table 2.

TABLE 1. Parameter Definitions

Design Parameters	Description
$\mathbb{X}_k$	the portion of data to be processed in k
Ψ_k	the total data stream of node k
Ψ^p_k	the processed data stream of node i
Ψ^u_k	the unprocessed data stream of node i
Π^p_k	the processing cost in node k
Π^r_k	the receiving cost in node k
Π_k	overall cost in node k
Π^o	total cost of whole network
N	Number of Nodes in the System
β	Data compression ratio of the target algorithm
$\mathcal{R}_{ik}$	Data Transfer rate from node i to node k
μ_k	Processing rate of the algorithm on a fog node
Φ_k	Raw data being received by node k from cellular device
C^P	Cost of Processing data
C^R	Cost of Receiving data

TABLE 2. System Configuration

Design Parameters	Values
Nodes	11,35,154
β	1/3
$\mathcal{R}_{ik}$	20Gbps
μ_k	4Gbps
Node Patterns	Random with max of 5 child nodes
Φ_k	Random with a max value of 0.5 Gbps
C^P/C^R	3.5
Generations	120

B. EVALUATION ACROSS VARIOUS PERFORMANCE PARAMETERS

Figs. 2 and 3 show how the different algorithms react to changes in the processing rates of the fog nodes. The

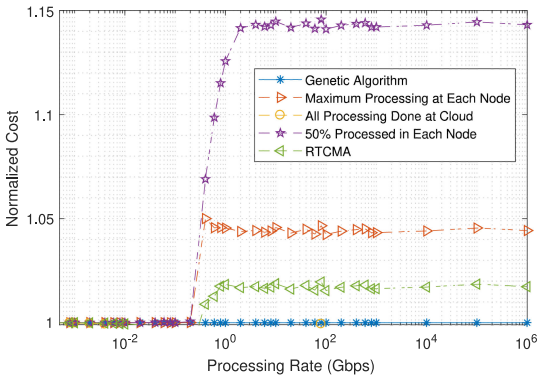


FIGURE 2. Cost values of a 154 node network with varying μ_k values. This shows the RTCMA is approximately 2% worse than the GA for systems with a large processing rate, but 3% better than the system that processes all data.

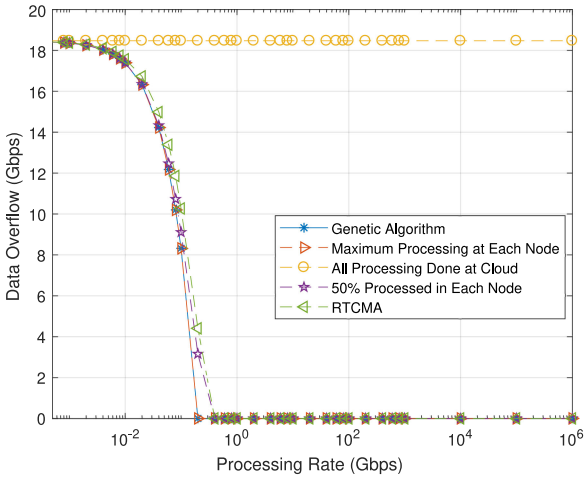


FIGURE 3. Overflow values of a 154 node network with varying μ_k values. The system which processed all data at the cloud failed to prevent overflow.

Cloud-based solution shows that it has a lower cost, but suffers from data overflow. This data overflow requires buffering and significantly increases the latency of the system. The Genetic algorithm is the best performing algorithm with minimal cost and overflow, but the GA can not be implemented in real-time. The fog-based solution has additional cost compared to the Real-time Continuous Data Cost-minimization Algorithm (RTCMA). These results are promising because they show that the RTCMA is the real-time algorithm which is most-resilient to changes in the processing speed.

In Figs. 4 and 5 we can see the results of varying the communication rate, given in Gbps. In this test, we can decipher that none of the algorithms can handle the tested load until 80 Gbps is achieved. As shown in Fig. 5, the only algorithm that takes more time is the cloud solution, which has no overflow after 100 Gbps. In terms of the cost, the cloud, GA, and RTCMA all have similar values, while the fog-based and 50% solutions both have an increase of about 6% and 15%, respectively.

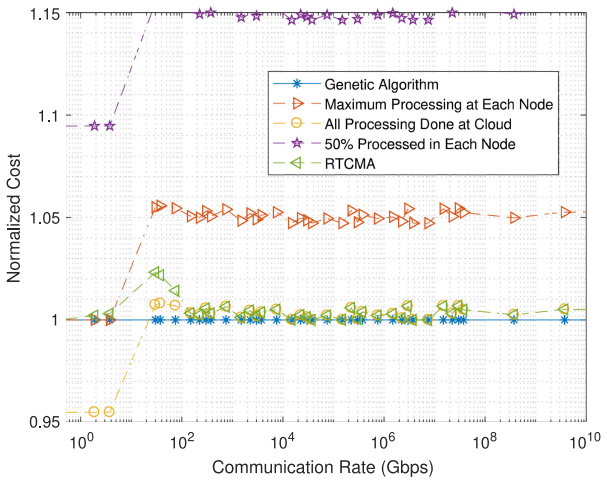


FIGURE 4. Cost values of a 154 node network with varying $\mathcal{R}$ values.For most of the communication rates, the RTCMA has a less than 1% increase in cost compared to the GA.

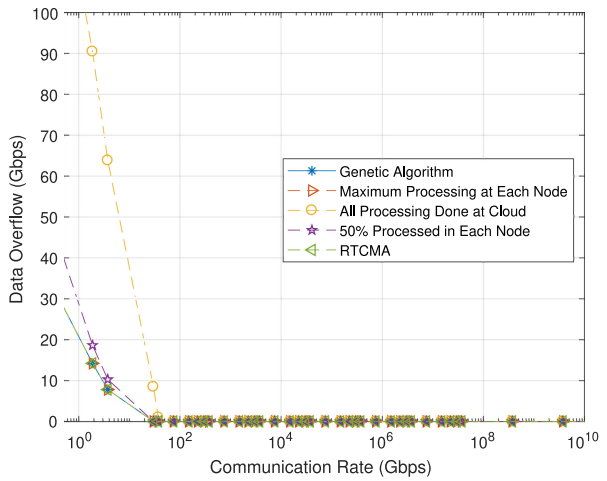


FIGURE 5. Overflow values of a 154 node network with varying $\mathcal{R}$ values. Again, the cloud-based solution fails to prevent overflow.

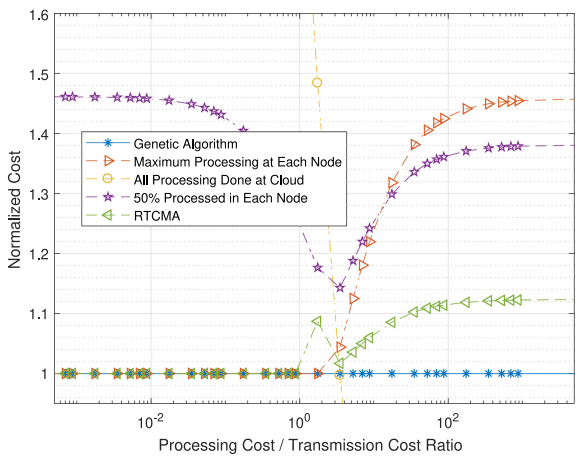


FIGURE 6. Cost values of a 154 node network with varying $\frac{C^P}{C^R}$ values. The RTCMA is the best solution other than the GA, and has a worst case increase of 11% compared to the GA, where the other conventional algorithms peaked over 45%.

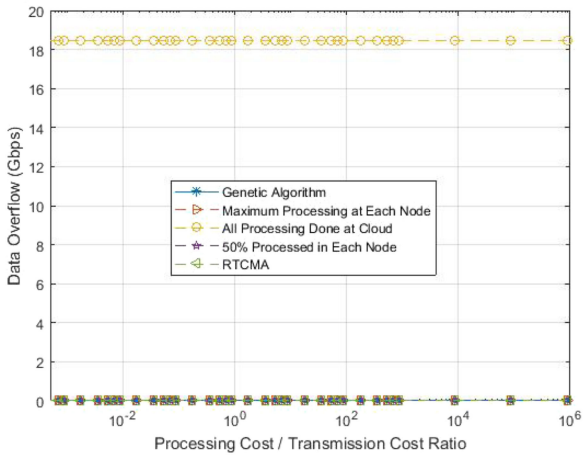


FIGURE 7. Overflow values of a 154 node network with varying $\frac{C^P}{C^R}$ values. All systems prevent overflow except for the cloud-based solution.

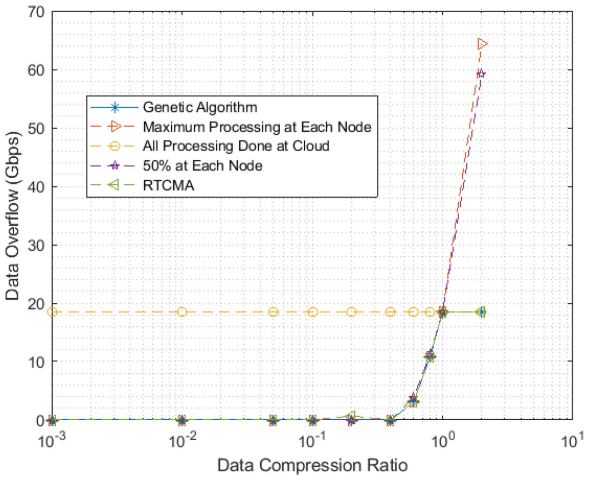


FIGURE 9. Overflow values of a 154 node network with varying β values.The RTCMA matches the GA in overflow as well.

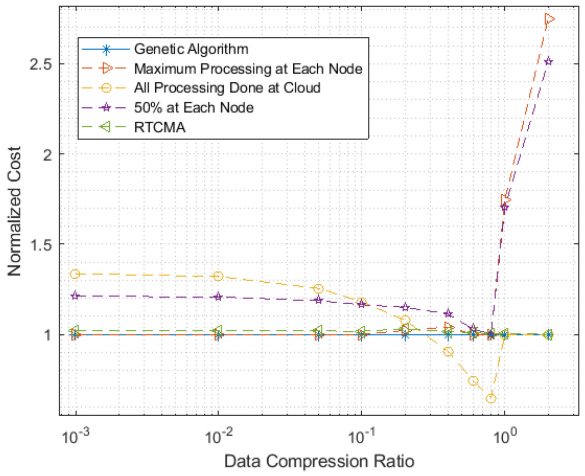


FIGURE 8. Normalized Cost values of a 154 node network with varying β values. The RTCMA matches the GA very well.

C. EVALUATION ACROSS VARIOUS COST RATIOS

In order to perform this test, we set the C^R to a realistic but fixed value. We then set the C^P equal to the ratio times the C^R value.

Because the cost does not affect the amount of data that overflows, we can see that in Fig. 7, that none of the values change, and that the cloud-based solution is the only algorithm to fail across all data points. Figure 6 shows us how each of the algorithms behaves to the changes in the cost ratio. The cost of the cloud-based solution is an extremely steep curve, which is very high for low values of the ratio, and is very low for the high values of the ratio. This is because if processing costs significantly more than transmission, then the cloud-based solution which only utilizes communication infrastructure will thrive. The fog-based solution starts with low costs in the areas where processing costs less than transmission, and eventually reaches a critical point where it slopes upward, and then it hits another critical point and plateaus. We see a similar shape when looking at the RTCMA, but the plateau is significantly lower. Where the fog-based solution plateaus at 145%, the RTCMA plateaus at 112%. There is one difference at around a communication rate of 2 Gbps, which is when the algorithm is getting falsely triggered to do some processing, even when its not the ideal case, but the cost remains below 110%. The 50% solution has the drawbacks of the cost of the cloud-based solution at the small cost ratios, and the fog-based solution at the large cost ratios. This is why the 50% solution exhibits a V-shaped curve in this test.

D. EVALUATION ACROSS WITH VARIED ALGORITHM CAPABILITIES

We want to ensure that the optimization algorithm can be applied to a wide variety of processing algorithms. Extremely efficient data processing algorithms can have a β value of 0.01 or less, and algorithms that increase the data size after processing have a β value greater than one. Therefore we simulated values from 0.001 to 2. From Fig. 9 We can see that the cloud solution is unaffected by the β value. When the performance is normalized to the GA, as in Fig. 8, we can see the RTCMA performs very similarly to the GA. This gives us confidence that the real-time algorithm is resilient to changes in the big data algorithm that it is optimizing the network for. Most of the algorithms have a significant dip when β is equal to 0.8. This is due to the fact that the GA is starting to be forced into a setting that processes significantly less data. With less choice, it starts to lose its benefit over the conventional systems.

E. ERROR EVALUATION

Figure 10 shows the average percent error across the different network sizes, while Fig. 11 shows the corresponding amount of data points that have overflow. The algorithm that is most sensitive to changes in the network size was the 50% solution which was not pictured so that other algorithms could

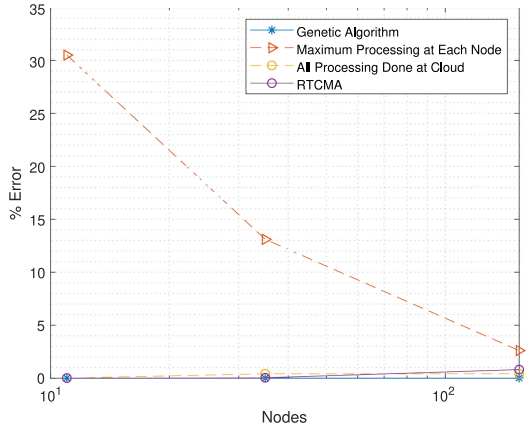


FIGURE 10. Percent error of the different algorithms across different network sizes (lower is better). The RTCMA peaks at around 1% error for cost.

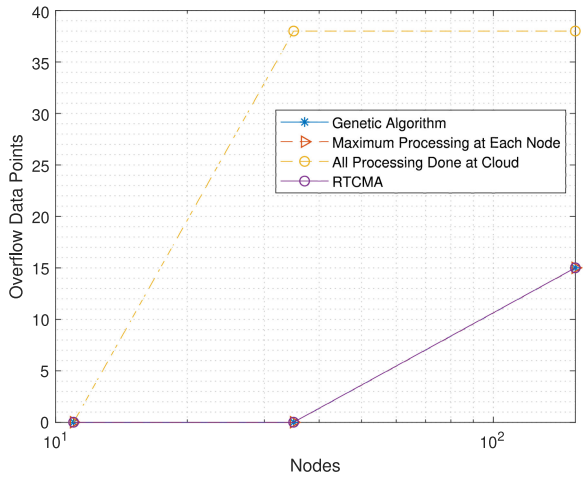


FIGURE 11. Number of data points that had data Overflow (lower is better). The RTCMA matches the GA for overflow.

be seen properly. The next most sensitive was the fog-based solution which was very inaccurate for small networks, which is unfortunate because most of the currently implemented fog networks are similar to this size. The cloud-based solution and the RTCMA are not as sensitive to changes in the network size, but each gets a little less accurate as the size grows. At the largest network size that was tested, both of these algorithms still had less error than the fog-based solution. In terms of overflow points, the cloud performed well on the small system, but it could not handle the medium or large systems at all. The fog-based solution and RTCMA had the same amount of overflow data points as the genetic algorithm, which we assume finds the ideal case, so for those few network parameters on the largest network, there is no viable solution that avoids overflow.

F. TIMING EVALUATION

One major claim from this paper is that the RTCMA is that it is real-time. In order to test this claim, we time how long it takes

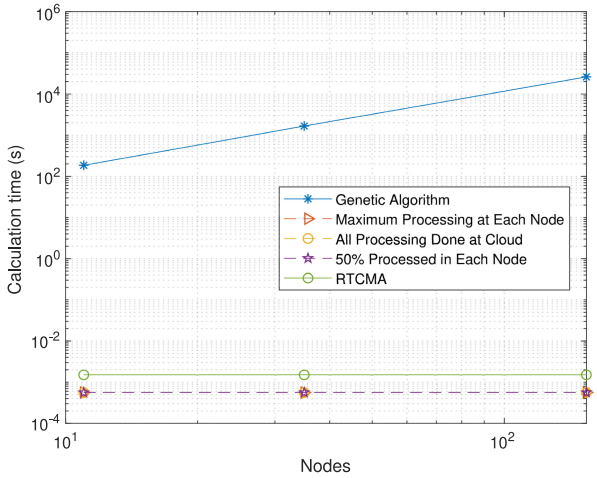


FIGURE 12. Timing values of the algorithm with different sized networks (lower is better). The RTCMA can be run in a few milliseconds, while the GA takes hours.

our simulator to run each of these algorithms. With specialized hardware that has the calculations done in circuitry instead of by an general purpose processor, the RTCMA can run even faster, but we wanted to see if it was fast enough of a regular CPU.

In Fig. 12, we can see that the GA takes a gross amount of time, and that that time only grows as the network gets larger. In the worst case, it breaks 25000 seconds, or approximately 7 hours, which is hardly real-time, and could not be implemented on any system that needs to adapt in real-time. The 3 fixed-rate algorithms take approximately half a millisecond, just to allocate the 0 value, but in reality can not adapt, and would simply be a latency when starting up the system. The final algorithm, RTCMA, runs all three network sizes at around 1.5 ms. This is not on specialized hardware but inside the simulation. A general purpose calculator would be faster in performing the calculation, and a field programmable gate array (FPGA) or application specific integrated circuit designed (ASIC) for the RTCMA algorithm could perform in one clock cycle, on the microsecond scale. But even in the worst-case scenario (inside a simulator), the algorithm can be considered to be real-time.

VI. CONCLUSION AND FUTURE WORK

This paper presents a set of equations and an algorithm to minimize the cost of running a big-data algorithm in a fog-computing wirelessly networked disaster area. We use previously proposed cost models and a simulator, and compare our proposed RTCMA algorithm with a genetic algorithm, and some other conventional methods.

The tests show that the RTCMA was the only viable real-time solution which could adapt properly to changes in network parameters. This means that running it in real-time at each node would allow the system to adapt to changes in the network structure or disconnections in real-time. The RTCMA successfully avoids any network overflow, and has minimal

cost increases compared to the GA, which had an approximate 10000x increase in calculation time.

In the future, we plan to implement the algorithm in an FPGA to improve the speed, and hopefully integrate it with MDRU technology. This will let us test the capabilities of the algorithm on a small scale, but bring it out of simulation. If the FPGA is successful, we would like to design and ASIC for fabrication, which is cheaper and smaller than an FPGA for each MDRU.

REFERENCES

[1] J. Wang, Y. Wu, N. Yen, S. Guo, and Z. Cheng, "Big data analytics for emergency communication networks: A survey," *IEEE Commun. Surveys Tut.*, vol. 18, pp. 1758–1778, Jul.–Sep. 2016.

[2] M.-P. Kwan and D. M. Ransberger, "Lidar assisted emergency response: Detection of transport network obstructions caused by major disasters," *Comput., Environ. Urban Syst.*, vol. 34, no. 3, pp. 179–188, 2010.

[3] B. Rochwerger *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM J. Res. Dev.*, vol. 53, pp. 4: 1–4:11, Jul. 2009.

[4] S. S. Gill *et al.*, "Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges," *Internet of Things*, vol. 8, pp 1–26, 2019, Art. no. 100118.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.,* pp. 13–16, 2012.

[6] R. Mahmud, R. Kotagiri, and R. Buyya, *Fog Computing: A. Taxonomy, Survey and Future Directions*, pp. 103–130. Singapore; Berlin, Germany: Springer Singapore, 2018.

[7] T. Sakano *et al.*, "Bringing movable and deployable networks to disaster areas: Development and field test of mdru," *IEEE Netw.*, vol. 30, pp. 86–91, Jan. 2016.

[8] S. S. Anjum, R. M. Noor, and M. H. Anisi, "Review on manet based communication for search and rescue operations," *Wireless Personal Commun.*, vol. 94, no. 1, pp. 31–52, 2017.

[9] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan, "How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users," in *Proc. IEEE Int. Conf, Pervasive Comput. Commun.*, 2012, pp. 122–127.

[10] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, pp. 810–819, May 2017.

[11] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, pp. 108–119, Jan. 2017.

[12] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[13] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. IEEE Veh. Technol. Conf.* (VTC Spring), 2015, pp. 1–6.

[14] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 74–83, Mar. 2015.

[15] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distribution Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[16] S. D. Ali Shah, H. P. Zhao, and H. Kim, "Distributed deep neural networks with system cost minimization in fog networks," in *Proc. TENCON IEEE Region 10 Conf.*, 2018, pp. 1193–1196

[17] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurrency Comput.: Pract. Experience*, vol. 29, no. 16, pp. 1–9, 2017, doi: 10.1002/CPE3975.

[18] Y. Wang, M. C. Meyer, J. Wang, and X. Jia, "Delay minimization for spatial data processing in wireless networked disaster areas," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[19] M. C. Meyer, Y. Wang, and J. Wang, "Cost minimization of data flow in wirelessly networked disaster areas," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.

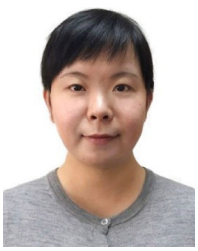
[20] M. of Internal Affairs and Communications, "information & Communications Statistics Database H28.1Q Report," Accessed: Mar. 29, 2017. [Online]. Available: http://www.soumu.go.jp/johotsusintokei/field/denpa01.html, 2016.

[21] T. Obara, S. Suyama, J. Shen, and Y. Okumura, "Joint fixed beamforming and eigenmode precoding for super high bit rate massive MIMO systems using higher frequency bands," in *Proc. IEEE 25th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Sep. 2014, pp. 607–611.

[22] N. Hinitt and T. Kocak, "Gpu-based fft computation for multi-gigabit wirelesshd baseband processing," *EURASIP J. Wireless Commun. Netw.*, vol. 2010, pp. 30: 1–30:13, Apr. 2010.

[23] B. Guo, W. Cao, A. Tao, and D. Samardzija, " LTE/LTE-a signal compression on the CPRI interface," *Bell Labs Tech. J.*, vol. 18, pp. 117–133, Sep. 2013.

MICHAEL CONRAD MEYER (Member, IEEE) graduated from the Rose-Hulman Institute of Technology, in Indiana, USA, with the B.S. in computer engineering in 2012, and then with the M.A. in engineering management in 2013. In 2017, he received the Ph.D. degree in computer science and engineering from the University of Aizu. He has worked for Texas Instruments before starting the Ph.D., and before that he had worked for Syntheon developing biomedical devices. He is currently an Assistant Professor with Waseda University. He was previously a Postdoctoral Researcher with the University of Aizu in Fukushima, Japan, as a member of the Data Networking Laboratory. His research interests cover on and off chip networks, reliability and photonics.



YU WANG (Member, IEEE) received the M.S. degree from the University of Aizu in 2016. She received the Ph.D. degree from the University of Aizu, Japan, in 2019, while studying in their Data Networking Laboratory. She is currently a Visiting Researcher with Waseda University. Her research interests include optimizations and algorithms for big data, fog computing, and disaster-area networks. She also did an Internship for ALPS Electric Company and ALPINE Electronics before studying with the University of Aizu.

TAKAHIRO WATANABE (Member, IEEE) was born in Ube, Japan, on October, 1950. He received the B.E. and M.E. in electrical engineering from Yamaguchi University, and the Dr. Eng. from Tohoku University. In 1979, he joined Research and Development Center of TOSHIBA Corporation, where he worked in the field of LSI design automation. In August 1990, he joined Yamaguchi University, the Department of Computer Science and Systems Engineering, and in April 2003, he moved to Waseda University, Graduate School of Information, Production and Systems. His current research interests include EDA algorithm, Microprocessor and MPSoC, NoC, FPGA and their applications. He is a member of IEICE, IPSJ.