Mobile Networks and Applications
https://doi.org/10.1007/s11036-023-02221-8

Computation Offloading for Image Compression in Mobile Edge Computing Using a Deep Belief Network Based on the Markov Approximation Algorithm

N. Noor Alleema[1] · Abhay Chaturvedi[2] · Ashok Kumar Nanda[3] · P. Joel Josephson[4] · Ahmed Mateen Buttar[5] · Dinesh Komarasamy[6]

Accepted: 29 June 2023

Abstract
The rising popularity of health-related wearables has increased the demand for computational offloading. Latency, battery life, and compute capabilities are all areas that could use significant improvement despite the massive expansion of wearable devices and offloading techniques. There are a number of issues caused by the limited battery life of mobile devices, so it is essential to conserve power by moving responsibilities for running applications to the cloud. This is especially crucial in situations where scheduling is being done in a dynamic mobile cloud computing environment. In this piece, we highlight the fact that most smart wearable devices can pair with smartphones via Bluetooth, and that this connection is significantly more power-efficient than 3G/LTE or Wi-Fi communication. To be more precise, we investigate the origins of this phenomenon. New computing technology, Mobile Edge Computing (MEC), may be the answer to capacity and performance problems in older systems like Mobile Cloud Computing (MCC). Some examples of these problems include excessive latency, a clogged core network, a lack of quality of experience (QoE), and the wasteful consumption of expensive resources like power and data transfer. To address these issues in the MEC setting, we introduced a Deep Belief Network (DBN) equipped with a Markov Approximation Algorithm (MAA). Because of this, a choice could be made that takes into account energy use, time constraints, and system load. AFCFS, MINET, and trade-off judgments for code offloading are all examples of state-of-the-art techniques that can be outperformed by this approach (TRADEOFF). Using entropy encoding as a post-processing step after quantization allows for lossless compression of an image. It allows for a picture to be shown in a way that is both more effective and requires less space for storage or transmission. Our proposed DBN-MAA reduces energy consumption while simultaneously increasing the number of completed jobs, as shown by the simulation experiment results.

Keywords Deep belief network · Markov approximation · Mobile edge computing · Energy efficiency and entropy encoding

1 Introduction

Mobile devices (MDs) are frequently utilized to gather and process data because of the growth of IoT technology. These gadgets are typically designed to be compact, with a meagre supply of computational power and energy. Concerns about the excessive energy consumption of mobile devices have been raised because the processing unit on these devices may take a long time to execute some computing jobs in some apps. To overcome the obstacle between the demand

✉ Ashok Kumar Nanda
ashokkumarnanda@yahoo.com;
ashok_kumarnanda78@outlook.com

[1] Department of Information Technology, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi, Chennai, India

[2] Department of Electronics and Communication Engineering, GLA University, Mathura, India

[3] Department of CSE, B V Raju Institute of Technology, Narsapur, Medak, Telangana, India

[4] Department of ECE, Malla Reddy Engineering College, Hyderabad, Telangana, India

[5] Department of Computer Science, University of Agriculture Faisalabad, Faisalabad 38000, Pakistan

[6] Department of CSE, Kongu Engineering College, Perundurai, India

Published online: 01 September 2023

for complicated applications and the availability of limited resources, mobile cloud computing (MCC) is offered. Computing tasks are typically carried out on the central cloud, which has abundant processing capabilities and huge storage capacity in MCC application scenarios [1]. Using this technique, MDs were able to handle difficult calculation tasks with just locally high energy consumption [2]. Usually, centralized servers are far from MDs. Increased use of real-time operations and the requirement for low latency for MCC are consequences of the proliferation of Internet and 5G mobile networks.

The problems have been solved with the establishment of MEC. Servers at the network's edge are typically located strategically throughout a mobile ad hoc network (commonly with the wireless base station in 5G networks). When servers are located closer to MDs, latency and energy consumption due to data transmission can be effectively decreased. When compared to a centralized server cluster, the resources available on MEC servers are limited. Given that MEC servers only cover a specific area, this shouldn't be an issue. It's simple to adjust the capacity to meet the needs of the situation. There's also the fact that MDs don't always produce huge computation jobs, so sending them to be processed on servers regardless of their size is a waste of resources. With the advent of MEC, MDs have greater flexibility in handling workloads, both in terms of when and by how much to outsource computational tasks. The quality of service greatly fluctuates depending on offloading decisions (QoS). Computation offloading choices, given the above, are an increasingly studied subfield of MEC.

Compression is the process of reducing the amount of data used to represent a file, image, or video without sacrificing the quality of the original data. Image compression is the use of data compression to compress digital images. The primary goal of image compression is to reduce redundancy and irrelevancy in images so that they can be stored and transferred more efficiently. When compared to the original, the compressed image uses fewer bits. As a result, the required storage size will be reduced, allowing for maximum image storage and faster transfer to save time and transmission bandwidth.

In theory, pictures taken on mobile devices should be able to be processed using the same algorithms that are used for other kinds of pictures. However, when directly applied to a problem, the effectiveness of such algorithms is diminished due to the many constraints and outside factors. The vast majority of mobile devices continue to have significantly less computing power than computers that have been specifically designed for image compression. They are also powered by a battery, and the limited energy that is stored in the battery needs to be conserved so that the device can fulfil its primary purpose. Image processing on mobile devices generally faces new challenges brought on by smartphones in particular. The competitive nature of the market has led to the development of smartphones with extraordinarily high-resolution cameras. The presence of a large number of pixels that need to be processed calls for algorithms that are particularly efficient. The necessity of living up to the anticipations of the end users is yet another important non-technical factor. Users of smartphones have come to anticipate that their devices will produce photographs and videos of a high quality, in addition to providing them with prompt access to processed content.

The motivation behind using compute offloading could be varied, including saving resources like time, energy, money, etc. The objective in terms of time is to lessen the lag time between operations, while the objective in terms of energy is to cut down on the amount of power consumed. The cost objective can be seen from both perspectives. One is the price of sending data, and the other is the price of computing power at the edge. Some tactics focus solely on one opponent. As an illustration, while [3] focuses on the time goal, [4] analyses the energy goal exclusively. Methods like [5] take into account the existence of two targets. Some studies take into account three or more targets, but these are rarely compared to those that focus on just one or two. Therefore, it's essential to study more than just two targets in depth.

Following the proposal of the computation offloading model, the optimal strategy for obtaining computation offloading decisions must be established. The decisions that are derived should produce the best possible model outcome given the constraints. Swarm intelligence is a term used to describe the collective intelligence behaviour of autonomous and distributed systems [6]. Further, many academics from different fields have taken an interest in a specific type of algorithm called a swarm intelligence algorithm. The given computation offloading paradigm has been shown to be solvable by many studies and applications of swarm intelligence optimization algorithms [7].

In this article, we will look at an MD that can adjust the frequency of its central processing unit (CPU) and split up computational tasks among several MEC servers. To reduce the time versus energy cost of finishing tasks on mobile devices, we use optimal control decisions based on job assignment and CPU frequency. The difficulty of solving this combinatorial optimization problem makes it NP-hard. Using this framework, which is derived from the Markov approximation framework, we were able to develop a nearly optimal approximation strategy. In a nutshell, this work mainly contributed these things:

- To make the trade-off between latency and energy as small as possible, it is suggested to use a nonlinear combinatorial optimization problem in a multimerger MEC system that makes use of the job assignment decision and the ability to scale up computing power.

- To swiftly address the stated problem, our approximation strategy takes advantage of the Markov approximation framework. This strategy can give the issue a close-to-ideal resolution using a Markov chain and state changes. Here, we look at the most noteworthy features of the resulting Markov chain and analyse the method's performance optimality, approximation gap, and error robustness.
- Our Markov approximation-based strategy is put through its paces with a wide range of input parameters in a series of simulations.. Results from computer simulations show that this algorithm outperforms the best benchmark algorithms and can produce results that are competitive with or better than the benchmarks.

The remaining sections of this paper are organised as follows. In Part 2, we'll discuss some related literature. Section 3 introduces the algorithm, based on the Markov Approximation Algorithm, that will be used to construct the Deep Belief Network (DBN) and the system model (MAA). In Section 4, we contrast the experimental outcomes achieved by HIBSA with those achieved by other algorithms. These contrasts show how much better HIBSA is than its rivals. In Section 5 we see the final results.

2 Literature survey

Zhang et al. [8] The proposed process of assigning jobs to edge servers with enough resources in accordance with specific offloading policies is referred to as task offloading. These rules define how effectively the MEC can compute as well as its efficiency. Computing migration or task offloading are other names for this practise. High quality of service (QoS) is attained by sending compute-incentivized tasks to MEC servers, like face recognition and video optimization. Mobile Edge Computing (MEC) servers can provide high quality of service (QoS) by processing compute-intensive tasks. This is because MEC servers are located closer to end-users, reducing latency and network congestion. Task offloading issues have piqued scholars' interests greatly in recent years.

A support vector machine-based offloading algorithm was put forth by Wu et al. [9]. (SVM). Applying a weight allocation strategy to the process of dismantling a task into its component subtasks constitutes the first step of the method that has been suggested. The next thing that needs to be done is to determine whether or not each individual subtask is going to be completed in-house or through outsourcing. This is a very important step that cannot be skipped.

In Youet al. [10], the authors investigated a problem with minimising energy consumption in multi-user MEC networks, where the problem was constrained by latency. They devised a method for the provisioning of resources that ended up saving a significant amount of energy over the course of the project. In addition, we present a novel strategy for dealing with the difficulties brought on by mobile computing's restricted access to power sources by integrating wireless power transfer (WPT) into mobile embedded computing.

To minimize power consumption and keep latency constraints, Cao et al. [11] described a cooperative optimization of computing and communication resource provisioning in MEC. They concluded that the cooperative method not only greatly improved performance but was also significantly more energy efficient than other methods that did not use cooperative design. Several previous studies have made the case for the MEC's techniques being energy efficient.

To lower users' overall energy consumption, Sardellitti et al. [12] created an adaptive strategies focused on succeeding convex approximation techniques. The previous work, on the other hand, exclusively considered the circumstance where an MD is only connected to a single edge server. This diversity can be used to give MDs more offloading options and the right amount of resource capacity in future mobile networks, keeping service latency to a minimum while giving users a good experience.

Jang et al. [13] modified the offloading ratios of multiple vehicles in order to decrease their collective energy footprint. This was done with the consideration that the nature of the communication landscape is in a state of constant change. The proposed energy-saving offloading method does considerably cut down on the vehicle's overall energy consumption; however, it does not account for the energy that is required to process data at the computing node. Consequently, the overall energy consumption of the vehicle remains relatively unchanged.

Deng et al. [14] outlined an original offloading system as a means of developing mobile services with offloading decisions that are reliable. This system takes into account the interdependent relationships that exist between the individual services that make up its components. Its goal is to reduce the amount of time needed to complete tasks and the amount of power that mobile devices require. The challenge of compute offloading in a single-server MEC system is the primary focus of the research projects that have been listed above.

Zhang et al. [15] conducted research on the process of computation offloading within a distributed MEC network that contained multiple edge servers. In this configuration, Internet of Things (IoT) devices were responsible for the generation of computational work that included variable requirements and was then offloaded to MEC servers to be processed. This work should, in the end, result in Internet of Things devices that have a lower power consumption and a higher rate of task completion.

In a decentralized IoT network, Anajemba et al. [16] use a collaborative offloading strategy for multi-access MEC that relies on the inefficient convergent computation offloading algorithm (LSCCOA). On the other hand, none of the methods took into account how important it was to transfer responsibilities from one person to another. When it comes to the importance of various activities, customers place varying amounts of weight on different pursuits.

The MET algorithm Li et al. [17] also known as MET Comm in the literature, improves on MINET. After being offloaded, each task is assigned to the resource that can complete it in the shortest amount of time overall. considering the amount of time spent executing the task as well as communicating about the task. In other words, it prioritizes the time spent communicating about the task over the time spent performing the task.

Sun et al. [18] proposed multi-objective task scheduling method with the goal of optimizing how much weight is given to the offloaded jobs. However, the following restrictions apply to this work: First, only an edge server can be used to offload jobs in this work. Second, the mobile terminal's energy usage was neglected. The optimization result in this work was obtained using the bat method. Since the bat algorithm doesn't have a mutation operation, the solutions may not be very different from each other.

According to the energy usage of the various implemented locations of the jobs, Hao et al. [19] suggest a method for offloading the mobile device in this study. This is done in an effort to conserve energy and account for the system load of mobile devices. After determining an initial value using the AFCFS method, the "Replace" and "Insert" offloading strategies are optimised using an adaptive greedy algorithm with the taboo mechanism. This is completed after the initial value has been determined. Only the records of feasible work schedules that contain the top-n values of multiple-target functions or those that are modifiable by taboo search are still included in these solutions. For cloud-based jobs, we continued to employ the same strategy.

In order to move code from mobile devices to the cloud, Tao et al. [20] proposed TRADEOFF as a smart data processing offloading system that can end up making tradeoff decisions. Mobile device storage has been cleared of this code. In order to maximise energy savings while still meeting job latency requirements, he developed a metric. Tasks are always completed as rapidly as possible by MINET. This strategy consistently ignores energy consumption, giving it the highest AEC score. Given that both methods constantly attempt to assess the tradeoff between execution time, and energy use,TRADEOFF and IGTMA both have a very stable AEC value. IGTMA considers a variety of goals. The simulation performance of such targets is guaranteed by the target function. The value of several target functions is also increased by "Replace" and "Insert."

3 Proposed system

A block diagram of the mobile cloud and mobile edge computing architectures optimized with the DBN-MAA algorithm is presented in Fig. 1. With a mobile cloud architecture, users can access the internet and pull the necessary information from a cloud server using only their mobile device and a communication network. The differentiation between the mobile cloud computing (MCC) and the mobile edge computing (MEC) is that MEC is utilized to process time-sensitive data, whereas MCC is used to process non-time-sensitive data.

It is evident in the mobile edge computing architecture that the MEC performs its process, which includes SBS and MBS, with the help of the central cloud and core network along with Huffman encoding for image compression. Small Base Stations (SBS) and Macro Base Stations (MBS) are vital components of cellular networks that provide wireless coverage to mobile devices. SBS and MBS work together to offer mobile devices seamless wireless coverage while preserving network efficiency, availability, and quality of service. To increase the coverage area, throughput, and data transfer rates of a cellular network, a miniature base station called a "small cell" can be installed. The use of small cells can lead to significant improvements in network capacity, coverage, performance, latency, and power consumption, making it a compelling solution for wireless communication systems. Macro stations are cellular base stations that transmit and receive radio signals via tall antenna towers. A deep belief network, which has undirected connections between some layers, is then used to classify the data. It's a generative hybrid graphical model that uses unsupervised probabilistic deep learning. Then, it was optimized with the help of MAA (Markov Approximation and Algorithm) before being validated to show that it performed better in various respects. The Markov Approximation and Algorithm (MAA) is a framework that utilizes a Markov chain representation of the state space to approximate high-dimensional dynamic programming issues. The MAA approach has been shown to perform better in several areas, including computational efficiency, flexibility, and so on.

Many people consider MCC essential because it facilitates the transition of computational workloads to the cloud. Mobile devices serve as the front end, while the cloud serves as the back end. The interaction between MCC and the cloud is enabled through the use of technological infrastructures like wireless connections, location-based tools, and mobility services to ensure a constant flow of resources. To get the most out of your mobile devices, consider using mobile cloud computing (MCC), which allows you to store and process data in a remote

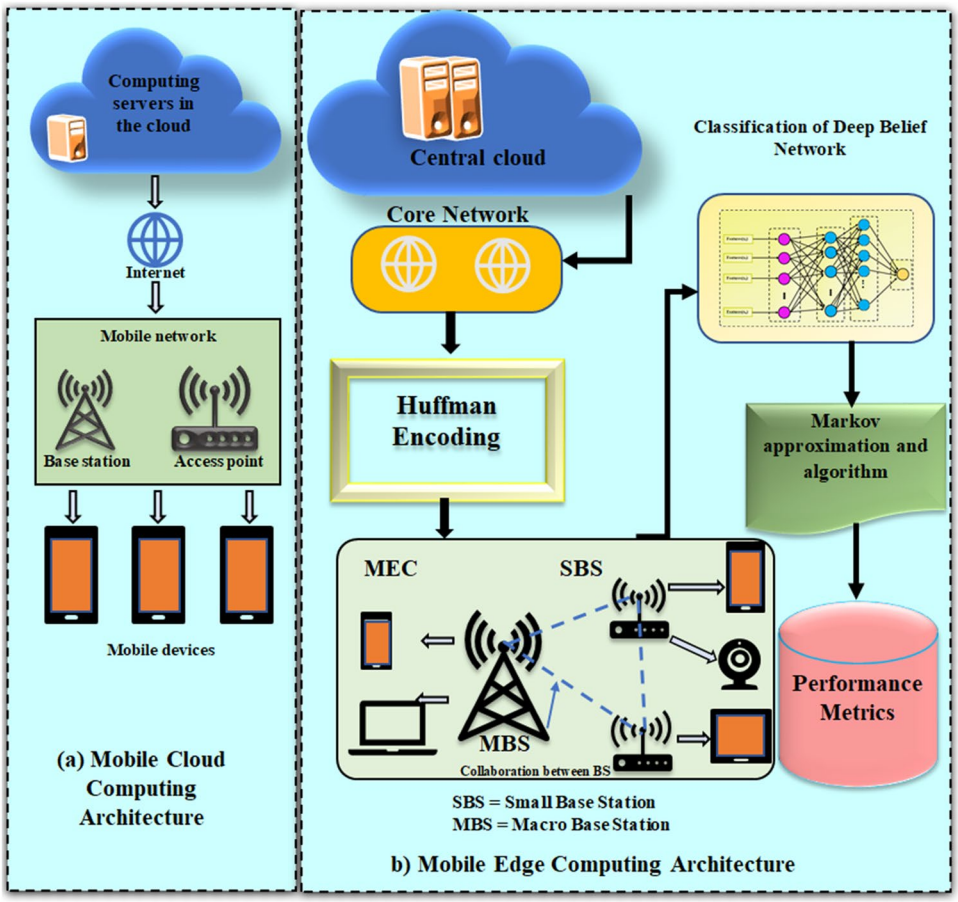

Fig. 1 Proposed diagram of DBN with MAA

server. The tasks are used to store and process data on a remote server is data synchronization, Data storage, Data processing, security and reliable network connection. Offloading the processing of resource-intensive mobile applications to the cloud is one way that cloud computing can boost their performance.

The 5G network is anticipated to produce more data than the 4G network as a result of the addition of more connected MDs to the network in the future. The primary difference between 4 and 5G is latency. While 4G latency ranges from 60 to 98 ms, 5G promises latency of less than 5 ms. Additionally, lower latency promotes improvements in other areas, like faster download rates.. Applications that are both resource-intensive and time-sensitive are putting increasing strain on the MCC system's capacity and performance. Due to limited battery life, CPU speed, and memory, doctors experience performance issues for resource-intensive mobile apps.

Previous studies have recommended architectures that bring computer resources closer to end users for the ease and effectiveness they provide, as explained above. According to the fog architectural design study conducted by According to Bonomi et al. in order to achieve low latency in Internet of Things (IoT) and big data applications, delay-sensitive operations should be performed at the edge while sending resource-intensive operations to the cloud via the core network. Fan and Ansari suggested a similar approach to enhance the labour allocation between BSs and fog nodes. Patel et al. suggested the Mobile Edge Computer (MEC) to put computing resources at cellular networks' Access Points (AP) or Base Stations (BS) closer to end users (BS).Energy efficiency, close proximity, fast reaction time, mobility, and the ability to track a user's location are among the MEC's most distinctive features.

As shown in Fig. 1, mobile devices serve as the front end of the MEC architecture, with MEC servers in the middle and the cloud serving as the back end. All essential processing and storage capacity is installed at BS or AP, which is more convenient for users. Depending on the intensity of the signal, each mobile device is assigned a BS or AP that is regarded as its "home BS." MEC can help relieve network congestion by offloading work, caching data, processing it, and delivering services, all of which have the potential to reduce bandwidth costs, energy consumption, and latency [20].

3.1 System model

Analyze an MD that has M tasks to do in sequence. These tasks can be executed locally or forwarded to one of the nearby MEC servers N. Different wireless channels ensure that two wireless bases will not collide with one another. Wireless channels are used to separate and distinguish between different wireless transmissions. They prevent two wireless bases, such as routers or access points, from colliding by using different frequencies. Three separate binary variables are used to represent task allocation $a_m, b_m, n \in [0, 1], \forall m \in M$ and $\forall n \in N$

$$\begin{matrix} a_m \left\{ \begin{matrix} 1, \textit{ if task m processed at local CPU} \\ 0, \textit{ if task m processed at local MEC server} \end{matrix} \right\} \\ b_{mn} \left\{ \begin{matrix} 1, \textit{ if task m is assigned MEC server n} \\ 0, \textit{ otherwise} \end{matrix} \right\} \end{matrix} \quad (1)$$

In other words, the |M| tasks could be partitioned into the sets |N| and |N|+ 1. As such, we place the following limitation on it:

$$a_m + \sum_{n \in N} b_{m,n} = 1 \ \forall m \in M \quad (2)$$

The triple (a_m,b_m, and c_m), characteristics a computational job m. where am is the number of processing unit cycles required to finish the task, b_m is the number of bits in the data used to perform the computation, and cm is the number of bits in the data that was generated as a result of the computation. Here, we assume that the MD has resources like offline measurements and call-graph analysis at his disposal in order to determine the parameters of interest (a_m,b_m, and c_m). Offline measurements involve collecting data on a device's usage, whereas call-graph analysis entails analyzing the sequence of function calls made by an application or program on a mobile device, that can offer insights into the application's behavior and performance and assist developers in identifying and resolving any issues. We have analyzed the power requirements and runtime of both on-premises and cloud-based options for coping with the computation's overhead.

1. The first option is local computing, in which case the MD would use its own central processing unit to carry out the computation task m. Let MD represent the MD's computational power (based on CPU cycles per second). The amount of time it takes for a local CPU to complete a batch of tasks, based on a given decision profile $a = \{a_m\} \in \{0, 1\}^{|M|}$ and ψ^{MD}, is:

$$T_{MD}(a, \psi_{MD}) = \sum_{m \in M} a_m \frac{p_m}{\psi_{MD}} \quad (3)$$

That is where we get our computational oomph.

$$E_{MD}(a, \psi_{MD}) = \left(\Upsilon_1 \psi^{\theta}{}_{MD} + \Upsilon_2\right) T_{MD}(a, \psi_{MD}) \quad (4)$$

where $\left(\Upsilon_1 \psi^{\theta}{}_{MD} + \Upsilon_2\right)$ represents the MD's ability to process data numerically. For example, the values for can be anywhere from two to three, and the values for B1 and B2 will vary with the design of the chip being used. The method of measurement described in allows for the extraction of values. The MD could dynamically scale ψ_{MD} using DVFS technology, cutting down on execution time and power consumption while doing so. In this work, we assume that ψ_{MD} is a real number with values in a finite, discrete set $\psi = \{\psi_{\min} \le \psi \le \psi_{\max}\}$, where $\psi_{\min}$ and $\psi_{\max}$ are the min and max CPU frequencies of the MD, respectively. Dynamic voltage and frequency scaling (DVFS) is an effective method for matching system power consumption to the required performance.

2. Using MEC offloading, the MD's compute task m is wirelessly transmitted to one of the MEC servers, where it is executed in the MD's stead. With this kind of computation offloading, more effort and resources are needed to send the data used in the calculation as input and as output. The maximum amount of service that the MD can receive from any given MEC server n is finite (CPU cycles per second are the unit of measurement.). The fee for MEC computing services is determined by the MD's contract with its mobile carrier. The MD is also expected to have a foundational knowledge of the typical uplink and downlink data rates before beginning processing on an assigned task (indicated by rUL n and rDL n). Because it is presumed that data transmission and task processing do not overlap or interfere with one another, the uploading, downloading and computing, processes are executed consecutively for the sake of simplicity. For a given decision profile, b={bm,n}∈{0, 1} |M|×|N| describes how long it will take for a batch of tasks to be computed on MEC server n.

$$T_n(Y) = \sum_{m \in M} B_{m,n} \left(\frac{p_m}{\psi_n} \frac{q_m}{r^{UL}{}_n} \frac{r_m}{r^{DL}{}_n} \right) \quad (5)$$

Energy used by the MD for wireless transmission can be determined as

$$E_{MEC}(B) = s_{tx} \sum_{m \in M} \sum_{n \in N} B_{m,n} \frac{q_m}{r^{UL}{}_n} + s_{rx} \sum_{m \in M} \sum_{n \in N} B_{m,n} \frac{r_m}{r^{DL}{}_n} \quad (6)$$

where s_{tx} the output power and s_{rx} the input power are indicated.

3.2 Problem formulation

In this project, we aim to lessen the time it takes to complete tasks and the amount of power needed by the MD. Considering the interdependence of MD and MEC servers, the time metric can be determined as follows:

$$T(a, b, \psi_{MD}) = max\{T_{MD}(a, \psi_{MD}, maxT_n(B)\} \quad (7)$$

One way to quantify energy use is to use:

$$E(a, b, \psi_{MD}) = \{E_{MD}(a, \psi_{MD}, E_{MEC}(B)\} \quad (8)$$

However, because xm, {ym, n} and ψMD link these two goals together, they cannot be improved separately or simultaneously. In order to do this, we develop a single objective function (or system utility).

$$U(a, b, \psi_{MD}) = \xi_T T(a, b, \psi_{MD}) + \xi_E E(a, b, \psi_{MD}) \quad (9)$$

where $\xi_T, \xi_E \in [0, 1]$ represents the weight that the MD places on the execution time and energy usage parameters. The decision can be made taking into account both the latency and energy metrics., with ξ_T and ξ_E reflecting their relative importance. If the MD is using a latency-sensitive application, it can adjust the values of $\xi_T \rightarrow 1$ and $\xi_E \rightarrow 1$ accordingly. When the MD's power is low, it will prioritise conserving energy, so it will adjust $\xi_T \rightarrow 1$ and $\xi_E \rightarrow 1$ accordingly. In order to model similar multi objective optimization problems, the weighted sum method has been widely employed.

As a final step, we formulate the optimization problem as

$var : \psi_{MD} \in \Psi,$

s.t. Limitation (2),

$a_m, \mathrm{b}_{m,n} \in \{0, 1\}, \forall m \in M, \forall n \in N.$

P1 is NP-hard because it requires solving a mixed-integer nonlinear programming problem. The problems of MINLP are a type of optimisation issue in which some variables can only have integer values and others can have continuous values. Furthermore, the optimal solution to this combinatorial optimization problem involves choices for both isolated computational tasks and the MD's central processing unit. Given the difficulty in computing the exact optimal solution, we propose employing the Markov approximation framework to develop a fast polynomial-approximated method for solving problem P1. The fast polynomial-approximated method is a technique used to efficiently compute mathematical functions such as trigonometric, exponential, and logarithmic functions. The method approximates the function using a polynomial, which is then evaluated using a specialized algorithm that reduces the number of arithmetic operations required. The basic idea behind FPA is to approximate the PI function with a low degree polynomial that can be solved using standard optimisation techniques.

3.3 Huffman encoding

These Huffman codes are built and stored in a tree structure using a greedy algorithm, which results in prefix codes with optimal average decode-length. The following lemma characterises Huffman encoding. A greedy algorithm is a technique of answering issues which determines the best option available at the time. It doesn't care whether the current best outcome leads to the overall best outcome. This algorithm is based on the principle of making the best decision possible at each step without regards for future consequences or general optimization.

Lemma 1 *The entropy of a set of characters, P, is denoted by* $H(M) \leq CL(M) \leq H(M) + 1$, *the average code length of Huffman codes being* $CL(M)$, *and the frequency distribution of those codes being* $H(M)$.

3.3.1 Encoding using Run-length Huffman (RLH)

Code length and entropy In RLH, we use Run-length encoding (RLE) to determine the distribution of characters. The size of a Run-length Huffman (RLH) code depends on several factors, including the input data and the encoding scheme used. The RLH encoding scheme is used to optimise the encoding scheme and achieve higher compression ratios by analyzing the frequency of occurrence of runs of repeated values. This means that the distribution of PRLH characters is a derivative of the distribution of P. The RLH entropy is denoted by the letter $H(M_{RLH}) \leq H(M)$. When coupled with the first lemma, we get:

$$CL(M_{RLH}) \leq H(M_{RLH}) + 1 \leq H(M) + 1 \leq CL(M) \quad (10)$$

As a matter of fact, the code is significantly shorter when put to use. Length-wise, RLH codes tend to be more substantial than Elias'.

All encoding techniques theoretically require at least $O(t)$ time for $Q(g) \in \mathbb{R}^t$. Iterative learning hierarchies (ILCs) are preferable to RLH because a second pass over $Q(g)$ is not required prior to code assignment. Here we propose a sampling method that, while increasing average code-length slightly, helps mitigate the additional burden RLH imposes. It employs a run-length compression method similar to that of RLH. The Run-length Huffman (RLH) encoding and the run-length compression method are two approaches that efficiently compress data containing long runs of repeated values. In the run-length compression method, consecutive repeated values are substituted with a count of the number of times they appear in a row. Similarly, RLH encoding compresses data with long runs of repeated values by grouping consecutive repeated values into runs and assigning a variable-length code to each run.

3.3.2 SH encoding (Sample Huffman)

Entropy and code length Let $M = \{m_1 m_2 m_3, .., m_s\}$ denote the original quantized gradient's probability distribution and

$M' = \{m'_1 m'_2 m_3', \ldots m'_s\}$ the sample size S. The entropy and average code-length of SH are then as follows:

$$H(M') = -\sum_{i=1}^{s} m'_i log(m'_i) \tag{11}$$

$$CL(M') = -\sum_{i=1}^{s} m'_i l_i, \tag{12}$$

where $\{l_1, l_2, \ldots, l_k\}$ denotes the character code word lengths with distribution M'. We get $CL(M') \leq H(M') + 1$ from Lemma 1 because we're using Huffman coding on M'. We have $|S| \to T$ and $\left|m'_i\right| \to m_i$, so we have $H(M') \to H(M)$ and thus $CL(M') \to CL(M)$, which is the average Huffman code-length.

Huffman coding is inefficient for sparse vectors because it encodes each character. Developing a sparse encoding method is essential. Sparse encoding is a crucial technique in Huffman coding as it allows for the efficient encoding of symbols that appear less frequently in the input data. In Huffman coding, each symbol is allocated a variable-length code based on its frequency of occurrence, with more frequent symbols allocated shorter codes and less frequent symbols assigned longer codes.

3.3.3 Huffman Sample with Minimal Information (SHS)

Entropy and the Best Possible Code The probability distribution of SHS $s + \gamma$ characters of length cap γ is given for a set $s + 1$ of distribution of characters $M_\gamma = \{m_1, m_2, m_{3,\ldots} m_s\}$, where P is the likelihood that the character will actually exist.

$$M_\gamma = \{m_1, m_2, m_{3,\ldots} m_s\} \cup \{m^i(1-m) | i \in [1|\gamma - 1]\} \cup \{m^\gamma\} \tag{13}$$

Therefore, the entropy is

$$H(m_\gamma) = -m^\gamma \ log(m^\gamma) - \sum_{i=1}^{s} m_i \ log(m_i) - \sum_{i=1}^{\gamma-1} (1-m) m^i \ log(m^i(1-m)) \tag{14}$$

Given the status of $\gamma \to \infty$, we obtain

$$H(M_\infty) = -\sum_{i=1}^{s} m_i \ log(m_i) - m \ log(1-m) - m\frac{log(m)}{1-m} \tag{15}$$

When $\gamma \to \infty$, the best possible SHS codes are generated. The following lemma is derived from the preceding definitions.

Lemma 2 *Specifically, we have for SHS with length cap* $\gamma > 0$

$$H(M_\gamma) = H(M_\infty) + (m^{\gamma+1} logm)/(1-m) + m^\gamma log(1-m) \tag{16}$$

to ensure that the average SHS code length achieved by a length cap is within $\Delta + 1$ *bits of* $CL(M_\infty)$.

Lemma 3 *We have SHS with length can* $\gamma > 0$.

$$H(m_\gamma) + 1 = H(M_\infty) + \Delta + 1 \leq CL(M_\infty) + \Delta + 1 \tag{17}$$

Average and length-constrained character set entropy The length of Huffman codes are denoted by $CL(M_\gamma)$ and $H(M_\gamma)$, respectively; and

$$\Delta = M^\gamma log(1-m) + (m^{\gamma+1} logm))/(1-m). \tag{18}$$

3.3.4 Time complexity analysis

Parallel implementation This study's encoders can be parallelized. If we have n workers, we can assign one to each of the n sub vectors that make up the gradient for RLH, and then calculate the frequencies of those sub vectors. We reduce the frequency of each partitioned sequence's length while enhancing the frequency of the merged size by one when dealing with partitioned sequences. If a run length sequence is split up into multiple parts, we can merge them together once they're finished. Let c_i^e represent the total length of the run of characters i^{th} at the termination of the i_{i+1}^b partition for all$i \in [0, n-1]$. It is assumed for all h that the run length order of the character $(i+1)^{th}$ at the start of the $i \in [1, n]$ divider has lengthc_{i+1}^b. The l_i^e, i_{i+1}^b frequency is lowered by one and the K sequence is raised by one to represent the character$c_i^e = c_{i+1}^b$. $l_i^e + i_{i+1}^b$ seconds are needed to locate the frequencies. It is possible to sample SH and SHS in n subsets over $O(d/n)$ time. Huffman tree construction for s characters takes log(s) time when done in parallel. The final tree has the letters (d'/n) for RLH, $O(logk)$ for SH, and $O(log(s+\gamma))$ for SHS. The highly parallel nature of accelerators like GPUs and FPGAs can be exploited by running these algorithms in parallel.

3.4 Markov approximation and algorithm design

Combinatorial network optimization problems can be solved using the Markov approximation, a method that was very recently proposed. The Markov approximation method is a mathematical method for simplifying complicated stochastic processes. The Markov approximation is utilised in a number of fields, including financing, biology, physics, and technology, to model various stochastic processes such as particle behaviour in a gas, fluctuations in stock prices in financial markets, and the spread of infectious illnesses.

The Markov approximation technique is an effective strategy for solving combinatorial network optimization problems that assume network traffic can be represented as a Markov process. Log-sumexp approximation and the construction of Markov chains tailored to a

specific problem yield efficient parallel implementations for approximating problem resolution in this context. It has been shown that the Markov approximation is both optimal and converging.

3.4.1 Approximation using the logarithm of the squared expense

Let f = {a, b, ψMD} ∈ F represent a workable answer to problem S1, with F representing the complete set of answers that can be found within the given constraints. Problem S1 is represented by $\min_{f\in F} W_f$ if we define utility W_f as the objective function of the system for a specific configuration f. Hence, S1's MWIS problem is equivalent to the following:

$$\min_{s\geq 0} = \sum_{f\in F} s_f W_f \tag{19}$$

$$s.t \sum_{f\in F} s_f = 1 \tag{20}$$

where pf is the probability that state f occurs and f is the percentage of time that state f occurs. The challenge here is to find the lightest possible weighted configuration, with W_f standing in for 'f's's relative importance. The log-sum-exp approximation of $\min_{f\in F} W_f$, using the Markov approximation framework, provides

$$W_{min} \approx -\frac{1}{\beta} log\left(\sum_{f\in F} exp(-\beta W_x)\right) \tag{21}$$

where β is a positive constant influencing the accuracy of the approximation. The following information identifies the approximation's accuracy: where |F| is the size of the set F:

$$\min_{f\in F} W_f - \frac{1}{\beta} log|F| \leq -\frac{1}{\beta} log\left(\sum_{f\in F} \exp\left(-\beta W_x\right)\right) \leq \min_{f\in F} W_f \tag{22}$$

The approximation gap decreases toward zero as $\beta \to \infty$ increases, and eventually reaches unity. As shown in, the following problem can be solved using the log-sum-exp approximation in (13) P2

$$\begin{aligned} &S2: \min_{p\geq 0} \sum_{f\in F} p_f W_f + \tfrac{1}{\beta} \sum_{f\in F} p_f \ log \ p_f \\ &s.t \sum_{f\in F} p_f = 1 \end{aligned} \tag{23}$$

By satisfying the Karush–Kuhn–Tucker (KKT) conditions, we can find the best answer to S2 since it is a convex problem.

$$s_f{}^* = \frac{exp\left(-\beta W_f\right)}{\sum_{f'\in F} \exp\left(-\beta W_{f'}\right)}, \forall f \in F \tag{24}$$

However, due to the vastness of the problem's solution space, it is unusual for someone to have complete knowledge of F, which is required for a direct solution of S2. Taking turns among various set ups in accordance with their subsections Taking a random sample from the distribution s * f in the configuration space F is a viable strategy for addressing both Problems S2 and S1 (24). It is challenging to design a Markov chain that is specific to a given problem, achieves stationary distribution s * f, and permits concurrent construction of the |M| tasks. one of the most difficult aspects is determining an appropriate set of states that precisely reflects the system under consideration.

3.5 Classification of deep belief network

3.5.1 Restricted Boltzmann Machine (RBN)

Two-layer stochastic networks are referred to as RBM in abbreviation. The two-layer stochastic networks are significant because they are capable of learning complex nonlinear relationships within high-dimensional data and can generalize effectively to new data. The visible layer is vv and the concealed layer is h. There are four nodes on the visible layer and three on the hidden layer of the Restricted Belief network seen in Fig. 2. The RBM network differs from the Boltzmann network in that nodes in a single layer have any impact on each other. RBMs are commonly used in reduction of dimensionality, learning features, and shared filtering, whereas BMs are used to model complex data distributions such as image and speech data.

A random value of 1 or 0 is given to each node in the RBM's evolution, based on the following posterior probability:

From a macro perspective, an RBM can be described in terms of its energy function and its probability distribution.

$$\begin{aligned} p\left(h_i = 1|v\right) &= f\left(b_i + w_i v\right) \\ p\left(v_i = 1/h\right) &= f\left(a_i + W_I h\right) \end{aligned} \tag{25}$$

Energy Function:

$$E(v,h) = -\sum\nolimits_{i\in v} a_i v_i - \sum\nolimits_{j\in h} b_j h_j - \sum\nolimits_{i,j} v_i h_j w_{ij} \tag{26}$$

Probability Distribution:

$$p(v,h) = \frac{1}{z} e^{-E(v,h)} \tag{27}$$

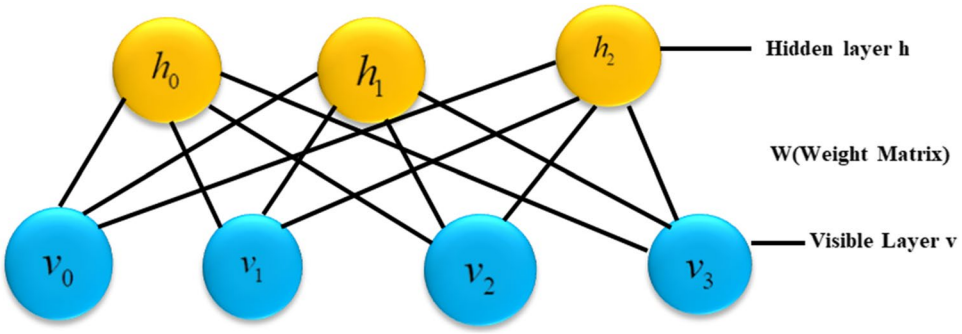

Fig. 2 A example of restricted Boltzmann machine

The edge distribution of the visible layer can be calculated using this method.

$$P(v) = \frac{1}{z}\sum_h e^{-E(v,h)} \tag{28}$$

(1) Applying the gradient method, we take $log\ p(v,\theta)$ to be the likelihood function, which does not alter the monotonicity. To improve our learning efficiency, we make adjustments to the parameters along the gradient $\frac{\partial\ log\ p(v,\theta)}{\partial\theta}$. Here are the specifics of the plan:

$$\begin{gathered}\theta(n+1) = \theta(n) + a * \left(\frac{\partial\ \log\ p(v,\theta)}{\partial\theta}\right),\ \theta \in \{W,a,b\} \\ -\frac{\partial\ \log\ p(v,W_{ij})}{\partial W_{ij}} = E_v\left[p\left(h_i|v\right)\right] - v_j^{(i)} * f\left(w_i * v^{(i)} + b_i\right) \\ -\frac{\partial\ \log\ (v,b_i)}{\partial b_i} = E_v\left[p\left(h_i|v\right) * v_j\right] - f\left(w_i * v^{(i)}\right) \\ -\frac{\partial\ \log\ p(v,a_j)}{\partial a_j} = E_v\left[p\left(h_i|v\right) * v_j\right] - v_j^{(i)}\end{gathered} \tag{29}$$

(2) Hinton proposes the Contrastive Divergence: CD algorithm in 2002. It's a quick way to learn RBM, and the process is laid out in the text.

4 Result and analysis

The presented system model and algorithm will be used as the basis for the numerical experiments in this section. All of the algorithms and tests are executed on a Windows 10 machine with 8 GB of RAM and MATLAB 2021a. Edge servers, located in the geographic centre of the service region, are part of the simulation scenario alongside a few mobile endpoints. There is a completely random distribution of mobile devices throughout the coverage area. Please take note that the task requirements, including the data size and required CPU cycles, are generated at random, making the assignment specific to each mobile device (400, 100). Computing power (F) of the edge servers is 40 GHz, while that of the mobile device is only 0.5–1 GHz. We have Po I set to transmit data at 100 mW and Pe I waiting for the result at 10 mW. The base price for the edge server is 1, and the baseline charging resource is 1 GHz Table 1.

4.1 Average energy consumption

Average Energy consumption $\mu(X) = \frac{\sum(xi-\mu)^2}{N-1}$

$\mu = \text{mean}(x)$

Table 1 and Fig. 3 compare the average amount of energy used by the DBN-MAA method to that of other approaches. Compared to other methods, the suggested method uses a very small amount of energy. For example, at the 10th node, the DBN-MAA method consumes only 2.134 J while the other techniques like AFCFS, MINET, TRADEOFF, and IGTMA consume 7.527 J, 6.489 J, 5.489 J, and 3.946 J, respectively. Similarly, at the 70th node, the DBN-MAA method consumes only 3.578 J while the other techniques like AFCFS, MINET, TRADEOFF, and IGTMA consume 8.108 J, 7.487 J, 6.319 J, and 5.298 J of energy, respectively. This proves that compared to other methods, the suggested method uses a very small amount of energy and shows higher performance.

4.2 System utility

The utility of the DBN-MAA method is compared to certain other methodologies in Table 2 and Fig. 4. The DBN-MAA algorithm incorporates the features of two influential algorithms: deep belief

Table 1 Average energy consumption of DBN-MAA Method with existing system

No of Nodes	AFCFS	MINET	TRADE-OFF	IGTMA	DBN-MAA
10	7.527	6.489	5.489	3.946	2.134
20	7.628	6.673	5.618	4.247	2.478
30	7.727	6.827	5.829	4.578	2.687
40	7.826	6.926	5.916	4.729	2.867
50	7.927	7.102	6.099	4.9176	3.098
60	8.025	7.278	6.278	5.190	3.168
70	8.108	7.487	6.319	5.298	3.578

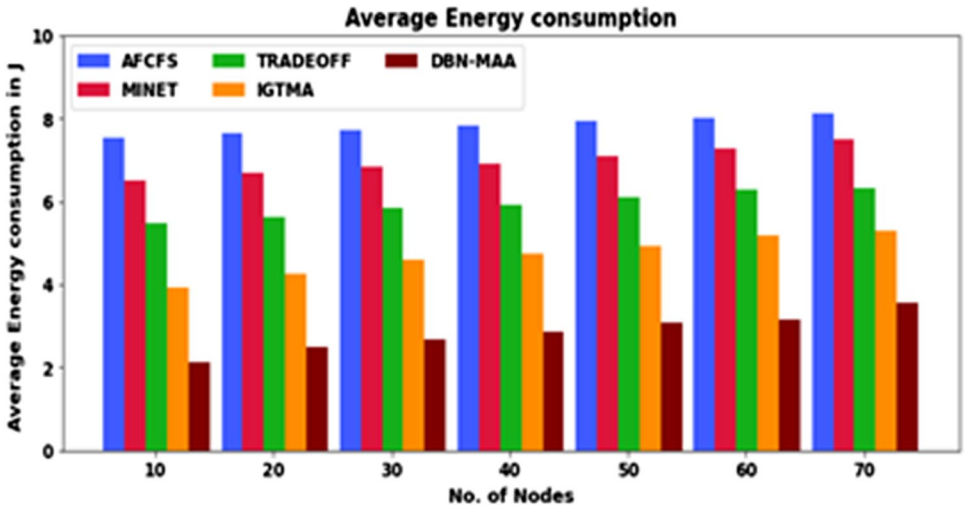

Fig. 3 Average energy consumption for DBN-MAA Method with existing system

Table 2 System utility for DBN-MAA method with existing system

No of iterations	AFCFS	MINET	TRADEOFF	IGTMA	DBN-MAA
100	25.56	31.22	35.29	39.15	44.26
200	26.41	31.62	35.82	40.28	45.28
300	26.85	32.78	36.43	41.38	46.18
400	27.11	33.17	36.75	41.93	47.82
500	27.89	33.82	37.11	42.18	48.56
600	28.45	34.38	37.95	43.28	49.27
700	30.17	34.95	38.74	43. 87	50.17

networks (DBNs) and the Markov approximation algorithm (MAA). DBNs are neural networks that have been developed to extract complex features from high-dimensional data, whereas MAA clarifies complicated stochastic procedures by considering that the future state is solely determined by the current state. When compared to other methods, the suggested method performs well. For example, with 100 data points from a dataset, the DBN-MAA method has a system utility of 44.26% while the other techniques like AFCFS, MINET, TRADEOFF, and IGTMA have 25.56%, 31.22%, 35.29%, and 39.15% of system utility, respectively. Similarly, with 700 data points, the DBN-MAA method has a latency of 50.175% while the other techniques like AFCFS, MINET, TRADEOFF, and IGTMA have 30.175%, 34.95%, 38.74%, and 43.87%, respectively. These findings demonstrate the superiority of the proposed method over its rivals. Time Lag, or Latency, 4.3

The latency comparison of the DBN-MAA method to other existing techniques is described in Table 3 and Fig. 5. The proposed method consumes very little energy compared to the other methods. For example, at the 10th node, the DBN-MAA method has

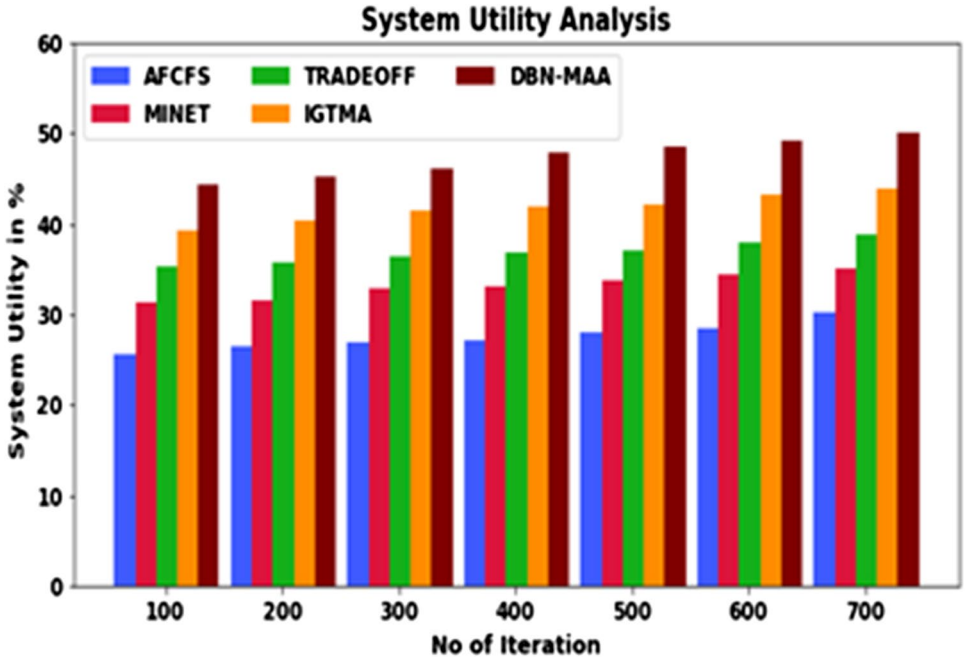

Fig. 4 System utility for DBN-MAA method with existing system

Table 3 Latency Analysis for DBN-MAA Method with existing system (sec)

No of nodes	AFCFS	MINET	TRADEOFF	IGTMA	DBN-MAA
10	123.46	85.36,	42.51	25.42	14.32
20	135.68	88.27	45.68	35.58	20.76
30	147.84	91.62	57.24	40.91	26.27
40	151.35	93.77	60.27	47.98	32.37
50	171.64	110.36	78.29	62.48	45.55
60	179.58	127.56	86.27	68.68	50.38
70	183.22	138.57	91.22	79.78	56.89

Table 4 Throughput Analysis for DBN-MAA Method with existing system(kbps)

No of nodes	AFCFS	MINET	TRADEOFF	IGTMA	DBN-MAA
100	925.79	958.33	992.23	1013.41	1053.67
200	9300.22	965.66	998.79	1019.25	1067.89
300	932.75	971.57	994.24	1023.77	1071.34
400	940.14	979.19	1003.31	1028.31	1079.50
500	943.46	984.75	1004.69	1035.64	1083.39
600	951.68	989.78	1011.69	1043.52	1090.86
700	957.66	993.32	1015.42	1049.69	1118.67

a latency of only 14.32 s while the other techniques like AFCFS, MINET, TRADEOFF, and IGTMA have 123.46 s, 85.36 s, 42.51 s, and 25.42 s of latency, respectively. Similarly, at the 70th node, the DBN-MAA method has a latency of 56.89 s while the other techniques like AFCFS, MINET, TRADEOFF, and IGTMA have 183.22 s, 138.57 s, 91.22 s, and 79.78 s of latency, respectively. This proves that the proposed method has very low latency compared to the other techniques and shows higher performance.

4.3 Throughput

AFCFS, MINET, TRADEOFF, IGTMA, and DBN-MAA methods are evaluated based on the number of nodes for throughput analysis in Table 4 and Fig. 6. When 100 nodes are employed, the DBN-MAA's throughput has been measured as 1053.67kbps. The network throughput of 925.79 Kbps, 958.33 Kbps, 992.23 Kbps, and 1013.41 kbps for AFCFS, MINET, TRADEOFF, and IGTMA methods, respectively. Similarly, at 700 nodes, the DBN-MAA method has a throughput of 1118.67 kbps while it is 957.66 kbps, 993.32 kbps, 1015.42 kbps, and 1049.69 kbps for AFCFS, MINET, TRADEOFF, and IGTMA, respectively. According to the study, DBN-MAA exceeds the other methods in terms of effectiveness (Fig. 7).

4.4 Time complexity

The time complexity of the DBN-MAA model is shown in Fig. 6 and Table 5. Bluetooth is currently used by the majority of smart devices to connect smartphones, and this connection uses far less energy than 3G/LTE or Wi-Fi does. The existing AFCFS, MINET, TRADEOFF, and IGTMA systems take longer time than the proposed DBN-MAA system. For 100 data points from the dataset, the proposed method takes 22.32 s while the other methods like AFCFS, MINET, TRADEOFF, and IGTMA take 39.28 s, 34.51 s,

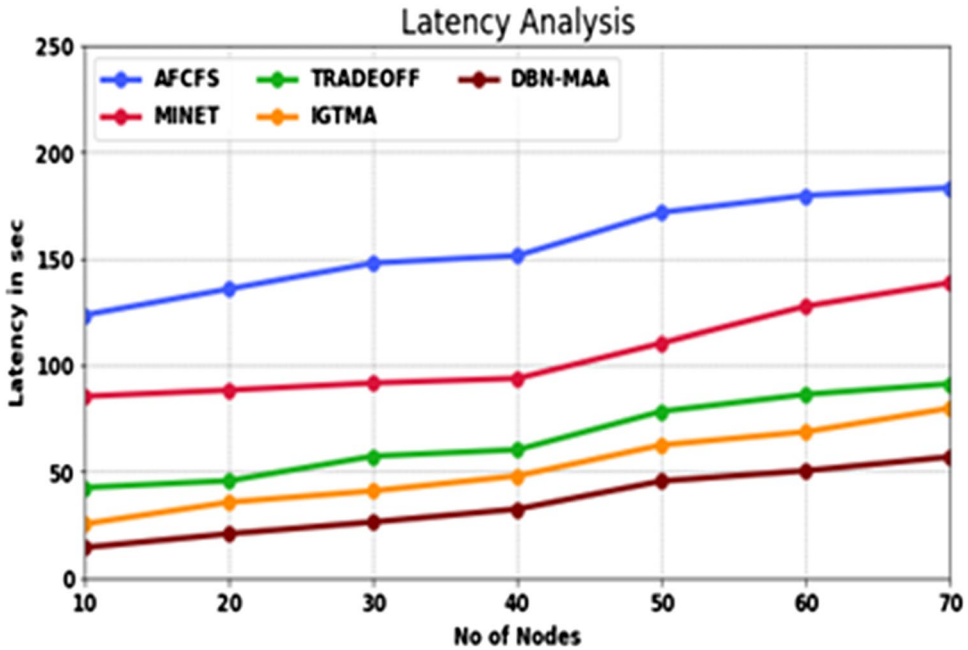

Fig. 5 Latency Analysis for DBN-MAA Method with existing system

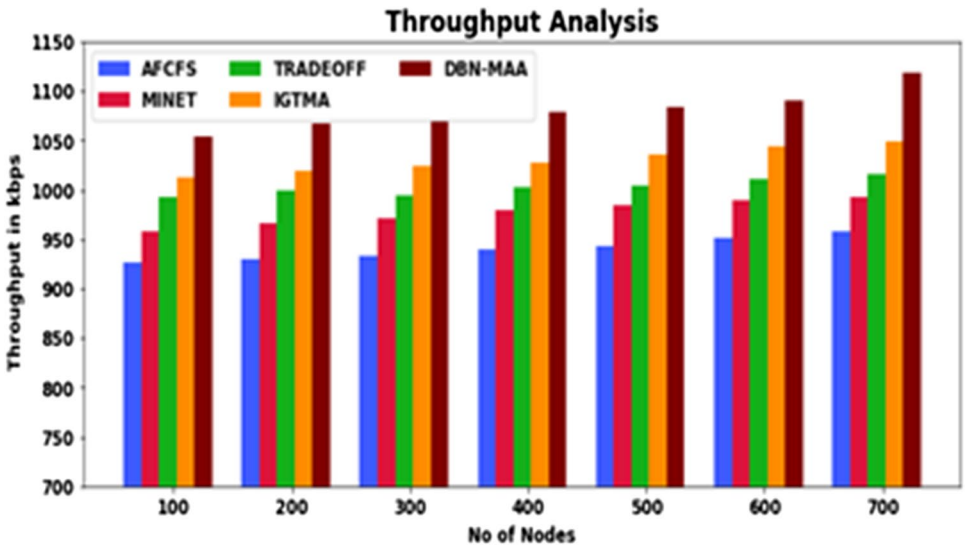

Fig. 6 Throughput Analysis for DBN-MAA Method with existing system

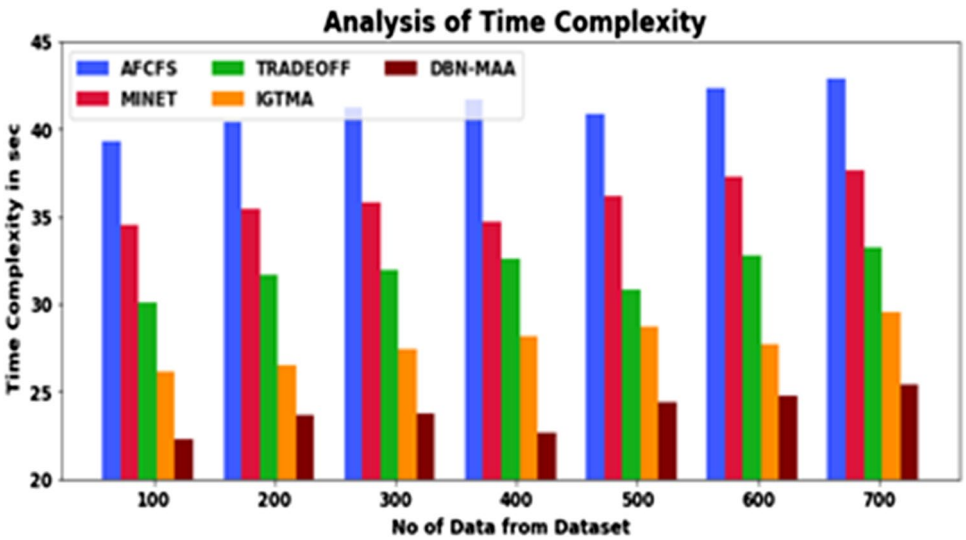

Fig. 7 Time complexity for DBN-MAA Method with existing system

30.11 s, and 26.13 s, respectively. Similarly, for 700 data points, the DBN-MAA method takes only 25.44 s while the other methods like AFCFS, MINET, TRADEOFF, and IGTMA take 42.87 s, 37.68 s, 33.25 s, and 29.53 s, respectively.

Table 5 Time complexity for DBN-MAA Method with existing system

No of data from dataset	AFCFS	MINET	TRADEOFF	IGTMA	DBN-MAA
100	39.28	34.51	30.11	26.13	22.32
200	40.37	35.44	31.63	26.48	23.69
300	41.23	35.79	31.94	27.44	23.75
400	41.68	34.74	32.54	28.15	22.64
500	40.83	36.16	30.85	28.71	24.44
600	42.33	37.25	32.78	27.73	24.79
700	42.87	37.68	33.25	29.53	25.44

4.5 Accuracy

Both Table 6 and Fig. 8 compare the DBN-MAA method's accuracy to that of other popular methods. When compared to alternative approaches, the proposed method performs exceptionally well. When applied to a sample of 100 data points from a dataset,

Table 6 Accuracy analysis for DBN-MAA method with existing system

No of data from dataset	AFCFS	MINET	TRADEOFF	IGTMA	DBN-MAA
100	75.46	79.84	83.63	88.61	92.64
200	76.25	80.73	84.79	89.32	93.46
300	77.15	81.27	85.53	90.63	94.53
400	78.53	82.37	86.85	91.29	95.32
500	79.27	83.18	87.42	92.14	96.19
600	79.54	83.44	87.77	92.51	97.43
700	80.18	84.19	88.28	93.09	97.87

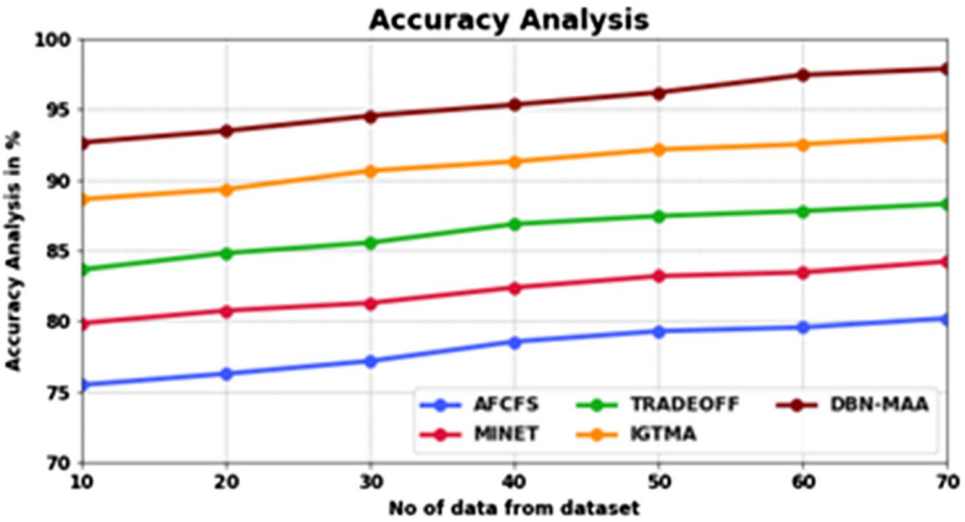

Fig. 8 Accuracy analysis for DBN-MAA method with existing system

the DBN-MAA method achieves an accuracy of 92.64%, while the AFCFS method achieves only 75.46% accuracy, the MINET method achieves only 79.84% accuracy, and the TRADEOFF method achieves only 83.63% accuracy, and the IGTMA method achieves only 88.61% accuracy. Comparatively, the DBN-MAA method has a latency of 97.87% with 700 data points, while the AFCFS method has 80.18%, the MINET method has 84.19%, the TRADEOFF method has 88.28%, and the IGTMA method has 93.09%. This demonstrates that the proposed method is more accurate and efficient than competing methods.

5 Conclusion

The cloud can alleviate task bottlenecks caused by a lack of computing resources on either the local machine or the cloud by offloading tasks to the mobile edge. In this paper, we propose a novel scheduling method (deterministic bias network multi-objective augmentation, or DBN-MAA) to simultaneously optimise multiple objectives by fusing the best aspects of immune-based and traditional scheduling approaches. The findings of this research contribute in three ways. The proposed system model consists of various components such as communication and computing resources, mobile device energy usage, and work weight. In addition, the scenario where a mobile device can create multiple jobs simultaneously is taken into account. And finally, the evolutionary algorithm incorporates both the bat and immune algorithms, so that convergence and diversity are both guaranteed. In this paper, we investigate computation offloading within the framework of mobile edge computing, where a single MD may employ multiple MEC nodes. Reducing the MD's energy consumption and task execution time requires optimal task assignment and frequency scaling. The Markov approximation is used to approximate the optimal solution to an NP-hard problem with a small, bounded error. Our algorithm is nearly optimal in terms of scalability, robustness, and performance, as demonstrated by simulations. The advantage of DBN-MAA is, it reduce image data irrelevance and redundancy in order to store or transmit data in an efficient manner. Its objective is to lower the amount of bits needed to represent an image. It performs better than exhaustive searching and local processing. Many-Markov-approximation (MD) research is planned for the future. In this online demonstration, we will implement our method (DBN-MAA) by simulating the arrival and departure of simulated medical doctors. A real-world mobile edge computing testbed can also be used to assess the work's efficacy. The computation offloading model will be further developed and improved in subsequent work by taking into account more realistic use cases. We'll also look into other potential approaches to offloading optimization beyond multi-objective swarm intelligence algorithms.

Authors' contributions All author is contributed to the design and methodology of this study, the assessment of the outcomes and the writing of the manuscript.

Data availability No datasets were generated or analyzed during the current study.

Code availability Not applicable.

Declarations

Conflicts of interests Authors do not have any conflicts.

References

1. Meng S, Wang Y, Miao Z, Sun K (2018) Joint optimization of wireless bandwidth and computing resource in cloudlet-based mobile cloud computing environment. Peer-to-Peer Netw Appl 11:462–472
2. Dinh HT, Lee C, Niyato D, Wang P (2013) A survey of mobile cloud computing: Architecture, applications, and approaches. Wirel Commun Mob Comput 13:1587–1611
3. Yang G, Hou L, He X, He D, Chan S, Guizani M (2021) Offloading time optimization via Markov decision process in mobile-edge computing. IEEE Internet Things J 8:2483–2493

4. Li Z, Chang V, Ge J, Pan L, Hu H, Huang B (2021) Energy-aware task offloading with deadline constraint in mobile edge computing. EURASIP J Wirel Commun Netw 2021:32569–32581
5. Bi J, Yuan H, Duanmu S, Zhou M, Abusorrah A (2021) Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization. IEEE Internet Things J 8:3774–3785
6. Hmimz Y, Chanyour T, Ghmary ME, Malki MOC (2021) Bi-objective optimization for multi-task offloading in latency and radio resources constrained mobile edge computing networks. Multimed Tools Appl 80:17129–17166
7. Mazouzi H, Boussetta K, Achir N (2019) maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud: a theoretical and an experimental study. Comput Commun 144:132–148
8. Zhang JM, Yang FY, Wu ZY (2019) Multi-access Edge Computing (MEC) and Key Technologies. Post & Telecom Press, Beijing
9. Wu S, Xia W, Cui W et al (2018) An efficient offloading algorithm based on Support vector machine for mobile edge computing in vehicular networks. In Proceedings of the 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6, Hangzhou, China
10. You C, Huang K, Chae H, Kim BH (2017) Energy-efficient resource allocation for mobile-edge computing offloading. IEEE Trans Wireless Commun 16(3):1397–1411
11. Cao X, Wang F, Xu J, Zhang R, Cui S (2016) Joint computation and communication cooperation for mobile edge computing. 2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt). IEEE, 2018
12. Sardellitti S, Scutari G, Barbarossa S (2015) Joint optimization of radio and computational resources for multicell mobile edge computing. IEEE Trans Signal Inf Process Over Netw 1(2):89–103
13. Jang Y, Na J, Jeong S, Kang J (2020) Energy-Efficient Task Offloading for Vehicular Edge Computing: Joint Optimization of Offloading and Bit Allocation. In: 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring). pp 1–5
14. Deng S, Huang L, Taheri J, Zomaya AY (2015) Computation offloading for service workflow in mobile cloud computing. IEEE Trans Parallel Distrib Syst 26(12):3317–3329
15. Ale L, Zhang N, Fang X, Chen X, Wu S, Li L (2021) Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. IEEE Trans Cogn Commun Netw 7(3):881–892
16. Anajemba JH, Yue T, Iwendi C, Alenezi M, Mittal M (2020) Optimal cooperative offloading scheme for energy efficient multi-access edge computation. IEEE Access 8:53931–53941
17. Li B, Pei Y, Wu H, Shen B (2015) Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds. J Supercomput 71(8):3009–3036
18. Sun J, Gu Q, Zheng T (2019) Joint communication and computing resource allocation in vehicular edge computing. Int J Distrib Sens Netw 15(3):1–13
19. Hao Y, Liu G (2015) Evaluation of nine heuristic algorithms with data-intensive jobs and computing-intensive jobs in a dynamic environment. IET Softw 9(1):7–16
20. Tao Y, Zhang Y, Ji Y (2015) Efficient computation offloading strategies for mobile cloud computing. Proc - Int Conf Adv Inf Netw Appl AINA 2015-April:626–633. https://doi.org/10.1109/AINA.2015.246
21. Wang F, Xu J, Wang X, Cui S (2016) Joint offloading, and computing optimization in wireless powered mobile-edge computing system. In: Proceeding of the 2016 IEEE ICC, IEEE, Paris, pp.1–6
22. Chen X, Jiao L, Li W, Fu X (2016) Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw 24(5):2795–2808
23. Jain DK, Liu X, Neelakandan S, Prakash M (2022) Modeling of human action recognition using hyperparameter tuned deep learning model. J Electron Imaging 32(1):011211. https://doi.org/10.1117/1.JEI.32.1.011211
24. Sreekala K, Cyril CPD, Chandrasekaran S, Walia R, Martinson EO Capsule network-based deep transfer learning model for face recognition. Wirel Commun Mob Comput 2022. https://doi.org/10.1155/2022/2086613
25. Sunitha G, Geetha K, Pundir AKS, Hemalatha S, Kumar V (2022) Intelligent deep learning-based ethnicity recognition and classification using facial images. Image Vis Comput 121. https://doi.org/10.1016/j.imavis.2022.104404
26. Wu G, Chen J, Bao W, Zhu X, Xiao W, Wang J (2017) Towards collaborative storage scheduling using alternating direction method of multipliers for mobile edge cloud. J Syst Softw 134:29–43
27. Pu L, Chen X, Mao G, Xie Q, Xu J (2019) Chimera: an energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications. IEEE Internet Things J 6(1):84–99
28. Fakhri ZH, Khan M, Sabir F, Al-Raweshidy HS (2018) A resource allocation mechanism for cloud radio access network based on cell differentiation and integration concept. IEEE Trans Netw Sci Eng 5(4):261–275
29. Tong Z, Deng XM, Chen HJ, Mei J (2021) DDMTS: a novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing. J Parallel Distrib Comput 149:138–148
30. Sethukarasi T, Prakash M, Baburaj E (2023) An Efficient Hybrid Job Scheduling Optimization (EHJSO) approach to enhance resource search using Cuckoo and Grey Wolf Job Optimization for cloud environment. PLOS ONE 18(3):e0282600. https://doi.org/10.1371/journal.pone.0282600
31. Ezhumalai P et al (2023) Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing. J Cloud Comp 12:24. https://doi.org/10.1186/s13677-023-00401-1
32. Li K (2021) Heuristic computation offloading algorithms for mobile users in fog computing. ACM Trans Embed Comput Syst 20:1–28
33. Wu Y, Cao J, Li Q, Alsaedi A, Alsaadi FE (2017) Finite-time synchronization of uncertain coupled switched neural networks under asynchronous switching. Neural Netw 85:128–139