A Distributed Deep Meta Learning based Task Offloading Framework for Smart City Internet of Things with Edge-Cloud Computing

Manvitha Gali[1*] and Aditya Mahamkali[2]

[1*]Lead Senior Software Engineer, Verizon, USA. manvithagalicloud@gmail.com

[2]Lead Senior Software Engineer, Goldman Sachs, USA. maditya6181@gmail.com

Received: August 17, 2022; Accepted: October 11, 2022; Published: November 30, 2022

Abstract

IoT (Internet of Things) and cloud computing are essential components of constructing smart cities and provide multiple smart services to end consumers. Because IoT devices are data-intensive and resource-constrained, using edge computing technologies might provide considerable benefits to the smart environment. However, heterogeneous clouds where ECSs (Edge Cloud Systems) and centralized clouds interact for satisfying demands of IoT applications, challenges in task offloading exist. When IoT system's environments change, such as the edge server's performances or the bandwidth, solutions based on DLTs (Deep Learning Techniques) must train from scratch. A meta-algorithm known as DDMTO (Distributed Deep Meta learning-driven Task Offloading) is presented to solve the issue of poor portability and ensure that DNNs (Deep Neural Networks) are utilised to make offloading decisions effectively and efficiently. These networks have their output which receives inputs from hidden layers in BP algorithm compute outputs. Inputs are compared to desired outputs and errors are traced from outputs to hidden layers and from hidden to input layers based on disparities. When flows are restored, neuron weights get altered. Epochs are cycles that traverse from inputs to outputs and backwards from outputs to inputs. Previously known inputs are fed into NNs (neural networks) which then generate known outputs called network training. Existing offloading systems ignore heterogeneous cloud co-operations which is overcome for providing better performances while significantly reducing computing complexities.

Keywords: Internet of Things, bandwidth, Edge Computing, Task Offloading, Deep Neural Network, Meta Learning and Offloading Strategy.

1 Introduction

In recent years, both counts of IoT devices connected to the Internet and the data these devices generate have significantly increased. Over 50 billion IoT devices are anticipated to be connected to the internet in the next years [1], reflecting the recent fast growth of the information technology sector. Furthermore, the availability of dependable, fast internet as well as communication technologies have been an encouraging factor in the creation of complicated and computationally intensive IoT applications that routinely collect and analyse massive amounts of data where internet gaming, augmented reality and video streaming are examples [2]. This massive expansion necessitates systems

*Corresponding author: Lead Senior Software Engineer, Verizon, USA.

to handle the expanded counts of IoT devices as well as organise and interpret generated data. However, reduced energies and computational capabilities of these devices (i.e., CPU and RAM) limit their capacities to execute resource demanding applications [3]. To circumvent these limitations and overcome communication/processing delay requirements, complex computations can be offloaded to more resourceful devices.

Cloud computing are viable strategies for application expansions since they provide on-demand accesses to large pools of computational resources used in service processes and data analytics [4]. Cloud computing resources which are centralized and remote from IoT devices demanding quick analyses of massive volumes of data generated by IoT sensors, but turn insufficient for handling low-latencies, real-time interactions, and high QoS (Quality of Service) applications [5].

Constraints exist in cloud computing which provide a pool of virtually operated processing and storage resources at network's edges that are close to IoT devices and bridge gaps in latencies [6]. Offloading IoT duties are a necessity to free resources for resource rich nodes like ECs. Additionally, they support distribution of topmost streaming services like Google Stadia and Netflix which are latency sensitive and bandwidth intensive. Additionally, edge computing architecture minimises the pre-processing of offloading activities and massive file uploading and downloading, which contributes to a reduction in overall service time [7]. However, efficiently managing computing workloads of latency sensitive applications and edge-cloud computing resources throw up substantial challenges [8]. These may entail offering efficient task schedules to improve overall service performances and reduce offloaded work delays.

The suggestions of practical scheduling models for operation offloads on ECs are also impacted by many factors. Infrastructure features and application characteristics make up the two groups of these criteria. The first category, which is titled "Infrastructure Characteristics," is concerned with infrastructure-related issues including choosing the best resource for a task, controlling utilizations of ECs and network’s conditions. For example, usage of CPUs may fluctuate based on activities and increasing IoT devices on shared networks causing variations in network bandwidths. The second category governs IoT application task aspects such as compute demand, needed data transfer for uploading and downloading, and the required deadline to finish tasks (application characteristics).

Based on the aforesaid descriptions, realistic models for scheduling activity offloads on ECs account for variety of parameters including resource heterogeneities/utilizations, computations, communications, and latencies. These factors which vary over a period of time can be described as dynamic multi-objective optimization issues [9]. Due to the complexities, ambiguities, and vagueness [10] of environments, it is difficult to implement effective scheduling approaches and obtain reliable mathematical models. For instance, a single region's IoT device density may increase or decrease as a result of IoT device mobility, influencing workloads on edge nodes and shared networks. Moreover, incoming tasks are unknown in advance, necessitating the use of a real-time system to manage them. Furthermore, the edge-cloud ecosystems are made up of a diverse collection of resources (e.g., different computation resource capabilities).

Several research projects launched in this domain to address latency concerns and resource unpredictability in edge clouds. For example, the study in [11], researchers investigated computing and communication factors in order to decrease total latencies. Similarly, the study [12] examined influences of resource heterogeneities in improving end-to-end service times in the context of ECSs [13] and focus on load balancing and server utilisations to reduce overcrowded edge nodes, which impair application service times. Most studies however target single situations or applications which make it less flexible and scalable [25]. Furthermore, this type of problem, where edge resources impose computational constraints is not addressed due to the complexity and processing time involved in applications. Majority of suggested solutions also rely on conventional approaches which assume that all optimization model's parameters are precisely known and thus may result in additional overheads at edge nodes making it more difficult to satisfy demands of service requirements for latency sensitive applications [14].

The work aims to offer a unique strategy for offloading work from latency sensitive IoT applications while maximizing resource utilisations in Edge Clouds. The proposed technique is also employed for scheduling task offloads for reducing total service times and maximizing resource utilisations. The contributions of this work are detailed below:

- Present an architecture for Edge Clouds based on DLTs that encompass necessary components to support scheduling of IoT application task offloads;
- To reduce characteristics of latency-sensitive application including CPU energies, executions, network demands, delays in addition to resource utilisations, heterogeneities, and total times, DDMTO, a novel approach that uses DNNs (Deep Neural Networks) with meat algorithm is presented
- A collection of techniques is presented for offloading work scheduling in ECSs to enhance service time and resource consumption.

This study is divided into four major portions. Section 2 discusses the research's background and purpose, as well as the conclusion. Section 3 goes into the materials and processes in great depth. The experimental findings on two benchmark datasets are shown in Section 4. Finally, Section 5 summarises this study with recommendations for further research.

2 Related Work

A thorough analysis of recent research in the area of compute offloading is provided in this section. ACO (Ant-Colony-Optimization) based computation offloading system proposed by Huang et al. [15] includes the two algorithms EA-OMIP and EA-RMIP. This section examined the performance of ACO approaches paired with mixed integer programming to achieve this goal (MIP). Model-based offloading methods either ignored delay and/or security constraints or overlooked energy usage.

Based on MO-BFO that was inspired by nature, Babar et al. [16] created a compute offloading method for load management across edge servers (multi-objective bacterial foraging optimization). The comprehensive findings reveal that, as compared to its competitors, the suggested method minimises reaction time, communication costs, and provides optimal load control. Obtaining the necessary QoS is, nevertheless, a difficult task.

In an IoT-edge-cloud network, long et al. [17] describe a mobility vehicle-based computation offloading technique. Sensing devices generate tasks and communicate them to automobiles, which decide whether to process them locally on the vehicle, remotely on MEC servers, or in the clouds. DLTs use reinforcement learning approaches to make judgments about computational offloads based on utility functions of energy usage and transmission latencies. However, many automobiles can interface with sensing devices in efficient and cost-effective ways.

Mu and Zhong's [18] novel cooperative computing approach fully utilized IoT device processing capabilities and edge cloud facilities where the study’s collaborators assist in computing offloads during idle and busy periods. The study treated the issue as a Markov decision issue with the goal of minimizing predicted energy consumptions of IoT devices and collaborators while adhering to computational deadlines. Most strategies, however, fail to take into account time varying characteristics of channels in computational offloads which restrict applications.

Mobile devices can't execute mobile applications because of their constrained resources. In order to provide mobile apps in edge/cloud networks, Shahidinejad and Ghobaei-Arani [19] suggested a metaheuristic-based job offloading mechanism called iNSGA-II that leverages the NSGA-II (non-dominated sorting genetic algorithm) technique. They determined task offloading to be an NP-hard issue. The crossover and mutation operators are also enhanced in this section, resulting in a faster convergence of the proposed technique than previous evolutionary algorithms. When compared to metaheuristic-based job offload solutions, the proposed method was found to be cost effective strategy that boosted average utilisation of ECSs while lowering energy consumptions and execution times in numerical results of the study’s simulated workloads.

Huang et al. [20] created a markov decision procedure based on fine-grained offloading in order to determine the best course of action for enhancing application reaction time while reducing energy consumption. Additionally, LFGO, a learning-aided fine-grained offloading for real-time applications was proposed in the study built using q-learning DLTs in edge-cloud enabled IoV to make offloads more adaptable to different application sizes, the efficacy of LFGO was then validated by DAGs (directed acyclic graphs) based on real-world applications which included a variety of sub-tasks, transmission rates, and processing capabilities. However, due to diversities of DAG based applications and complexities of dynamic environments of ECSs, vehicle intelligent management systems find it difficult to efficiently schedule offloads resulting in higher transmission latencies and energy expenditures on wireless channels and backhaul links.

Jayaram and Prabakaran [21] executed real time secure processing with data offloads amongst edge nodes in healthcare patient record processing. Their Parkinson disease prediction model was based on the AFO-k-NN (Adaptive Fuzzy Optimized k-Nearest Neighbor) classifier where their proposed system provided additional diagnoses and rehabilitations based on the severity. The demand for real-time healthcare applications have raised requirements for higher calibre expertise to do jobs including sickness predictions and low latencies in computing processes.

Peng et al. [22] modelled multi-objective optimizations for resource utilisations of ECSs as well as mobile user times and energy usages. The study’s end-edge-cloud collaborative computing offloading approach was based on upgraded Strength Pareto Evolutionary Algorithm for managing this mode. Experiments of the suggested method was found to be effective and efficient when compared to benchmark alternatives and could be widely applied for heterogeneous ECSs and multi-objective optimizations. These applications however can strain if the form of delays in multi-objective optimizations.

In order to provide precise offloading decisions, Qu et al. [23] presented DMRO (Deep Meta Reinforcement Learning) approach, which combines numerous concurrent DNNs with Q-learning. The optimum offloading technique may be quickly and flexibly learned from a dynamic environment by integrating the perceptual capabilities of DLTs, reinforcement learning's decision-making capacity, and meta-rapid learning's environment learning ability. The problem of optimal offloading decision-making is NP-hard, though, thus conventional optimization techniques are ineffective.

Yousafzai et al [24].'s investigation of the effects of platform-dependent native apps on computational offloading in edge networks resulted in the proposal of a simple framework for computational offloading based on process migration. The proposed architecture enables native programmes to be easily transferred because edge servers do not require application binaries. The suggested framework is evaluated using a testbed for experiments. However, due to their limited processor, battery, and storage space resources, these devices struggle to run computation-intensive applications that regularly require strong computing power, wide bandwidth, and quick response times.

Wu et al. [25] suggested DDTO (Distributed Deep Learning-driven Task Offloading), a near-optimal offloading technique for mobile devices, edge cloud servers, and central cloud servers. The results of the experiments reveal that the DDTO algorithm is accurate and can produce near-optimal offloading decisions in edge and cloud computing contexts. Furthermore, as compared to existing offloading systems that ignore heterogeneous cloud cooperation, it provides outstanding performance while significantly reducing computing complexity. However, solving task offloading difficulties in heterogeneous cloud computing systems, where ECSs and central clouds collaborate to meet the needs of municipal IoT applications, remains tough.

Inferences: Several studies have examined task offloading experiments. However, most studies concentrated on issues of work offloads with edge devices in MEC or cloud computing, disregarding issues of sensing devices. In contrast to existing efforts which provide workload offloading techniques based on MECs, this study focuses on IoT-edge-cloud networks, which incorporates several sensing devices.

3 Proposed Methodology

This research work proposes a hybrid task offload framework for heterogeneous clouds and detailed in this section. Mobile devices can execute workflows locally or offload in entirety or partially to the central cloud and/or the edge cloud for execution. DLTs are promising strategies in the absence of optimal alternatives for work burdens of users due to their ability to suggest solutions based on labelled data. Solutions based on DLTs can be useful for decisions, dynamic resource allocations, and content caching as communication and computation volumes increase in municipal IoT applications. Currently, it is unclear how to effectively adapt DLTs for job offloads in IoTs and examinations of intelligent technologies and efficient parallel algorithms are a necessity to handle complex offloads which are bound by high dimensionalities and require reductions in completion of weighted sums, delays, and energy consumptions while maintaining QoS. The goal of this study's DDMTO approach is to find optimal learning in diverse MEC and MCC situations, as well as to overcome computationally expensive challenges in decisions of offloads. The proposed approach combines many concurrent DNNs with Meta algorithms for its offloading choices. It encompasses inner and outer models where inner models use distributed DRLs to identify best options. The latter is trained using meta-learning to provide inner models with warm-start initializations as shown in Fig. 1. This part merges MCCs and MECs for task offloads, taking into consideration the broad processing capabilities in MCCs and the low transmission latency in MECs.

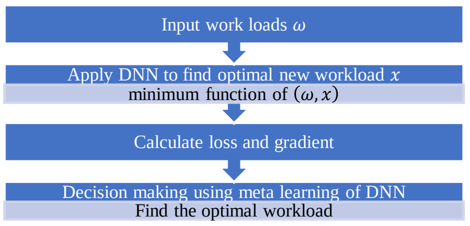
Input work loads ω
Apply DNN to find optimal new workload x
minimum function of (ω, x)
Calculate loss and gradient
Decision making using meta learning of DNN
Find the optimal workload

Figure 1: Smart Art Diagram of Proposed Methodology

3.1. System Model

Figure 2 depicts the system for IoT-edge-cloud computing scenarios task offloads which are made up of CSs (cloud servers), ECSs, and other IoT devices. IoT devices can run locally or outsource their process to central or ECSs. Consider single edge clouds, single central clouds, and large number of MDs which may outsource computationally intensive operations to either edge or central clouds. The goal of this section is collaborating MEC and MCC computing systems where edge and central clouds may be connected and integrate heterogeneous computing resources efficiently.

Using application partitioning strategies, mobile applications are separated into multiple tasks which are allocated to CSs. Task offloads are done based on job complexities and existing network environments. For example, offloading computationally -intensive operations to centralized CSs and data intensive activities to ECSs, reducing load bottlenecks, delays, and guaranteeing fault tolerances. The system concept consists of a collection of mobile IoT devices $N = \{1, 2, \cdots, N\}$, centralized CSs, edge CSs, wireless access points and independent computational tasks denoted by sets $M = \{1, 2, \cdots, M\}$. Given that each MD must manage multiple jobs, the size of the nth MD's task is denoted throughout this work by the notation (n,m). Additionally, each MD has the option of doing their activities locally or offloading them to the CSs for additional processing. When a job is decided to be offloaded to the CS, it can either be offloaded to the edge CS or the central CS. Create two binary variables with the names $\mathfrak{x}^1_{(n,m)}$ and $\mathfrak{x}^2_{(n,m)}$ to reflect the offloading choices in this section. On one hand, $\mathfrak{x}^1_{(n,m)} \in \{0, 1\}$ stands for the offloading decision for the mth task, which is measured as:

$$\mathfrak{x}^1_{(n,m)} = \begin{cases} 1 & if\ task\ is\ executed\ on\ the\ n\ th\ MD \\ 0 & if\ task\ is\ offloaded\ to\ the\ CS \end{cases}$$

where $\mathfrak{x}^1_{(n,m)}$ stands for m^{th} task that is processed locally on the n^{th} MD, and indicates the offloaded m^{th} task to CSs. Alternatively, m^{th} task is offloaded to CSs and $\mathfrak{x}^2_{(n,m)} \in \{0, 1\}$ is further defined to represent offloading destinations for m^{th} task measured as:

$$\mathfrak{x}^2_{(n,m)} = \begin{cases} 1 & if\ offloaded\ to\ edge\ cloud\ \&\mathfrak{x}^1_{(n,m)} \\ 0 & if\ offloaded\ to\ central\ cloud\ \&\mathfrak{x}^1_{(n,m)} \end{cases}$$

where $\mathfrak{x}^2_{(n,m)} = 1$ implies m^{th} task is offloaded to edge CSs, and $\mathfrak{x}^2_{(n,m)} = 0$ implies m^{th} task is offloaded to centralized CSs.

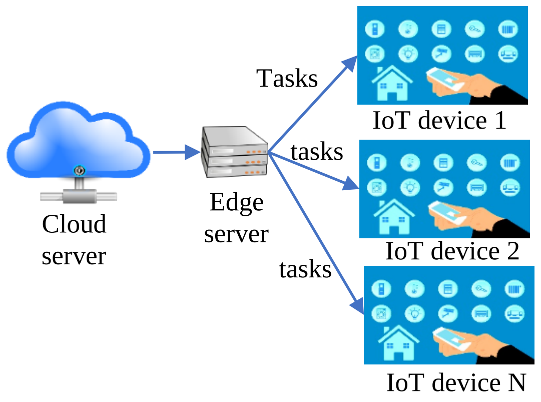

Figure 2: System Framework of Edge-cloud Computing with Multiple IoT Devices

This study first proposes the local computing paradigm, in which MDs opt to perform their jobs locally. MDs can only do simple functions due to their limited resources, such as battery capacity. c should be defined as the total number of CPU cycles used to calculate the mth job of the nth MD c(n,m). Given that CPU cycles are inversely proportional to workloads and are expressed as:

$$\mathfrak{c}(n,m) = \varphi\omega(n,m)$$

where φ signifies the positive proportionality coefficient. The energy expended while doing the mth job on the nth local device may be stated as follows:

$$E(n,m) = e \cdot \mathfrak{c}(n,m)$$

where e signifies the local device's energy consumption per unit of workloads The local device's execution time ET may be computed as follows:

$$ET(n,m) = \mathfrak{c}(n,m) \cdot \mathfrak{tp}$$

where $\mathfrak{tp}$denotes the task processing rate of the MDs. Therefore, the total computation time $CT$of the nth MD can be derived as:

$$CT(n) = \sum_{m=1}^{M} \mathfrak{x}_{(n,m)}^{1} \cdot ET(n,m)$$

The transmission's energy usage may be stated as follows:

$$et(n,m) = \varphi\omega(n,m)$$

The edge CS will perform the tasks when they have been transmitted to the edge cloud. The overall progress completion delay D may be expressed as:

$$D(n,m) = et(n,m) + \frac{\mathfrak{c}(n,m)}{\mathfrak{tp}_{EC}}$$

where $\mathfrak{tp}_{EC}$ implies task processing rates of ECSs. It is assumed that all users have equal amounts of tasks as software can be divided into several tasks and additional workloads can be treated as zeros. Moreover, it is a challenging unsupervised learning problem to resolve since it is not possible to directly extract the best judgements in this portion. Offer a method for acquiring offloading judgements and transforming it into a supervised learning issue as a consequence in this part. In this part, the neural networks' output will be the best offloading choices. Think of the workloads as their input. Importantly, preserve the least value of the function $\mathfrak{m}(\omega, \mathfrak{x})$ and the best options x together in this section in a database. The labelled data is then utilised to train several DNNs simultaneously, creating fresh data that is used to update the outdated data in the database in this stage. As a consequence, database can be updated and trained with DNNs to resolve the NP-hard problem.

3.2. Training processes of DNNs

Propose a strategy for obtaining the near optimum offloading decisions in this section. In DLTs, MSEs (mean square errors) functions are used to find the best offloading option by minimising the loss function. The MSE function is defined as follows:

$$MSE = \frac{1}{n}\sum_{t=1}^{n} |lv_t - ov_t|^2$$

where lv_t stands for labels and ov_trepresents predicted values. Since, decision elements are binary, logits can only be 0 or 1. It is easy to show that the logits will be 1, else 0, if the output is more than 1/2. The MSE function is minimised in this way, indicating that the accuracy of the model is maximised. You will choose whether to train multi-distributed DNNs to address the optimization problem in this part. DNNs generally are made up input, dual hidden, and output layers. After obtaining best offloading choices x*, the best offloading decisions x*, and the value of the functions m (, x*) are recorded in a database. There are m distributed units of NNs and their action units are parallel where one of them target parameter freezing. The frozen network's parameters are not iterated showing that the networks have the same structures. The settings are replicated to frozen networks when the other network learns a specific number of times. Reduced learning importance are the aim of parameter freezes [40]. The stated action values are assigned to unit in NNs based on their m values, which are calculated by a greedy algorithm. Additionally, local OF (objective function) is established as:

$$OF(\mathbb{mm}_i, a) = ET(n,m) + D(n,m) + \varphi\omega(n,m)$$

where $OF(\mathbb{mm}_i, a)$ may also be used to compare $OF(\mathbb{mm}_i, a)$ values produced by the actions picked by various DNNs as a measure of the consequences of the actions selected by different DNNs. It can be read as the weighted sum of the delay and energy consumption for picking action an in state "st" i. The best course of action is predetermined in the state st i to be the one with the lowest OF. When it comes to the reward function $RF(\mathbb{mm}_i, a)$ in DDMTO, depending on whether actions are action values of optimal solutions a, reward values are either negative values of minimum optimizations functions or negative values of maximum optimization functions which use cross entropy loss functions given by:

$$L(\delta_k) = -\,\mathfrak{x}^T \, log\, \mathfrak{tp}\delta_k(\omega) \;-(1-\mathfrak{x})^T \, log\big(1 \;-\; \mathfrak{tp}\delta_k(\omega)\big)$$

where δ_k reflects the DNN's parameter value. Gradient descents are used to minimize cross-entropy losses, after which all DNNs' parameters are updated. The convergence is described as the process of moving toward a defined value, i.e., the extremum, after which offloading decisions are generated as the lv t, and the database structure is continually updated. More specifically, identify the function $\mathfrak{m}(\omega, \mathfrak{x}\,)$ minimal value throughout the generation of offloading decisions as m 1. Obtain another $\mathfrak{m}_2$ optimum value by selecting a batch of data at random from the database and repeating the previous step. Then the ratio $\mathfrak{K}_1$ can be formulated as: $\mathfrak{m}(\omega, \mathfrak{x}\,)$

$$\mathfrak{K}_1 = \frac{\min(\mathfrak{m}_1, \mathfrak{m}_2)}{\max(\mathfrak{m}_1, \mathfrak{m}_2)}$$

where the ratio $\mathfrak{K}_1$ enumerates all cases for determining actual minimum values, it can be seen as convergences of DDMTO and cannot be seen as true minimums or maximum values $\mathfrak{m}(\omega, \mathfrak{x}\,)$expressed as $\mathfrak{m}_1^*$ and compared with the result of optimal values $\mathfrak{m}_2^*$. $\mathfrak{m}_1^*$ is obtained using time-consuming greedy algorithm where all offloading decision combinations are enumerated and true optimal ones are identified. The ratio of the minimum values to optimal values (defined as $\mathfrak{K}_2$) are computed as:

$$\mathfrak{K}_2 = \frac{\mathfrak{m}_1^*}{\mathfrak{m}_2^*}$$

where $0 < \mathfrak{K}_2 \leq 1$ shows how near the method's result is to the genuine optimum solution determined by the greedy algorithm. When K 2 = 1, it means that the genuinely optimal solution has been found (relative optimality). Algorithm 1 depicts the whole process of the DDMTO algorithm for the MCC and MEC hybrid offloading models. The inner model is based on an approach for Distributed Deep Learning-driven Task Offloading., as shown in Fig. 3, and it integrates environmental data, labelled beginning parameters, and process. Use a I to represent the i-th sub-task of the workflow's offloading decision in this section, which is specified as:

$$a_i = \begin{cases} 0 & if\ subtask\ i\ is\ executed\ on\ IoT\ device, \\ 1 & if\ subtask\ i\ is\ offloaded\ to\ edge\ server, \\ 2 & if\ subtask\ i\ is\ offloaded\ to\ cloud\ server, \end{cases}$$

where a_i = 0, 1, and 2 denote that the i-th subtask is carried out locally on edge servers, cloud servers, and IoT devices, respectively. Multiple DNNs are first created with random parameter values with database structures being initially empty. The suggested DDMTO approach automatically modifies the parameters to give close to optimum offloading possibilities after learning from prior offloading experiences in varied MEC and MCC scenarios. Because of the wide search space, it also avoids the curse of dimensionality, which lessens the need to address challenging MIP problems. The computational cost of the DDMTO approach won't significantly rise as the number of users and jobs rises since it just needs to select one out of a small number of potential offloading alternatives each time.

Good convergence performance may be reached as a result of notable created offloading decisions. The first approach suggests a meta-algorithm for figuring out the initial parameters of DNNs. An image classification network training algorithm, based on the fundamental method described in [38], Adapting to the reinforcement learning training technique requires learning initial parameters, and we present a novel way for doing so in this section. This portion uses a deep meta-learning strategy to train the decision-making engine and this section uses IoT edge-cloud computing settings to make speedy offloading decisions. The meta-idea algorithm's objective is to provide the training model with the results of decision-making and process execution across many circumstances. The training model selects training samples at random both during learning in one environment and following learning in another environment. Making sure that the model's parameters aren't too close to the ideal response in a particular situation is the aim of training sample learning. Use the parameters that you learnt in this way as the inner model's starting points in this part.

Algorithm 1: The Algorithm Steps of DDMTO for Optimal Workload Prediction

```
Input: Workflow $\omega$ of local MDs, Meta parameter $\rho$
Output: Optimal offloading decision
    1:  Initialize $\mathbb{m}$ DNNs with parameters $\delta$
    2:  Empty memory pools
    3:  for $i = 1, 2, 3, \cdots, N$ do
    4:  Replicate state $st_i$to all $\mathbb{m}$ DNNs
    5:  Generate $\mathbb{m}$-th offloading actions $a$ using greedy
        policies
    6:  for $j = 1, 2, 3, \cdots, M$ do
    7:  Input $a$ to decision sets
    8:  Evaluate local objective functions $OF$ and $RF$
    9:  Procedure calls meta-algorithms (DNNs)
    10: if data to memory $n$ times are added then
    11: Randomly select environments
    12: Extracts $OF(\mathbb{m}_i, a) =$ from memories at random
    13: Do step 4 to 8
    14: Update s DNNs weights $\delta$
    15: end if
    16: end for
    17: Output parameters of DNNs as meta-parameters $\rho$
    18: if trained then
    19: do
```

```
20: Initialize $\mathfrak{m}$ DNNs with meta-parameters $\rho$
21: Replicate ith offload decision candidates $\mathfrak{x}_i$ from $i$th
    DNNs
22: Select optimal offloading decisions $\mathfrak{x}^*$ by minimizing
    $\mathfrak{m}(\omega, \mathfrak{x})$ and compute $\mathfrak{m}(\omega, \mathfrak{x}^*)$ as $\mathfrak{m}^*$
23: end if
24: if database is not full then
25: Store $(\omega_i, \mathfrak{x}^*, \mathfrak{m}^*)$ into the database
26: else
27: Discard old data while saving new ones
28: end
29: end if
30: end for
31: do steps 6 to 18
32: result of optimal workload detection with updating of
    DNNs parameter $\delta$
33: end for
```

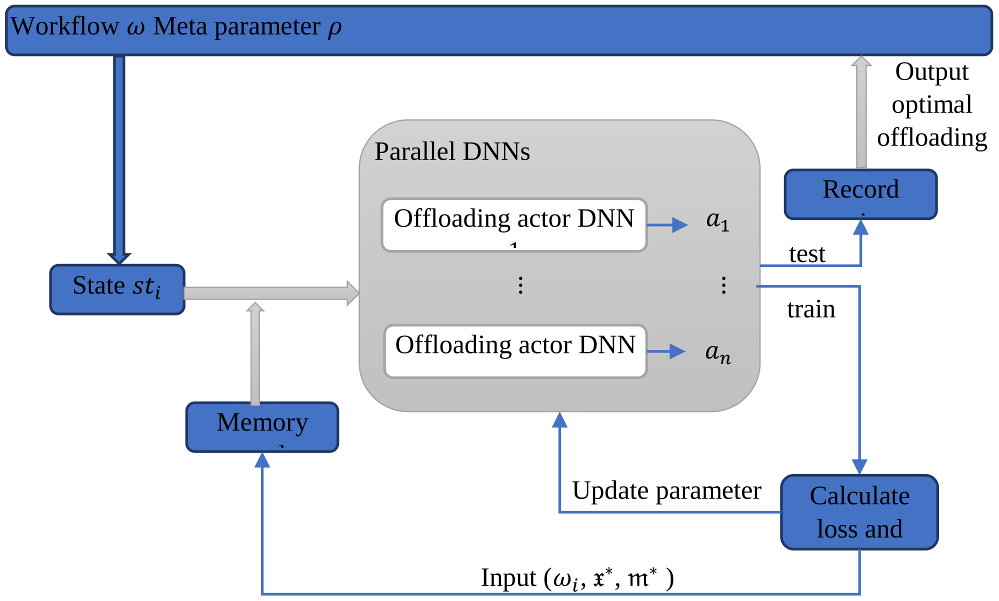

Figure 3: Illustration of the Proposed DDMTO-based Offloading Scheme

4 Experimental Results and Discussion

The proposed DDMTO method as well as existing offloading decision algorithms such as MO-BFO [16], DMRO [23], and DDTO [25], This work uses Tensor flow and libraries of MLTs to implement and analyse these methods in Python. With N = 3 users or MDs and M = 3 concurrently performing compute workloads per user or MD, heterogeneous edges and cloud computing environments are created. In addition, assuming input workloads of tasks are distributed at random between 0 and 30 MB, user's bandwidths are set to 50 Mbps. Set at 0.5, the weighing parameter emphasize on balancing performances and power usages and DNNs are trained from the database using batches of 500.

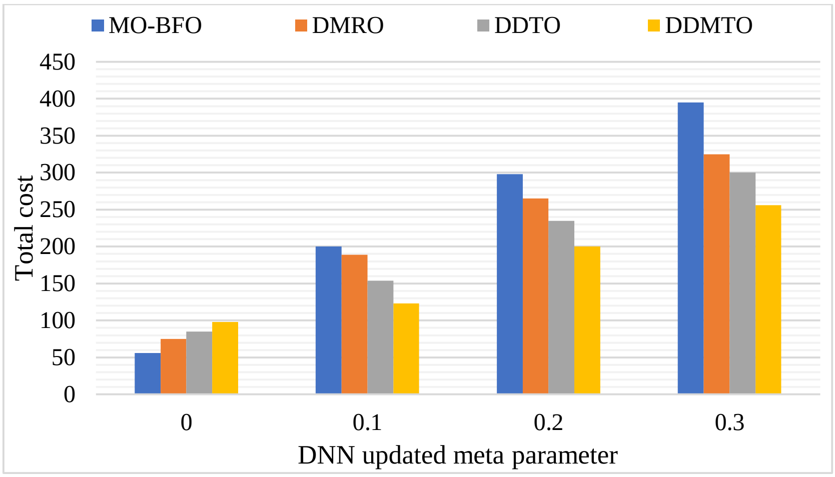

Figure 4: Performance Comparison of Different Offloading Schemes Under Different Weights

The experiment uses a variety of learning offloading techniques, including as MO-BFO, DMRO, DDTO, and DDMTO, to regulate the variables. This scheme's beginning parameters are identical to those of that scheme, including the same state spaces, action spaces, and structures of NNs. The results of the comparison are given in Fig. 4, where ordinates represent values of goal functions and abscissas represent weight ratios of delay to energy consumptions. The goal function's value is dimensionless since feature scaling was used in this section. Shorter delays and less energy use are indicative of a larger offloading effect when the total cost is lower. When the weight value is 0, it means that only the delay is considered. The graph shows that, of the three methods, the DDMTO algorithm has the lowest total cost. As seen in the graphic, the DDMTO method often outperforms other algorithms. For instance, the DDMTO algorithm surpasses the MO-BFO, DMRO, and DDTO algorithms by obtaining a rate of 200 when the weight is 0.2. Additionally, overall consumptions of local executions increase weight ratios of energy consumptions demonstrating local device's sensitiveness to energy consumptions.

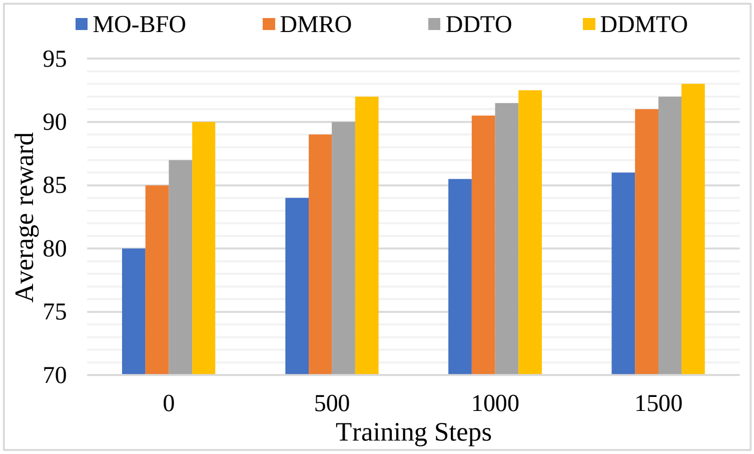

Figure 5: Performance Comparison of Different Offloading Schemes in a Multi-server Environment

Figure 5 examines the impacts of several algorithms in a multi-edge environment, including MO-BFO, DMRO, DDTO, and DDMTO, where vertical coordinates are algorithm's average reward values and horizontal coordinates are training step counts. Because the reward value of this model is the inverse of the objective function value, a larger average reward indicates a greater model offloading effect. The graph illustrates that as the number of training steps rises, all DDMTO algorithms improve, demonstrating that an appropriate state space and reward function were defined in this section. Additionally, the proposed DDMTO approach provides the highest average reward value and quickly converges, proving that DDMTO offloading outcomes are significantly better than those of existing methods like MO-BFO, DMRO, and DDTO.

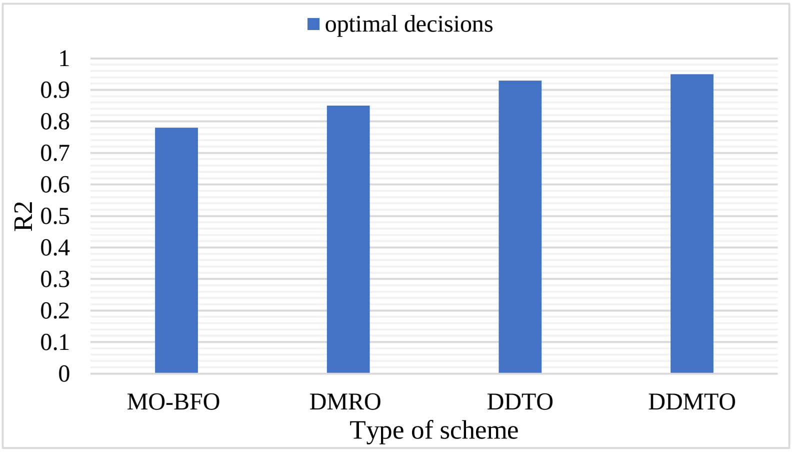

Figure 6: Comparison with Several Offloading Schemes

A comparison of the outcomes of several offloading procedures is shown in Figure 6. Since the relative optimality, or ratio R2, of the proposed DDMTO algorithm exceeds 0.95, which is substantially higher than for any of the other three schemes, it can be shown that the algorithm's optimum judgments perform very well. The R2value of the local-only strategy, for example, is only about 0.78, 0.85, and 0.93 for MO-BFO, DMRO, and DDTO, respectively. This is because, unlike edge-only and central-only techniques, DDMTO dynamically offloads jobs based on heterogeneous computing environments which include task workloads, communication data, and network circumstances. Offloading duties to the edge/cloud server may not be desirable, especially if network bandwidths are limited. As a result, the proposed DDMTO technique achieves near to optimal offloading decisions in heterogeneous edge and cloud computing scenarios. This demonstrates that in the new MEC context, both the meta-learning and the learned parameters exhibit strong convergence.

5 Conclusion and Future work

In order to solve the issue of neural network portability, this research provides a unique DDMTO architecture to manage the task offloading decision-making dilemma in heterogeneous edge and cloud collaborative computing contexts. It is composed of a distributed DNN-based task offloading decision model and a deep meta-learning-based training initial parameter model, and it may be capable of rapidly adapting to a dynamic MEC environment and addressing the challenge of offloading decision-making for edge-cloud computing. The DDMTO algorithm determines if a job should be done locally or offloaded to the clouds, and if so, whether it should be offloaded to the edge or the central cloud. The numerical findings show that the DDMTO approach is valid and they outperform the outcomes of many other well-known earlier methods. This section will continue to look at more aspects of the hybrid offloading model in subsequent studies to improve the algorithm's ability to handle real-world mobile offloading scenarios. Additionally, in this part, we'll offer a framework for assessing the DDMTO algorithm's effectiveness throughout the work offloading procedure. According to the experimental findings, In terms of work offloading alternatives, DMRO outperforms binary offloading approaches and traditional DRL-based partial offloading systems. Because of the addition of meta parameters, the model is also more portable and capable of swiftly learning its environment. As the MEC environment changes, the model may quickly converge, and only a few learning steps are necessary to determine the optimum offloading possibilities at the lowest feasible cost. Future research in this area will concentrate on increasing the meta learning algorithm's capacity to adapt to task offloading options in large-scale MEC settings, particularly by making adaptive modifications to the initial parameters in response to environmental conditions. This research also looks at server-less edge computing frameworks for overcoming challenges like resource allocation and bandwidth correction. The offloading model may also offer resource scheduling tactics in addition to the outcomes of task offloading decision-making.

References

[1] Alam, F., Mehmood, R., Katib, I., & Albeshri, A. (2016). Analysis of eight data mining algorithms for smarter Internet of Things (IoT). *Procedia Computer Science*, *98*, 437-442.

[2] Kipper, G., & Rampolla, J. (2012). *Augmented reality: An emerging technologies guide to AR*. Elsevier.

[3] Othman, M., Madani, S.A., & Khan, S.U. (2013). A survey of mobile cloud computing application models. *IEEE communications surveys & tutorials*, *16*(1), 393-413.

[4] Rao, B.P., Saluia, P., Sharma, N., Mittal, A., & Sharma, S.V. (2012). Cloud computing for Internet of Things & sensing based applications. *In IEEE Sixth International Conference on Sensing Technology (ICST)*, 374-380.

[5] Shannigrahi, S., Mastorakis, S., & Ortega, F.R. (2020). Next-generation networking and edge computing for mixed reality real-time interactive systems. *In IEEE International Conference on Communications Workshops (ICC Workshops)*, 1-6.

[6] Salman, O., Elhajj, I., Kayssi, A., & Chehab, A. (2015). Edge computing enabling the Internet of Things. *In IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 603-608.

[7] Varshney, P., & Simmhan, Y. (2017). Demystifying fog computing: Characterizing architectures, applications and abstractions. *In IEEE 1st international conference on fog and edge computing (ICFEC)*, 115-124.

[8] Hu, P., Dhelim, S., Ning, H., & Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications*, *98*, 27-42.

[9] Ruan, G., Yu, G., Zheng, J., Zou, J., & Yang, S. (2017). The effect of diversity maintenance on prediction in dynamic multi-objective optimization. *Applied Soft Computing*, *58*, 631-647.

[10] Meng, S., Li, Q., Wu, T., Huang, W., Zhang, J., & Li, W. (2019). A fault-tolerant dynamic scheduling method on hierarchical mobile edge cloud computing. *Computational Intelligence*, *35*(3), 577-598.

[11] Ren, J., Zhang, D., He, S., Zhang, Y., & Li, T. (2019). A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)*, *52*(6), 1-36.

[12] Slamnik-Kriještorac, N., Kremo, H., Ruffini, M., & Marquez-Barja, J. M. (2020). Sharing distributed and heterogeneous resources toward end-to-end 5G networks: A comprehensive survey and a taxonomy. *IEEE Communications Surveys & Tutorials*, *22*(3), 1592-1628.

[13] Beraldi, R., Canali, C., Lancellotti, R., & Mattia, G.P. (2022). On the impact of stale information on distributed online load balancing protocols for edge computing. *Computer Networks*, *210*, 108935.

[14] Intharawijitr, K., Iida, K., Koga, H., & Yamaoka, K. (2017). Practical enhancement and evaluation of a low-latency network model using mobile edge computing. *In IEEE 41st annual computer software and applications conference (COMPSAC)*, *1*, 567-574.

[15] Huang, X., Yang, Y., & Wu, X. (2019). A meta-heuristic computation offloading strategy for IoT applications in an edge-cloud framework. *In Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*, 1-6.

[16] Babar, M., Din, A., Alzamzami, O., Karamti, H., Khan, A., & Nawaz, M. (2022). A Bacterial Foraging Based Smart Offloading for IoT Sensors in Edge Computing. *Computers and Electrical Engineering*, *102*, 108123.

[17] Long, J., Luo, Y., Zhu, X., Luo, E., & Huang, M. (2020). Computation offloading through mobile vehicles in IoT-edge-cloud network. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 1-21.

[18] Mu, S., & Zhong, Z. (2020). Computation offloading to edge cloud and dynamically resource-sharing collaborators in Internet of Things. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 1-21.

[19] Shahidinejad, A., & Ghobaei-Arani, M. (2022). A metaheuristic-based computation offloading in edge-cloud environment. *Journal of Ambient Intelligence and Humanized Computing*, *13*(5), 2785-2794.

[20] Huang, Q., Xu, X., & Chen, J. (2021). Learning-aided fine-grained offloading for real-time applications in edge-cloud computing. *Wireless Networks*, 1-16.

[21] Peng, K., Huang, H., Wan, S., & Leung, V. (2020). End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment. *Wireless Networks*, 1-12.

[22] Jayaram, R., & Prabakaran, S. (2021). Adaptive cost and energy aware secure peer-to-peer computational offloading in the edge-cloud enabled healthcare system. Peer-to-Peer Networking and Applications, 14(4), 2209-2223.

[23] Qu, G., Wu, H., Li, R., & Jiao, P. (2021). DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing. *IEEE Transactions on Network and Service Management*, *18*(3), 3448-3459.

[24] Yousafzai, A., Yaqoob, I., Imran, M., Gani, A., & Noor, R.M. (2019). Process migration-based computational offloading framework for IoT-supported mobile edge/cloud computing. *IEEE internet of things journal*, *7*(5), 4171-4182.

[25] Wu, H., Zhang, Z., Guan, C., Wolter, K., & Xu, M. (2020). Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. *IEEE Internet of Things Journal*, *7*(9), 8099-8110.

[26] Schmidbauer, T., & Wendzel, S. (2022). Detection Of Computational Intensive Reversible Covert Channels Based On Packet Runtime. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 13*(1), 137-166.