Grenze International Journal of Engineering and Technology, June Issue

Energy Optimized and Dynamic Design of Task Offloading in Mobile Fog Computing

M.Vedaraj[1], A.Satwika[2], B.Monisha[3] and I.Namratha[4]
[1-4]R.M.D Engineering College/Computer Science and Engineering Department, Chennai, India.
Email: vedaraj84@gmail.com,ucs20110@rmd.ac.in,ucs20117@rmd.ac.in,ucs20207@rmd.ac.in

***Abstract*— In this study, we provide a dynamic and energy-optimized task offloading approach for MFC. Our method takes into account how much energy FNs, cell phones, and communication cables use. We employ a dynamic programming technique to choose the job offloading strategy that minimises energy use while satisfying the application's latency requirements. We conduct a simulation-based experiment to assess our suggested strategy. The results show that our solution performs better than others in terms of energy consumption and delay efficiency. Additionally, our method may balance energy usage and delay performance while adapting to dynamic changes in the network topology.Our proposed energy-optimized and dynamic design for task offloading in MFC can enable efficient and effective utilization of computational resources in mobile devices and FNs, This may result in better functionality and longer battery life for portable electronics.**

***Index Terms*— MEC(Mobile Edge Computing), QOS(Quality of Service)**

I. INTRODUCTION

The volume and complexity of data have increased to previously unheard-of levels as a result of the widespread deployment of 5G wireless networks and the subsequent installation of countless connected devices. Due to limited bandwidth and latency concerns, using centralised and cloud data centres has become inefficient as a result of straining the network's capacity and infrastructure capabilities. New applications with higher quality-of-service (QoS) needs, like autonomous driving, augmented reality, and virtual reality, have been made possible by the continued development of 5G networks. However, due to their constrained computational power and battery life, mobile customers may find it challenging to meet these stringent QoS requirements. These issues have been addressed by the development of mobile edge computing (MEC), a promising technology that enables mobile users to offload compute-intensive tasks to nearby base stations that support MEC.

Concerns about energy usage, network latency, and dynamic changes in the network topology are brought up by task offloading in MEC. In this study, we offer an energy-optimized and dynamic task offloading strategy for mobile fog computing (MFC). This design makes use of the superior computational capacity of neighbouring fog nodes (FNs) to offload jobs from mobile devices. In our method, the best task offloading choice that minimises energy consumption while satisfying the application's delay limitations is determined using a dynamic programming algorithm. The results show that our strategy beats other ways in terms of energy usage and delay performance. We test our approach using simulation-based studies.

Concerns about energy usage, network latency, and dynamic changes in the network topology are brought up by task offloading in MEC. In this study, we offer an energy-optimized and dynamic task offloading strategy for

Grenze ID: 01.GIJET.10.2.57_1

mobile fog computing (MFC). This design makes use of the superior computational capacity of neighbouring fog nodes (FNs) to offload jobs from mobile devices. In our method, the best task offloading choice that minimises energy consumption while satisfying the application's delay limitations is determined using a dynamic programming algorithm. The results show that our strategy beats other ways in terms of energy usage and delay performance. We test our approach using simulation-based studies.

Our proposed energy-optimized and dynamic solution for job offloading in MFC may enable a more effective and efficient use of computing resources in mobile devices and FNs. Mobile devices may operate better and have longer battery lives as a result of this. High quality and low-latency applications are in greater demand in the era of 5G networks, and the contributions made in this study may help pave the way for the development of more advanced and dependable ones. MEC devices that can meet this need. As fifth-generation (5G) mobile networks proliferate, connected devices are being widely deployed and continuously used. These devices generate data that is too vast and complex for the infrastructure and network to handle. The network's latency issues and capacity restrictions make using centralised and cloud data centres remain inefficient. Our way of life has been significantly affected by the continued development of fifth generation networks, which has also given rise to a number of novel applications including augmented reality, virtual reality, and autonomous driving, which have significantly higher quality-of-service (QoS) requirements than conventional applications.

The tight QoS requirements may be difficult for mobile users to achieve, however, because to their limited processing power and battery life. Mobile edge computing (MEC), which enables mobile users to wirelessly transfer their compute work to nearby base stations (BSs) outfitted with MEC servers, has been presented as a solution to these problems. Smart phones, tablet computers, laptops, and smart assistants are a few examples of mobile terminals that have greatly increased in popularity as a result of the rapid development of mobile Internet technology and the ongoing development of mobile communication technology. However, there are restrictions on the mobile terminal's size, weight, usability, power, etc. It is unable to satisfy the population's rising demand because its operating capacity is still in a horrible and tiring state. Despite the regular replacement of the CPU and GPU, the advancement of chip manufacturing from 28nm to 14nm to the present 7nm, 5nm, etc., mobile device hardware technology still falls short of what consumers require. Additionally, because new ideas like autonomous driving, telemedicine, and Industry 4.0 require exceptional reliability and very little delay, standard technology is becoming less and less capable of supporting them. In between, applications for machine learning, artificial intelligence, and other new technologies are growing rapidly along applications for speech recognition, image identification, and other new technologies.

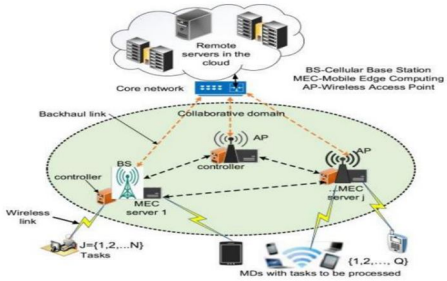
The development of video games for virtual reality and augmented reality is expanding as well quickly. All of these programmes require substantial expenditures in computer and storage resources due to their analysis requirement. Because of the limitations of mobile terminals or other devices, running computationally expensive programmes on smart terminals significantly reduces the lifespan of the terminal and the program's performance. The typical method for sending tasks and data to a cloud centre is for a third party to request computing resources through the Internet. Mobile cloud computing, for instance, was created to improve the operation of portable electronics. This centralised strategy offers infinite processing capability because to the clustering of tens of thousands or more servers at the cloud's end. Due to the network traffic and speed, delivering large amounts of data requires a lot of latency and energy, and the cloud is frequently far from a mobile device.

Therefore, mobile cloud computing is constrained by operational efficiency. An answer to the problem caused by centralised computing is sought by the distributed computing paradigm known as edge computing (EC). This shared design offers a more successful and effective approach. By evaluating the data in terms of nearby computing resources, such as those in the same region, a speedier response time is attained. Consequently, EC allows for a large reduction in latency while simultaneously lowering congestion on the central servers and at the main link. and backhaul levels of the network. The advantages of EC are best seen in large, complicated networks, such as Internet of Things (also known as IoT) networks. By improving the transmission time, latency, and bandwidth usage, the IoT network's efficiency may be significantly increased. Microsoft, for example, made advantage of these. You may design a system that collects movies from a wide area and instantaneously evaluates them using edge infrastructures.

Furthermore, EC provides a flexible job offloading approach that lengthens the battery life of end devices. Cell phones including smartphones, tablets, laptops, and smart assistants are increasingly widely utilised as a result of the speedy development of mobile communication technology and the fast growth of mobile internet. These devices' inability to satisfy the rising expectations of people is still hampered by aspects including volume, weight, performance, and power. Furthermore, conventional devices are unable to support the ultra-reliable and low-latency equipment needed by new technologies like autonomous driving, telemedicine, and Industry 4.0. The

demand for processing power and storage has increased as a result of the development of computationally demanding applications such as machine learning, artificial intelligence, speech recognition, picture recognition, and virtual reality. However, it can be difficult to operate these applications effectively due to the constrained resources and high energy consumption of mobile terminals.

This issue has been addressed by the development of edge computing (EC), a distributed computing paradigm that uses nearby computer resources to analyse data and create speedier response times and lower latency. The performance of complex networks like the Internet of Things (IoT) networks could be significantly improved by EC as compared to centralised computing. It also offers an adjustable workload offloading technology that lengthens the battery life of end devices. In conclusion, the rise of computationally intensive applications and the spread of mobile devices have highlighted the demand for improved and more efficient computing paradigms like EC. The performance of mobile devices can be greatly enhanced by EC by utilising nearby computer capabilities, which also solves the problem of limited resources and energy usage.



II. Related Work

A. Literature Survey

Several Research Studies have focused on mobile Edge Computing in various fields .one of the main challenge faced by them are Resource constraints: Mobile devices have limited resources such as processing power, battery life, and memory. Therefore, designing an energy-efficient task offloading system that optimizes resource utilization while meeting application requirements can be a challenging task. Security and privacy: Offloading tasks to external computing resources introduces security and privacy concerns. The system needs to ensure the confidentiality, integrity, and availability of data, as well as prevent unauthorized access and data leakage.

B. Mobile edge computing

Mobile Edge Computing (MEC) is a distributed computing paradigm that brings computing resources closer to the end-users by deploying computing resources and services at the edge of the network. In MEC, computing resources, such as servers, storage, and networking infrastructure, are deployed in close proximity to the end-users, typically within the radio access network (RAN) of the mobile network. MEC enables the development of new applications and services that can benefit from the low latency, high bandwidth, and contextual information available at the edge of the network. By moving processing and storage closer to the end-users, MEC can reduce the data transfer and processing overheads associated with cloud computing, which can result in faster response times, improved network efficiency, and better user experiences.

C. Existing system

Mobile devices increasingly require more resources and computing power, as evidenced by the quick evolution of mobile applications. By fusing cloud computing with mobile devices, or mobile cloud computing (MCC), advanced and resource-intensive apps can run on mobile devices despite their performance constraints (such as battery life, memory use, and processing). Computation offloading Unloading labor-intensive work to cutting-edge computers and receiving the results from these servers is a frequent strategy for getting around these constraints. Numerous studies on mobile code offloading have been conducted in an effort to get around the restrictions and cut down on execution time or battery consumption by using single- or multiple site unloading. Unloading is difficult, nevertheless, because of the majority of technologies currently in use. The same site must be used to discharge an object and signal a stable network environment when decisions are based on profile data.

Due to these challenges, the current study proposes a novel approach that enhances the multisite offloading mechanism by combining a Niching Genetic Algorithm (NGA) and a Markov Decision Process (MDP). Using

MDP, it is decided where each application module should be executed. GA was used to determine the appropriate transition probability for components operating across many sites. The present strategy makes use of the context-based clearing (CBC) method to broaden the variety of the genes contained within the chromosome and reduce their linkage, which aids in the first process of population selection. According to the simulation of the current system, the suggested method quickly chooses the best offloading option while consuming fewer power generational counts. The current method uses a niching genetic algorithm to choose the best probability from an initial population pool.

In order to prevent viable solutions from concentrating in one section of the feasible environment, it is essential to maintain a large collection of viable solutions when dealing with problems that cut over several feasible zones. Numerous solutions in the population can be simultaneously assessed by applying niching techniques to lessen the effects of genetic drift in the standard GA selection operator. The advantages of this approach are simplicity of use and effectiveness for parallel processing Cleared individuals have minimal ability to understand the evolutionary process in the conventional clearing system since they are unable to take part in mutations or crossings in the clearing-based niche strategy. The Context-Based Clearing (CBC) method is applied to solve this problem. Instead of using a specific radius to omit a portion of the population, CBC uses a predetermined number of applications. As opposed to the traditional clearing procedure, the CBC technique makes use of local knowledge to direct the cleaning procedure. A set radius also does away with the requirement for unnecessary overhead processing.

D. Problems with the Current System

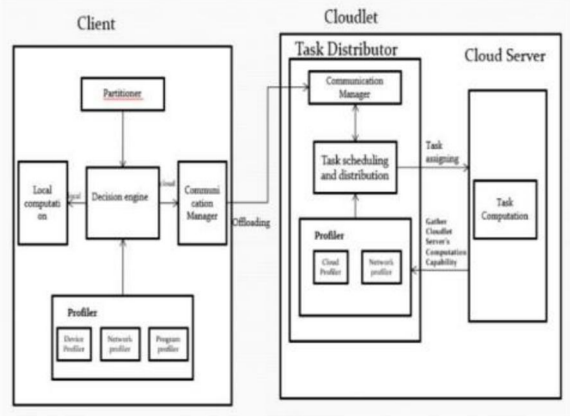
1. User Quality of Experience (QoE) isn't taken into account User Quality of Experience (QoE) is an essential aspect of mobile fog computing since it relates to user satisfaction with the service provided. However, some existing algorithms do not take into account the user's QoE in their computation offloading solutions. Ignoring QoE can lead to user dissatisfaction, low adoption rates, and loss of revenue for the service provider. Thus, it is important to consider QoE as a key performance metric when designing and evaluating computation offloading solutions in mobile fog computing.
2. Most schedulers consider only single client in their computation offloading solution: Many existing computation offloading solutions in mobile fog computing consider only a single client in their scheduling algorithm. However, in real world scenarios, there are multiple clients accessing the same fog node, and each client has its own requirements and preferences. Neglecting other clients can lead to unfairness and inefficiency in resource allocation, resulting in poor performance and user experience. Hence, it is necessary to develop scheduling algorithms that consider the needs of all clients simultaneously.
3. Cannot be adopted as a realistic solution: Some existing computation offloading solutions in mobile fog computing are not practical because they rely on unrealistic assumptions or constraints. For example, some solutions assume that the fog nodes have unlimited computational resources, which is not realistic. Additionally, some solutions require accurate information about the network conditions, which may not always be available in real-world scenarios. Therefore, it is necessary to develop solutions that are practical, realistic, and applicable in real-world scenarios.
4. Cloudlet of limited server needs to schedule its computational resources efficiently: Cloudlets are small-scale data centers that can host fog computing resources and services. However, cloudlets have limited computational resources and need to schedule their resources efficiently to provide the required service quality. Therefore, efficient scheduling algorithms are needed to optimize resource utilization and minimize response time, energy consumption, and cost.
5. Related to task overload or job scheduling problem: In mobile fog computing, task overload or job scheduling problem refers to the issue of how to assign computational tasks to the available resources in a way that optimizes performance and resource utilization. It is a challenging problem because the resources are distributed across multiple fog nodes, and the network conditions are dynamic and uncertain. Hence, efficient algorithms are required to address this problem, which can balance the workload across different fog nodes and ensure that tasks are executed within the specified time and resource constraints.

III. METHODOLOGY

A. System Architecture

In a multi-user MEC system, the challenge of optimising the least number of offloaded bits is achieved using a provably efficient and convergent algorithm. The primary argument is presented in the following. We take into account a group of hosts that are connected by a network and have the capacity to all create and/or carry out soft

real-time jobs over time. Then, a host may develop or run only jobs, or it may run both its own work and jobs that are transferred to different hosts. Offloading decisions allows them to be well-informed and flexible to the requirements of the moment.



1. System Architecture

Information broadcast among hosts on factors including network bandwidth and latency, host job load, and available energy offers the necessary feedback. We assume that each job has a soft realtime nature, meaning that it has a relative deadline that represents the shortest amount of time that may be used to complete the task while maintaining a satisfactory QoS. We also assume that in the case of offloading, each job needs communication across hosts in order to provide work inputs (before the computation may start) and collect job outputs (after the calculation is completed). The framework's modular design allows for the approach and offloading strategies to be implemented effortlessly, and it provides users the tools they need to decide how to execute offloading based on system response.

IV. ALGORITHM

Dynamic Computation Offloading Algorithm Dynamic Computation Offloading Algorithm is a technique used in mobile fog computing that involves the delegation of computationally demanding tasks to a distant fog node or cloud server from a mobile device. In order to decide when and where to offload tasks, the algorithm keeps track of the processing power of both the mobile device and the fog nodes as well as the network circumstances. The goal of the Dynamic Computation Offloading Algorithm is to optimize the energy consumption of the mobile device while also ensuring that tasks are executed within the desired time frame. This is achieved by dynamically adjusting the offloading decision based on the current network and device conditions. The algorithm can be implemented in different ways, but it typically involves the following steps:

1. Task Partitioning: The algorithm breaks the task down into smaller, more manageable components that can be carried either locally on the mobile device or online on an external server.
2. Computation Analysis: The algorithm analyzes the computational requirements of the task and determines the best strategy for executing each subtask based on the available resources and network conditions.
3. The algorithm selects whether to execute the subtask locally or offload it to an external server based on the analysis. This choice is made in the moment and is subject to dynamic modification as circumstances change.
4. Task Execution: The sub-tasks are executed either locally or remotely, depending on the offloading decision. The results are then combined to produce the final output.

A. Code Profiler

Code profiling is an essential module in offloading systems that allows for the identification of resource-intensive portions of code that can be offloaded to remote servers to reduce the computational burden on mobile devices. However, there are several other techniques that can be used to improve code profiling and offloading efficiency.one such technique is dynamic profiling, which involves monitoring the application's execution at runtime to identify hotspots or frequently executed code segments. This technique can capture the actual execution behaviour of the application, making it more accurate than static profiling. Machine learning techniques are another method for code profiling. Machine learning algorithms can learn the application's behaviour and identify which

parts of the code are most likely to benefit from offloading, based on historical usage data.Additionally, code partitioning strategies can be improved by taking into account the network and server conditions at the time of offloading. This can be done by considering the available bandwidth, latency, and server workload to determine the optimal offloading strategy.Finally, the use of hybrid offloading techniques, such as edge-cloud offloading, can improve the performance of offloading systems. These techniques involve using both local and remote resources to execute the application, thereby reducing the overall latency and improving the ser experience. Offloading is an opportunistic technique that employs distant servers to execute code that has been assigned by a mobile device.By providing the mobile with local decision logic to identify highly resource-intensive code sections, this method enables the mobile to decide where executing a piece of code will require the least amount of processing power when there is network access. The code profiler determines what should be offloaded. Offloading candidates are consequently discovered within Method, Thread, or Class. It is necessary to decide what software should be offloaded when using code partition. There are several ways to split up a piece of code.. for example, a software developer can choose specifically which code should be offloaded using static annotations like @Offloadable and @Remote. Other methods use an automated system to perform an implicit analysis of the code while it is running. Thus, the process chooses the code to be offloaded after the application has been installed on the device. The mechanism use techniques like static analysis and historical traces to determine whether or not a certain section of code is intense. Static mechanisms are preferable to automated ones because they can adjust the code to run on various platforms.

B. System Profiler

System profilers can also monitor the CPU usage, memory usage, and battery status of the smartphone. This information is critical to determining when to offload to the cloud. If the CPU usage or memory usage is high, offloading certain tasks to the cloud can help reduce the workload on the smartphone and improve its performance. However, if the battery status is low, offloading tasks to the cloud may not be feasible as it would consume more energy to transmit data to the cloud. Additionally, system profilers can also monitor the network availability and latency to determine whether offloading to the cloud is feasible and beneficial in terms of reducing the computation time. Finally, system profilers can also track the user's location and context (e.g., indoors/outdoors, driving/walking) to adapt the offloading strategy based on the user's behaviour and preferences.

C. Decision Engine

The Decision Engine is a critical component of the Mobile Fog Computing system, which determines whether to offload the computation to the fog or cloud or to execute it locally. The Decision Engine collects data from various sources, including the System Profiler, Code Profiler, and Network Profiler, to make informed decisions about offloading. The engine applies certain rules and policies to the collected data to make the decision. The main goal of the Decision Engine is to optimise Quality of Service (QoS) while reducing energy consumption. The engine considers various factors such as the available resources, network bandwidth, computational capabilities, and application requirements before making the decision. It utilizes the collected data to create a comprehensive model of the system and make accurate predictions about the system's behaviour. The Decision Engine uses machine learning algorithms to make better decisions based on the collected data. It learns from the user's behaviour and adjusts its policies accordingly. The engine is also capable of adapting to changing conditions such as varying network bandwidth, changes in the available resources, and other dynamic factors that affect the system's behaviour. Overall, the Decision Engine plays a critical role in optimizing the performance of Mobile Fog Computing systems. It helps to minimize energy consumption while ensuring optimal QoS by making informed decisions about when to offload computation to the cloud or execute it locally

D. Software Testing

Software developer artefacts produced during or before testing are documented in testing documentation. The paperwork demonstrates how important processes are to the company, the client, and the individual. Only-document projects exhibit a high level of maturity. With good documentation, the organisation might be able to save time, money, and resources. The testing or development team will require a document to identify the problem if it receives software that isn't working properly but was generated by someone else. Now, if the documentation is available, the team will analyse it to determine the error's root cause. The tester must create fresh blackbox and white-box tests if the documentation is not accessible. which will be a waste of the organization's time and resources. Additionally, the absence of documentation makes acceptance difficult. Because they test the software programmer from the perspective of the user, testers must act as though they are the user in the test scenario. The

most important step is scenario preparation, and in order to do this, it is necessary to consult with customers, stakeholders, or developers.

1. Test strategy

Test strategy is a high-level plan for testing a software application or system. It outlines the testing approach, techniques, tools, and resources to be used to ensure the quality and reliability of the software. A welldefined test strategy helps to identify potential issues, risks, and dependencies in the testing process, and provides a roadmap for how to overcome these challenges.

2. Test data

The data that testers enter into the programmer to validate particular features and their outcomes. Examples of this type of data include fictitious user profiles, statistics, and media files that resemble those that an end user may provide to a ready solution. A document defining the plan, resources, environment, restrictions, and schedule for the testing process is referred to as planned tests. The most comprehensive testing document, it is crucial for well-informed planning. Each team member receives a copy of this document, which is then shared with all stakeholders.

3. Test scenarios

Throughout the testing process, scenarios are used by testers to provide real-time progress updates and to segment the functionality and interface of the product into modules. A module may simply require a single statement to describe it or may require hundreds of statuses, depending on the size and breadth of the module.

4. Test cases

A scenario defines a process (how), but a test scenario describes the thing being tested (what). These documents include step-by-step instructions, thorough conditions, and the most recent inputs for a testing task. Depending on the testing method, test cases might be functional, user interface, physical, logical, etc Each item has a unique ID, Consequently, stakeholders and team members can monitor the status of any job by just entering its ID into the search. The secret to fully comprehending all testing procedures is the combination of internal and external documentation. Stakeholders usually work with external files because they are more brief and focus on genuine issues and results, even though they have access to most of the documentation. Teammates, on the other hand, expedite the testing procedure by utilising internal files. The idea of unit testing is not new. It has existed since since programming first emerged. Unit tests, also known as test-driven development, TDD, or test-first development, are primarily created by developers and occasionally by White box testers to improve code quality by verifying each and every piece of code used to implement functional requirements. A type of testing called software performance testing is used to assess a system's performance in terms of key performance indicators like responsiveness, speed, scalability, and stability under various load situations. A variety of stress events are delivered to the system in order to observe how quickly it responds to different workloads. In order to make sure that the code is working as intended under varied load levels, software performance testing also entails testing the application that is being tested. Performance bottlenecks and faults in the system are sought after and attempted to be fixed during performance testing.

V. CONCLUSION

In that work, it was investigated how to shorten job offloading delays and boost local device energy savings. To lessen the amount of tasks that need to be broadcast, a novel task offloading technique that takes task correlation into account in both the time domain and the task domain is being offered. Additionally, a combined optimization for job distribution, transmission power, and clock frequency has been researched. We have developed an optimization problem for the system model under discussion, with constraints on latency, transmit power, and energy requirements, to maximize the minimal number of offloaded bits among the users. A provably convergent algorithm is developed by solving the original problem successively, and each iteration of the method can be solved effectively.

REFERENCES

[1] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, Computation offloading in multi-access edge computing: A multi-task learning approach, TMC.2020.2990630.

[2] B. Yang, X. Cao, X. Li, C. Yuen, and L. Qian, Lessons learned from accident of autonomous vehicle testing: An edge learning-aided offloading Aug. 2020. [30] B. Yang, X. Cao, C. Yuen, and L. Qian, Offloading optimization in edge computing for deep learning enabled target tracking by Internet10.1109/JIOT.2020.3016694..
[3] Z. Song, Y. Liu, and X. Sun, Joint radio and computational resource allocation for NOMA-based mobile edge computing in heterogeneous networks, Dec. 2018..
[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, CloneCloud: Elastic execution between mobile device and cloud, in Proc. 6th Conf..
[5] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, Femto clouds: Leveraging mobile devices to provide cloud service at the edge, in Proc.
[6] N. M. Razali and J. Geraghty, Genetic algorithm performance with different selection strategies in solving TSP, in Proc. 19th World Congr.
[7] A Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and 168, Nov. 2019. J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicu-Aug. 2019.
[8] Y. Mao, J. Zhang, and K. B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in Proc. [30] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile published. Canu , Pierre Vera and Su Ruan, IEEE 2021.
[9] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004. [10] H. Xiang, W. Zhou, M. Daneshmand, and M. Peng, Network slicing in
[10] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, mCloud: A context-aware offloading framework for heterogeneous Sep./Oct. 2019.