POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A Stackelberg Game approach for managing an AI application in Mobile Edge Cloud systems

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Roberto Sala, 965311

Advisor:
Prof. Danilo Ardagna

Co-advisors:
Dr. Hamta Sedghani

Academic year:
2021-2022

Abstract: The growth, development, and commercialization of artificial intelligence-based technologies is leading to the need for more and more computing power. Some clear examples are self-driving cars, augmented reality viewers, chat-bots, and virtual assistants. Most of these applications are based on Deep Neural Network (DNN) which leads to an increase in the computing power required to meet user demands. This demand, however, can be met neither by users' local devices because of insufficient processing capacity, nor by cloud data centers, because of high transmission latency enforced by long distance. Therefore, edge cloud computing overcomes this problem by handling user requests through 5G reducing transmission latency from local devices to computational resources and providing sufficient computing power.
In this thesis, we model a Mobile Edge Cloud system for an application based on a single DNN. The interaction among multiple mobile users and the edge platform is modelled as a one-leader multi-follower Stackelberg game. The platform provider is the leader who aims to maximize its profit by setting proper application fee and minimizing the cost of resources while a response time constraint is guaranteed, and the users are the followers with the goal of running the application locally as far as their device capacity allows. The formulation results in a non-convex mixed integer nonlinear programming (MINLP) problem. We propose a heuristic approach, thanks to KKT condition, that quickly solves the MINLP by obtaining closed formulas for the approximate number of edge and cloud resources. Our approach is compared with BARON state-of-the-art-solver under the scenario that the platform knows all the users' parameters. We considered for validation a multiplicity of realistic scenarios, stressing the code and verifying its scalability. Results show that our approach gives a sub-optimal solution in less than 2.5 seconds, while BARON does not converge in two hours for more than 100 users. In addition, the difference in profit in small scale problem instances is less than 1%. Moreover, we propose an algorithm to estimate the best profit for the platform provider in case it does not know the sensitive parameters of users. In this case, the profit estimate loses less than 1% from the scenario with full-knowledge with a worst-case execution time of about 3 seconds.
Finally, we analyze the convenience for an edge provider in solving the Stackelberg game rather than setting a single price for its users. We observed that by solving the game the platform makes an extra profit of between 49% and 195%, with a cost reduction for resource usage between 40% and 50%.

Key-words: Stackelberg game, Mobile Edge Cloud system, DNN partitioning

Abstract in lingua italiana

La crescita, lo sviluppo e la commercializzazione di tecnologie basate sull'intelligenza artificiale stanno portando alla necessità di una sempre maggiore potenza di calcolo. Ne sono chiari esempi le auto a guida autonoma, i visori di realtà aumentata, i chatbot e gli assistenti virtuali. La maggior parte di queste applicazioni si basa su reti neurali profonde, le quali comportano un aumento della potenza di calcolo necessaria a soddisfare la domanda degli utenti. Questa domanda, tuttavia, non può essere soddisfatta né dai dispositivi locali, a causa dell'insufficiente capacità di elaborazione, né dai cloud data center, a causa dell'elevata latenza di trasmissione dovuta alla lunga distanza. Pertanto, l'edge cloud computing pone rimedio a questo problema gestendo direttamente le richieste degli utenti consentendo l'accesso a data center local attraverso reti 5G, riducendo così la latenza di trasmissione tra i dispositivi locali e le risorse computazionali, e fornendo un'adeguata potenza di calcolo.
In questa tesi, modelliamo un sistema di Mobile Edge Cloud per un'applicazione basata su una singola rete neurale. L'interazione tra molteplici utenti mobili e la piattaforma edge viene definita come un gioco di Stackelberg ad un leader e più follower. Il fornitore della piattaforma è il leader che ha l'obiettivo di massimizzare il proprio profitto, proponendo una corretta tariffazione per l'applicazione e minimizzando il costo delle risorse, garantendo al tempo stesso un vincolo sul tempo di risposta. Gli utenti hanno il ruolo di follower con l'obiettivo di eseguire l'applicazione in locale secondo quanto consentito dalle capacità del proprio dispositivo. Tale formulazione risulta in un problema di programmazione non lineare intera mista non convesso. Proponiamo un approccio euristico, grazie alle condizioni di KKT, che risolve rapidamente il problema, ottenendo soluzioni in forma chiusa per la stima del numero di risorse edge e cloud. Il nostro approccio viene confrontato con il solver BARON nello scenario in cui la piattaforma è a conoscenza di tutti i parametri degli utenti. Abbiamo considerato una molteplicità di scenari realistici per la validazione del metodo proposto, verificando la scalabilità del codice sviluppato.
I risultati mostrano che il nostro approccio fornisce una soluzione sub-ottimale in meno di 2.5 secondi, mentre il solver BARON non converge ad una soluzione in due ore per più di 100 utenti. Inoltre, la differenza di profitto in casi significativi risulta inferiore all'1%. Proponiamo, inoltre, un algoritmo per stimare il maggior profitto per il fornitore della piattaforma nel caso in cui quest'ultima non sia a conoscenza dei parametri sensibili degli utenti. In tal caso, la stima del profitto risulta inferiore di meno dell'1% rispetto allo scenario in cui la piattaforma sia a piena conoscenza dei parametri sensibili degli utenti, con un tempo di esecuzione di circa 3 secondi nello scenario peggiore.
Infine, analizziamo la convenienza per un fornitore di servizi di edge computing nel risolvere il gioco di Stackelberg piuttosto che imporre un singolo prezzo agli utenti. Si osserva che la risoluzione del gioco comporta un extra profitto compreso tra il 49% e il 195% per la piattaforma, con una riduzione dei costi compresa tra il 40% e il 50% dimostrando l'importanza pratica per i fornitori di servizi dell'approccio proposto.

Parole chiave: Gioco di Stackelberg, sistema Mobile Edge Cloud, partizionamento DNN

Contents

1. Introduction

Artificial Intelligence (AI) and Deep Learning (DL) have experienced significant growth in recent years, thanks in large part to the advancements made in cloud computing and related technologies, which have helped to accelerate progress in these areas. In fact, the International Data Corporation (IDC) predicts total spending on AI Services to be \$52.6 billion by 2025, at a Compound Annual Growth Rate (CAGR) of 21.9% [28]. By combining AI with cloud computing, it becomes possible to create a vast network that can handle massive amounts of data and generate predictions, all while constantly learning and improving its performance.

In recent times, AI applications have been increasingly geared towards mobile computing and the Internet of Things (IoT). IoT has become prevalent in both business and daily life, opening up numerous opportunities for new and innovative services across a variety of domains such as virtual reality (VR) [36], augmented reality (AR) [16], autonomous vehicles [11], multiplayer games [20], image processing for facial recognition, and natural language processing for real-time translation systems [2]. Industrial IoT, logistics [37], e-health, smart cities, smart agriculture, and smart factories are just a few other examples of domains that are benefiting from the growth of IoT. The rapid proliferation of IoT devices serves as a clear indication of this progress, with the number of connected IoT devices having reached approximately 23 billion in 2018. This number is projected to exceed 75 billion by 2025 and may even reach 125 billion by 2030, which would mean that each consumer would have access to around 15 connected devices [1].

The rise of AI-based IoT applications has brought to light the challenges of processing big data on mobile devices, which are often limited in terms of processing power, memory, and storage. Cloud computing has emerged as a prevalent solution to support these applications for resource-constrained devices by offloading tasks to the cloud, where specialized hardware for AI acceleration can provide powerful task inference capabilities. However, this approach introduces the challenge of long-distance transmission of data, which can result in high transmission delays and costs, and even network and cloud server overload if excessive task offloading occurs [3]. Although cloud computing offers unlimited storage and a large processing capacity via powerful virtual servers, it is unable to meet the needs of real-time processing applications due to the centralized fashion, which results in large end-to-end delay and high network bandwidth utilization. Moreover, the centralized approach leads to increasing delays, which are not tolerated by most IoT applications. To address these challenges, more computation power and real-time response are needed. While cloud computing is a reliable solution to provide computation power, the real-time response is not guaranteed. As the amount of data generated by IoT devices continues to increase rapidly, with 1.7 MB of data created every second for every person on Earth in 2020 [10] and self-driving cars generating about 1 GB of data per second [25], the need for processing this data in real-time has become more critical. Most current applications have strict response time requirements, and today's users tend to be delay and jitter sensitive. Violating real-time requirements substantially degrades their Quality of Experience (QoE) and drives down service providers' revenue.

Edge computing represents a new era of computing paradigm that aims to move AI and machine learning closer to where data generation actually takes place. Some applications generate large amounts of data that can overload networks, making centralized cloud computing unsuitable. Mobile edge computing (MEC) is a promising approach in the 5G era, as it utilizes edge cloud, which has the benefit of low communication costs and relieves the burden of the core network [19]. While edge computing is becoming a promising approach in mitigating issues in cloud computing, it cannot be a complete substitute because of limited computing and storage capacities of edge nodes compared to traditional clouds. However, MEC aims to bring computation closer to the end-user, improving management and efficiency of geographically distributed mobile devices and applications. By shifting computation from the cloud to edge, response time is reduced, and timely decisions can be made, which is of high priority. In fact, edge computing is already processing a significant amount of data. According to the Cisco Global Cloud Index, in 2020, 45% of the data produced by things, people, and machines were analyzed, processed, stored, and acted upon by edge computing [32]. As a result, Linux Foundation believes that by 2025, edge computing will overtake cloud computing [32].

Recent AI applications, particularly in the field of Augmented Reality (AR), heavily rely on neural networks and Deep Neural Networks (DNN) with numerous layers. This is evident in the use of AI technologies such as machine learning and deep learning, which are ideal for AR applications [12] due to their ability to gather data from a camera as well as integrate other information from the device's sensors, accelerometers, gyroscopes, and GPS. Cloud processing in navigation systems is an example of a novel AI application that utilizes scene descriptions and integrates data from gyroscopes and GPS positions to ensure a secure route. The scarcity of resources in IoT devices can have a significant impact on the performance of applications with Quality of Service (QoS) response time constraints. To address this issue, the edge computing paradigm is an effective solution that can enhance the performance of AI-enabled applications while keeping operating costs low. One common approach is to partition the DNN of an AI application component and assign the partitions to the available resources on the edge or cloud. This approach optimizes resource utilization while meeting QoS constraints [21]. DNNs have natural partition points in their various layers, making it easier to partition them. Tasks can

be offloaded from IoT devices to the associated edge device and then to the cloud. The multiple segments of the DNN are suitable with of different partition points which require different amounts of computation and intermediate data transmission between IoT devices and the cloud. The ideal partition point for a DNN depends not only on the topology but also on the communication and computing resources of the system. Consequently, an efficient resource allocation strategy that maximizes resource utilization in edge-cloud-assisted IoT environments must be closely tied to the DNN partition strategy [15]. Therefore, to fully utilize the available resources in edge-cloud-assisted IoT environments, it is critical to use DNN partitioning strategies to optimize resource allocation and meet QoS requirements. This approach can help to overcome the challenge of resource scarcity in IoT devices while ensuring that AI-enabled applications can deliver high-quality performance.
The difficulty of allocating resources across the computing continuum arises from the limited capacity of mobile devices and high costs of running applications in edge/cloud environments. To address this challenge, game theory models are necessary to support the execution of AI applications. These models must consider two main factors: first, the memory and energy available on IoT devices and the user's budget when executing on the field; and second, the costs of guaranteeing a specific latency threshold and the competition for resources at the edge provider site (e.g., a set of small datacenters that can be accessed through 5G towers) and in the remote cloud. Many existing works have focused on partitioning DNNs for AI applications, assuming that some parts can be executed on mobile devices while others require edge or cloud resources.
The focus of this work is a Mobile Edge Computing (MEC) system that comprises an edge platform, with a limited number of edge servers, connected to a cloud-based pool of unlimited virtual machines (VMs) through a fast network, and a large number of mobile users with smart devices. These users want to run an AI application and they are connected to the edge platform through a network with non-negligible latency. Each user has energy and memory limitations to run the AI application, and a predefined response time performance constraint must be met for all users. For the sake of simplicity, we consider a single DNN-based application with different *deployments*, which represent different ways of partitioning the DNN. We have developed two models, one with direct communication between each device and the edge platform, consisting of at most two partitions of the DNN, and one where smart devices require the support of a mobile phone for both running the application and communicating with the edge platform, considering at most three partitions of the DNN application. Although the models differ in formulation, we will use the same approach to solve the related problems. The users can either choose an appropriate deployment based on their budget, energy, and memory or not run the application. On the platform side, the edge platform has to, on one hand, provide the sufficient resources for the users to guarantee the performance constraint and on the other hand, propose an appropriate price as application fee to incentivize the users to run the application. According to the selfish behaviour of the edge platform and users, we model the interaction among the edge platform and users as a Stackelberg game, where the platform, as a leader, aims to maximize its own profit by charging a reasonable price and minimizing the number of resources while meeting the performance constraint, and the users, as followers, want to minimize their cost by choosing a deployment with lowest price as far as their device capacity allows.
Furthermore, we examine two distinct scenarios in our study. The first scenario assumes that the platform has access to all user parameters through an agent running on user devices. In contrast, in the other scenario, the platform is unaware of the sensitive parameters of the users. To the best of our knowledge, this is the first work aiming at designing a mechanism that encourages users to run an AI application based on their satisfaction level, with the assumption that each user assigns a value to indicate their satisfaction. Users will only be motivated to run the application if the cost is less than their satisfaction value.
In summary, the main contributions of this work are the following:

1. We formulate the interaction among multiple mobile users and edge platform as a one-leader multi-follower Stackelberg game. It results in a non-convex mixed integer nonlinear programming problem (MINLP), which is very difficult to be solved.
2. We develop an heuristic approach that relies on KKT condition to find a closed form for providing the optimal number of edge and cloud resources, and on the fact that the platform profit results in a piecewise linear increasing function. The resulting approach solves the mixed integer nonlinear problem quickly.
3. We evaluate the performance of the proposed approach by comparing with BARON state-of-the-art solver [5] under the assumption of full knowledge of users' parameters. Experimental results show that our approach could find the optimal solution much faster than BARON. In particular, the heuristic algorithm returns a sub-optimal solution in less than 2.5 seconds, while BARON does not converge for more than 100 users in two hours. Moreover, the difference in profit in small scale problem instances is less than 1%.
4. Finally, we develop an algorithm to estimate the best profit for the edge provider under the assumption of partial knowledge of sensitive users' parameters. Experimental results show that our proposed approach takes about 3 seconds for largest scale system with 1000 users.

The thesis is structured as follows. Section 2 discusses related works. Section 3 outlines our approach, specifically, we describe the MEC system model and formulate the problems from both the platform and user perspectives for both the two and three-partition scenarios. In Section 4, we formulate the problem as a Stackelberg game. The proposed resolution approach is explained in Section 5, and we reformulate an approximate model for the Stackelberg game as a convex problem. In addition, we propose a solution method in the case where the platform is unaware of users' sensitive data. Experimental results are presented in Section 6. Finally, Section 7 provides conclusions and possible developments.

2. Related Work

The topic of placing components in edge and cloud computing is becoming more popular in both academic research and industry. In [6] a taxonomy of optimization problems related to edge/cloud computing is presented. These problems are evaluated by considering various factors such as their objective functions (linear or non-linear, black or white box) and constraints (e.g., on latency, privacy and security of data, throughput, energy consumption of local and edge devices), as well as costs and profits.

Several studies have investigated the placement of multi-component applications, such as [35], [4], and [30]. In [35], the authors represented the application components as a linear graph and the physical devices as physical graph nodes. They proposed an offline algorithm that used a Mixed Integer Linear Program (MILP) solved by CPLEX to map the application graph to a tree physical graph while considering performance guarantees. They also proposed online approximation algorithms with polynomial-logarithmic time complexity for tree application graph placement using the solution of the offline problem. Likewise, [4] approached the multi-component application placement problem as a MILP and solved it with CPLEX in the offline version. For the online version, they initially determined the best matching between components and the edge servers using the Hungarian algorithm without considering the component communication cost. Subsequently, they used a local search algorithm to enhance the solution by taking into account the communication cost among the components. In [30], the authors represented the application components as a Direct Acyclic Graph (DAG) and took into account the presence of diverse resources in edge and cloud environments. They formulated the problem of component placement and resource selection as a Mixed Integer Linear Program (MINLP) and suggested a randomized greedy algorithm to address it.

Another area of research in component placement involves AI applications, in particular deep neural networks (DNNs), with a focus on finding the optimal partition placement. For instance, in [23], a hierarchical AI learning framework on the Mobile-Edge-Computing (MEC) paradigm was proposed, which employs a novel hybrid parallelism method to partition and assign DNN model layers and data samples across edge devices, edge servers, and cloud centers. The authors cast the problem of scheduling DNN training tasks in both layer partitioning and data partitioning as an integer linear problem (ILP). By solving this optimization problem, they were able to achieve the minimum training time. To tackle the partition placement problem for DNN inference on mobile web and edge, Huang et al. [18] assume that there is only one partitioning point, dividing the DNN model into two parts. They aim to optimize two objectives, namely latency and mobile energy, while constraining each objective to an upper bound. To solve this multi-objective optimization problem, the authors use the weighted sum of latency and mobile energy, which yields a linear complexity solution, and find the optimal partition point. The authors of [17] built upon their previous work presented in [30] and focused on DNN components. They formulated the problem of partition placement and resource selection as a MINLP and proposed a random greedy algorithm to find the optimal solution.

Other researchers have also investigated DNN partition placement in order to optimize both training and inference time. For example, Huang et al. [19] proposed a lightweight DNN using a binary neural network (BNN) to execute the DNN on mobile devices, with the aim of reducing the size of the DNN and thus minimizing mobile energy consumption and latency. They also developed a training method for the lightweight DNN and an inference library to run the BNN. To maximize the resource utilization of edge devices, they proposed an online scheduling algorithm based on deep reinforcement learning (DRL). In their paper [14], the authors present a method for executing DNNs on both mobile devices and the cloud in a cooperative manner to optimize mobile cloud computing scheduling while reducing mobile energy consumption and latency. They represent a DNN as a DAG and reduce the mobile cloud computing optimal scheduling problem to the shortest path problem. If there are no constraints, such as mobile battery limitations or QoS constraints, the optimal scheduling can be obtained by solving the shortest path problem. However, if there are limited resources on the cloud server, the authors formulate an ILP for both training and inference phases. They compute the execution time for computation and communication on both the mobile device and cloud server, and minimize them by considering a bound for the execution time constraint on the cloud server to alleviate the server load. In scenarios with QoS constraints, the authors minimize the required energy consumption on the mobile device while meeting a specified deadline for the total time spent on both the cloud and mobile device.

Finally, [24] proposes a framework for optimizing server placement and workload allocation in MEC environments using Reinforcement Learning (RL). The proposed framework uses a two-stage RL algorithm that first decides the optimal placement of servers, and then allocates workloads to the servers based on their capacities and workload requirements. The framework is evaluated through simulations, and the results show that it outperforms other benchmark algorithms in terms of workload completion time, resource utilization, and cost. The

N	**Execution time** (s)											
	$\lvert\mathcal{D}\rvert = 3$				$\lvert\mathcal{D}\rvert = 4$				$\lvert\mathcal{D}\rvert = 5$			
	N/5	N/10	N/20	N/40	N/5	N/10	N/20	N/40	N/5	N/10	N/20	N/40
100	0.10	0.06	0.04	0.06	0.11	0.06	0.04	0.05	0.11	0.06	0.04	0.07
250	0.33	0.18	0.11	0.08	0.36	0.20	0.13	0.09	0.41	0.22	0.14	0.10
500	0.87	0.50	0.29	0.20	1.02	0.62	0.33	0.22	1.19	0.69	0.39	0.24
750	1.70	0.97	0.60	0.33	2.01	1.12	0.69	0.39	2.36	1.31	0.80	0.46
1000	3.01	1.62	0.93	0.53	3.56	1.98	1.16	0.64	4.15	2.20	1.31	0.69

Table 22: Execution time Follow the line Algorithm.

Results show that with few users we need a larger fraction of points to have a low profit ratio, i.e. a more precise estimate of the best profit, such as for $N = 100$. We observed that in the three-partition scenario the algorithm needs more points to obtain a good estimate of the optimal profit than in the two-partition case. On the other hand, for a large N it is sufficient to pick $N/20$ points to have a profit ratio lower than 1%, for $|\mathcal{D}|= 3$. Note that for high N a large number of analyzed points does not guarantee a better estimate of profit. This happens because the platform profit function presence of local maxima, as shown in Figure 35, for which an increase in the number of analyzed points is indifferent. Therefore, with $N/5$ points a different parameter setting may be necessary for the purpose of obtaining a better profit estimate. As for the execution time, it decreases with the number of analysed point and in the worst case, with $N = 1000$, $|\mathcal{D}| = 5$ and $fraction\ of\ total\ points = N/5$ it turns out to be just over 4 s. Considering $N/10$ total points, as in the two-partition scenario, the execution time remains under 3 s, which is less than the execution time of the chosen heuristic with full knowledge of users' parameters (see Figure 32).

7. Conclusions and future developments

In this thesis, we modeled a Mobile Edge Cloud (MEC) system with a large number of users who are willing to run an AI application based on a single DNN with multiple alternative deployments. The users might choose one of the deployments according to their budget constraints and the constraints imposed by their local devices. We formulated two slightly different models, which differ in the number of partitions in the neural network. In the first scenario, with two partitions, smart devices communicate directly with the edge platform, while in the second scenario, with three partitions, smart devices such as AR glasses need other devices, such as smartphones, both to distribute the load and to communicate with the platform. For both scenarios, we formulated the interaction between the users and the edge platform as a Stackelberg game in which the edge platform is the leader and the users are the followers. The users aim to maximize their profit by choosing the cheapest deployment as far as their device capacity allows while the platform decide about the price of deployments and the number of resources in the edge and cloud with the goal of maximizing its profit while paying attention to the performance constraint. The model results in a non-convex mixed integer nonlinear programming problem (MINLP).
We, therefore, developed a heuristic algorithm that solves the problems very quickly by tracing it back to an approximated convex subproblem that would determine the optimal number of edge servers and cloud VMs, based on the KKT conditions. The core idea on which our approach is based is that the maximum value of an increasing, piecewise linear function, such as the edge platform profit, is a point of discontinuity. The latter are due to drops from the system or changes in the choice of deployments, both caused by price increases.
Given the heuristic nature of our approach, there is no guarantee for optimality of the solutions. Therefore, we compared the solutions of the heuristic approach with those of the BARON global solver. The results show that for a significant number of users, the heuristic approach loses less than 1% in terms of platform profit compared with BARON, but in an execution time of a few seconds compared to the hours taken by the global solver. Specifically, in the worst case, i.e., 1000 users and 5 deployments, the heuristic algorithm solves the problem in less than 2.5 seconds in the two-partition scenario and in less than 2 seconds in the three-partition scenario, while BARON does not converge to an optimal solution with more than 100 users in two hours.
Finally, we developed the *Follow the Line* Algorithm 5, which estimates the best profit for the edge platform without knowing sensitive users' parameters. In this case, the edge platform is not able to anticipate users' choices and, therefore, to calculate discontinuity points. The proposed algorithm selects in a smart way the points to be analyzed to maximize the profit function. Experimental results show that in both the two-partition and the three-partition scenarios, the Follow the Line algorithm 5 succeeds in estimating a profit less than 1% lower than the heuristic approach with full knowledge, in an execution time of about 3 s in the former scenario and less than 2.5 s in the latter. In addition, a sensitivity analysis on the parameters has demonstrated the robustness of the algorithm itself. In particular, we observed that by varying the parameters by ±50% from the

tuned ones, the range of variability of the profit ratio of the Follow the Line algorithm with respect to the full knowledge scenario is less than 1.5%.
Since the proposed approach assign the load of deployments, in a specific order, to the edge up to saturate it, and then to the cloud, there is only one deployment that is served both by the edge and cloud while the optimal solution could be the one with multiple deployments served both by the edge and cloud. Future work, can improve the current solution by moving the users with different deployments from edge to cloud and vice versa. In addition, the proposed model could be further developed for an online setting, where some users join the system while some others are finishing running the application and release their resources. In this case, it might be interesting to consider a scenario where the users who have already access to the resources and are already priced do not participate in the Stackelberg game, but still can access their resource pool. On the other hand, new users could be allocated to servers/VMs that are not fully saturated, without increasing the number of computational resources. If this is not possible, new users would be offered a price that corresponds to a new solution of the Stackelberg game, in which the remaining computational resources are kept unchanged. Finally, the performance of the algorithms, in terms of execution time, can be further improved by porting our Python implementation to C++.

References

[1] Shadi Al-Sarawi, Mohammed Anbar, Rosni Abdullah, and Ahmad B Al Hawari. Internet of things market analysis forecasts, 2020–2030. In *2020 Fourth World Conference on smart trends in systems, security and sustainability (WorldS4)*, pages 449–453. IEEE, 2020.

[2] Taha Alfakih, Mohammad Mehedi Hassan, Abdu Gumaei, Claudio Savaglio, and Giancarlo Fortino. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa. *IEEE Access*, 8:54074–54084, 2020.

[3] AR Arunarani, Dhanabalachandran Manjula, and Vijayan Sugumaran. Task scheduling techniques in cloud computing: A literature survey. *Future Generation Computer Systems*, 91:407–415, 2019.

[4] T. Bahreini and D. Grosu. Efficient placement of multi-component applications in edge computing systems. In *IEEE SEC*, 2017.

[5] MINLP: BARON. `https://minlp.com/baron-solver`.

[6] J. Bellendorf and Z. Ádám Mann. Classification of optimization problems in fog computing. *FGCS*, 107:158–176, 2020.

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[8] Cheng Chen, Xiaogang Chen, Dibakar Das, Dmitry Akhmetov, and Carlos Cordeiro. Overview and performance evaluation of wi-fi 7. *IEEE Communications Standards Magazine*, 6(2):12–18, 2022.

[9] Eduard Ionut Chirica. An approach and a tool for the performance profiling and prediction of partitioned Deep Neural Networks in Computing Continua Environments. *Politecnico di Milano thesis*, 2022.

[10] "data our new natural resource.". `https://www.demandcaster.com/blog/data-our-new-resource/`.

[11] Alisson Barbosa De Souza, Paulo AL Rego, Tiago Carneiro, Jardel Das C Rodrigues, Pedro Pedrosa Reboucas Filho, Jose Neuman De Souza, Vinay Chamola, Victor Hugo C De Albuquerque, and Biplab Sikdar. Computation offloading for vehicular environments: A survey. *IEEE Access*, 8:198214–198243, 2020.

[12] J. S. Devagiri, S. Paheding, Q. Niyaz, X. Yang, and S. Smith. Augmented reality and artificial intelligence in industry: Trends, tools, and future challenges. *Expert Systems with Applications*, 207:118002, 2022.

[13] Average monthly electricity wholesale price in italy from january 2019 to january 2023. `https://www.statista.com/statistics/1267548/italy-monthly-wholesale-electricity-price/`.

[14] A. E. Eshratifar, M. S. Abrishami, and M. Pedram. Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services. *IEEE Transactions on Mobile Computing*, 20(2):565–576, 2021.

[15] Wenhao Fan, Li Gao, Yi Su, Fan Wu, and Yuan'an Liu. Joint dnn partition and resource allocation for task offloading in edge-cloud-assisted iot environments. *IEEE Internet of Things Journal*, 2023.

[16] Paula Fraga-Lamas, Tiago M Fernandez-Carames, Oscar Blanco-Novoa, and Miguel A Vilar-Montesinos. A review on industrial augmented reality systems for the industry 4.0 shipyard. *Ieee Access*, 6:13358–13375, 2018.

[17] D. Ardagna H. Sedghani, F. Filippini. A Random Greedy based Design Time Tool for AI Applications Component Placement and Resource Selection in Computing Continua. In *IEEE International Conference on Edge Computing (EDGE)*, 2021.

[18] Y. Huang, X. Qiao, S. Dustdar, and Y. Li. AoDNN: An Auto-Offloading Approach to Optimize Deep Inference for Fostering Mobile Web. In *IEEE INFOCOM*, 2022.

[19] Y. Huang, X. Qiao, P. Ren, L. Liu, C. Pu, S. Dustdar, and J. Chen. A lightweight collaborative deep neural network for the mobile web in edge cloud. *IEEE Transactions on Mobile Computing*, 2020.

[20] Mike Jia, Jiannong Cao, and Weifa Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, 5(4):725–737, 2015.

[21] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *ACM ASPLOS '17*, 2017.

[22] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models.* Prentice-Hall, 1984.

[23] D. Liu, X. Chen, Z. Zhou, and Q. Ling. Hiertrain: Fast hierarchical edge ai learning with hybrid parallelism in mobile-edge-cloud computing. *IEEE Open Journal of the Communications Society*, 1:634–645, 2020.

[24] Anahita Mazloomi, Hani Sami, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. Reinforcement learning framework for server placement and workload allocation in multiaccess edge computing. *IEEE Internet of Things Journal*, 10(2):1376–1390, 2022.

[25] Lucas Mearian. Self-driving cars could create 1gb of data a second. *Computerworld*, 23, 2013.

[26] Open Signal, Italy, May 2022, 5G Experience Report. `https://www.opensignal.com/reports/2022/05/italy/mobile-network-experience-5g`.

[27] Oscar - open source serverless computing for data-processing applications. `https://github.com/grycap/oscar`.

[28] David Schubmehl. Worldwide Artificial Intelligence Software Platforms Forecast, 2019–2023. `https://www.idc.com/getdoc.jsp?containerId=prUS48881422`, 2019.

[29] H. Sedghani, D. Ardagna, M. Passacantando, M. Z. Lighvan, and H. S. Aghdasi. An incentive mechanism based on a Stackelberg game for mobile crowdsensing systems with budget constraint. *Ad Hoc Networks*, 123:102626, 2021.

[30] H. Sedghani, F. Filippini, and D. Ardagna. A Randomized Greedy Method for AI Applications Component Placement and Resource Selection in Computing Continua. In *IEEE International Conference on Joint Cloud Computing (JCC)*, 2021.

[31] H. Sedghani, M. Z. Lighvan, H. S. Aghdasi, M. Passacantando, G. Verticale, and D. Ardagna. A stackelberg game approach for managing ai sensing tasks in mobile crowdsensing. *IEEE Access*, 10:91524–91544, 2022.

[32] Zubair Sharif, Low Tang Jung, Imran Razzak, and Mamoun Alazab. Adaptive and priority-based resource allocation for efficient resources utilization in mobile edge computing. *IEEE Internet of Things Journal*, 2021.

[33] The Camera Basics for Visual SLAM. `https://www.kudan.io/blog/camera-basics-visual-slam/`.

[34] U. Tadakamalla and D. A. Menasce. Autonomic resource management for fog computing. *IEEE TCC*, pages 1–1, 2021.

[35] S. Wang, M. Zafer, and K. K. Leung. Online placement of multi-component applications in edge computing environments. *IEEE Access*, 5:2514–2533, 2017.

[36] Renchao Xie, Qinqin Tang, Qiuning Wang, Xu Liu, Fei Richard Yu, and Tao Huang. Collaborative vehicular edge computing networks: Architecture design and research challenges. *IEEE Access*, 7:178942–178952, 2019.

[37] Peiyun Zhang, Yutong Chen, Mengchu Zhou, Ge Xu, Wenjun Huang, Yusuf Al-Turki, and Abdullah Abusorrah. A fault-tolerant model for performance optimization of a fog computing system. *IEEE Internet of Things Journal*, 9(3):1725–1736, 2021.

A. Appendix A

This appendix contains the proofs of the theorems stated within this thesis. In particular, Section A.1 shows that (106) and (117) are convex response time constraints. The proof of the theorem for the two-partition scenario is given in Section A.2, while Section A.3 shows that, considering the transmission delay between edge and cloud within the response time constraint, there is no closed solution to the associated KKT system. Sections A.4 and A.5 provide proofs for the theorems of the OnlyEdge and OnlyCloud scenarios, respectively, with global response time. Similarly, Sections A.6 and A.7 provide proofs considering the local response time. Finally, in Section A.8 we show that the optimal price solution for the platform lays at one of the points of discontinuity or in a left neighborhood of a point of discontinuity of the profit function.

A.1. Proof of **Theorem 5.3** and **Theorem 5.6**

Proof. We denote the response time of edge resources for deployment k as follows:

$$f(n_e^{(k)}, \lambda_e^{(k)}) = \frac{\lambda_e^{(k)}}{\lambda^{(k)}} \cdot \frac{D_e^{(k)} n_e^{(k)}}{n_e^{(k)} - D_e^{(k)} \lambda_e^{(k)}} \tag{136}$$

The first derivative of f respect to $n_e^{(k)}$ is as follows:

$$\frac{\partial f(n_e^{(k)}, \lambda_e^{(k)})}{\partial n_e^{(k)}} = -\frac{1}{\lambda^{(k)}} \frac{(\lambda_e^{(k)})^2 (D_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^2} \tag{137}$$

And the second derivative of f respect to $n_e^{(k)}$ is as follows:

$$\frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial {n_e^{(k)}}^2} = \frac{1}{\lambda^{(k)}} \frac{2(\lambda_e^{(k)})^2 (D_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3} \tag{138}$$

The first derivative of f respect to $\lambda_e^{(k)}$ is as follows:

$$\frac{\partial f(n_e^{(k)}, \lambda_e^{(k)})}{\partial \lambda_e^{(k)}} = \frac{1}{\lambda^{(k)}} \cdot \frac{D_e^{(k)} (n_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^2} \tag{139}$$

And the second derivative of f respect to $\lambda_e^{(k)}$ is as follows:

$$\frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial {\lambda_e^{(k)}}^2} = \frac{1}{\lambda^{(k)}} \cdot \frac{2(D_e^{(k)})^2 (n_e^{(k)})^2}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3} \tag{140}$$

The Partial derivatives is:

$$\frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial \lambda_e^{(k)} \partial n_e^{(k)}} = \frac{\partial^2 f(n_e^{(k)}, \lambda_e^{(k)})}{\partial n_e^{(k)} \partial \lambda_e^{(k)}} = -\frac{1}{\lambda^{(k)}} \left(\frac{2\lambda_e^{(k)} (D_e^{(k)})^2 n_e^{(k)}}{\left[n_e^{(k)} - \lambda_e^{(k)} D_e^{(k)}\right]^3} \right) \tag{141}$$

So we can write the Hessian of $f(n_e^{(k)}, \lambda_e^{(k)})$ as follows: