الجامعة اللبنانية الأميركية
Lebanese American University

Lebanese American University Repository (LAUR)

Post-print version/Author Accepted Manuscript

Publication metadata

Title: Latency and Reliability-Aware Workload Assignment in IoT Networks with Mobile Edge Clouds

Author(s): Nouha Kherraf, Sanaa Sharafeddine, Chadi Assi, Ali Ghrayeb

Journal: IEEE Transactions on Network and Service Management

DOI/Link: https://doi.org/10.1109/TNSM.2019.2946467

How to cite this post-print from LAUR:

Kherraf, N., Sharafeddine, S., Assi, C., & Ghrayeb, A. (2019). Latency and Reliability-Aware Workload Assignment in IoT Networks with Mobile Edge Clouds. IEEE Transactions on Network and Service Management, DOI, 10.1109/TNSM.2019.2946467, http://hdl.handle.net/10725/11509

© Year 2019

"© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works."

This Open Access post-print is licensed under a Creative Commons Attribution-Non Commercial-No Derivatives (CC-BY-NC-ND 4.0)

cc BY NC ND

This paper is posted at LAU Repository

For more information, please contact: archives@lau.edu.lb

Latency and Reliability-Aware Workload Assignment in IoT Networks with Mobile Edge Clouds

Nouha Kherraf, Sanaa Sharafeddine, Chadi Assi, Ali Ghrayeb

***Abstract*—Along with the dramatic increase in the number of IoT devices, different IoT services with heterogeneous QoS requirements are evolving with the aim of making the current society smarter and more connected. In order to deliver such services to the end users, the network infrastructure has to accommodate the tremendous workload generated by the smart devices and their heterogeneous and stringent latency and reliability requirements. This would only be possible with the emergence of ultra reliable low latency communications (uRLLC) promised by 5G. Mobile Edge Computing (MEC) has emerged as an enabling technology to help with the realization of such services by bringing the remote computing and storage capabilities of the cloud closer to the users. However, integrating uRLLC with MEC would require the network operator to efficiently map the generated workloads to MEC nodes along with resolving the trade-off between the latency and reliability requirements. Thus, we study in this paper the problem of Workload Assignment (WA) and formulate it as a Mixed Integer Program (MIP) to decide on the assignment of the workloads to the available MEC nodes. Due to the complexity of the WA problem, we decompose the problem into two subproblems; Reliability Aware Candidate Selection (RACS) and Latency Aware Workload Assignment (LAWA-MIP). We evaluate the performance of the decomposition approach and propose a more scalable approach; Tabu meta-heuristic (WA-Tabu). Through extensive numerical evaluation, we analyze the performance and show the efficiency of our proposed approach under different system parameters.**

I. Introduction

The Internet of Things (IoT) paradigm has emerged to shape the future of the Internet since 1999 [1]. It is quickly evolving to be an intrinsic part of our daily lives, where everyday's objects are embedded with transceivers, protocols and microcontrollers turning them into smart things that can communicate with each other [2]. In the current era of smart things, novel use cases (e.g., Tactile Internet, Intelligent Transportation Systems (ITS), Tele-surgery, etc.) are contributing to the evolution of mega smart cities [3] [4]. Such cities exploit those applications to enhance the quality of life by providing seamless services (i.e., e-health care, intelligent transportation, smart everything, etc) to end users. However, in order to offer the immersive experience these applications promise, low latency (within milliseconds) and high reliability (less than 10^{-5} packet loss rate) are indispensable [5]. Such requirements would only be possible with the emergence of ultra-reliable low latency communications (uRLLC) promised by the fifth generation mobile communication system ($5G$).

The terms reliability and latency are indeed broad terms which encompass different meaning based on their definitions. For instance, latency could be defined as the end-to-end latency caused by transmission, queuing, and processing delays [6]. Further, Reliability - in general - denotes the probability of successfully transmitting packets over a period of time. Reliability could, however, also refer to the availability of a service a particular application demands [7]. Hence, it is essential to properly define both terms. For the sake of clarity, in the rest of this paper the term $latency$ is defined as the end-to-end latency and $reliability$ as the availability of the service; for instance, an availability of 99.999% means that the service is available to end users 99.999% of the time [8].

Within this uRLLC realm, future networks will need to be designed upon three building blocks; $i)Scale$: accommodation of the huge number of devices and the volume of data they produce $ii)Risk$: robustness towards uncertainty and sudden system changes and $iii)Tail$: dealing with the tail behavior of traffic distribution considering the heterogeneity and randomness of latency and reliability requirements [6]. Building such network is plausible by leveraging technologies such as pervasive AI, massive network densification, new radio (NR) and advanced wireless access, exploiting new spectrum at high frequency bands (e.g. mmWave) [6], [9], [10]. One of the rather appealing enablers for this vision is Mobile Edge Computing (MEC) [11], [12]. MEC is an emerging computation paradigm that was introduced to overcome the high latency and low reliability of communication resulting from the large distance between the end users and the cloud [13]. In an MEC infrastructure, the cloud capabilities are brought closer to the IoT devices (e.g. smart cameras, industrial sensors, etc.) by equipping the nearby 4G/5G Base Stations (BSs) or WiFi Access Points (APs) with computing and storage capabilities, creating an MEC node. Traditionally, the MEC nodes host Virtual Machines (VMs) running different IoT applications to serve the IoT devices. Hence, IoT devices can offload their computationally intensive workloads to the MEC nodes located at a close proximity to them, rather than the distant cloud, and thus, reducing latency and ensuring a more reliable network [12] [14].

Now, to realize the vision of a smart city with massive IoT integration, provisioning necessary MEC resources to fulfil the anticipated uRLLC requirement would ensure a seamless delivery of services and a satisfactory Quality of Experience (QoE) for end users. However, a critical aspect of this adoption is considering the various underlying trade-offs in a uRLLC environment. Some of these trade-offs could arise between latency, reliability, energy consumption, spectral efficiency, SNR, etc,. For instance, a fundamental trade-off, where intensive research has been done, is between latency and energy consumption in which the device would consume more energy by periodically checking for packet delivery; the more frequent checks, the lower the latency but the higher energy consumption [6]. One of the less tackled trade-offs is between latency and reliability. For instance, different devices (smart sensors, smart cameras, etc.) generate a shear volume of data to be offloaded to the MEC nodes to provide a specific type of service (Tactile Internet, Process Automation, etc.). Since each service has both reliability and latency requirements, offloading the workload to an MEC node that satisfies the latency constraint does not guarantee achieving the required reliability and vice versa. Hence, a decision needs to be made on which MEC node the workload should be processed to satisfy and optimize both requirements. Moreover, the generated workload is subject to different failure scenarios either through accessing the network; communication link failures (due to jamming and equipment failure) or being processed on an MEC node; MEC node failure (DoS attacks, hardware failure) [15], causing the service to be unavailable to the end user. Therefore, achieving the ultra high reliability demanded by the IoT services coupled with the aforementioned failure scenarios may require repeated transmissions which would incur higher latency [6]. Some work has been done to address the reliability and latency in mobile edge computing either jointly or separately. For instance, the authors in [15] considered the probability of occurrences of failure scenarios for both communication and MEC nodes. Further, the work in [13] addressed the problem of workload assignment considering the latency and transmission reliability. However, to the best of our knowledge, no work has considered the reliability of the individual MEC nodes along with the specific reliability and delay requirements of the IoT services.

In this paper, we consider a smart environment integrated with IoT capabilities and devices which enable intelligent and innovative applications and services to various emerging verticals. We assume each enabled service generates a workload with particular requirements in terms of latency and reliability. IoT applications are hosted at the network edge on MEC nodes and we assume that MEC nodes, being hardware with software elements, have their own availability (a hardware/software may fail due to a multitude of reasons as mentioned earlier). We assume the generated IoT load is dispersed across a wide area covered by a communication infrastructure (e.g., 5G and/or WiFi). The generated load demands services with QoS requirements (latency and reliability) which may be provided by IoT applications hosted at the network edge. For instance, a request for process automation generated by a sensor in a smart factory would require to be authenticated (among other possible security processing), and the authentication function is placed in close proximity at the network edge.

We hence address the heterogenous *Workload Assignment (WA)* problem taking into account the various incurred delays as well as the reliability of MEC nodes. We formulate the *WA* problem as a Mixed Integer Program (WA-MIP) with the objective of maximizing the admitted load with respect to the latency and reliability requirements. Intuitively, higher reliability is achieved by replicating the load onto applications on one or more of the MEC nodes; however, this yields higher loads and accordingly higher latencies for servicing the load. This interplay between reliability and latency is also explored. We show that the WA-MIP is NP-Hard, and hence, we propose an efficient decomposition approach (WA-D) that first solves the resiliency problem and then the latency problem. Thus, WA-MIP is decomposed into two sub-problems, 1) The *Reliability Aware Candidate Selection (RACS)* sub-problem which is tackled by a heuristic search to determine the set of potential MEC nodes satisfying the demanded ultra high reliability requirements per IoT service. 2) The *Latency Aware Workload Assignment (LAWA)* sub-problem which is formulated as a MIP that takes the solution of RACS as an input and determines the optimal workload assignment with respect to the latency requirements. Through extensive numerical evaluation we show that the WA-D is more scalable than the WA-MIP, but nonetheless remains unscalable for very large instances. Therefore, we propose a Tabu-search-based meta-heuristic approach (WA-Tabu) to solve the WA problem.

The remainder of this paper is organized as follows. Section II presents the related work, the system model is introduced in III, the problem is formulated as a MIP in IV, the proposed WA-D approach is discussed in V and the WA-Tabu in VI . The numerical evaluation and conclusion are presented in VII and VIII respectively.

II. Related work

One of the rather disruptive advancements in the IoT industry is the introduction of 5G and its promise of uRLLC. Some work has been done on the feasibility of 5G and uRLLC within the context of IoT. Specifically, the authors in [16] studied the use of uRLLC in factory automation and proved its feasibility in factory automation with latency of sub milliseconds and failure rate of 10^{-9}. Further, the work in [17] studied the feasibility of 5G mm-wave as an enabler to Connected Autonomous Vehicles (CAV) applications. It was concluded that the 5G mm-wave satisfied the latency requirements of safety-critical applications and achieved high data rates sufficient for real-time applications (e.g. video streams processing for in-vehicle infotainment system). Further, extensive work has been done exploring the possibilities and use cases of MEC as a key technology for the inception of uRLLC in an IoT environment. For instance, the authors in [18] presented a detailed survey on MEC and its integral part in the development of 5G. The authors explored the various MEC use cases and challenges within the context of IoT and smart cities. Some of the challenges the authors presented was the service orchestration; optimizing the synergies between the different entities in an edge network (MEC nodes dimensioning, applications placements and workload assignment), as well as service enhancements; improving the users' quality of experience (QoE) and achieving resiliency.

A. Latency in MEC & IoT infrastructure

Many research has focused on workload offloading in an MEC infrastructure with an emphasis on the end-to-end latency. For instance, Xiang et al. in [19] considered a network where multiple users are offloading their workload to the geographically distributed MEC nodes. They proposed a latency aware offloading framework that minimizes the total response time incurred by the users' workload. Further, the work done in [14] discussed the provisioning of resources (edge servers and applications) as well as the workload assignment. The authors formulated the problem as a Mixed Integer Program (MIP) with the objective of minimizing the cost with respect to latency requirements of different industry verticals. A more realistic model is when latency is coupled with another system design parameter. For example, in [20], the authors considered the trade-off between energy consumption and latency by addressing workload offloading problem in an IoT environment. Their proposed framework optimizes the energy consumption and the system utility while respecting the latency requirement.

B. Reliability in MEC & IoT infrastructure

Unlike latency, little research has considered the reliability issues in MEC and IoT context, let alone considering the tradeoff between reliability and latency together.

In [13] the tradeoff between latency and reliability was studied. The problem was formulated to jointly minimize the end to end latency and the failure probability of offloading tasks to MEC nodes. Further, the authors only considered the transmission reliability(offloading failure probability), and only one user with one task to be offloaded. The task is partitioned into subtasks, where each is transmitted using the whole channel bandwidth in a sequential manner. It was concluded that the higher channel quality, the better achieved reliability. On the other hand, the authors in [15] formulated an ILP to minimize the operational cost of placing Virtual Process Control Functions (VPFs) on MEC nodes with respect to capacity and resiliency constraints for different failure scenarios. The failures are due to either MEC node failure or communication link failure. Each failure scenario is assigned a probability based on historical data. Due to the complexity of the problem, the authors developed an iterative algorithm that uses the generalized Benders Decomposition and linear relaxation to reduce the search space. Further, the work in [21] considered an environment with uRLLC and targeted the task offloading and allocation problems with the objective of minimizing the users' power consumption. While the authors in [22] assumed that all tasks are offloaded to the network edge and they targeted the tasks assignment problem with the objective of minimizing the tasks' total experienced delay with respect to reliability constraints. Both works [21] and [22] were satisfying the reliability by imposing a probabilistic constraint that ensures that the latency is bounded by a threshold with a specific probability. However, non-of them considered the specific reliability requirements of the services demanded by the workloads.

C. Novelty of our work in comparison to the literature

In the aforementioned work, most of the authors often considered the latency requirements and overlooked the reliability demands. However, the few works that considered the reliability; communication link failure or MEC node failure, considered a single user task and its end to end delay [13], or the latency was neglected [15]. Further, in both works [13] and [15], the specific service reliability and latency requirements were not considered and they were not tailored towards an IoT context. To the best of our knowledge, our work is the first to consider the workload assignment problem in a densely populated, MEC-enabled IoT environment with multiple workloads/IoT devices while considering both ultra high reliability and low latency requirements of the IoT services, and the availability of the MEC nodes.

III. System Model

We depict our system model in Figure 1 which consists of a smart environment where different IoT devices are spatially distributed and connected through a communication network (e.g., nearby WiFi APs or cellular BSs). IoT devices generate a tremendous workload by requesting IoT services enabled by IoT applications of different types and capabilities hosted on MEC nodes. Serving the IoT devices does not only depend on their requested services' types, but also on the service's latency and reliability requirements. In what follows, we formally explain the network, reliability and latency models.

A. Network model

Formally, the network is represented by a graph $G(N, E)$ where E is the set of communication links connecting a set of nodes N. $N(= L \cup R)$ is composed of communication elements, such as APs/BSs, dispersed at different locations $l \in L$, and the networking equipment R (routers, switches, etc.). We assume a set M of MEC nodes deployed nearby the APs/BSs, and hence, we denote by π_m^l that MEC node $m \in M$ is located at $l \in L$. IoT devices are connected to their nearby WiFi APs or cellular BSs located at locations $l \in L$ to access services provided by IoT applications hosted on MEC nodes. Therefore, a set A of IoT applications is assumed to be hosted on Virtual Machines (VMs) running on MEC nodes. Each application $a \in A$ is assigned a processing capacity p_a to process the corresponding workload generated by IoT devices. Further, we define σ_a^m to indicate which MEC node $m \in M$ is hosting application $a \in A$. Each application $a \in A$ is providing a specific service type requested by the IoT load. We use T to denote the set of all types of IoT applications. In addition, the parameter μ_a^t represents whether IoT application $a \in A$ is of type $t \in T$. For the sake of simplicity, we assume that each MEC node $m \in M$ is hosting one instance of each IoT application providing a service of type $t \in T$, and hence, all MEC nodes are capable of supporting all IoT applications' types. For instance, in a smart hospital environment, IoT devices are collecting various environmental data (temperature, humidity, etc.) to regulate the surroundings and vital data to monitor the patients. In order to deliver such functionalities, the IoT devices would offload their data to the MEC nodes requesting it to be processed by IoT applications providing 1) temperature monitoring and 2) patient tracking services. Hence, all MEC nodes would be running two applications, each providing one of the previously mentioned IoT services.

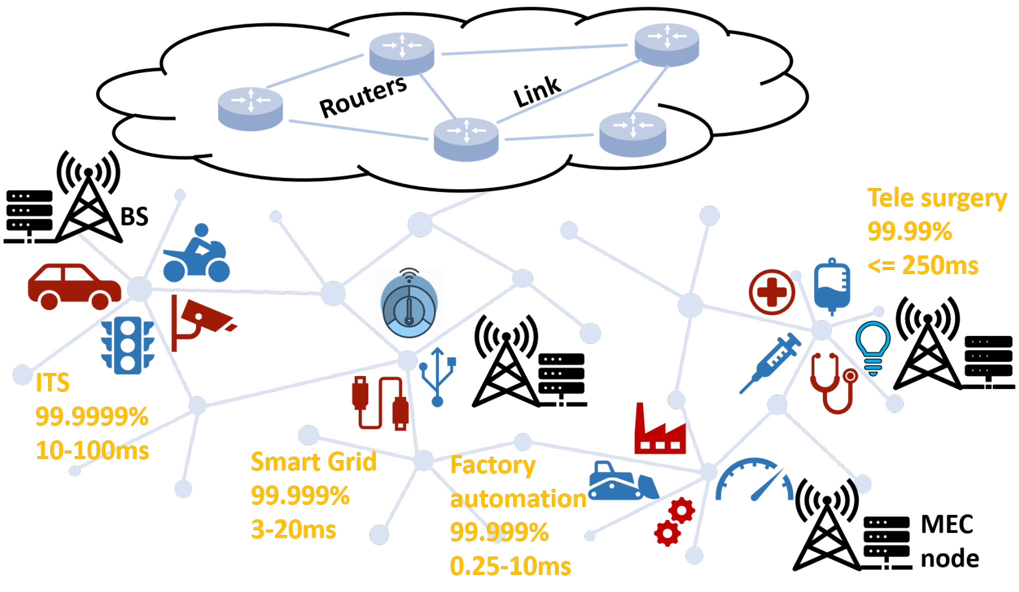


Fig. 1: IoT enabled smart environment

B. Reliability and latency model

IoT devices offload their workload onto MEC nodes to be processed by IoT applications providing a service of the corresponding type. For simplicity and without loss of generality, we consider the aggregate demand generated by all IoT devices located at $l \in L$ and requesting IoT application of type $t \in T$. This aggregate demand is assumed to follow a Poisson process with an arrival rate of λ_l^t $(requests/sec)$, and has an average computing size of w_t $(CPUcycles)$ per request [19][14]. Different demanded IoT services, with stringent latency and reliability figures, are offered by applications providing the same service type $t \in T$ to ensure a seamless experience and a satisfactory QoS. For instance, in order to provide the ultimate experience promised by ITS, latency has to be as low as $10-100ms$ and an almost guaranteed reliability of 99.9999% [5]. Consequently, we denote by δ_t and r_t the maximum allowable response time when using an IoT application of type t, and the minimum reliability required by the requested service of type $t \in T$ respectively. Within this framework, we model each IoT application as an $M/M/1$ queue with an average arrival rate of IoT devices' requests (λ_l^t) and a service rate determined by p_a and w_t. Further, we consider scenarios where the IoT devices' workload may not be assigned to the MEC node at which it was generated (home node), but it is routed to another MEC node. This could be due to insufficient computing capacity and/or failure to satisfy the latency or reliability requirements. Hence, we define $h_l^{l'}$ to depict the network delay incurred from redirecting the load from its home MEC node at location l to the MEC node at location l'. Therefore, the total delay experienced by a workload offloaded and processed by an application running on an MEC node is calculated as in Eq.(1) that will be explained in details in section IV.

$$Delay_{total} = 2(h_l^{l'}) + \frac{1}{ServiceRate - ArrivalRate} \quad (1)$$

Similarly, some failure scenarios could result in the IoT workload not being processed or transmitted. These failures could be due to transmission link failures resulting from jamming, denial of service attacks or hardware failure (MEC node failures) which could happen due to equipment error, cyber attacks, etc [15]. Some other work has already addressed the transmission failures [13]. Given that the IoT applications are hosted on VMs running on the MEC nodes, a failure of the MEC node would result in the failure of executing the IoT application. In other words, the IoT applications inherit the reliability of their hosting MEC nodes. Hence, we assign a reliability θ_m for each MEC node $m \in M$ to depict its availability. This probability is assigned based on historical data of the average repair time and time between failures [23]. In order to achieve the required service reliability demanded by the workload, we replicate the load and assign it to one or more applications hosted by one or more MEC nodes such that the required reliability is met. Therefore, the overall achieved reliability of a certain workload is the probability that at least one MEC node that can process it, is available as shown in (2).

$$Reliability_{achieved} = 1 - \prod_{m \in M} (1 - \theta_m) \quad (2)$$

Hence, the more MEC nodes accepting the workload replicas, the higher the achieved reliability. This is demonstrated in the following illustrative example.

Illustrative example: Consider five locations; l_1, l_2, l_3, l_4, l_5 in a smart environment where workloads are generated requesting different IoT services. Specifically, workload generated from l_1 is requesting tele-surgery services, while the workloads generated from l_2 and l_3 are requesting process automation services. The arrival rate (λ_l^t) for each generated load from location $l \in L$ requesting IoT service of type $t \in T$

is given below:

$$\begin{array}{c|ccccc} & l_1 & l_2 & l_3 & l_4 & l_5 \\ t_1 & 100 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 250 & 40 & 0 & 0 \end{array}$$

To ensure a satisfactory quality of service, the workload from each location has to be completed within a specific window of time and it has to be served with an ultra high reliability. The response times and reliabilities of the IoT services are listed in Table I. Particularly, workload requesting tele-surgery service has to be processed within $50ms$ and guaranteed an availability of 99.99%, and workload requesting process automation service requires a response time of $100ms$ and a reliability of 99.9%. To accommodate these requirements, we consider five MEC nodes m_1, m_2, m_3, m_4, m_5 each with reliability θ_m of 0.96, 0.96, 0.9, 0.9, and 0.9 respectively. Each MEC node hosts two IoT applications, each providing tele-surgery and process automation services. Specifically, let a_1, a_3, a_5, a_7, a_9 be the applications providing tele-surgery services while process automation services are provided by a_2, a_4, a_6, a_8 and a_{10}. Moreover, let m_1, m_2, m_3, m_4 and m_5 be located at l_1, l_2, l_3, l_4 and l_5 respectively. For the sake of simplicity we assume that the network delay is $1.5ms$ for all MEC nodes. In addition, we assume that the service rate $\frac{p_a}{w_t}$ $(requests/sec)$ of all applications of type tele-surgery is $150 requests/sec$ and that of applications of type process automation is $300 requests/sec$. In this example, we start off by considering only two workloads are generated from locations; l_1 and l_2 as shown in Figure 2(a). With the aforementioned assumptions, workload generated from l_1 could not be sent to only one MEC node as none satisfies its required reliability individually. Consequently, the workload has to be replicated to multiple MEC nodes to achieve its required reliability. The achieved reliability resulting from replicating and sending the workload generated from l_1 to MEC nodes m_1, m_2, m_3 and m_4 is 0.99998, thus, satisfying its requested service reliability requirement (0.9999). In fact, the workload generated from l_1 could be assigned to any combination of MEC nodes satisfying its reliability according to Eq.(2). Moreover, the maximum total delay incurred by the workload when processed by one of the latter MEC nodes, according to Eq.(1), is:

$$2(1.5ms) + \frac{1}{150 - 100} = 23ms \leq 50ms$$

Hence, the workload generated at l_1 is assigned to m_1, m_2, m_3 and m_4. Similarly, the workload generated from l_2 could be sent to any combination of MEC nodes that satisfies its reliability and latency requirements (e.g.$\{m_2, m_3, m_4\}$, $\{m_1, m_2, m_3\}$, etc.). Let m_2, m_3 and m_4 be the MEC nodes that the load is assigned to since the maximum total delay incurred by the workload is $23ms$ $(\leq 100ms)$ and the achieved reliability is 0.9996 (≥ 0.999).

IoT service	Reliability (%)	Allowable response time (ms)	Used δ_t (ms)
Factory-automation	99.999	0.25-10	10
Smart Grids	99.999	3-20	20
ITS	99.9999	10-100	30
Tele-surgery	99.99	<=250	50
Process-automation	99.9	50-100	100

TABLE I: IoT QoS requirements for different industry verticals

Now, we consider another load that is generated from location l_3 requesting process automation service as shown in Figure 2(b). The requested service reliability (0.999) would be met by replicating the load and mapping it to any combination of MEC nodes that satisfies its required reliability such as $\{m_2, m_3, m_4\}$, $\{m_1, m_2, m_3\}$ and $\{m_1, m_3, m_5\}$. However, since $\{m_2, m_3, m_4\}$ has already a load assigned to it from l_2 requesting the same service process automation, assigning the new workload generated from l_3 to the same MEC nodes would incur an additional queuing delay at the applications providing service of type process automation running on those MEC nodes. Therefore, the total delay incurred by the workload generated at l_3 if assigned to $\{m_2, m_3, m_4\}$ would be given by

$$2(1.5ms) + \frac{1}{300 - (250 + 40)} = 103ms \nleq 100ms$$

Similarly, this workload can not be assigned to any combination of MEC nodes that contains m_2 or m_4 as it would incur $3ms$ additional network delay which would result in a total delay of $103ms$ violating the latency requirement of $100ms$. Thus, the workload from l_3 is replicated and sent to $\{m_1, m_3, m_5\}$ which satisfies the latency and reliability requirements with a total delay of $100ms$ and an achieved reliability of 0.9996.

From this example, it can be seen that determining an optimal workload assignment is challenging where the objective is to satisfy most of the users' requests along with their low latency and ultra high reliability requirements.

IV. The uRLLC-aware Workload Assignment Problem

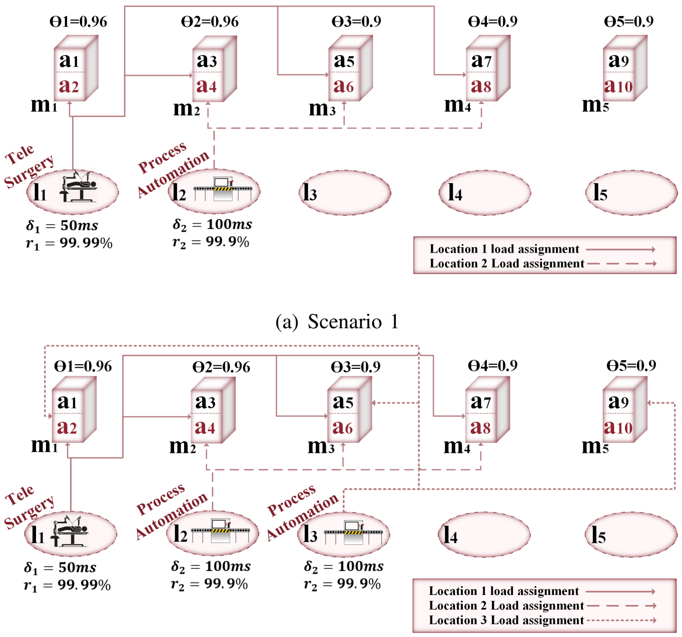
A. Problem definition

Definition 1. *Given $G(N, E)$, a set M of deployed MEC nodes, a set A of IoT applications of different types $t \in T$ hosted on the given MEC nodes, and a set of heterogeneous IoT loads requesting services of different types provided by the IoT applications, determine the optimal assignment of the generated workloads to IoT applications that maximizes the admitted load, satisfying each type's latency δ_t and reliability r_t requirements.*

B. Problem formulation

Table II shows the parameters used throughout the formulation of the defined workload assignment problem that is referred to as WA-MIP and presented below. We define a variable $x_l^t \in [0, 1]$ to determine the fraction of load generated from location l and requesting service of type t, that can be admitted to the network, and hence our objective becomes:

$$Maximize \quad \sum_{l \in L} \sum_{t \in T} \lambda_l^t . x_l^t \tag{3}$$



(a) Scenario 1

(b) Scenario 2

Fig. 2: Illustrative example

Network Inputs	
$G(N, E)$	Network of N nodes where $N = L \cup R$ and E links connecting them.
L	Set of locations where APs/BSs are mounted.
R	Set of network equipment.
M	Set of MEC nodes in $G(N, E)$.
A	Set of IoT applications hosted on $m \in M$.
T	Set of IoT applications' types.
$\pi^l_m \in \{0, 1\}$	Parameter which depicts that MEC node $m \in M$ is deployed at location $l \in L$ (1) or not (0).
$\mu^t_a \in \{0, 1\}$	Parameter which depicts that application $a \in A$ is of type $t \in T$ (1) or not (0).
$\sigma^m_a \in \{0, 1\}$	Parameter to depict that application $a \in A$ is hosted on MEC node $m \in M$ (1) or not (0).
$\delta_t \in \mathbb{R}^+$	Maximum allowable response time required by an application providing service of type $t \in T$.
$r_t \in \mathbb{R}^+$	Minimum required reliability of IoT service of type $t \in T$.
$\theta_m \in \mathbb{R}^+$	Reliability of MEC node $m \in M$.
$p^a \in \mathbb{R}^+$	Processing capacity of application $a \in A$ hosted on $m \in M$.
$\lambda^t_l \in \mathbb{Z}^+$	Arrival rate of requests for an application of type $t \in T$ generated by IoT devices located at $l \in L$.
$w_t \in \mathbb{Z}^+$	Average number of CPU cycles per request for an application of type t .
$h^{l'}_l \in \mathbb{R}^+$	Network delay of a request from its home MEC node at $l \in L$ to its assigned MEC node at $l' \in L$.

TABLE II: Parameters of WA-MIP.

That is to maximize the percentage of admitted load subject to the reliability and latency constraints. To realize our objective, we introduce a binary decision variable $z^a_{lt} \in \{0, 1\}$ to determine if a workload generated from location $l \in L$ requesting IoT service of type $t \in T$ is mapped to application $a \in A$ that is hosted on an MEC node.

$$z^a_{lt} = \begin{cases} 1 & \text{if generated workload at location } l \text{ demanding service of type } t \text{ is mapped to application } a, \\ 0 & \text{otherwise.} \end{cases}$$

Further, a new decision variable $y^m_{lt} \in \{0, 1\}$ is introduced to determine if workload generated from location $l \in L$ requesting service of type $t \in T$ is mapped to MEC node $m \in M$.

$$y^m_{lt} = \begin{cases} 1 & \text{if generated workload at location } l \text{ demanding service of type } t \text{ is assigned to MEC node } m, \\ 0 & \text{otherwise.} \end{cases}$$

Let $r^m_{lt} \in [0, 1]$ indicate the achieved reliability when processing the workload generated at location $l \in L$ requesting IoT application of type $t \in T$ at MEC node $m \in M$. Hence, the following constraints are considered.

1) **Workload assignment**

We need to make sure that whenever there is a generated load ($\lambda^t_l > 0$), it is mapped to an IoT application $a \in A$; i.e., $z^a_{lt} = 1$

$$\sum_{a \in A} z^a_{lt} \geq \frac{\lambda^t_l}{H} \quad \substack{\forall l \in L \\ \forall t \in T} \tag{4}$$

where H is a large integer number.
Similarly, the load λ^t_l is mapped to an MEC node $m \in M$; $y^m_{lt} = 1$

$$\sum_{m \in M} y^m_{lt} \geq \frac{\lambda^t_l}{H} \quad \substack{\forall l \in L \\ \forall t \in T} \tag{5}$$

Furthermore, the generated load is mapped to an application providing the same requested service type. This is ensured by Eq.(6).

$$z^a_{lt} \leq \mu^t_a.\lambda^t_l \quad \substack{\forall l \in L \\ \forall t \in T \\ \forall a \in A} \tag{6}$$

Eq.(4)-(6) together ensure that loads would always be assigned to IoT applications and if there exists no load, there will be no assignment.

Moreover, Eq.(7) and (8) ensure that whenever a load λ_l^t is mapped to MEC node $m \in M$ ($y_{lt}^m = 1$), the load is also mapped to an application $a \in A$ hosted on the same m ($z_{lt}^a = 1$), and vice versa.

$$z_{lt}^a \leq \sum_{m \in M} y_{lt}^m.\sigma_a^m \qquad \begin{smallmatrix}\forall l \in L\\ \forall t \in T\\ \forall a \in A\end{smallmatrix} \tag{7}$$

$$y_{lt}^m \leq \sum_{a \in A} z_{lt}^a.\sigma_a^m \qquad \begin{smallmatrix}\forall l \in L\\ \forall t \in T\\ \forall m \in M\end{smallmatrix} \tag{8}$$

2) **Reliability constraints:**

In order to admit a workload requesting service of type t to an MEC node m, its requested service requirements should be met. Hence, the required reliability r_t has to be satisfied. Considering both the service required reliability r_t and the reliability of the MEC node θ_m, one of two possible outcomes would occur. The first possibility is that for an MEC node $m \in M$ with reliability θ_m, the service required reliability r_t is achieved, that is $\theta_m \geq r_t$. Hence, if the workload is sent to that MEC node, the achieved reliability would solely depend on the reliability of the one MEC node processing it. Therefore, Eq.(9) ensures that the achieved reliability is at least the required reliability.

$$r_{lt}^m \geq r_t \qquad \begin{smallmatrix}\forall t \in T\\ \forall l \in L\\ \forall m \in M\end{smallmatrix} \tag{9}$$

Where r_{lt}^m is the achieved reliability and is defined as: $r_{lt}^m = y_{lt}^m.\theta_m$.

When considering all MEC nodes in the network, Eq.9 becomes:

$$\sum_m y_{lt}^m.\theta_m \geq r_t \qquad \begin{smallmatrix}\forall t \in T\\ \forall l \in L\end{smallmatrix} \tag{10}$$

On the other hand, the other possibility would be that the required service reliability is not met, that is non of the MEC nodes in the network could satisfy the required reliability (i.e $\theta_m < r_t, \forall m$). In this case, the workload is replicated and sent to multiple MEC nodes taking advantage of the independency of their reliabilities. Hence, the failures of one MEC node would not influence the availability of the other MEC nodes. This means that replicating and sending the workload to multiple MEC nodes would increase the overall achieved reliability as it would become dependent on the reliability of the MEC nodes that can accept the workload and its replicas. Hence, the new achieved reliability becomes $r_{lt}^m = 1 - \prod_{m \in M}(1 - y_{lt}^m.\theta_m)$. Eq.(11) makes sure that the IoT service required reliability is guaranteed:

$$1 - \prod_{m \in M} (1 - y_{lt}^m.\theta_m) \geq r_t \qquad \begin{smallmatrix}\forall l \in L\\ \forall t \in T\end{smallmatrix} \tag{11}$$

Thus, a decision needs to be made to select the subset of MEC nodes that will satisfy the required reliability of the requested service.

3) **Latency constraints**

The offloaded workload from IoT devices incurs different types of delays. These delays could be due to accessing the network (access delay), redirecting the workload from the home MEC node to another node (network delay) and queuing and processing delays (system delays). In this paper, for the sake of simplicity, we consider the access delays to be negligible. Hence, the total delay incurred by the offloaded workload is represented solely by the system and network delays.

We use $d_{network}^{mlt}$ to depict the network delay experienced by workload generated from location l requesting service of type t to be transferred to MEC node m and is given by:

$$d_{network}^{mlt} = \sum_{l' \neq l} h_l^{l'}.y_{lt}^m.\pi_m^{l'} \qquad \begin{smallmatrix}\forall l \in L\\ \forall t \in T\\ \forall m \in M\end{smallmatrix} \tag{12}$$

Further, we define d_{system}^{alt} to represent the system delay. Given that each IoT application is modeled as M/M/1 queue with an average arrival rate of $\sum_{l' \in L} z_{l't}^a.x_{l'}^t.\lambda_{l'}^t$ and service rate of $\frac{p_a}{w_t}$, the system delay is given by:

$$d_{system}^{alt} = z_{lt}^a \left(\frac{1}{\frac{p_a}{w_t} - \sum_{l' \in L} z_{l't}^a.x_{l'}^t.\lambda_{l'}^t} \right) \qquad \begin{smallmatrix}\forall l \in L\\ \forall t \in T\\ \forall a \in A\end{smallmatrix} \tag{13}$$

To avoid congestion at the application, the service rate should be greater than the arrival rate as in Eq.(14).

$$\frac{p_a}{w_t} - \sum_{l' \in L} z_{l't}^a.x_{l'}^t.\lambda_{l'}^t \geq 0 \qquad \begin{smallmatrix}\forall a \in A\\ \forall t \in T\end{smallmatrix} \tag{14}$$

Combining both delays together, the total delay D_{lt}^{ma} incurred by offloading a workload to application a hosted on MEC node m is:

$$D_{lt}^{ma} = 2d_{network}^{mlt} + d_{system}^{alt} \qquad \begin{smallmatrix}\forall l \in L\\ \forall t \in T\\ \forall m \in M\\ \forall a \in A\end{smallmatrix} \tag{15}$$

And finally, in order to meet the delay requirements δ_t of each IoT service provided by an application of type t, we have:

$$D_{lt}^{ma} \leq \delta_t \qquad \begin{smallmatrix}\forall t \in T\\ \forall a \in A\\ \forall l \in L\\ \forall m \in M\end{smallmatrix} \tag{16}$$

C. *Complexity analysis*

The WA is a mixed integer programming problem (MIP) which is complex and hard to solve. The NP-Hardness can be easily shown through a reduction from the Generalized Assignment Problem (GAP) (known to be NP-Hard), where the workloads represent the items to be assigned to bins (MEC nodes) [24], [25]. Given its complexity, we devise two different approaches to solve it.

V. WA-D Approach

Solving the workload assignment problem with respect to both reliability and latency requirements is challenging. To deal with this complexity, we exploit the independency of the reliability and latency requirements and decompose the problem into two subproblems; the Reliability Aware Candidate Selection subproblem (RACS) and the Latency Aware Workload Assignment subproblem (LAWA).

A. RACS heuristic

Given the heterogeneity of the MEC nodes reliabilities, not all the MEC nodes can admit the workloads coming from the different locations. Hence, for all workloads generated from different locations demanding the same service type t, a set S_t of potential MEC nodes candidates is generated, where each element in S_t includes a combination of MEC nodes that together can provide the reliability required by service type t. The algorithm starts by identifying the set of requested types by the generated workloads. It then constructs a larger set of MEC nodes combinations. To avoid generating all possible combinations ($2^M - 1$), the size of each combination ranges between one MEC node and a predefined number N of them. We choose N based on a worst case scenario; that is when all MEC nodes have the lowest possible reliability θ_m and a generated load requesting a service having the highest required reliability r_t. Hence, N is the minimum number of MEC nodes needed to satisfy the service with the highest required reliability. Each combination is represented in binary to simplify the computation and to depict which MEC nodes are in the set; 0 for $m \notin set$ and 1 otherwise. For each requested type, the achieved reliability is computed for each combination according to Eq.(11). If the achieved reliability is $\geq r_t$, the combination is added to S_t. The i^{th} element in the set S_t is denoted by S_i^t, and represents either a potential MEC node or a subset of them, and hence, $S_i^t \in S_t$. We denote by I the set of all elements belonging to S_t. Further, each element $i \in I$ is weighted and all elements are sorted in an ascending order according to the weight function defined in Eq.(17). A pre-defined number of subsets $i \in I$ with the minimum weights are selected for each S_t and passed to the second subproblem LAWA MIP.

$$\begin{aligned} &W(MEC\ nodes\ subset) = \\ &\quad w_1\left(reliability(MEC\ nodes\ subset) - r_t\right) \\ &\quad + w_2\left(|MEC\ nodes\ subset|\right) \\ &\quad s.t:\ w_1 + w_2 = 1 \end{aligned} \tag{17}$$

In other words, the weight function in Eq.(17) ensures using the available resources (MEC nodes) efficiently by selecting the elements that precisely satisfy the required reliability of a specific service of type t. For instance, consider two elements in S_t with the same number of MEC nodes and a type t workload requesting a service with a reliability of 0.999. Mapping the load to the first element would achieve a reliability of 0.9999 and mapping it to the second element would achieve a 0.999999 reliability. Hence, according to Eq.(17), the load should be assigned to the first subset. Thus, a predefined number of elements, minimizing the difference between the required reliability and achieved reliability and consisting of the lowest number of resources (MEC nodes), is selected from each S_t.

B. LAWA MIP

Given the set S_t of the potential MEC nodes for the generated workloads from different locations demanding service of type t obtained from the RACS heuristic, the LAWA MIP determines the optimal candidate S_i^t for each generated load λ_l^t. The optimal candidates are chosen to maximize the fraction of admitted load while satisfying the workloads' latency requirements. Here, we should note type t workload from different locations may contend for the same MEC nodes (to achieve higher reliability), and as a result the load at one particular application may increase, and as a result, affecting the latency. Therefore, selecting the best candidate of MECs for each IoT workload is what we seek to find. Formally, we use the decision variable $x_l^t \in [0, 1]$ as defined in section IV-B to determine the fraction of admitted load. The LAWA objective is as depicted in Eq.(18).

$$Maximize \quad \sum_{t \in T} \sum_{l \in L} \lambda_l^t . x_l^t \tag{18}$$

that is to maximize the percentage of admitted load subject to latency constraints. In order to meet our objective, we define g_i^{tl} to depict whether the i^{th} element in the set S_t is selected to process workload λ_l^t or not.

$$g_i^{tl} = \begin{cases} 1 \text{ if the } i^{th} \text{ element in } S_t \text{ is selected,} \\ 0 \text{ otherwise.} \end{cases}$$

Further, we use the decision variable $z_{lt}^a \in \{0, 1\}$ as defined in section IV-B to determine if workload generated from location l demanding service of type $t \in T$ is mapped to application $a \in A$. Now, Eq.(19) makes sure that at most one element should be selected from S_t.

$$\sum_i g_i^{tl} \leq 1 \quad \begin{smallmatrix} \forall t \in T \\ \forall l \in L \end{smallmatrix} \tag{19}$$

We need to make sure that whenever there is a generated workload demanding service of type t ($\lambda_l^t > 0$), it is mapped to an IoT application $a \in A$ hosted on an MEC node ($z_{lt}^a = 1$).

$$\sum_{a \in A} z_{lt}^a \geq \frac{\lambda_l^t}{H} \quad \begin{smallmatrix} \forall t \in T \\ \forall l \in L \end{smallmatrix} \tag{20}$$

Furthermore, the generated load is mapped to an application providing the same requested service type. This is ensured by Eq.(21).

$$z_{lt}^a \leq \mu_a^t . \lambda_l^t \quad \begin{smallmatrix} \forall t \in T \\ \forall a \in A \\ \forall l \in L \end{smallmatrix} \tag{21}$$

Eqs. (20) and (21) together ensure that loads would always be assigned to IoT applications and if there exists no load, there will be no assignment.

Moreover, we need to make sure that the workload λ_l^t is assigned to an application $a \in A$ that is hosted on an MEC node $m \in M$ that is in the i^{th} element in S_t. This is ensured by Eq.(22).

$$z_{lt}^a \leq \sum_{i \in I} \sum_{m \in S_i^t} g_i^{tl} . \sigma_a^m \quad \begin{smallmatrix} \forall t \in T \\ \forall l \in L \\ \forall a \in A \end{smallmatrix} \tag{22}$$

Whenever an element i (subset of MEC nodes) is selected $g_i^{lt} = 1$, the load is assigned to all applications of type t hosted on the MEC nodes in S_i^t. This is ensured by Eq.(23).

$$\sum_{i \in I} |S_i^t| g_i^{tl} \leq \sum_{a \in A} z_{lt}^a \quad \begin{smallmatrix} \forall t \in T \\ \forall l \in L \end{smallmatrix} \tag{23}$$

Further, we need to make sure that each individual MEC node in the selected subset S_i^t meets the delay requirements δ_t of workload λ_l^t. This is verified by Eq.(24).

$$2\left(\sum_{l'\neq l} h_l^{l'}.z_{lt}^a.\sigma_a^m.\pi_m^{l'}\right)+ z_{lt}^a\left(\frac{1}{\frac{p_a}{w_t}-\sum_{l'\in L} z_{l't}^a.x_{l'}^t.\lambda_{l'}^t}\right)\leq \delta_t \quad \substack{\forall t\in T\\ \forall a\in A\\ \forall l\in L\\ \forall m\in M} \tag{24}$$

To avoid congestion, the service rate should be greater than the arrival rate. This is ensured by Eq.(25).

$$\frac{p_a}{w_t}-\sum_{l'\in L} z_{l't}^a.x_{l'}^t.\lambda_{l'}^t \geq 0 \quad \substack{\forall a\in A\\ \forall t\in T} \tag{25}$$

WA-D still requires to solve a MIP, which makes it challenging to solve the problem. Hence, in the following section we also propose a meta-heuristic approach (WA-Tabu) to accelerate the performance of the WA-D.

VI. WA-Tabu

Tabu search is a meta-heuristic search method that has been used to solve NP hard optimization problems, one of which is the task assignemnt problem [26]. The Tabu search-based algorithm consists of the following components [27]:

1) **Initial solution:** The WA-Tabu starts by an initialization step of constructing an initial solution for the workload assignment to the MEC nodes. This is done by first generating a set of potential candidates S_t for each requested IoT service type $t \in T$ as described in section V-A. The load assignment is then performed by selecting the best subset of MEC nodes for each workload. The selection is based on maximizing the weight function given in Eq.(26) that is conditioned to satisfy the latency requirements of the workload's requested service.

$$\begin{aligned} W(Mec\ nodes\ subset) &= w_1(min_{x_l^t}) \\ &- w_2\left(reliability(MEC\ nodes\ subset) - r_t\right) \\ &- w_3\left(|MEC\ nodes\ subset|\right) \\ \\ s.t:\ \ w_1 + w_2 + w_3 &= 1 \end{aligned} \tag{26}$$

where $min_{x_l^t}$ is the fraction of the load that a subset can admit, determined by the MEC node admitting the least fraction of load. In other words, the best subset of MEC nodes is the one satisfying the latency requirement of the workload's requested service, maximizing the percentage of admitted load, minimizing the difference between the required and the achieved reliability, and using the lowest number of resources (MEC nodes). The priority of load assignment is then calculated for each load based on the priority function given in Eq.(27) and the load with the highest priority is assigned. The process repeats until all loads are considered.

$$\begin{aligned} P(\lambda_l^t) &= W(Best\ MEC\ nodes\ subset) \\ &- W(2^{nd}\ Best\ MEC\ nodes\ subset) \end{aligned} \tag{27}$$

2) **Neighborhood solutions:** Given the initial load assignment, the algorithm searches for improving the workload assignment in the neighborhood of the current solution based on the weight function defined in Eq.(26). Within this context, a neighborhood is defined as any solution that involves shifting a workload from the MEC nodes subset that is assigned to, to another. In order to reduce the search space, we consider shifting loads from the most loaded subsets to the least. Hence, if an improved workload assignment is found, the initial load assignment is updated. If, however, no improving assignments were found, the algorithm finds the first non-improving assignment by allowing shifting a workload to the first subset yielding less weight. This raises the chances of reaching a global maximum.
3) **Tabu list:** A tabu move is defined as shifting the load assignment of a workload from the more loaded subsets to the less loaded. Once the shift is performed, the tabu move is added to the tabu list where it is not considered for the next $tabuListSize$ iterations. This prevents the workload from cycling back to its original subset before allowing other possible moves to be considered. Further, choosing a solution with a lower weight than the current solution is also considered as a tabu move.
4) **Aspiration criterion:** In certain scenarios, we allow the violation of the tabu status of moves if the move gives a better solution than the best solution found so far.
5) **Stopping criteria:** The algorithm iterates until:
   - A maximum number of iterations is reached.
   - All the loads are admitted.

   The pseudocode for the Tabu-search is shown in Algorithm 1.

VII. Numerical Evaluation

In this section, we compare the performance of WA-MIP, WA-D and WA-Tabu through extensive numerical evaluation. Further, we evaluate the efficiency of our proposed WA-Tabu approach under varying system parameters.

A. Experimental setup

To evaluate our algorithms, we consider a network with $L = 25$ locations (unless stated otherwise) where at each location an MEC node is deployed. Each MEC node has a reliability θ_m that is randomly generated between $[0.9 - 0.96]$ [28]. Further, from each location a workload is generated with an average arrival rate λ_l^t taking random values between $[70 - 300]requests/sec$. The generated workloads request different types of IoT services with various latency and reliability requirements. Hence, we consider $T = 4$ types of IoT services corresponding to the industry verticals and their QoS requirements presented in Table I, which yields an aggregate load of $[7k - 30k]requests/sec$. In addition, for each of the requested types, we assume an average computing size w_t generated randomly between 1×10^6 and $2 \times 10^6 CPUcycles$ per request. In order to accommodate the generated workload, we consider applications providing different IoT services hosted

Algorithm 1 WA-Tabu

```
Input:
$y_{lt}^{m} current, x_{l}^{t} current$: initial solution
$tabuMove : (l, t, subset_{original}, subset_{new})$
$TabuList$: holds tabu moves
$TabuListSize$: indicates size of $TabuList$
$y_{lt}^{m} best, x_{l}^{t} best$: indicates the best Assignment so far
while stopping criteria is not met
    $firstImrpovAssign \leftarrow getFirstImprovAssign()$
    if ($firstImprovAssign$ is found and $\notin tabuList$ )
        $y_{lt}^{m} current, x_{l}^{t} current \leftarrow firstImrpovAssign$
    else
        $firstNonImprovAssign \leftarrow$
                $getFirstNonImrpovAssign()$
        if ($firstNonImprovAssign \notin tabuList$ )
            $y_{lt}^{m} current, x_{l}^{t} current \leftarrow$
                $firstNonImrpovAssign$
    end if
    if $\sum x_{l}^{t} current > \sum x_{l}^{t} best$
        $y_{lt}^{m} best, x_{l}^{t} best \leftarrow y_{lt}^{m} current, x_{l}^{t} current$
    end if
    Add $tabuMove$ to $tabuList$
    if $tabuList$ is full
        $remove$ first element added to $tabuList$
    end if
    $iter++$
end while
```

on the MEC nodes. We then assume that all the MEC nodes support all types of applications, and hence, the number of applications A is equal to $T \times M$. Each of the applications is assigned computing resources p_a chosen randomly within the range of $[1.7 - 1.9] GHz$. Furthermore, since some of the generated loads might migrate to different MEC nodes other than their home MEC nodes, we assume the network delay to be generated between 1 and $2ms$ at random. Moreover, since the loads could be replicated and offloaded to a subset of MEC nodes satisfying the QoS requirements of its requested type, we choose the size of the set S_t of the potential subsets that a load could be assigned to, to be between 200 and 900 (unless stated otherwise). All our numerical evaluations are averaged over 5 sets. The WA-MIP and the WA-D are evaluated using IBM ILOG CPLEX Optimization Studio v.12.8.

B. WA-MIP vs. WA-D vs. WA-Tabu

We first compare the performance of our proposed solutions; WA-MIP, WA-D and WA-Tabu in terms of optimality (total admitted load) and scalability (CPU run time). To do so, we vary the network size by increasing the number of locations L, the number of MEC nodes M and the number of applications A. In fact, increasing the number of locations in the network implies increasing the aggregate generated load. Thus, more MEC nodes subsets are needed to accommodate the added load, and hence, increasing I (the size of S_t) as the size of the network increases for both WA-D and WA-Tabu. Further, we consider that the applications are of $T = 4$ different types belonging to smart grid industry vertical with $\delta_t = 20ms$ and $r_t = 99.999\%$. The evaluation results are presented in Table III.

Instance	**Execution Time (sec)**			**Admitted Load (%)**		
<L, M, T, A, I>	**WA-MIP**	**WA-D**	**WA-Tabu**	**WA-MIP**	**WA-D**	**WA-Tabu**
<5, 5, 4, 20, 10>	0.025	0.020	0.017	100%	100%	100%
<8, 8, 4, 32, 50>	0.99	0.36	0.047	100%	100%	100%
<11, 11, 4, 44, 100>	2.1	1.31	0.13	100%	100%	100%
<14, 14, 4, 56, 200>	20	2.37	0.40	100%	100%	100%
<17, 17, 4, 68, 300>	288.4	15.2	0.87	100%	100%	100%
<20, 20, 4, 80, 500>	Out of Mem.	27.07	2.40	-	100%	100%
<23, 23, 4, 92, 700>	-	20.1mins	4.61	-	100%	100%

TABLE III: WA-MIP vs. WA-D vs. WA-Tabu.

- **Scalability:** From Table III it is shown that as the size of the network increases, the execution time (CPU run time) increases exponentially for the WA-MIP. This behavior continues until it fails to give a solution when the size of the network is $L = M = 20,\ T = 4$ and $A = 80$. On the other hand, WA-D proved to be more scalable compared to the WA-MIP. This can be seen from the execution time of the WA-D where it increases exponentially as the size of the network increases, but at a slower rate compared to the WA-MIP. It can be observed that the WA-D gave a solution when the network size was $L = M = 23,\ T = 4$ and $A = 92$, but the run time jumped to $20.1mins$. Alternatively, the WA-Tabu proved to be the most scalable as its execution time increases linearly as the size of the network increases.
- **Optimality:** As can be seen from Table III, the algorithm WA-MIP was able to accept all the load up to the network size $L = M = 17,\ T = 4$ and $A = 68$. It failed however, to give a solution for the last two instances. On the other hand, the algorithms WA-D and WA-Tabu were able to admit all the generated load which yields to an optimality gap of 0% in the considered instances where I was large enough to accommodate all the generated workload. In fact, varying the size of S_t which is denoted by I, has a significant impact on the admission rate and CPU run time of both WA-D and WA-Tabu. We further illustrate the impact of varying the parameter I on the performance of WA-D and WA-Tabu in terms of admission rate and run time in VII-C.

C. WA-D vs. WA-Tabu

In the previous section, we showed that WA-MIP is not scalable. Further, we showed that WA-D is more scalable than WA-MIP, and WA-Tabu algorithm is the most scalable for the chosen values of I in Table III. In this section, we further evaluate the performance of the WA-D and WA-Tabu. We thus select the instance from Table III with the network size $L = M = 17,\ T = 4$ and $A = 68$ to investigate the performance under varying the size of S_t. We vary the size of S_t between $(5 - 300)$. Our evaluation is in terms of execution time and total admitted load. The results are shown in Figures 3 and 4.

From Figure 3, we observe that for small values of I ($I = 5,\ I = 20$) that represent the size of S_t, WA-D fails to give a feasible solution after running for a couple of hours as the

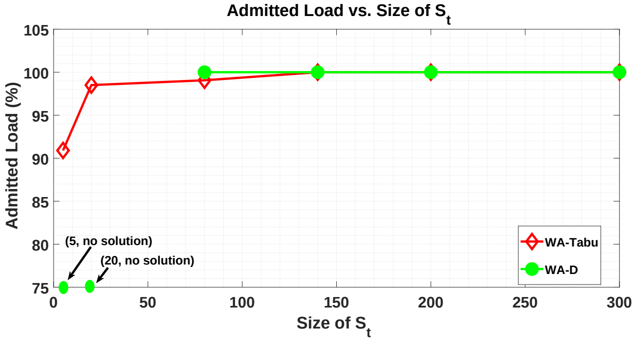


Fig. 3: Admitted load under varying size of S_t

LAWA-MIP was hard to solve. While on the other hand, WA-Tabu admits 90.9% and 98.5% of the load for $I = 5$ and $I = 20$ respectively. Further, as the size of I increases, the admission rate increases for both WA-D and WA-Tabu. This is explained by the fact that increasing the size of S_t means increasing the number of potential subsets of MEC nodes that the generated loads could be assigned to, and hence, admitting more load. Moreover, for $I = 80$, WA-D performed slightly better than the WA-Tabu in terms of the total admitted load with a difference less than 1%. In addition, for $I > 80$, both algorithms admit 100% of the total generated load.

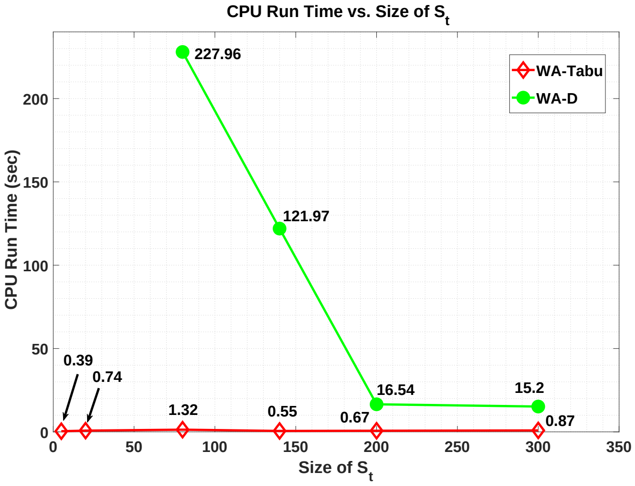


Fig. 4: Execution time under varying size of S_t

Moving to Figure 4, it can be observed that the CPU run time for WA-D decreases dramatically as the size of S_t increases. This is due to the fact that increasing I makes it easier for the WA-D to find a solution as more potential candidates (subsets) become available to it, and hence, the lower the execution time. More interestingly, the CPU run time for WA-Tabu algorithm increases slightly as I increases up to $I = 80$ where it starts decreasing until I is equal to 140, then it starts increasing again. The first increasing behavior is because WA-Tabu iterates over the potential subsets to construct the initial solution and find neighboring solutions. Hence, increasing the size of I would increase the run time. When the size of I exceeds 80, the initial solution of the WA-Tabu gives a 100% admitted load, and hence, the algorithm terminates before iterating over the subsets in S_t to improve the initial solution as the stopping criterion is met. Finally, the execution time hardly increases for $I > 140$ as the WA-Tabu would only iterate over the subsets to construct the initial solution.

D. Performance evaluation of WA-Tabu

1) Varying the workloads for different industry verticals and its impact on the admission rate: We vary the workloads and study the impact on the admission rate for different industry verticals with different requirements. Thus, we increase the generated workload and choose the values of λ_l^t to be $\{100,\ 200,\ 300\}$ requests/sec. The results are presented in Figure 5. It can be seen from the figure that for each industry vertical, as the generated workload per location l per IoT service type t increases, the admission rate decreases. This is due to the fact that as λ_l^t increases, more load is requesting to be processed by the same available resources $a \in A$, which are not sufficient to accommodate the newly generated load with the given QoS requirements. Thus, this results in admitting less load. Moreover, for the same value of λ_l^t, the admitted load increases as the latency and reliability requirements become less strict. For instance, for $\lambda_l^t = 200$, the total admitted load for the industry vertical factory automation with the QoS requirements ($\delta_t = 10ms$, $r_t = 99.999\%$) is 89.56%. While on the other hand, the admitted load for the smart grid industry vertical with the QoS requirements ($\delta_t = 20ms$, $r_t = 99.999\%$) increased to 92.44%. The admission rate keeps increasing to reach 100% for the industry vertical process automation with the least strict QoS requirements ($\delta_t = 100ms$, $r_t = 99.9\%$).

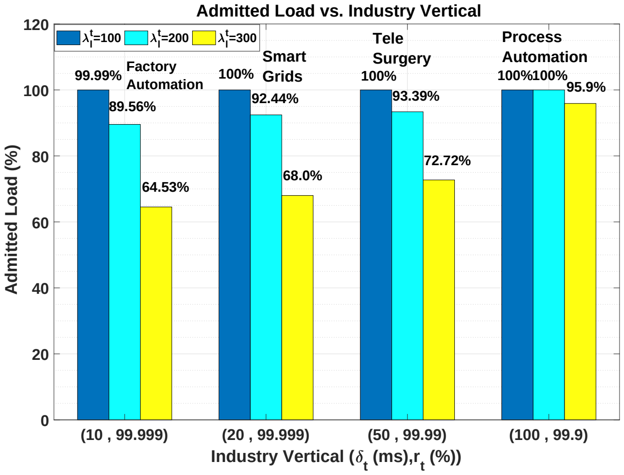


Fig. 5: Admission rate under varying workloads for different industry verticals

2) Varying the network delay for ITS industry vertical and its impact on the admission rate: We evaluate the impact of increasing the network delay on the admission rate for the intelligent transportation systems (ITS) industry vertical with $\delta_t = 30ms$ latency requirement. We thus vary the network delay between 6 and $16ms$. The results are depicted in Figure 6. It can be observed from the figure that as the network

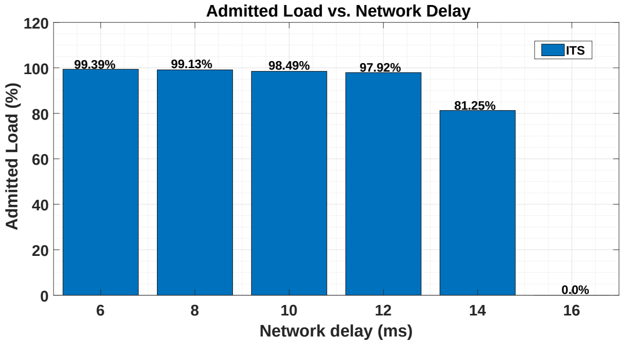


Fig. 6: Admission rate under varying the network delay

delay increases, the admission rate decreases. For instance, the total admitted load decreased by almost 17% when the network delay went from $12ms$ to $14ms$. This behavior is due to the fact that increasing the network delay leads to a more limited system delay at the MEC nodes to process the requests within the deadline ($30ms$). Further, for the network delay $16ms$, no load was admitted as no system delay remained to process the workload within the maximum allowable response time.

3) Varying the required reliability for different response times and its impact on the admission rate: We consider the reliability requirements for different IoT industry verticals and overlook their latency requirements. We then study the impact of varying the required reliability (r_t) on the admission rate for two different deadlines, $10ms$ and $100ms$. Our results depicted in Figure 7 show that for a specific maximum allowable response time (δ_t), as the required reliability becomes more strict, the admission rate decreases. In fact, increasing the reliability requirements is coupled with replicating the workload to more MEC nodes, which leads to an increase in the queuing delay at the MEC nodes. Hence, the available resources (applications) would not be sufficient to completely process the workload. For instance, for $\delta_t = 10ms$, the admission rate decreased from 98.9% to 85.9% when the required reliability went from 99.9% to a more strict value of 99.9999%. Moreover, it can be seen that for a less strict deadline ($100ms$), more system delay at the MEC nodes remains to process the workload, and hence, the admission rate is higher than that of $10ms$.

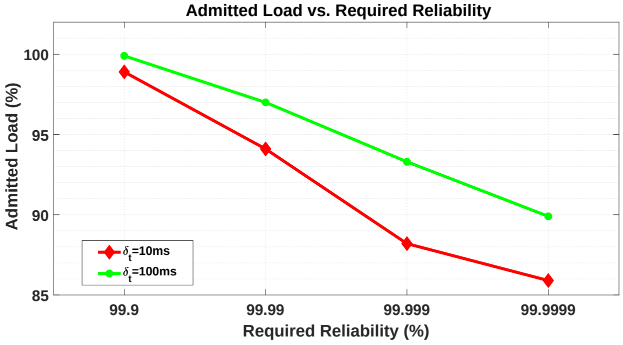


Fig. 7: Admission rate under varying the required reliability

4) RACS-heuristic vs. random candidate selection: As part of our evaluation, we explore different strategies for the selection of the subsets composing S_t. Particularly, as discussed in section V-A, we use the RACS heuristic as our selection methodology to select the subsets based on minimizing the achieved reliability with respect to the reliability required, and the used resources (number of used MEC nodes). Alternatively, we devise another selection strategy that randomly selects the subsets. We evaluate the performance of the WA-Tabu algorithm for different industry verticals under both strategies in terms of admission rate and resource utilization. The results are shown in Figures 8 and 9.

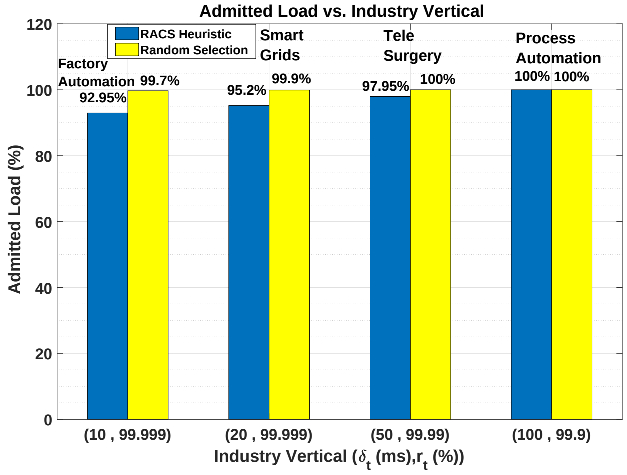


Fig. 8: Admission rate for different subsets selection strategies

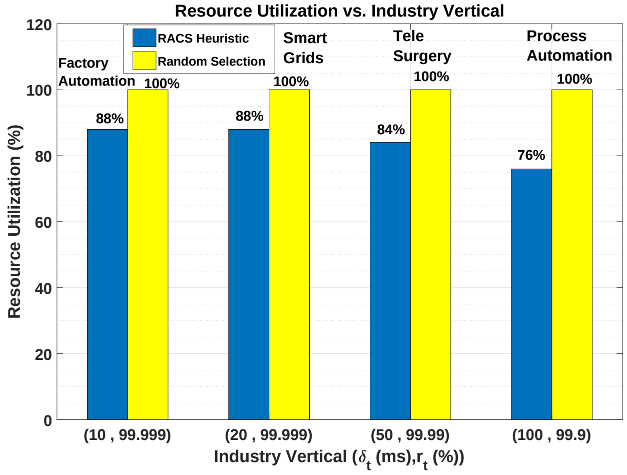


Fig. 9: Resource utilization for different subsets selection strategies

From Figure 8, it is observed that for the same selection strategy, the admission rate increases as the QoS requirements become less strict from one industry vertical to the other. More interestingly, for a given industry vertical, the random selection strategy gives a higher admission rate compared to the RACS method, with an insignificant difference between 0 and 7%. While on the other hand, the difference in the resource utilization is remarkable (between 12 and 24%) in favor of the RACS heuristic as demonstrated in Figure 9. More precisely, for the industry vertical factory automation with the the most strict QoS requirements, the WA-Tabu with the random selection strategy obtained around 7% more admitted load than the RACS heuristic. However, it utilized 12% more resources to admit the load. Moreover, for the industry vertical process automation with the least stringent QoS requirements, the resource utilization with the RACS heuristic was significantly lower than the random selection with a difference of 24%, while both yield the same admitted load (100%). In fact, the random selection method provides more diverse subsets composing S_t, the fact that yields to having less number of common MEC nodes among the subsets, and hence, resulting in more admitted load. However, this diversity increases the probability of using more resources, and hence yielding higher resource utilization. One can note that there is a trade-off between admitting more load and utilizing less resources. Moreover, from the network operator perspective, using the RACS strategy would allow saving energy, as he/she could shut down the unused MEC nodes. Another scenario where the RACS selection method would be more favorable is when an additional load is generated in the network, the unused resources could then be utilized to admit it.

5) Impact of varying the network size on the admitted load and execution time: For further exploration, we vary the size of the network by varying the number of locations, MEC nodes, IoT applications and size of S_t, and evaluate its impact on the admitted load and execution time. As can be seen from Figure (11), as the network size increases, the execution time increases at a very slow rate. Further, from Figure (10), one can note that the admitted load has a decreasing trend. This decrease was not significant until network size $< 60, 60, 240, 4, 600 >$. This is due to the fact that the size of S_t was fixed to 600 for the rest of the network sizes. Hence, one can conclude that the admitted load heavily depends on the size of S_t. Further, the execution time would increase by increasing the size of S_t. Therefore, there is a trade-off between the execution time and admitted load.

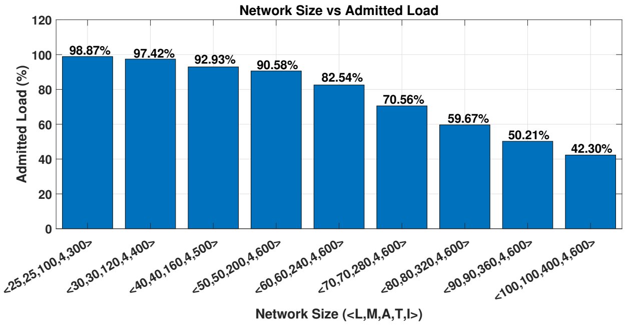


Fig. 10: Admission rate under varying Network Size

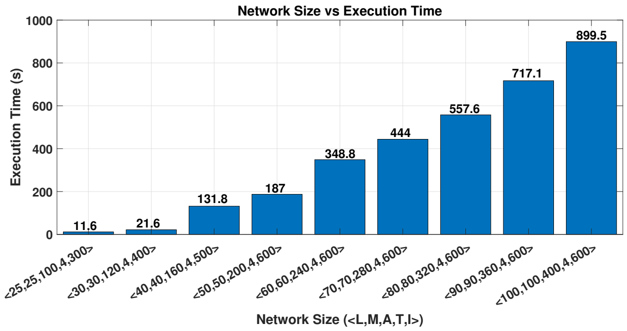


Fig. 11: Execution Time under varying Network Size

VIII. Conclusion

In this paper, we studied the workload assignment problem with latency and reliability constraints and evaluated different approaches to solve it. We first mathematically formulated the problem as a mixed integer program (WA-MIP) with the objective of maximizing the total admitted load to the network while satisfying the QoS requirements of the supported IoT services. We then proved the non-scalability of the WA-MIP in addition to the NP-Hardness of the considered workload assignemnt problem. Therefore, we developed the WA-D that decomposes the problem into two subproblems; namely, RACS and LAWA-MIP. The decomposition approach showed a significant improvement in scalability as compared to WA-MIP. Although WA-D proved to be more scalable, it is not scalable enough for very large networks. Thus, a meta-heuristic approach (WA-Tabu) was developed to efficiently solve the problem for larger networks with heterogeneous QoS requirements. Through extensive simulations under various system parameters, we evaluated the performance of our proposed approach (WA-Tabu). Our proposed WA-Tabu aids the network operators to efficiently use their available resources to serve the maximum number of end users in an smart IoT environment. Based on our findings, this work can be extended to consider the deciding on the number of IoT applications to be deployed as well as the amount of computing resources allocated to them which would complicate the problem more. Another future direction could be the dynamic workload assignment with an online arrival of workloads.

Appendix

A. Linearization of Eq.11

Eq.11 is not linear and can be linearized by first rearranging the terms as follows:

$$1 - r_t \geq \prod_{m \in M} (1 - y_{lt}^m.\theta_m) \quad \substack{\forall t \in T \\ \forall l \in L} \tag{28}$$

and then taking the natural logarithm of both sides, we get:

$$\ln(1 - r_t) \geq \ln\left(\prod_{m \in M} (1 - y_{lt}^m.\theta_m)\right) \quad \substack{\forall t \in T \\ \forall l \in L} \tag{29}$$

Using the natural logarithm properties, Eq.29 becomes:

$$\ln(1 - r_t) \geq \sum_{m \in M} \ln\left((1 - y_{lt}^m.\theta_m)\right) \quad \substack{\forall t \in T \\ \forall l \in L} \tag{30}$$

Now, Eq.30 can be linearized by observing the two possible outcomes of its right hand side:

$$\ln(1 - y_{lt}^m.\theta_m) = \begin{cases} \ln(1 - \theta_m) \text{ if } y_{lt}^m = 1, \\ 0 \text{ if } y_{lt}^m = 0. \end{cases}$$

Hence, the right hand side of Eq.30 could be rewritten as in Eq.31.

$$\sum_{m\in M} y^m_{lt}(\ln(1-\theta_m)) \tag{31}$$

Thus, Eq.11 can then be replaced by Eq.32.

$$\ln(1-r_t) \geq \sum_{m\in M} y^m_{lt} \ln\left((1-\theta_m)\right) \quad \substack{\forall t\in T\\ \forall l\in L} \tag{32}$$

B. Linearization of Eq.16 and Eq.24

Equations (16) and (24) are not linear and can be linearized by rewriting them as follows:

$$z^a_{lt}\Bigg[2\Bigg(\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m\Bigg)+ \left(\frac{1}{\frac{p_a}{w_t}-\sum_{l''\in L} z^a_{l''t}.x^t_{l''}.\lambda^t_{l''}}\right)\Bigg] \leq \delta_t \quad \substack{\forall l\in L\\ \forall t\in T\\ \forall a\in A\\ \forall m\in M} \tag{33}$$

$$\frac{z^a_{lt}}{\frac{p_a}{w_t}-\sum_{l''\in L} z^a_{l''t}.x^t_{l''}.\lambda^t_{l''}}\Bigg(2.\frac{p_a}{w_t}\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m - 2.\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m.\sum_{l''\in L} z^a_{l''t}.x^t_{l''}.\lambda^t_{l''}+1\Bigg) \leq \delta_t \quad \substack{\forall l\in L\\ \forall t\in T\\ \forall a\in A\\ \forall m\in M} \tag{34}$$

$$\Bigg(z^a_{lt}.2.\frac{p_a}{w_t}\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m - 2.\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m.\sum_{l''\in L} z^a_{lt}.z^a_{l''t}.x^t_{l''}.\lambda^t_{l''}+z^a_{lt}\Bigg) \leq \delta_t.\frac{p_a}{w_t}-\sum_{l''\in L} z^a_{l''t}.x^t_{l''}.\lambda^t_{l''} \quad \substack{\forall l\in L\\ \forall t\in T\\ \forall a\in A\\ \forall m\in M} \tag{35}$$

Eq.35 is nonlinear and can be linearized by declaring two new decision variables $\beta^a_{ll''t} \in \{0,1\}$ and $\psi^a_{l''t} \in [0,1]$ such that:

$$\psi^a_{l''t} = z^a_{l''t}.x^t_{l''} \quad \substack{\forall l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{36}$$

$$\beta^a_{ll''t} = z^a_{lt}.\psi^a_{l''t} \quad \substack{\forall l,l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{37}$$

Eq.35 can then be replaced by the following equations:

$$\Bigg(z^a_{lt}.2.\frac{p_a}{w_t}\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m - 2.\sum_{l'\neq l} h^{l'}_l.\sigma^m_a.\pi^{l'}_m.\sum_{l''\in L} \beta^a_{ll''t}.\lambda^t_{l''}+z^a_{lt}\Bigg) \leq \delta_t.\frac{p_a}{w_t}-\sum_{l''\in L} \psi^a_{l''t}.\lambda^t_{l''} \quad \substack{\forall l\in L\\ \forall t\in T\\ \forall a\in A\\ \forall m\in M} \tag{38}$$

$$\psi^a_{l''t} \leq z^a_{l''t} \quad \substack{\forall l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{39}$$

$$\psi^a_{l''t} \leq x^t_{l''} \quad \substack{\forall l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{40}$$

$$\psi^a_{l''t} \geq x^t_{l''} + z^a_{l''t} - 1 \quad \substack{\forall l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{41}$$

$$\psi^a_{l''t} \geq 0 \quad \substack{\forall l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{42}$$

$$\beta^a_{ll''t} \leq z^a_{lt} \quad \substack{\forall l,l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{43}$$

$$\beta^a_{ll''t} \leq \psi^a_{l''t} \quad \substack{\forall l,l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{44}$$

$$\beta^a_{ll''t} \geq z^a_{lt} + \psi^a_{l''t} - 1 \quad \substack{\forall l,l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{45}$$

$$\beta^a_{ll''t} \geq 0 \quad \substack{\forall l,l''\in L\\ \forall t\in T\\ \forall a\in A} \tag{46}$$

C. Linearization of Eq.14

Eq.14 is nonlinear and can be linearized by replacing it by Eq.39, 40, 41, 42 and 47.

$$\frac{p_a}{w_t} - \sum_{l'\in L} \psi^a_{l't}.\lambda^t_{l'} \geq 0 \quad \substack{\forall t\in T\\ \forall a\in A} \tag{47}$$

References

[1] Jason Pontin. Bill joy's six webs, 2005.

[2] Kumar Avijit and R Chinnaiyan. Iot for smart cities. 2018.

[3] Jingjing Yao and Nirwan Ansari. Qos-aware fog resource provisioning and mobile device power control in iot networks. *IEEE Transactions on Network and Service Management*, 2018.

[4] José Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. Resource provisioning for iot application services in smart cities. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–9. IEEE, 2017.

[5] 5G Americas. New services & applications with 5g ultra-reliable low latency communications. Technical report, 5G Americas, 2018.

[6] Mehdi Bennis, Mérouane Debbah, and H Vincent Poor. Ultra-reliable and low-latency wireless communication: Tail, risk and scale. *arXiv preprint arXiv:1801.01270*, 2018.

[7] Hyoungju Ji, Sunho Park, Jeongho Yeo, Younsun Kim, Juho Lee, and Byonghyo Shim. Introduction to ultra reliable and low latency communications in 5g. *Computing Research Repository (CoRR) abs/1704.05565*, 2017.

[8] David Öhmann, Meryem Simsek, and Gerhard P Fettweis. Achieving high availability in wireless networks by an optimal number of rayleigh-fading links. In *Globecom Workshops (GC Wkshps), 2014*, pages 1402–1407. IEEE, 2014.

[9] Faisal Tariq, Muhammad Khandaker, Kai-Kit Wong, Muhammad Imran, Mehdi Bennis, and Merouane Debbah. A speculative study on 6g. *arXiv preprint arXiv:1902.06700*, 2019.

[10] Walid Saad, Mehdi Bennis, and Mingzhe Chen. A vision of 6g wireless systems: Applications, trends, technologies, and open research problems. *arXiv preprint arXiv:1902.10265*, 2019.

[11] Guangxu Zhu, Dongzhu Liu, Yuqing Du, Changsheng You, Jun Zhang, and Kaibin Huang. Towards an intelligent edge: Wireless communication meets machine learning. *arXiv preprint arXiv:1809.00343*, 2018.

[12] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing: a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[13] Jianhui Liu and Qi Zhang. Offloading schemes in mobile edge computing for ultra-reliable low latency communications. *Ieee Access*, 6:12825–12837, 2018.

[14] Nouha Kherraf, Hyame Assem Alameddine, Sanaa Sharafeddine, Chadi Assi, and Ali Ghrayeb. Optimized provisioning of edge computing resources with heterogeneous workload in iot networks. *IEEE Transactions on Network and Service Management*, 2019.

[15] Peiyue Zhao and György Dán. Resilient placement of virtual process control functions in mobile edge clouds. In *IFIP Networking Conference (IFIP Networking) and Workshops, 2017*, pages 1–9. IEEE, 2017.

[16] Osman NC Yilmaz, Y-P Eric Wang, Niklas A Johansson, Nadia Brahmi, Shehzad A Ashraf, and Joachim Sachs. Analysis of ultra-reliable and low-latency 5g communication for a factory automation use case. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 1190–1195. IEEE, 2015.

[17] Sakib Mahmud Khan, Mashrur Chowdhury, Mizanur Rahman, and Mhafuzul Islam. Feasibility of 5g mm-wave communication for connected autonomous vehicles. *arXiv preprint arXiv:1808.04517*, 2018.

[18] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[19] Xiang Sun and Nirwan Ansari. Latency aware workload offloading in the cloudlet network. *IEEE Communications Letters*, 21(7):1481–1484, 2017.

[20] Ziling Wei and Hai Jiang. Optimal offloading in fog computing systems with non-orthogonal multiple access. *IEEE Access*, 6:49767–49778, 2018.

[21] Chen-Feng Liu, Mehdi Bennis, Mérouane Debbah, and H Vincent Poor. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Transactions on Communications*, 2019.

[22] Mohammed S Elbamby, Mehdi Bennis, Walid Saad, Matti Latva-Aho, and Choong Seon Hong. Proactive edge computing in fog networks with latency and reliability guarantees. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):209, 2018.

[23] Long Qu, Maurice Khabbaz, and Chadi Assi. Reliability-aware service chaining in carrier-grade softwarized networks. *IEEE Journal on Selected Areas in Communications*, 36(3):558–573, 2018.

[24] Mutsunori Yagiura and Toshihide Ibaraki. The generalized assignment problem and its generalizations.

[25] Sven O Krumke and Clemens Thielen. The generalized assignment problem with minimum quantities. *European Journal of Operational Research*, 228(1):46–55, 2013.

[26] Manuel Laguna, James P Kelly, JoséLuis González-Velarde, and Fred Glover. Tabu search for the multilevel generalized assignment problem. *European journal of operational research*, 82(1):176–189, 1995.

[27] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[28] Long Qu, Chadi Assi, Khaled Shaban, and Maurice Khabbaz. Reliability-aware service provisioning in nfv-enabled enterprise datacenter networks. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 153–159. IEEE, 2016.