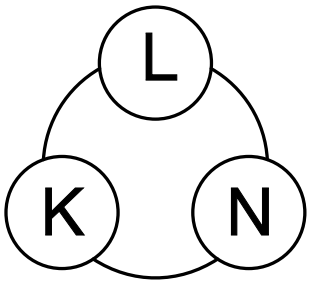


Technische Universität München
Lehrstuhl für Kommunikationsnetze

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Kommunikation und Navigation

Master's Thesis

A Timeliness-based Data Computing/Gathering Offloading Model for Internet of Things Devices

Author:	Guloglu, Umut
Address:	Sintpertstr. 42, 81539 Munich, Germany
Matriculation Number:	3751016
Supervisor:	Prof. Dr.-Ing. Wolfgang Kellerer (TUM)
Advisor:	Dr. Andrea Munari (DLR)
Co-Advisor:	M.Sc. Polina Kutsevol (TUM)
Begin Date:	01. August 2023
End Date:	01. February 2024

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 01.02.2024	
Place, Date	Signature

München, 01.02.2024	
Place, Date	Signature

Abstract

Numerous Internet of Things (IoT) applications demand accurate and timely information to enable effective actuation. Nonetheless, in computation-intensive status update systems or data gathering systems, the capabilities of IoT devices may fall short in computing/acquiring data with high accuracy, thereby necessitating multiple trials. Offloading data computation or acquisition tasks to robust units mitigates this inaccuracy. However, these units can be positioned far from the user, and latency becomes an issue for offloading due to some factors, such as propagation delays and resource sharing with other possible tasks. In this thesis, we study how to keep the information accurate and fresh, introducing a cost function representing the staleness of the recently obtained data that is accurate enough for actuation. We propose a timeliness-based model striking a balance between employing local and remote resources. We consider two settings to treat the problem, namely blind and informed decision settings. In the blind setting, we utilize a stochastic decision-making strategy where the user makes the offloading decisions without knowledge of the current value of the cost function. We conduct a steady-state analysis and solve the problem through convex optimization. We also extend this setting to multi-user scenarios sharing the same remote resources. In the informed decision setting, we address the optimization problem as a Markov Decision Problem (MDP), in which the user leverages the current cost function value. We resolve the issue using finite horizon Dynamic Programming (DP). We state that while the informed decision setting yields superior results, it also has several drawbacks, such as consuming extensive memory and flexibility.

Acknowledgement

Without the support of my mentors, colleagues, friends and family, this thesis would not be possible. First and foremost, I am deeply thankful to my thesis supervisor, Prof. Dr.-Ing. Wolfgang Kellerer, for providing me with the opportunity to be a part of the Chair of Communication Networks, as well as for his guidance and efforts.

I am deeply appreciative of Dr. Andrea Munari for the crucial support from the German Aerospace Center, funding me as a researcher throughout my thesis, and for his mentoring. His contributions have been invaluable.

It would not be possible to express my gratitude to my thesis advisor Polina Kutsevol for her guidance, her patience in dealing with my questions, providing me with the most valuable feedback.

I would like to also thank all my classmates and colleagues that were with me during the hardest of times, and helped me get back on track each time I felt lost.

Lastly, my family and friends have always been the greatest source of motivation and inspiration for me in my life. Without them, I would not be here today, and this thesis would not exist. Thank you all for everything!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the contemporary digital age, the Internet of Things (IoT) plays a vital role in establishing a hyper-connected ecosystem with a myriad of devices. The number of IoT devices connecting to the Internet is expected to reach 83 billion by 2024 [1], more than ten times the human population. It will create a massive amount of data; however, realizing IoT's transformative potential hinges upon the availability of current and actionable data rather than voluminous data. For instance, the timely procurement of accurate data is the primary purpose in some real-time application areas, such as healthcare [2], transportation [3], agriculture [4], and Industrial IoT [5]. In these settings, keeping information up-to-date is crucial.

In computation-intensive status update systems or data-collecting systems, the accuracy and latency of computed/collected data directly influence the efficacy of user actions. Cloud servers, having powerful computational units, storage areas, and resources, offer significant accuracy to these systems with higher latency. However, prior to the introduction of Multi-Access Edge Computing (MEC) [6, 7], which advocates for placing the resource-rich units near to the user, such as base stations, the significant distance between the cloud server and the user was a considerable issue for time-sensitive applications. MEC facilitated the offloading concept and, accordingly, garnered significant interest not only within academic literature but also among different corporations, such as IBM and Nokia [8, 9]. Contrastingly, data can be procured using solely local resources, in which comparatively inaccurate data is acquired with low latency [10]. Both computation-intensive status update systems and data collecting systems experience a hard choice between accuracy and latency. Within the context, the terms local and remote create a dichotomy.

Within this thesis, the term "Remote" refers to the robust computational units and vast data repositories strategically positioned outside the immediate proximity of the user or the operational environment, like a cloud server. They can provide accurate and detailed data at the cost of latency, which may impede the time-sensitive decision-making processes depending on the system. Users can also suffer from network congestion whenever the

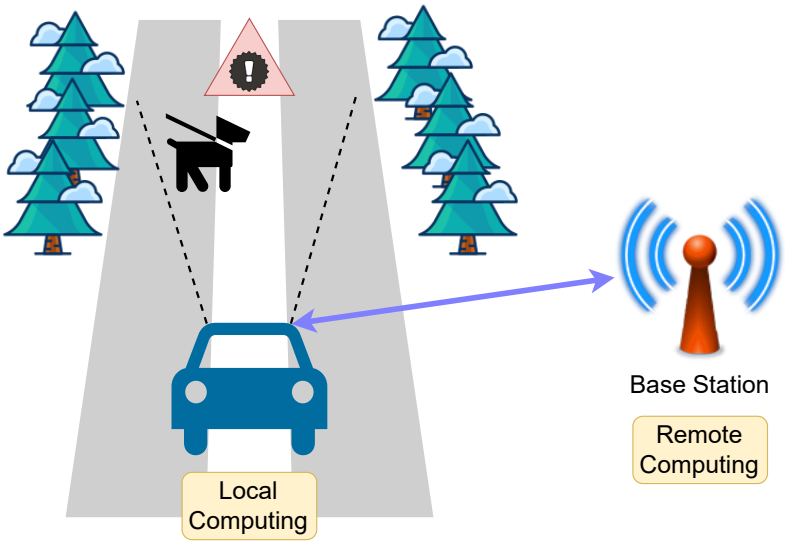


Figure 1.1: Vehicle computation use case

processing units of the server or bandwidth are insufficient to handle the high use. On the other hand, the term "Local" refers to computationally limited resources in proximity, such as embedded sensors and computational units inside an IoT node or the sensors in the vicinity and the processing resources on the user. The service extended to the user is marked by its notably low latency but may be beset by inaccuracy. The users must maintain a delicate equilibrium between latency and accuracy, bearing in mind the local computational limitations of IoT devices due to power, processor sharing, and duty cycle factors [11]. This study offers an analytical framework in which all related parameters are included and presents optimal user behavior in user-driven IoT offloading scenarios.

1.1 Motivation

The setting under study captures a key trade-off that applies to different application scenarios. First, assume an autonomous vehicle surveillance system where the goal is to gather data about the road, traffic, and possible dangers and take action in case of jeopardies, as pictured in Fig. 1.1. The vehicle's controller needs the latest information about the situation, which will be revealed after computing the data collected via various sensors such as LiDAR, radar, and cameras. The sensors in the vehicle collect pre-processed data of the region and may (i) process them using local computation units or (ii) offload the computation task to a server outside of the vehicle such as roadside units (RSU) [12]. The latter requires much time due to the integration of data from other vehicles or more refined data processing; however, the accuracy of its computation result exceeds that of the former.

In some IoT applications, garnering indispensable information for actuation might be difficult in the absence of external sources [13]. Procuring and storing this type of information

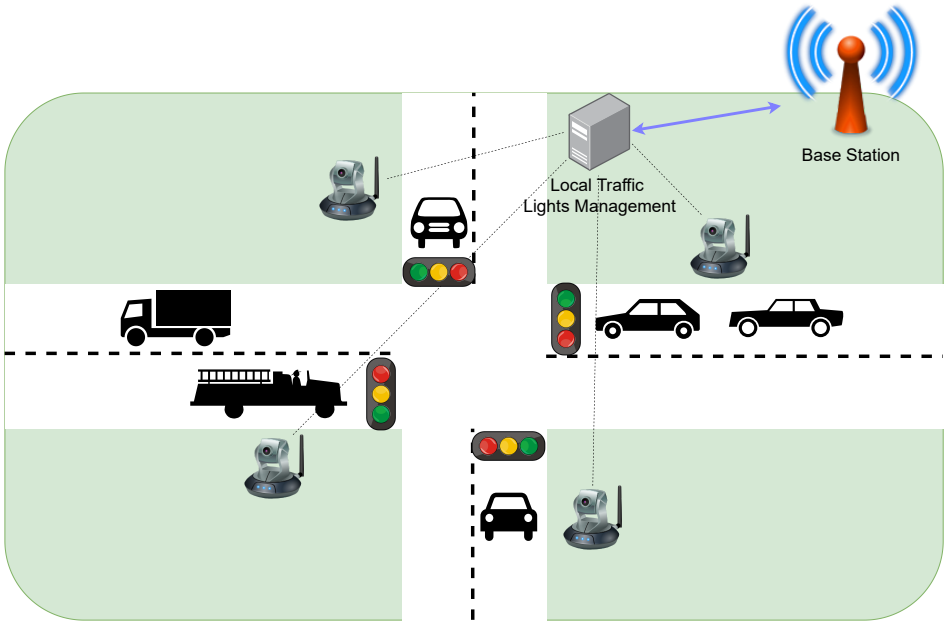


Figure 1.2: Traffic lights use case

can be challenging because of the limited resources of the IoT node [14]. The second use case instance, portrayed in Fig. 1.2, is inspired by this setting. Suppose a traffic lights management system desires to direct the traffic as efficiently as possible, requiring traffic information acquired either by (i) locally computing pre-processed data from cameras or sensors located in proximity or (ii) requesting traffic data of a related position from a remote server via the base station. The results obtained via local units may not be fully reliable due to low computation power. Conversely, the traffic data from the remote server is accurate as it can employ data from various sources to obtain traffic information, but high latency is an issue. In addition to the Round-Trip Time (RTT) delay, the specific information might not be available in the connected edge server, obligating the transfer of the request to another edge, which indirectly increases the latency.

In both scenarios, the fundamental problem is to strike a balance between latency and accuracy under the constraint of having limited local resources due to factors such as energy or other independent tasks' computational power consumption. The idea can be generalized to settings in which the local option stands for inaccuracy and low latency, and the remote option symbolizes accuracy and high latency. For instance, local action can be regarded as gathering data from another vehicle, i.e., Vehicle-to-Vehicle (V2V) [15]. In contrast, remote action can be using roadside units (RDU) to acquire the same data in a Vehicle-to-Everything (V2X) [16] data gathering scenario. Additionally, the interaction of the nodes within a cluster can be seen as local against offloading to fog or edge servers, accepted as remote in the dichotomy [17]. This study responds to a fundamental question: When shall the user offload?

Within the context of this thesis, the offloading term is used for using remote resources, not positioned in the vicinity, for both data computing and gathering scenarios. Many parameters, such as the distance between the user and the remote server and the units' capabilities, influence the decision. For instance, when the remote server is far away or numerous users are connecting to the same server, **not** offloading is the wisest choice. However, it is pertinent to underscore that there might be some limitations on employing local resources. Hence, **not** using neither of the resources might be the ideal move.

Another critical point is to map the accuracy into the planned model. Quantifying the output of a task can be complicated. While the output of Task A may be more accurate, i.e., high-quality, than that of Task B, both can still be categorized as low-quality. According to Juran's definition in [18], data is considered high-quality if they are fit for its uses in operations, decision-making, and planning. This definition serves as the foundation for the model we will develop. We define the output of a task as high-quality if it can be used to actuate and assume the user, or remote server, can differentiate between low-quality and high-quality output. Suppose the user obtains low-quality output while using local data collection. In that case, it is regarded as a failure (non-usable). Conversely, if it has high quality (local or remote), it is accepted as a successful task execution.

This thesis aims to design a timeliness-targeted model for offloading scenarios in time-critical data computing or gathering applications. We favor harnessing a user-driven decision-making model, defining a cost function representing the staleness, and employing convex optimizations and dynamic programming.

1.2 Related Work

The scientific literature on utilizing a server located far away from the user, along with a resource within the proximity, often overlooks data gathering use cases and primarily focuses on the computation offloading scenario. Within the data gathering context, the dichotomy of local and remote is generally noted in caching systems [19–23]. Caching reduces the experienced latency and the network load by locating a storage unit closer to the user end [19].

Many studies delve into different aspects of caching strategy and architecture. For instance, [20] explores hit rate and latency in a vehicle caching system, whereas [21] establishes an optimization model to jointly minimize delay and energy consumption in Mobile Edge Caching systems. The primary challenge is that the classical caching concept is developed to reduce latency and is generally unsuitable for time-sensitive applications. Even so, studies have investigated the timeliness of a flow in a caching system. For example, the duality between delay and freshness is investigated in [22], and fresh caching for dynamical content changes is studied in [23]. Even though there are two centers, there is a single source in these systems, and both caching nodes and the original remote server store data without any accuracy differences, and the user does not make any choice between the server and caching node. Thus, it is significantly different from our data-gathering use scenario.

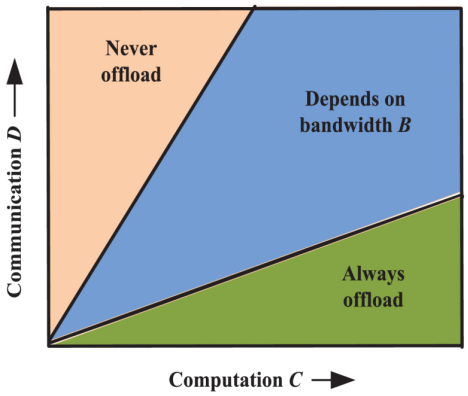


Figure 1.3: Duality of communication and computation (reproduced from [28])

Conversely, there are numerous studies investigating data computing architecture. The initial research on computation offloading primarily focused on Mobile Cloud Computing (MCC), where mobile devices utilize a distant (remote) server. Despite the success of traditional cloud applications like Siri and iCloud, they still face challenges with response time [24]. As a solution for the issue, Satyanarayanan et al. [25] publicized Cloudlet as an idea of mitigating the device's resource poverty with the help of nearby resource-rich units. In 2013, IBM and Nokia Siemens defined Mobile Edge Computing to portray computing facilities in a base station [8]. Subsequently, the European Telecommunications Standards Institute (ETSI) established an Industry Specification Group (ISG) to standardize this concept in December 2014 [9]. The name of this concept was later changed to Multi-Access Edge Computing to encompass various communication technologies. Throughout the evolution of the concept, the main idea, placing powerful computation units on the network's edge, remains unchanged. Utilizing MEC allows users to leverage remote computing capabilities with minimal latency, making MEC convenient for delay-sensitive practices.

After introducing the MEC paradigm, the concept of offloading engaged the attention of various studies, each exploring it from different perspectives. Researchers aimed to address questions of "When" (or "Whether") [11, 26–28], "Where" [27, 29, 30], "What" [26, 30–33] and "How" [11, 24, 34] to offload. We shall now delve into these questions to provide a comprehensive exploration.

Q1. **When to offload?**

The response to the question "When" pertains to the time instants of offloading. At times, offloading every task may be the better option, while in some cases, not offloading might be preferable. Kumar et al. [28] suggest offloading in scenarios where the application demands much computation and little communication (see Fig. 1.3). Conversely, not offloading is the wisest choice when the computation is minimal, and communication is significant. In cases where the required computation and communication are roughly equivalent, the decision depends on other factors. Throughout this thesis, we aim to address this question by taking into account various influencing factors.

Q2. **Where to offload?**

The response to this inquiry is context-dependent, potentially encompassing at least one MEC server, Fog server, another vehicle, and so forth. In [27], the user selects between an edge and cloud server to offload, whereas another research [29] models an offloading system where an edge server can offload the incoming tasks to other edge or cloud servers. The common feature is that the options available for offloading are entities not located within the immediate operational range. In our proposed model, the offloaded server, named the remote server, is the only option for the user to offload.

Q3. **What to offload?**

There are different studies regarding this issue, yet they predominantly fall into two distinct categories [31]. On the one hand, full offloading [32], which is also called binary offloading or coarse-grained, represents the unity of each task. The user performs the task locally or offloads the task to the remote server. On the other hand, partial offloading [33], also called fine-grained, stands for offloading solely the computation-hungry parts of the tasks. We favor using the former since it is appropriate for both data computing and data gathering use cases.

Q4. **How to offload?**

The right offloading path is another issue to be addressed. Various studies explore the preferable wireless interfaces, such as cellular and WiFi, for offloading [24,34], while certain other research use the queuing theory approach to answer this question [24]. It is essential to note that these specific approaches are outside the scope of consideration in this thesis.

The presented contributions generally center their attention on different objectives, such as maximizing throughput, enhancing the quality of experience (QoE) or quality of service (QoS), or minimizing latency, consumed energy or computation cost [31]. For instance, [35,36] focus on improving QoE or QoS by offloading computations. On the other hand, [37] aims to maximize the offloaded task, or throughput, in a downlink scenario where the data computations can be done at the edge server or in the end user, whereas [38] explores an uplink scheme problem by allocating power and resources to the users to maximize throughput.

Potential computational limitations, primarily arising from constraints such as limited energy, processor availability, and duty cycle factors, are typically examined under the energy-harvesting (EH) problem or the optimization based on energy consumption. For instance, Mao et al. [39] devised an EH framework to determine a computation offloading strategy minimizing both latency and computation failure. They employed Lyapunov optimization to decide whether to offload, compute locally, or drop, along with determining power and CPU-cycle frequencies. On the other hand, the researchers in [29] treated this limitation issue as a constrained multi-objective optimization problem (CMOP), aiming to jointly minimize processing delay and consumed energy. Nevertheless, potential limiting factors other than energy are generally overlooked.

The in-depth examination of latency within offloading scenarios is another area of research. Zhao et al. [30] focused on the interplay between accuracy and latency, ferreting local, edge, and cloud computing, and proposed offloading only a fraction of data to the remote servers (edge or cloud) instead of an all-or-nothing paradigm. They could lessen the latency by 25% while obtaining minor reductions, or even advancements in some cases, in accuracy. A similar research [26] investigates a centralized offloading decision-making strategy aimed at minimizing overall latency. The authors tackle the problem by decomposing it into two sub-problems: communication and computation resource allocation, which they solve using convex optimization tools.

Addressing a different facet of the issue, Mancuso et al. [27] explored different routing selection algorithms for virtualized services over edge or cloud servers under latency constraints. The algorithms contrasted based on the amount of knowledge of the system. The researchers found that stateless methods, where the user did not know the servers' occupancy, gave comparable results to more complex state-based algorithms in the case of perfect information. However, stateless methods outperformed their more complex counterparts in the case of inaccurate state information, as incorrectness brought forth erroneous offloading choices.

Among all computation offloading studies, only a limited number of them scrutinize the notion of information freshness despite its significance as one of the Key Performance Indicators (KPI) for time-sensitive actuating applications. Although freshness and latency are related, they are distinct concepts. Freshness pertains to an information flow, whereas the delay, or latency, is about a data packet. Transmitted data can become stale over time due to propagation, queuing, or processing delay. However, there are other factors on which freshness depends. Substantially, minor latencies might even increase the staleness depending on the updating rate [40].

In order to picture the freshness of a data flow process while taking cognizance of these factors, the concept of the Age of Information (AoI) was introduced [41–43]. Within the AoI framework, the age of status denotes the time difference between the current time and the generation timestamp of the last acquired data, indicating the flow's freshness or timeliness. In [44], this definition was extended to Age of Processing (AoP), where the age represents the elapsed time of the latest processed data, highlighting additional processing delays for edge computing-enabled applications. Studies from various sectors utilized the age concept to enhance the freshness in time-sensitive systems.

Ndikumana et al. [34] studied the offloading problem for autonomous vehicles, where the vehicles could choose one of the defined Radio Access Technology (RAT) to send the preprocessed data to the edge. They underlined the energy problem of the vehicle in the case of purely operating local computation and selected their goal as reducing AoP. To achieve this, they employed unsupervised machine learning techniques. Another study, detailed in [11], examined computation offloading for the Internet of Medical Things (IoMT). The authors aimed to develop an offloading strategy based on AoI, criticality, and energy consumption. Employing a game-theoretic approach, the authors derived a decentralized solution.

A general queuing theory approach, applicable to various use cases, was adopted in [45]. The authors analyzed a multi-user downlink system, seeking to minimize average AoI. They utilized an entirely local computing scheme and a fully remote computing scheme, also known as full offloading [32], and obtained closed-form expressions for average AoI.

Another study [46], which bears similarities to ours, examined a single-user computation-intensive status update system. This study employed three different computing schemes: (i) Fully edge server computation, (ii) Fully local server computation, and (iii) Partially edge, partially server computation (akin to [33]). The authors underlined that partial offloading consistently yielded equal or superior results. Unlike our study, the work did not account for local computation limitations or assess the accuracy of local resources.

Finally, it shall be noted that AoI is not the sole metric introduced to capture freshness. Many studies in the literature, such as [47,48], adopted a different timeliness representation. The crucial aspect is the capability to express dissatisfaction with untimeliness. In this thesis, we define a cost function signifying our dissatisfaction regarding the staleness of the most recently computed or gathered data. Based on this function, we devise a framework for scenarios involving both data computing and gathering offloading, considering local computation limitations and the accuracy of the computations (or the quality of data in gathering scenarios).

1.3 Our Contributions

In this study, we propose a user-driven timeliness-based framework, applicable for computation offloading or dual data gathering situations in which the user experiences a dilemma between *Local* and *Remote* resources. Our research contributions, outlined in connection with the thesis' structure, can be summarized as follows:

- In Chapter 2, we introduce a slotted-time model and outline the potential states of a user. Moreover, we define a cost function representing the staleness of the flow and system parameters that characterize the system.
- In Chapter 3, we study a blind decision-making setting where the user determines the next move, i.e., local or remote action, independent of the current cost value. We conduct a steady-state analysis and derive a closed-form expression for the average cost function. The optimal closed-form offloading solution is determined using convex optimization for single-user scenarios. We extend the model for multiple-user scenarios and observe the system's characteristics.
- In Chapter 4, we focus on an informed decision-making setting where the user leverages the current state and time. We employ Dynamic Programming (DP) over a finite horizon to obtain optimal offloading sequences and assess the power of knowledge.
- In Chapter 5, we compare the outcomes between blind and informed decision-making settings and address several issues regarding the settings. The study is concluded in Chapter 6.

Chapter 2

System Model

This chapter introduces a simple time-slotted model to study the settings and use cases presented in Chapter 1. Each time slot is of equal length within the model and denoted by an index $k \in \mathbb{N}_0$. Before explaining the model in detail, several terms are defined.

Definition 1 (Task)**.** A task represents an atomic activity that the user has to perform. This may encompass either data collection, processing, or both, depending on the context. The assumption is that a new task arises once the previous one is executed. Hence, there is always a task that must be addressed.

Definition 2 (Availability State, (ς))**.** The availability state is the user's status representing availability for performing tasks. In other words, it describes whether the user is *Idle* $(\varsigma = 0)$ or *Busy* $(\varsigma = 1)$. In *Busy* state, the user cannot take any action and has to wait until becoming *Idle*, whereas *Idle* state indicates the user is currently available and can take one of three possible *Actions*.

Definition 3 (Action, (u))**.** The user's decision pertaining to performing tasks is called an action. In this context, three possible actions can be taken when the user's availability state is *Idle*.

- **Remote:** Send a request to the remote server (High percentage of success with possible high latency);
- **Local:** Try to perform the task via local resources (Lower percentage of success with possible low latency);
- **Wait:** Do nothing.

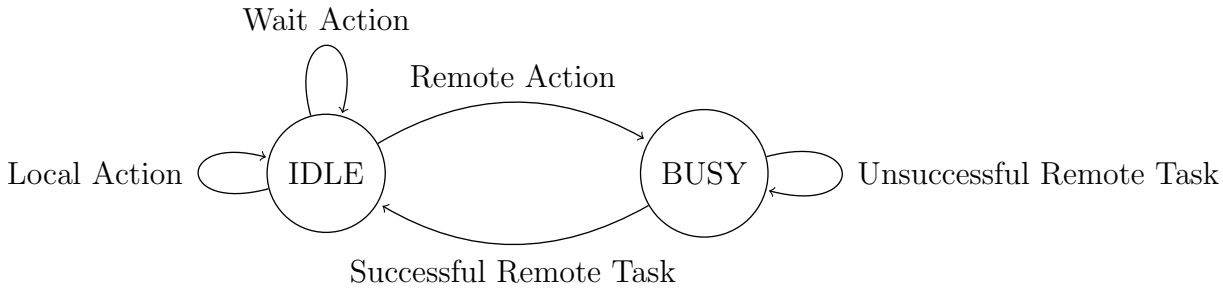


Figure 2.1: Availability state transition diagram of the model

When the user's availability state is *Idle* and the user takes a **Remote** action, the availability state becomes *Busy* in the subsequent time slot. It stays there until the remote server successfully completes the task, and when this happens, the user's state reverts to *Idle*. Taking **Local** or **Wait** action does not change the availability state, and the user remains *Idle*. In other words, an attempt to perform the task locally takes one slot. As mentioned, the attempt may be successful or not. A graphical representation of these state transitions is provided in Fig. 2.1.

Definition 4 (Cost (Penalty) Function, (Δ))**.** The cost function, also called the penalty function, pictures the staleness of the most recent task completion. It quantifies the time difference between the current time and the instant of the last successful task execution. It grows linearly until a new task is successfully performed, at which point it drops to 0. The cost function maps the time slot index (k) to a natural number, including 0, and the value of the cost function at time index k, denoted by Δ_k, is defined as:

$$\Delta : \mathbb{N}_0 \longrightarrow \mathbb{N}_0$$

$$\Delta_k = \begin{cases} 0 & \text{if a new task is successfully performed at } (k-1)^{th} \text{ time slot,} \\ \Delta_{k-1} + 1 & \text{otherwise} \end{cases}$$

Without loss of generality, we shall assume $\Delta_0 = 0$. It is worth noting that the cost function is discrete. It is assumed that the action is taken at the start of a time slot, while the effect of that action is observed by the end. In simpler terms, Δ_k equals the cost right before k^{th} slot begins. A simple realization and cost changes based on the actions taken can be seen in Fig. 2.2. In this scenario, the cost function equals m just before $(k-1)^{th}$ slot. At the onset of this slot, the user selects **Wait** action, resulting in an increase of the cost by the end of the time slot, setting $\Delta_k = m + 1$. Next, the user chooses to perform the task locally and succeeds. As a result, the function decreases to 0 by the end of the time slot, and Δ_{k+1} becomes 0. At the onset of $(k+1)^{th}$ time slot, the user makes another decision, and this sequence continues.

The cost function we consider resembles classical AoI or AoP functions [41, 44]. In the literature, similar studies about freshness [49–52] generally decrease the cost function to 1

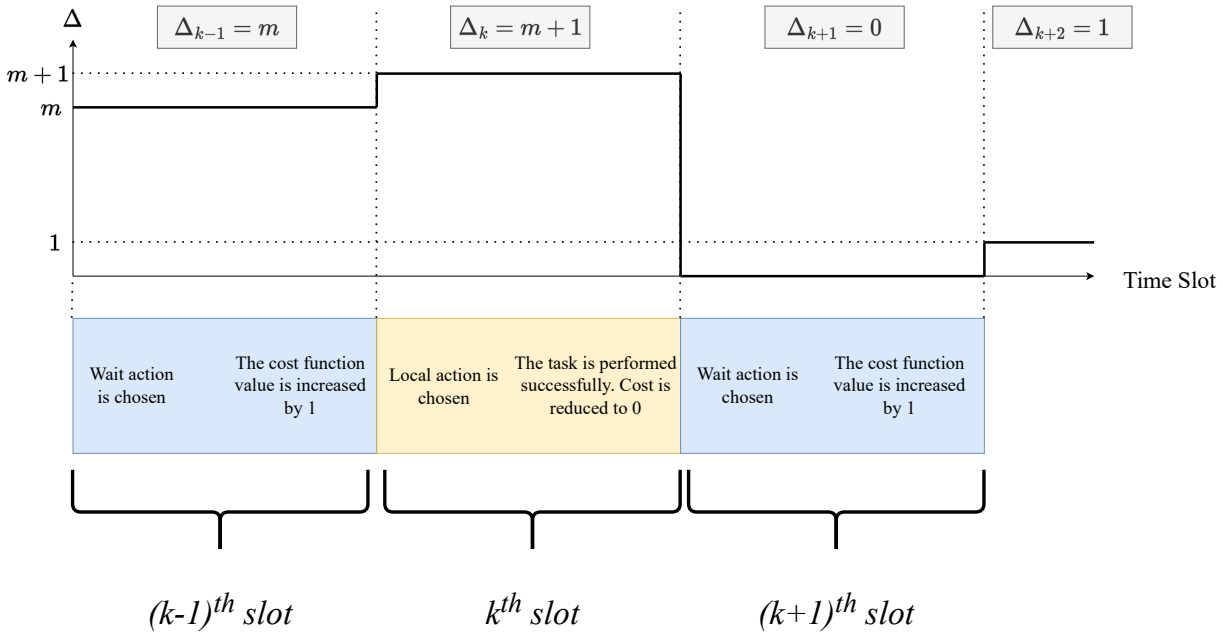


Figure 2.2: Time slots representation

instead of 0. However, this does not affect anything except raising the cost function value by 1 at each time slot and does not change the optimization results.

When designing a cost function, one could use an approach that assumes it is continuous and increases linearly even within a time slot until a new task is completed, after which it drops to zero. Figure 2.3 shows an illustration comparing discrete and continuous cost functions. Notably, a 0.5 (time slot $\times$ cost) difference in the area under the curves at each time slot creates a bias when calculating average cost, for instance. However, this is the only difference; ultimately, the simple discrete model explained previously is preferred.

Definition 5 (Reset)**.** A reset denotes completing a task successfully and returning to the *Idle* ($\varsigma = 0$) availability state with a cost function equal to 0. Reset is the only way to diminish the cost function. Visual representations of these resets are illustrated in Fig. 2.3. For example, at time slot $k = 5$, the user, or remote server, performed the task successfully, initiating the fall at the end of that time slot.

Definition 6 (Inter-Refresh Time (χ))**.** Inter-refresh time is the time between two consecutive resets. For example, in the specific case of Fig. 2.3, the χ values follow the sequence $6, 1, 4, 3, 5$, assuming that there is a reset prior to the initial time slot $k = 0$.

In order to create a model that accurately reflects real-world conditions, considering numerous variables and factors is necessary. However, achieving a perfect reflection can be unfeasible in many cases. Therefore, several assumptions are made in order to simplify calculations while striking a balance between accuracy and practicality.

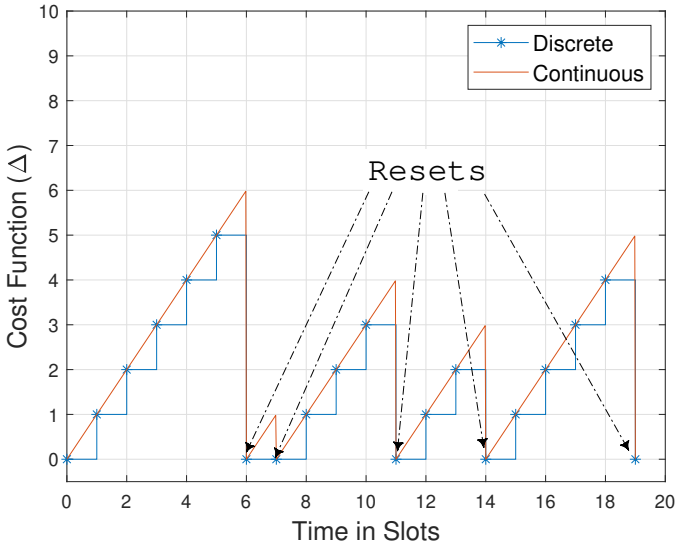


Figure 2.3: Comparison of discrete and continuous cost functions

Assumption 1. Each time slot has a duration equal to or longer than the time necessary to complete the local task, including the time needed to collect raw data from local sensors, process it, obtain desired data, and assess the success of the task. For remote tasks, the propagation delay is smaller than the length of a time slot, guaranteeing that the request arrives at the server by the end of the time slot.

Assumption 2. It is assumed that the remote server executes the task successfully within a time slot with a probability $p \in (0, 1]$, named remote latency parameter. With probability $1-p$, the task is not completed and will continue to be dealt with remotely in the next slot. This may occur because the server might be busy with other tasks or due to processing time. In general, it is also possible that the server cannot process the task and forward it to another available server. These details are hidden within our model. In the end, the remote server delay, i.e., the number of time slots spent in *Busy* state, is **geometrically** distributed, and the system is memoryless, simplifying the calculations on the model.

To illustrate a general overview of the changes in the cost function with different actions, Fig. 2.4 is given. The green and red shaded areas represent the times when the user is *Idle* and *Busy*, respectively. The black circles indicate chosen **Local** actions, while the red stars represent the time slots when a **Remote** action is selected, i.e., the transition points from the *Idle* to the *Busy* state. On the other hand, the blue stars represent the time slots when the remote server completes the task successfully, i.e., the transition points from the *Busy* to the *Idle* state. All markers are placed at the beginning of the related time slot to depict that users make decisions at the beginning of a time slot. Hence, the transitions occur by the end of the transition slots. This example assumes the user always performs

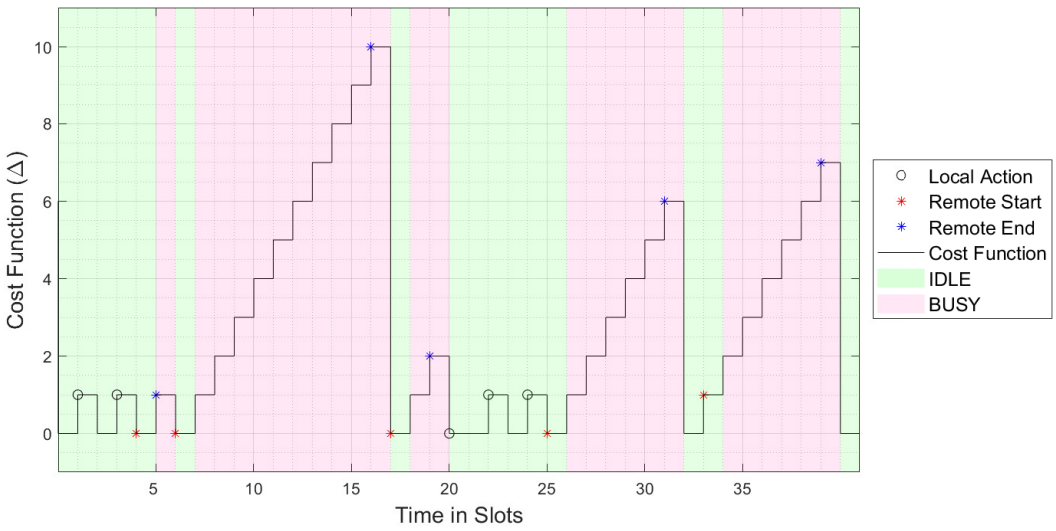


Figure 2.4: Cost function versus time

the tasks successfully when using a **Local** action. However, this generally does not hold, and the model must take this into account.

Definition 7 (Local Success Probability, (α))**.** The probability of successfully executing a task through **Local** action is termed as the local success probability, denoted by $\alpha \in [0, 1]$. Values of $\alpha = 0$ and $\alpha = 1$ signify consistent failure or success in task completion, respectively. The trials are independent and identically distributed (i.i.d.) across different time slots. In other words, the outcomes of the prior trials do not affect the outcomes of the incoming ones.

The local success probability parameter is subject to the influence of several internal and external factors. For instance, deploying high-resolution sensors and high-computing power can enhance the parameter's value, while adverse weather conditions might impede it.

The primary objective of the research is to optimize the behavior of the nodes by minimizing the cost function. We consider two different settings: blind and informed decision optimizations. In the first case, the average cost function over an infinite horizon is examined, where the user employs stochastic decision-making independent of the current status. We aim to determine the optimal action probabilities, thereby minimizing the average cost. To tackle this setting, we employ a steady-state analysis as discussed in Chapter 3. In the informed decision case, the user uses the current cost and time values, facilitating informed decision-making. We apply the Dynamic Programming approach to obtain optimal action sequence, as elaborated in detail in Chapter 4. Additionally, one of the critical components of the model is the restriction on using local tasks, which is implemented differently depending on the optimization approach used and is explained in the corresponding chapter.

Chapter 3

Blind Decision

The blind decision setting serves as our initial strategy, employed when the user determines the next move independent of the current cost function value. This setting is particularly valuable in scenarios where memory constraints are an issue. In such scenarios, the main goal is to lessen the average cost over an infinite time horizon. The user utilizes stochastic decision-making and assigns a probability to each action. The question that arises is how a user should pick these probabilities.

3.1 System Model Extensions

Chapter 2 introduced the key elements of the system model. To tackle the case under study, we incorporated some additional modifications. First, let us assign a probability to each of the available actions. These actions include **Wait**, **Local**, and **Remote**, whose probabilities are denoted as P_W, P_L, and P_R, respectively.

- $P(\text{Remote Action} \mid \text{User is Idle}) = P_R$
- $P(\text{Local Action} \mid \text{User is Idle}) = P_L$
- $P(\text{Wait Action} \mid \text{User is Idle}) = P_W$

It is important to note that these probabilities must sum up to 1 at each time slot. That is, $P_W + P_L + P_R = 1$. Fig. 3.1 reports the availability state transition diagram, illustrating the availability states and transitions that can occur. Additionally, the stationary probability of being in the idle state π_I or the busy state π_B is seen in Eq. (3.1), whereas its derivation is reported in Appendix A.1.

$$\pi_I = \frac{p}{P_R + p} \quad \text{and} \quad \pi_B = \frac{P_R}{P_R + p} \tag{3.1}$$

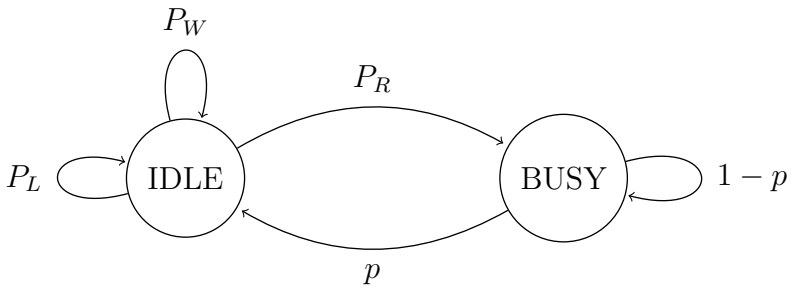


Figure 3.1: Availability state transition diagram for blind decision approach

It is essential to highlight that as the value of P_R rises, the likelihood of the user being in the *Busy* state also increases. This is because the user is more likely to use **Remote** action, which keeps it busy for extended periods. Similarly, higher remote latency parameter (p) values, signifying lower remote latencies, increase π_I since the remote server executes the tasks more quickly, resulting in the user being *Idle* more often.

A representation of the blind decision setting is pictured through the Markov Chain model, seen in Fig. 3.2. Each state represents a pair of cost (Δ) and availability state (ς). The superscript of nodes presented in the Markov Chain symbolizes the cost value, while the subscript symbolizes the availability state.

As previously mentioned, when a user is *Idle*, there are three potential actions to consider. First, **Remote** action has probability P_R. In this case, the user sends a request to the remote server and becomes *Busy* in the next time slot, awaiting a successful task execution. Secondly, the **Local** action has probability P_L. In this situation, the user uses local resources to perform a task with a probability α of being successful. As choosing an action and successful completion are independent events, the user successfully completes the local task execution with a probability of αP_L if *Idle*. The user then resets the cost and returns to the *Idle* availability state with a cost function value 0. Conversely, the user may employ a **Local** action and fail with a probability $(1-\alpha)P_L$. Finally, the **Wait** action has probability $P_W = 1 - P_L - P_R$. In this case, the user waits until the next time slot. In cases where the user waits or fails to perform the task locally, the cost value increases by one, and the availability state remains *Idle*, with a total probability of $1 - P_R - \alpha P_L$.

Assumption 3. We restrict to the cases $P_R + \alpha P_L > 0$. Other situations are inconsequential and unreflected.

When the user is *Busy*, a remote server finishes the task successfully with a probability of p, leading the user to reset the cost. Alternatively, the remote server may fail with a probability of $1-p$ and continue performing the task in the next slot. In this case, the user stays in the *Busy* availability state and waits until the remote units complete the task. The availability state transitions are shown in Fig. 3.2 with red and blue lines, with the former representing a transition from the *Idle* state to the *Busy* state and the latter representing the transition from the *Busy* state to the *Idle* state.

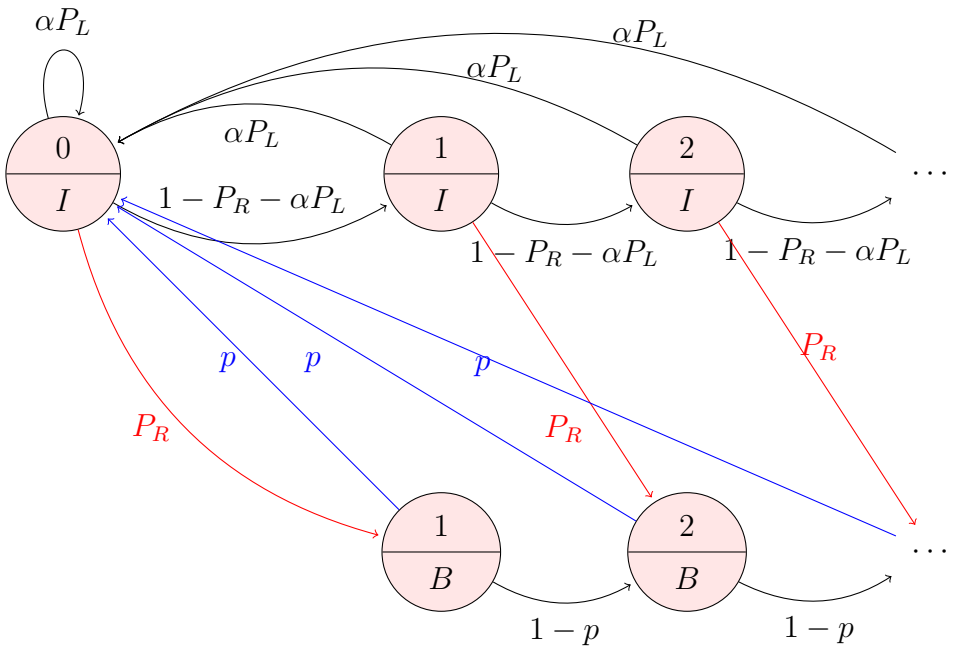


Figure 3.2: Markov chain of the system

As stated, the execution of local tasks may be constrained due to several factors, such as energy limitations or the need to perform other operations. For the blind decision approach, we introduce the following definition to specify how frequently a user can utilize local resources.

Definition 8 (Local Action Limitation Rate, (P_{lim}))**.** The Local Action Limitation Rate refers to the frequency at which a user can employ a **Local** action, regardless of whether the action is successful. The user is allowed to use **Local** actions in a certain percentage of time slots, denoted by $P_{lim} \in [0, 1]$.

Notably, the user can use **Local** actions only if it is *Idle*, and the frequency of using **Local** action is

$$\pi_I P_L = \frac{pP_L}{P_R + p}, \tag{3.2}$$

where π_I comes from Eq. (3.1). This frequency is limited by P_{lim}, i.e., $\frac{pP_L}{P_R+p} \leq P_{lim}$. After simple manipulations, P_L is thus constrained as:

$$P_L \leq \frac{P_{lim}(p + P_R)}{p}. \tag{3.3}$$

If the value of P_{lim} equals 1, there are no limitations, and the user can take the **Local** action in every time slot. Conversely, if P_{lim} is equal to 0, the user cannot use the **Local** action at all, and the only option available is to use **Remote** or **Wait** actions. Another

critical point is that P_L shall always be smaller than $1 - P_R$ since the sum $P_R + P_L + P_W$ cannot exceed 1.

$$P_L \leq 1 - P_R \tag{3.4}$$

The P_L limitations in (3.3) and (3.4) set a boundary for the local action probability. Taking this into account, we aim to determine P_R and P_L minimizing the cost. Given a fixed P_R, the user must distribute the remaining $1 - P_R$ probability to **Wait** and **Local** actions. In this case, the local action probability is preferred over **Wait** action as long as P_{lim} allows. The reason is that the user resets the cost with a probability α in case of employing a **Local** action, but the cost function always increases with the **Wait** action. Thus, the ideal P_L value, that is the optimal P_L for a fixed P_R, is calculated as shown in Eq. (3.5).

$$P_{L,ideal}(P_R) = \min\left\{1 - P_R, \frac{P_{lim}(p + P_R)}{p}\right\} \tag{3.5}$$

Our analysis will center on the steady state of the process, proved to be ergodic [53]. This framework aims to find the optimal P_R minimizing the average cost function $\overline{\Delta}$[1]. In short, the objective is as follows:

$$P_{R,optimal} = \min_{P_R} \mathbb{E}[\Delta_k], \quad \text{s.t. } P_L = \min\left\{1 - P_R, \frac{P_{lim}(p + P_R)}{p}\right\} \tag{3.6}$$

For this purpose, we first deduce the Probability Mass Function (PMF) of inter-refresh time (see Def. 6). The general closed form of the PMF of inter-refresh time, i.e., when $p \neq 1$, $\alpha P_L + P_R \neq 1$ and $\alpha P_L + P_R \neq p$, denoted as $P_\chi(x)$, is seen in Eq. (3.7), whereas the closed form of the extreme cases is in Eq. (3.8). Here, $h[x]$ is the unit step function[2], and $\delta[x]$ is the unit impulse function as defined in [54]. The derivations are in Appendix A.2.

$$P_\chi(x)_{x\in\mathbb{N}^+} = \frac{(\alpha P_L + P_R)(\alpha P_L - p)}{\alpha P_L + P_R - p}(1 - \alpha P_L - P_R)^{x-1} + \frac{pP_R}{\alpha P_L + P_R - p}(1-p)^{x-1} \tag{3.7}$$

$$P_\chi(x) = \begin{cases} \alpha P_L(1-p)^{x-1}h[x-1] + pP_R(x-1)(1-p)^{x-2}h[x-2] & \text{if } p = \alpha P_L + P_R \neq 1 \\ pP_R(1-p)^{x-2}h[x-2] + \alpha P_L\delta[x-1] & \text{if } p \neq 1, \alpha P_L + P_R = 1 \\ (\alpha P_L + P_R)(1-\alpha P_L)(1-\alpha P_L - P_R)^{x-2}h[x-2] & \text{if } p = 1, \alpha P_L + P_R \neq 1 \\ \quad +\alpha P_L\delta[x-1] & \\ \alpha P_L\delta[x-1] + P_R\delta[x-2] & \text{if } p = \alpha P_L + P_R = 1 \end{cases} \tag{3.8}$$

[1] Eq. (3.5) shows $P_{L,ideal}(P_R)$ for each $P_R \in [0, 1]$. While finding optimal P_R, we automatically derive the optimal P_L. Notably, $P_{L,optimal} = P_{L,ideal}(P_{R,optimal})$.

[2] The notation of u is used to denote actions. Hence h is preferred for the unit step function.

Lemma 1. *Let $P_\chi(x)$ be the Probability Mass Function of inter-refresh time, and assume that $\mathbb{E}[\chi]$ exists and is finite. Then, the expected value of the defined cost function $\mathbb{E}[\Delta]$, which is symbolically represented as $\overline{\Delta}$, is equal to $\frac{\mathbb{E}[\chi^2]}{2\mathbb{E}[\chi]} - \frac{1}{2}$.*

Proof. As the process is ergodic, $\overline{\Delta}$ equals the ensemble average of the defined cost function, which is the average of the cost function over time and calculated by dividing the area under the cost function by the total length.

$$\begin{aligned}\overline{\Delta} &= \lim_{k\to\infty} \frac{1}{\sum_{n=1}^{k}\chi_n} \sum_{n=1}^{k} \frac{\chi_n(\chi_n-1)}{2} = \lim_{k\to\infty} \frac{k}{\sum_{n=1}^{k} X_n} \lim_{k\to\infty} \frac{1}{k} \sum_{n=1}^{k} \frac{\chi_n(\chi_n-1)}{2} \\ &= \frac{1}{\lim_{k\to\infty} \frac{1}{k}\sum_{n=1}^{k}\chi_n} \lim_{k\to\infty} \frac{1}{k} \sum_{n=1}^{k} \frac{\chi_n(\chi_n-1)}{2} = \frac{1}{\mathbb{E}[\chi]} \mathbb{E}\left[\frac{\chi(\chi-1)}{2}\right] \\ &= \frac{1}{\mathbb{E}[\chi]} \frac{\mathbb{E}[\chi^2]-\mathbb{E}[\chi]}{2} = \frac{\mathbb{E}[\chi^2]}{2\mathbb{E}[\chi]} - \frac{1}{2}\end{aligned}$$

□

Theorem 1. *Let $P_R \in [0,1]$ be the remote action probability, $P_L \in [0,1]$ be the local action probability, $\alpha \in [0,1]$ be the local success probability (see Def. 7), and $p \in (0,1]$ be the remote latency parameter (see Assumption 2). Assuming $0 < P_R + \alpha P_L \leq 1$, then the average cost function $\overline{\Delta}$ (refer Lemma 1) is calculated as:*

$$\overline{\Delta} = -1 + \frac{1}{p} + \frac{1}{\alpha P_L + P_R} - \frac{1}{p + P_R} \tag{3.9}$$

Proof. The proof of Theorem 1 is given in Appendix A.3. □

We aim to confine the optimal P_R and P_L pair that minimizes the average cost for different parameters, such as local success probability (α), remote latency parameters (p), and local action limitation rate (P_{lim}). In the subsequent sections, the optimization issue is scrutinized for (i) one user connecting to the remote server and (ii) multiple users, where the total number of users in the system is denoted as N, utilizing remote server resources. The summary of defined parameters for the blind decision paradigm can be seen in Table 3.1.

Parameter Name	Symbol	Explanation
Availability State	ς	The user's status representing availability for performing tasks (*Busy* or *Idle*)
Cost Function	Δ	The function depicting the staleness of the most recent task completion
Inter-Refresh Time	χ	Time duration between two consecutive resets
Remote Latency Parameter	p	The probability of remote server executing the task successfully within a time slot
Local Success Probability	α	The probability of successfully executing a task through local action
Remote Action Probability	P_R	The probability of using **Remote** action if the user is in Idle availability state
Local Action Probability	P_L	The probability of using **Local** action if the user is in Idle availability state
Wait Action Probability	P_W	The probability of using **Wait** action if the user is in Idle availability state
Local Action Limitation Rate	P_{lim}	The maximum frequency at which the user can employ local resources
PMF of χ	$P_\chi(x)$	Probability Mass Function of inter-refresh time
Average Cost Function	$\overline{\Delta}$	Average of the cost function
Number of users	N	Number of users connecting to the remote server (For multiple user scenario)

Table 3.1: Blind case nomenclature

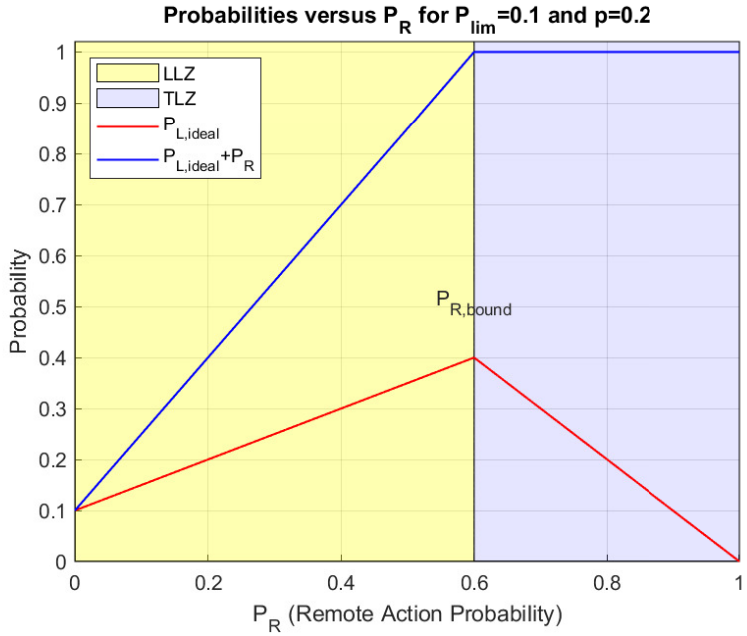


Figure 3.3: Relation between $P_{L,ideal}$, $P_{L,ideal} + P_R$ and P_R

3.2 Single User Case

As stated before, the local action probability is constrained by the local action limitation rate parameter. An illustration of $P_{L,ideal}(P_R)$ and corresponding $P_{L,ideal}(P_R) + P_R$ sum, based on Eq. (3.5), are shown in Fig. 3.3 with red and blue lines, respectively. One can observe that $P_{L,optimal}$ value equals P_{lim} at $P_R = 0$. Smaller P_R values lead to staying idle more, so P_{lim} restricts $P_{L,optimal}$ as in (3.3). As P_R grows, $P_{L,ideal}$ rises as well until P_R hits a boundary point, termed $P_{R,bound}$, satisfying $P_{L,ideal} = 1 - P_{R,bound} = P_{lim}\frac{(p+P_{R,bound})}{p}$. From this equation, $P_{R,bound}$ is calculated as:

$$P_{R,bound} = \frac{p(1 - P_{lim})}{p + P_{lim}}. \tag{3.10}$$

The remote action probability interval $[0, P_{R,bound}]$ is defined as "Local Limited Zone (LLZ)", which can be seen as the yellow area in Fig. 3.3. After $P_{R,bound}$ point, $P_{L,ideal}(P_R)$ is equal to $1 - P_R$. The interval $(P_{R,bound}, 1]$, named as "Total Limited Zone (TLZ)", is observed as the blue area in the figure. It is important to note that $P_{R,bound}$ correlates with P_{lim} parameter inversely. When $P_{lim} = 0$, $P_{R,bound} = 1$, meaning that the system is in the LLZ $\forall P_R \in [0, 1]$. Conversely, when $P_{lim} = 1$, $P_{R,bound} = 0$, meaning that the system is in the TLZ $\forall P_R \in [0, 1]$. We consider both zones mentioned above while determining an optimal P_R and P_L pair.

3.2.1 Local Limited Zone

The first region to be examined is the LLZ where $P_R \in [0, P_{R,bound}]$. Throughout the zone, P_L is restricted by P_{lim} parameter as in Ineq. (3.3). Thus, the average cost equation in Eq. (3.9) becomes:

$$\overline{\Delta}_{LLZ}(P_R) = -1 + \frac{1}{p} + \frac{1}{\alpha P_{lim} + P_R(\frac{\alpha P_{lim}}{p} + 1)} - \frac{1}{p + P_R}. \tag{3.11}$$

There are two values of P_R that cause $\overline{\Delta}_{LLZ}$ to diverge: $-\frac{\alpha P_{lim} p}{\alpha P_{lim}+p}$ and $-p$. These values are the same if and only if $p = 0$, undefined in our context. Therefore, these two P_R values are always distinct and $-\frac{\alpha P_{lim} p}{\alpha P_{lim}+p} > -p$. Moreover, both values are smaller than 0. Let us define $\aleph = -\frac{\alpha P_{lim} p}{\alpha P_{lim}+p}$ for the sake of simplicity. With the objective to comprehend the $\overline{\Delta}_{LLZ}$ dynamics, P_R space is dilated to $P_R \in (\aleph, \infty)$ while the sole possible logical P_R values are in $[0, P_{R,bound}]$. In order to find the extreme points of the cost function, the partial derivative of $\overline{\Delta}_{LLZ}$ over P_R is calculated.

$$\frac{\partial \overline{\Delta}_{LLZ}}{\partial P_R} = \frac{1}{(p + P_R)^2} - (\frac{p}{p + \alpha P_{lim}})(\frac{1}{P_R + \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^2 \tag{3.12}$$

$\frac{\partial \overline{\Delta}_{LLZ}}{\partial P_R}$ exists $\forall P_R \in (\aleph, \infty)$. The extreme points make this function equal to 0.

$$\begin{aligned}
\frac{1}{(p + P_{R,extrema})^2} &= (\frac{p}{p + \alpha P_{lim}})(\frac{1}{P_{R,extrema} + \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^2 \\
\pm\frac{1}{(p + P_{R,extrema})} &= \sqrt{\frac{p}{p + \alpha P_{lim}}}(\frac{1}{P_{R,extrema} + \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}) \\
P_{R,extrema}(\pm 1 - \sqrt{\frac{p}{p + \alpha P_{lim}}}) &= p\sqrt{\frac{p}{p + \alpha P_{lim}}} \mp \frac{p\alpha P_{lim}}{p + \alpha P_{lim}} \\
P_{R,extrema,1} = \frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}{1 - \sqrt{\frac{p}{p+\alpha P_{lim}}}} &\text{ and } P_{R,extrema,2} = \frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} + \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}{-1 - \sqrt{\frac{p}{p+\alpha P_{lim}}}}
\end{aligned} \tag{3.13}$$

Eq. (3.13) shows the extreme points, which can be local minimum or maximum. With the assistance from the second derivative test, $P_{R,extrema,1}$ proved to be the local minimum and bigger than $\aleph$ (see Appendix A.4), whereas $P_{R,extrema,2}$ is smaller than $\aleph$ (see Appendix A.5) and stays out of the interested region. One can interpret that $\overline{\Delta}_{LLZ}$ decreases between $\aleph$ and $P_{R,extrema,1}$, and rises after $P_{R,extrema,1}$. The position of $P_{R,extrema,1}$ on the P_R axis depends on the system parameters, shifting the optimal P_R ($P_{R,optimal}$) and P_L ($P_{L,optimal}$) pair. Hence, the position of this local minimum must be located.

- $\aleph < P_{R,extrema,1} \leq 0$ case

It is proven that $P_{R,extrema,1}$ is always bigger than $\aleph$. The inequality of $P_{R,extrema,1} \leq 0$ is investigated in Appendix A.6, and the condition required for $P_{R,extrema,1}$ to be non-positive is given as:

$$p \leq \alpha P_{lim}(\frac{\sqrt{5}-1}{2}) = p_{min} \approx \alpha P_{lim}(0.618). \tag{3.14}$$

In systems where the remote latency parameter p is smaller than $\alpha P_{lim}(\frac{\sqrt{5}-1}{2})$, denoted as p_{min}, the value of $P_{R,\text{extrema},1}$ is negative. Thus, $\overline{\Delta}_{LLZ}$ strictly increases along $P_R \in [0, P_{R,bound}]$, and $P_{R,optimal}$ value that minimizes the cost function equals to 0. In other words, the wisest move is not choosing any **Remote** action.

- $0 < P_{R,extrema,1} \leq P_{R,bound}$ case

This case indicates a local minimum inside LLZ, making that minimum point the optimal P_R value. First, we know the remote latency parameter p must be greater than p_{min} for $P_{R,extrema,1}$ to be greater than 0. Appendix A.7 details the derivation of the necessary conditions satisfying $P_{R,extrema,1} \leq P_{R,bound}$, given in Ineq. (3.15).

$$p^4+p^3(2+\alpha P_{lim})+p^2[2P_{lim}-P_{lim}^2(1-\alpha)^2]+p(2\alpha P_{lim}^2(1-\alpha)-\alpha P_{lim})-(\alpha^2 P_{lim}^2) \leq 0 \tag{3.15}$$

Assuming α and P_{lim} is fixed, let us denote the left side of this inequality as $f(p)$. Only one of the roots of f function is positive and between 0 and 1 (see Appendix A.8). This root is denoted as $p_{max,LLZ}$. If the p value of the system satisfies the condition that it is less than $p_{max,LLZ}$ and greater than p_{min}, $P_{R,extrema,1}$ is the optimal value for LLZ.

- $P_{R,bound} < P_{R,extrema,1}$ case

p values bigger than $p_{max,LLZ}$ makes $P_{R,extrema,1} > P_{R,bound}$, entailing a strictly decreasing $\overline{\Delta}_{LLZ}$ function throughout LLZ. Therefore, the optimal P_R value that minimizes the cost function becomes $P_{R,bound}$. One can see the visualization of the general relation between $P_{R,optimal}$ and p in Fig. 3.4.

To sum up:

- ▷ If $p \leq p_{min}$, then $P_{R,optimal} = 0$
- ▷ If $p_{min} < p \leq p_{max,LLZ}$, then $P_{R,optimal} = \frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}{1-\sqrt{\frac{p}{p+\alpha P_{lim}}}}$
- ▷ If $p_{max,LLZ} < p$, then $P_{R,optimal} = P_{R,bound} = \frac{p(1-P_{lim})}{p+P_{lim}}$

As a final note, it is observed that as P_{lim} approaches 1, $P_{R,bound}$ goes to 0 and $p_{max,LLZ}$ converges to p_{min}.

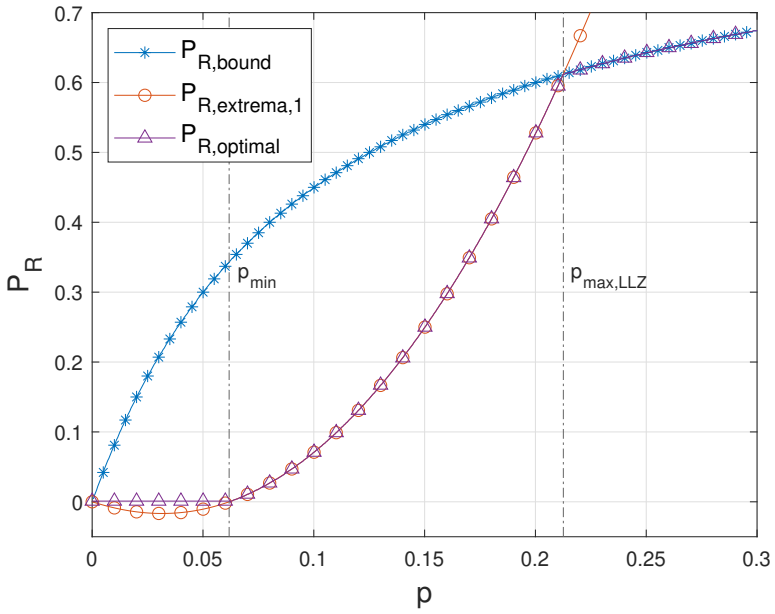


Figure 3.4: Optimal P_R vs p in LLZ ($P_{lim} = 0.1$ and $\alpha = 1$). In minimal p values where $p \leq p_{min}$, $P_{R,optimal}$ equals 0. In other words, the best option is not choosing any **Remote** action. However, if $p_{min} < p \leq p_{max,LLZ}$, the optimal P_R value follows the first extreme point we derived. Finally, when $p > p_{max,LLZ}$, the optimal remote action probability is equal to $P_{R,bound}$. This is the only case where $P_{R,optimal} + P_{L,optimal} = 1$, signifying that the user should not wait.

3.2.2 Total Limited Zone

Let us now focus on TLZ, where $P_R \in (P_{R,bound}, 1]$. Again, we aim to attain the $P_{R,optimal}$ and $P_{L,optimal}$ pair to minimize $\overline{\Delta}$. Throughout the zone, P_L is restricted by Ineq. (3.4); thus, the average cost equation in Eq. (3.9) becomes:

$$\overline{\Delta}_{TLZ}(P_R) = -1 + \frac{1}{p} + \frac{1}{\alpha + P_R(1-\alpha)} - \frac{1}{p + P_R}. \tag{3.16}$$

Akin to the LLZ, the partial derivative is used to explore the characteristics of the function over the TLZ, and the P_R domain is extended to $P_R \in \mathbb{R} \setminus \{\frac{-\alpha}{1-\alpha}, -p\}$. Accordingly,

$$\begin{aligned} \frac{\partial \overline{\Delta}_{TLZ}}{\partial P_R} &= \frac{1}{(p+P_R)^2} - (\frac{1}{1-\alpha})\frac{1}{(P_R + \frac{a}{1-\alpha})^2} \\ &= \frac{(1-\alpha)(P_R + \frac{\alpha}{1-\alpha})^2 - (p+P_R)^2}{(1-\alpha)(P_R + \frac{\alpha}{1-\alpha})^2(p+P_R)^2}. \end{aligned} \tag{3.17}$$

$\frac{\partial\overline{\Delta}_{TLZ}}{\partial P_R}$ exists $\forall P_R \in \mathbb{R} \setminus \{\frac{-\alpha}{1-\alpha}, -p\}$[3]. Notably, two P_R values make $\frac{\partial\overline{\Delta}_{TLZ}}{\partial P_R} = 0$, and denoted as $P_{R,extrema,3} = \frac{\alpha - p\sqrt{1-\alpha}}{\sqrt{1-\alpha}-(1-\alpha)}$ and $P_{R,extrema,4} = \frac{\alpha + p\sqrt{1-\alpha}}{-\sqrt{1-\alpha}-(1-\alpha)}$. The latter is always negative. Conversely, the former can be either positive or negative, and it can be bigger or smaller than $P_{R,extrema,4}$ depending on the parameters α and p. In this subsection, we identify four distinct intervals of interest based on these considerations.

- $P_{R,extrema,3} \leq P_{R,extrema,4} < 0$ case

As previously mentioned, $P_{R,extrema,4}$ is always negative. For $P_{R,extrema,4}$ to be greater than or equal to $P_{R,extrema,3}$, p value must exceed or equal to $\frac{\alpha}{1-\alpha}$ in which case $\frac{\partial\overline{\Delta}_{TLZ}}{\partial P_R}\Big|_{P_R=0} = (\frac{1}{p^2} - \frac{1-\alpha}{\alpha^2}) < 0$. Since both extrema are less than 0, it can be deduced that:

$$\frac{\partial\overline{\Delta}_{TLZ}}{\partial P_R} < 0, \quad \forall P_R > max(P_{R,extrema,3}, P_{R,extrema,4})$$

Consequently, if $p \geq \frac{\alpha}{1-\alpha}$, then $max(P_{R,extrema,3}, P_{R,extrema,4}) = P_{R,extrema,4} < 0$. Hence, $P_R \in [0,1] : P_R > max(P_{R,extrema,3}, P_{R,extrema,4})$ and $\frac{\partial\overline{\Delta}_{TLZ}}{\partial P_R} < 0$. In short, all p values higher than $p \geq \frac{\alpha}{1-\alpha}$ guarantees a strictly decreasing average cost function. Thus, the optimal P_R value of TLZ is the highest P_R value, i.e., $P_{R,optimal} = 1$.

- $P_{R,extrema,4} < P_{R,extrema,3} < 0$ case

Given the condition $p < \frac{\alpha}{1-\alpha}$, $P_{R,extrema,3}$ is always higher than $P_{R,extrema,4}$. Furthermore, as proved in Appendix A.9, $P_{R,extrema,3}$ is a local maximum when $p < \frac{\alpha}{1-\alpha}$. Hence, ensuring $P_{R,extrema,3} < 0$ makes the average cost function strictly decrease between 0 and 1. To determine the necessary condition for $P_{R,extrema,3} < 0$:

$$\begin{aligned} \frac{\alpha - p\sqrt{1-\alpha}}{\sqrt{1-\alpha}-(1-\alpha)} &< 0 \\ \alpha &< p\sqrt{1-\alpha} \\ p &> \frac{\alpha}{\sqrt{1-\alpha}} \end{aligned}$$

In essence, this case is obtained when $\frac{\alpha}{1-\alpha} > p > \frac{\alpha}{\sqrt{1-\alpha}}$. Analogously to the prior situation, the optimal remote action probability $P_{R,optimal}$ equals 1.

[3]When $\alpha = 1$, the value of $\frac{-\alpha}{1-\alpha}$ becomes undefined. Therefore, we use $\alpha = 1^-$ to define consistent success in the local task execution.

- $P_{R,extrema,4} < 0 \leq P_{R,extrema,3} \leq 1$ case

In this case, $P_{R,extrema,3}$, proved to be local maximum in Appendix A.9, lies within the interval $[0, 1]$. However, our objective is to identify the minimum average cost value. Since no other local extrema exists within $[0, 1]$, our focus should be on examining the function at the boundary points of TLZ. The required condition for $P_{R,bound} \neq 1$ to be optimal, along with its derivation, is elaborated on Ineq. (3.18). Moreover, the condition stipulating $P_{R,extrema,3} \leq 1$ is detailed in Ineq. (3.19).

$$
\begin{aligned}
\overline{\Delta}_{TLZ}(P_R = P_{R,bound}) &\leq \overline{\Delta}_{TLZ}(P_R = 1) \\
-1 + \frac{1}{p} + \frac{1}{\alpha + P_{R,bound}(1-\alpha)} - \frac{1}{p + P_{R,bound}} &\leq -1 + \frac{1}{p} + \frac{1}{\alpha + (1-\alpha)} - \frac{1}{p+1} \\
\frac{1}{\alpha + P_{R,bound}(1-\alpha)} - \frac{1}{p + P_{R,bound}} &\leq \frac{p}{p+1} \\
&\vdots \\
0 \leq (P_{R,bound} - 1)(P_{R,bound} - \frac{\alpha}{p(1-\alpha)} + p + 1) & \\
\frac{p(1-P_{lim})}{p + P_{lim}} &\leq \frac{\alpha}{p(1-\alpha)} - p - 1 \\
p^3 + 2p^2 + p(P_{lim} - \frac{\alpha}{1-\alpha}) - \frac{\alpha P_{lim}}{(1-\alpha)} &\leq 0
\end{aligned}
\tag{3.18}
$$

$$
\begin{aligned}
P_{R,extrema,3} &\leq 1 \\
\frac{\alpha - p\sqrt{1-\alpha}}{\sqrt{1-\alpha} - (1-\alpha)} &\leq 1 \\
\frac{1 - \sqrt{1-\alpha}}{\sqrt{1-\alpha}} &\leq p
\end{aligned}
\tag{3.19}
$$

Utilizing Descartes' Sign Rule, we ascertain only a single root of the equation on the left side in Ineq. (3.18) is positive. This root is denoted as $p_{max,TLZ}$. Briefly, if $p \leq p_{max,TLZ}$, then $\overline{\Delta}_{TLZ}(P_R = P_{R,bound}) \leq \overline{\Delta}_{TLZ}(P_R = 1)$, leading to $P_{R,optimal} = P_{R,bound}$. Conversely, if $p > p_{max,TLZ}$, then $\overline{\Delta}_{TLZ}(P_R = P_{R,bound}) > \overline{\Delta}_{TLZ}(P_R = 1)$, resulting in $P_{R,optimal} = 1$. It must be noted that in order for $P_{R,extrema,3} \in [0, 1]$, the remote latency parameter p must be in $[\frac{1-\sqrt{1-\alpha}}{\sqrt{1-\alpha}}, \frac{\alpha}{\sqrt{1-\alpha}}]$ interval.

- $P_{R,extrema,4} < 0 < 1 < P_{R,extrema,3}$ case

In this situation, where $p < \frac{1-\sqrt{1-\alpha}}{\sqrt{1-\alpha}}$, the local maximum exceeds 1. Therefore, the average cost function strictly increases along $P_R \in [0,1]$, and the optimal value is the minimum feasible P_R value, which is $P_{R,bound}$. One can see the overall picture of TLZ in Fig. 3.5.

To sum up, $p_{max,TLZ}$ is the pivotal threshold distinguishing between $P_{R,optimal} = P_{R,bound}$ and $P_{R,optimal} = 1$ outcomes. Remarkably, this threshold can be bigger than 1. To illustrate, if $\alpha = 1$, then based on Eq. (3.16), $\overline{\Delta}_{TLZ} = \frac{1}{p} - \frac{1}{p+P_R}$. Under this condition, $P_{R,optimal}$ always equals $P_{R,bound}$. It can be inferred that $p_{max,TLZ}$ goes to infinity as α approaches 1. Conversely, as α approaches 0, $p_{max,TLZ}$ converges to 0.

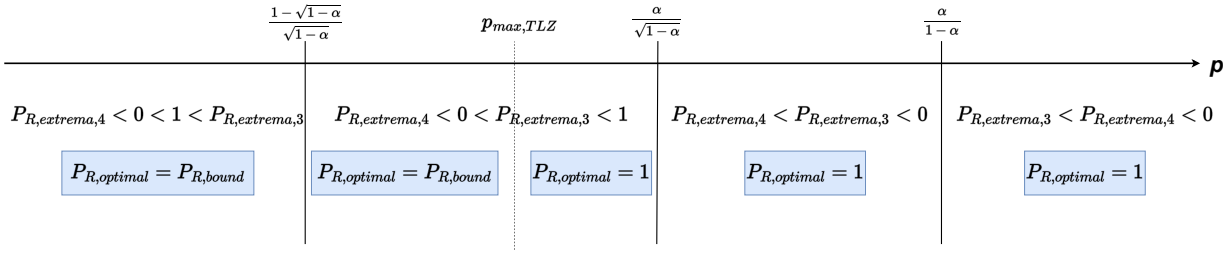


Figure 3.5: Overall picture of TLZ

3.2.3 Overall System and Results

Thus far, we have delved into two distinct P_R zones. At this point, it is important to merge these zones and obtain a general overview. Several possible scenarios exist, but we introduce a conjecture before detailing these scenarios.

Conjecture 1. Let $\alpha \in [0,1)$ be the local success probability and $P_{lim} \in [0,1]$ be the local action limitation rate. Under this condition, $p_{max,TLZ} > p_{max,LLZ}$.

Although there is no mathematical proof, we tested the conjecture for numerous parameters and are confident that it holds. In light of this information, the situation depicted in Fig. 3.6 is attained.

The overall system encompasses four partitions, or regions, each following a distinct optimal remote probability ($P_{R,optimal}$) function. A system setup with known remote latency parameter p, local success probability α, and local action limitation rate P_{lim} parameters falls under one of these four regions, and the optimal P_R value that the user should use is derived accordingly. Fig. 3.6 is based on altering the parameter p; however, all boundary points, namely p_{min}, $p_{max,LLZ}$ and $p_{max,TLZ}$, depend on α and P_{lim}. A similar figure can be obtained for changes of α and P_{lim}. The crucial aspect is identifying which system parameters are associated with which region.

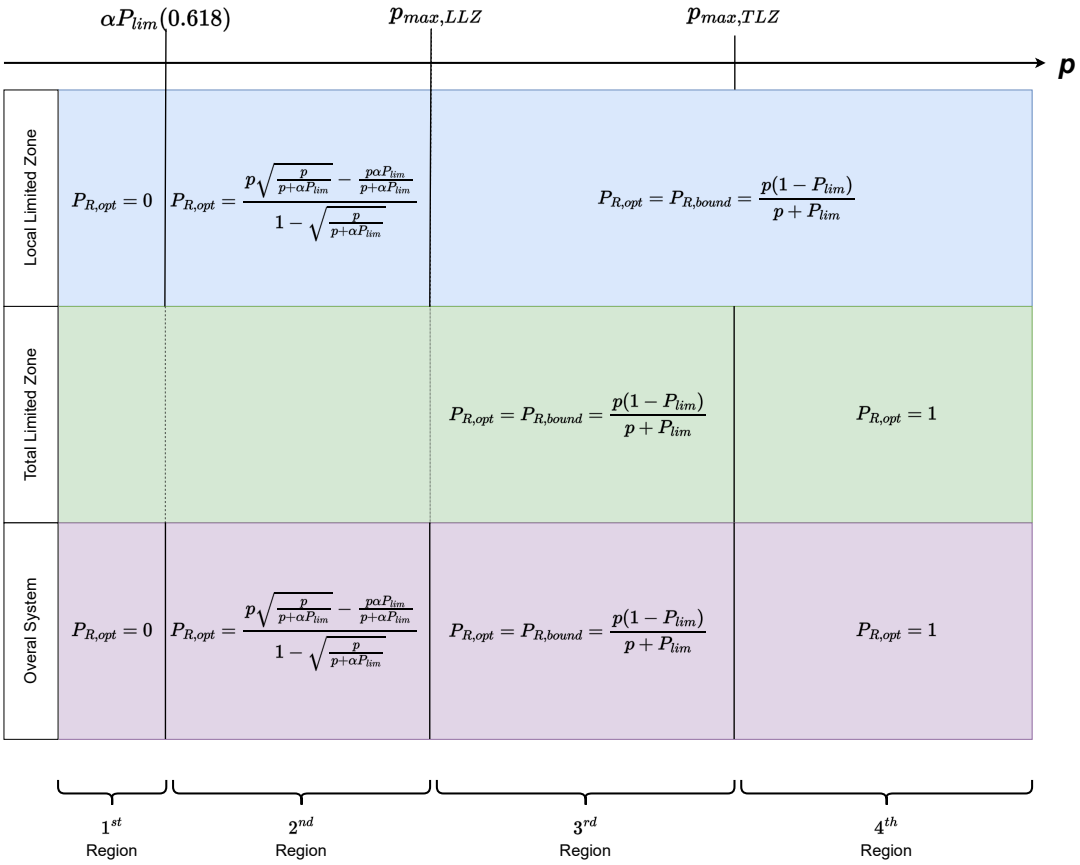
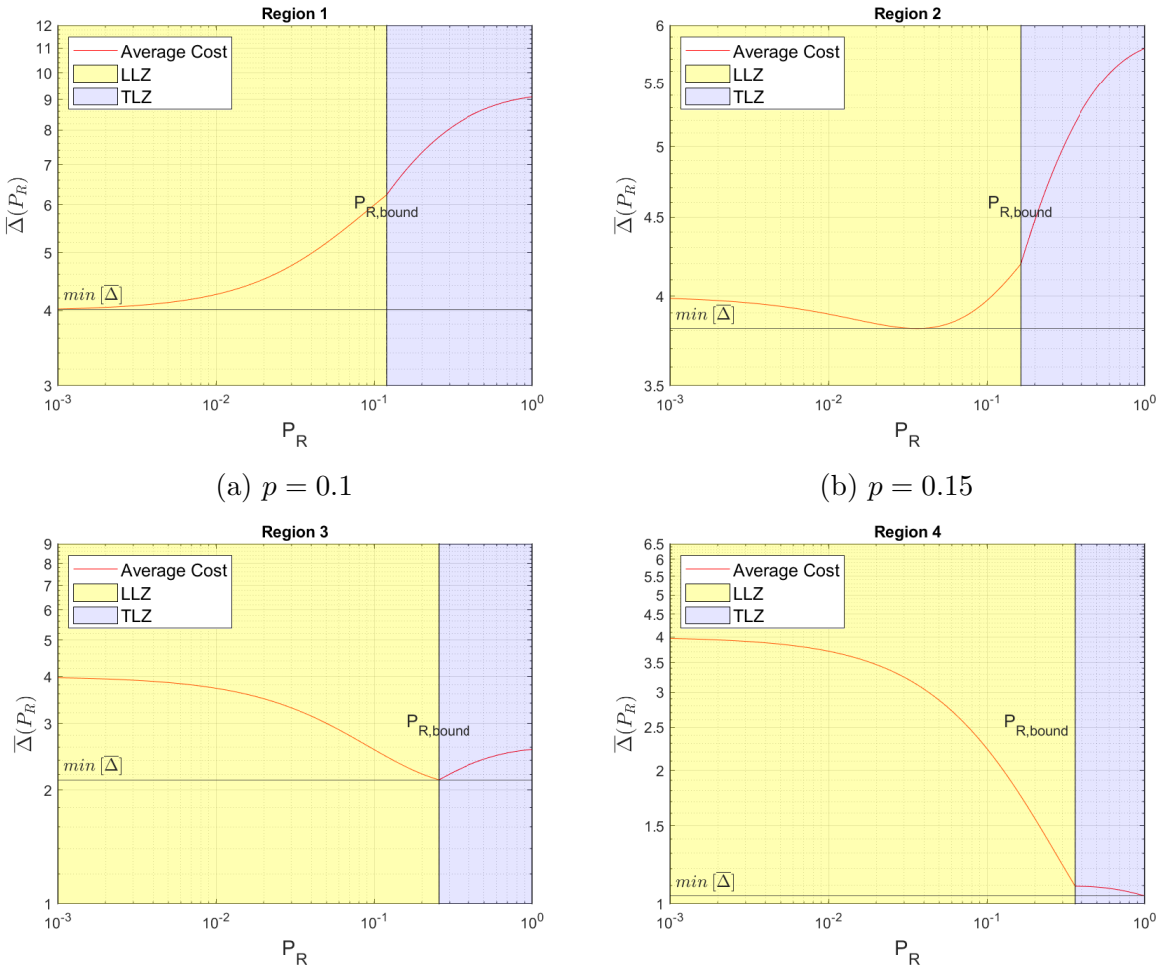


Figure 3.6: Overall blind decision-making system

The first region covers the systems characterized by parameters satisfying Ineq. (3.14), indicating where $p \leq p_{min}$. The attributes of this region are depicted in Fig. 3.7a. Within this domain, the optimal values for LLZ and TLZ are 0 and $P_{R,bound}$, respectively. As $P_{R,bound}$ is also included in LLZ, we can deduce that the overall $P_{R,optimal} = 0$. This region signifies an extremely small p value, leading to relatively extensive remote latencies after employing a **Remote** action. Instead, the wisest move is to utilize local resources to perform tasks and wait for the remaining time to adhere to local limitation constraints.

The second region includes the systems where the parameters do not satisfy Ineq. (3.14), but satisfy Ineq. (3.15). In other words, $p_{min} < p \leq p_{max,LLZ}$. Similar to the first region, the optimal value of TLZ, $P_{R,bound}$, is covered by LLZ. Thus, the optimal value of LLZ, $P_{R,extrema,1}$, becomes dominant and overall $P_{R,optimal}$ equals $P_{R,extrema,1}$. An example of average cost versus P_R graph is shown in Fig. 3.7b. One can interpret that the models with relatively higher remote latencies than Region 1 belong to this region. The p value is not minimal; hence, utilizing a **Remote** action can prove advantageous. Nonetheless, the optimal P_R still resides within LLZ, emphasizing that occasionally waiting is necessary to minimize the cost, thereby improving freshness.



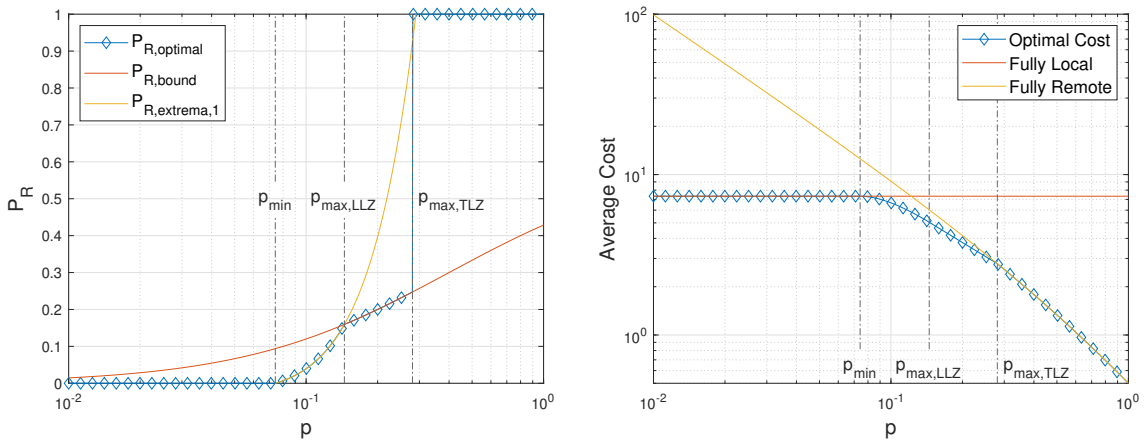
(a) $p = 0.1$ (b) $p = 0.15$

(c) $p = 0.3$ (d) $p = 0.6$

Figure 3.7: Average cost versus P_R graphs ($P_{lim} = 0.4$ and $\alpha = 0.5$)

The third region stands for the only one in which both LLZ and TLZ share the same $P_{R,optimal}$, equivalent to $P_{R,bound}$, as illustrated in Fig. 3.7c. Systems with remote latency parameter p values between $p_{max,LLZ}$ and $p_{max,TLZ}$ belong to this region. In this context, the p value is sufficiently high to consistently use **Remote** actions instead of waiting but not high enough to fully utilize the remote servers. Consequently, the wait action probability $P_W = 1 - P_R - P_L$ equals 0.

The fourth region encompasses the frameworks with better remote task completion capabilities than the local ones; thus, the user always employs remote servers without waiting or taking any **Local** action. An illustration of this region, including the zones, is seen in Fig. 3.7d. Within this framework, the optimal P_R for LLZ is $P_{R,bound}$. However, the average cost is smaller at $P_R = 1$ in TLZ, thereby $P_{R,optimal} = 1$.



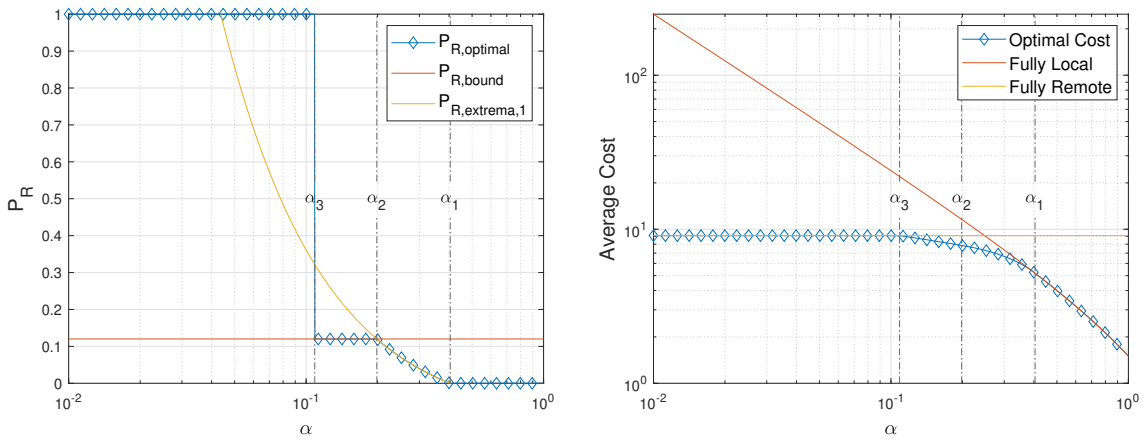
(a) P_R versus p (b) $\overline{\Delta}$ versus p

Figure 3.8: $P_{R,optimal}$ and average cost versus p graphs ($P_{lim} = 0.4$ and $\alpha = 0.3$)

Fig. 3.8a portrays the $P_{R,optimal}$ versus p graph when α and P_{lim} are fixed. As seen, the optimal remote action probability is 0 for extremely small p values (Region 1). Beyond a certain threshold value, p_{min}, the framework transitions into Region 2, and $P_{R,optimal}$ begins to follow $P_{R,extrema,1}$ until it hits $p_{max,LLZ}$. Before reaching this threshold, waiting was a part of the optimal strategy. However, when $p > p_{max,LLZ}$, this strategy is invalid, and P_W equals 0. In the third region, $P_{R,optimal} = P_{R,bound}$. Subsequently, increasing the p value leads the system to Region 4, where $P_{R,optimal} = 1$. Notably, there is a jump from $P_{R,bound}$ to 1, and it is impossible to obtain $P_{R,bound} < P_{R,optimal} < 1$ in a single-user case.

The graph presented in Fig. 3.8b highlights the corresponding cost value when the user uses $P_R = P_{R,optimal}$, explained via Fig. 3.8a. The yellow line signifies the fully remote case, in which task completion is performed exclusively by a remote server, i.e., $P_R = 1$. In contrast, the red line stands for using solely local resources as long as the local action limitation rate allows, that is, $P_L = P_{lim}$ and $P_R = 0$. The cost value for the entirely local case does not change over p, as enhancing or deteriorating remote latencies does not affect the user if they do not use **Remote** action. Conversely, average cost strictly decreases in the fully remote case. Observe that the optimal cost obtained by following $P_R = P_{R,optimal}$ is always less than or equal to the ones associated with entirely local or remote cases.

It has been observed that $p_{max,TLZ}$ can exceed 1, positioning the system in the third region even when $p = 1$, signifying that the remote server always performs the task successfully within a time slot. In such cases, the optimal strategy does not entail exploiting remote resources whenever the user is idle. If α and P_{lim} are high enough, using **Local** actions reduces the average cost function. Notably, the left side of Ineq. (3.15) exceeds 0, $\forall \alpha, P_{lim} \in [0, 1]$ and $p = 1$, which means the optimal P_R is in TLZ and $P_W = 0$. Additionally, as P_{lim} approaches to 1, $p_{max,LLZ}$ converges to $\alpha P_{lim}(0.618)$ and $P_{R,optimal}$ converges to 0 in the first three region.



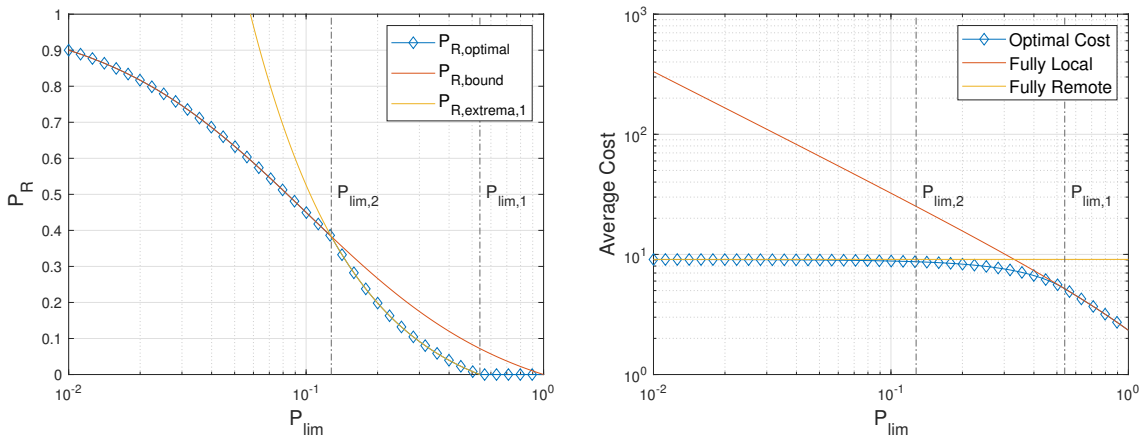
(a) P_R versus α (b) $\overline{\Delta}$ versus α

Figure 3.9: $P_{R,optimal}$ and average cost versus α graphs ($p = 0.1$ and $P_{lim} = 0.4$)

Thus far, we have conducted analyses to optimize user actions and derived the inequalities based on varying p values. Intuitively, if expected remote latency diminishes, i.e., p grows, an increase in **Remote** action use is expected. Alternatively, given p and P_{lim} are constant, an increase in α makes the **Local** actions more "Reliable" and tends to shift the system towards Region 1.

As depicted in Fig. 3.9a, the systems with extremely small α values are associated with Region 4, i.e., $P_{R,optimal} = 1$. Beyond a threshold value, let us denote α_3 for simplicity, $P_{R,optimal}$ drops to $P_{R,bound}$—a constant value throughout α. The specific value of α_3 depends on the parameters p and P_{lim} and can be calculated from Ineq. (3.18). Amplifying α eventually puts the system into Region 2 and then Region 1. The boundary α values where the transition from Region 3 to Region 2 and Region 2 to Region 1 happen are denoted as α_2 and α_1. The specific values of α_2 and α_1 can be calculated from the Ineq. (3.15) and Ineq. (3.14), respectively.

Another thing to be examined is the obtained average cost function values and their comparison with fully local and fully remote cases, detailed in Fig. 3.9b. The yellow and red lines represent fully remote and local scenarios, respectively. In contrast, the blue line marked with diamond shapes is the attained optimal cost via the optimal P_R values. Altering α values does not influence the remote case as they are independent; however, it is inversely correlated with the average cost in the case of employing only local resources. Thus, the user should prefer utilizing solely remote servers if α is very small and local resources if α is high.



(a) P_R versus P_{lim} (b) $\overline{\Delta}$ versus P_{lim}

Figure 3.10: $P_{R,optimal}$ and $\overline{\Delta}$ versus P_{lim} graphs ($\alpha = 0.3$ and $p = 0.1$)

Lastly, the evaluation of optimal P_R for different P_{lim} values is depicted in Fig. 3.10a. Although not apparent in the graph, excessively small P_{lim} values, where the local limitation is too high, make $P_{R,optimal} = 1$. Similar to the α case, an increase in P_{lim} decreases $P_{R,optimal}$. However, in this instance, the reason is not reliability but rather limitations.

In this section, we aimed to obtain the optimal trio of P_R, P_L, and P_W, diminishing the average cost function. Evaluating the outage, or age violation, probability [42,55–57], that is, the probability of exceeding a specific value, is another point of view. For that purpose, we conducted a Monte Carlo simulation experiment and attained empirical CDFs of the observed cost function values for different P_R values, as seen in Fig. 3.11. This figure is an example from a system in Region 2, i.e., $P_{R,optimal} = P_{R,extrema,1}$. One can observe that the optimal P_R minimizing the average cost **does not** necessarily minimize the outage probability for some threshold values. For example, the probability of the cost function being higher than 2 is higher at $P_{R,optimal}$ compared to $P_{R,bound}$. Therefore, an in-depth analysis is required if outage probability is a critical indicator for the system.

3.3 Multiple Users

In the previous section, we assumed a solitary user was operating within the model. However, real-world situations often deviate from this simplification. In practice, multiple users may request task executions from the same remote units, creating an excessive load. Consequently, this can lead to additional delays in getting a response and, by extension, increase the staleness. The main objective of this chapter is to integrate the multiple-user case into our model and discover the most favorable remote action probability P_R.

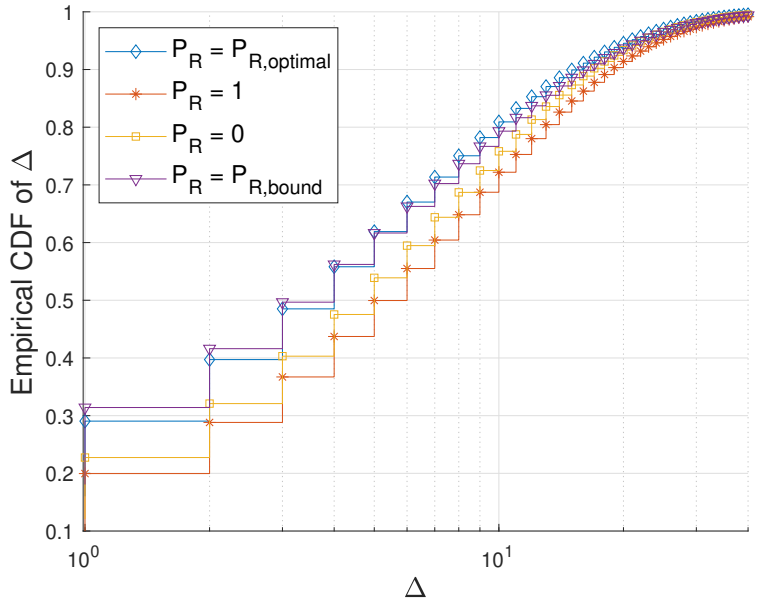


Figure 3.11: Empirical CDF of the cost function

Definition 9 (Number of Users, (N))**.** The total number of users connected to the same remote units is denoted as $N \in \mathbb{N}, N \geq 1$. When multiple users are present, they share not only the first contacted remote server but also the subsequent servers behind this task execution process, along with the same channel. Whenever more than 1 users request task execution from the remote units, the available resources are allocated among them.

Assumption 4. All users exhibit identical action probabilities, that is, P_R, P_L, and P_W are the same for each user. The primary aim is to optimize performances across all users.

From a heuristic standpoint, designating high P_R signifies handling multiple users simultaneously and sharing resources among them. Thus, the experienced latency increases, forcing the remote latency parameter (p) value to diminish. Ultimately, the p parameter is inversely correlated to P_R and n.

Assumption 5. The resources, such as bandwidth and processing units, are shared **equally** among users connected to the remote units. Therefore, the remote latency parameter equation is:

$$\frac{1}{p} = T_{base}\tilde{N} \tag{3.20}$$

where $T_{base} \in [1, \infty)$ is the expected remote latency for the single user case and $\tilde{N}$ is the number of users allocated with resources and awaiting a response from the remote server.

Deriving the exact number of users awaiting a response from the remote server is complex and challenging. Accordingly, we approximate $\tilde{N}$ from a user's perspective as follows:

$$\mathbb{E}[\tilde{N}_{user}] = 1 + (N-1) \cdot \pi_B \tag{3.21}$$

This equation represents how many users are in the *Busy* availability state from one of the user's point of view. Within the equation, the "1" term signifies the user connecting to the remote server, while each of the remaining $N-1$ users is in the *Busy* availability state, i.e., awaiting a response from the remote server, with a probability of π_B. The $(N-1)\pi_B$ term shows the expected number of busy users except the one that already took a **Remote** action. Applying the formula of π_B determined in Eq. (3.1), the remote latency parameter equation becomes Eq. (3.22).

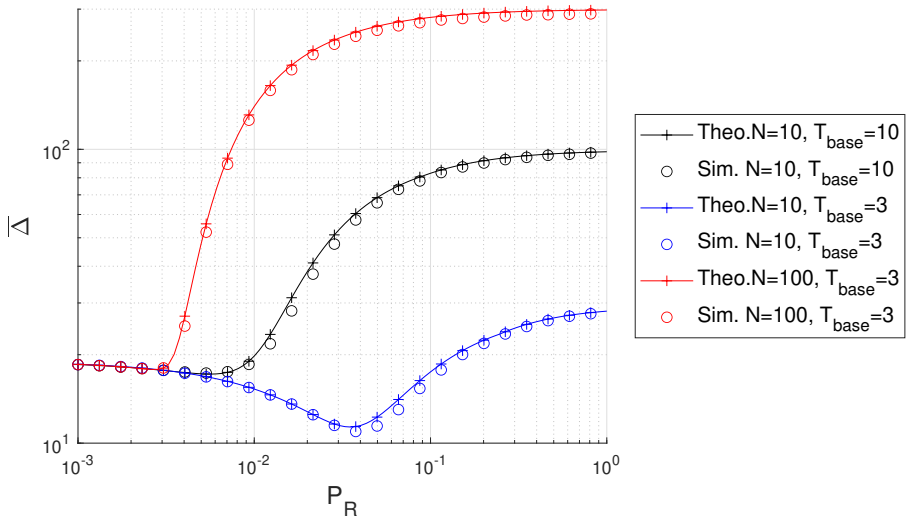
$$\begin{gathered}
\frac{1}{p} = T_{base}[1 + (N-1) \cdot \frac{P_R}{p + P_R}] \\
\frac{1}{p} = T_{base}[\frac{p + NP_R}{p + P_R}] \\
p^2 T_{base} + p(NP_R T_{base} - 1) - P_R = 0 \\
\vdots \\
p = \frac{\frac{1}{T_{base}} - NP_R + \sqrt{(\frac{1}{T_{base}} - NP_R)^2 + 4\frac{P_R}{T_{base}}}}{2}
\end{gathered} \tag{3.22}$$

As seen from Eq. (3.22), the p value depends on T_{base}, N and P_R. Unlike the single-user case, we used T_{base} and N parameters to represent the remote task execution characteristics of the system instead of employing the p parameter. Thus, we have four distinct parameters defining a framework in total.

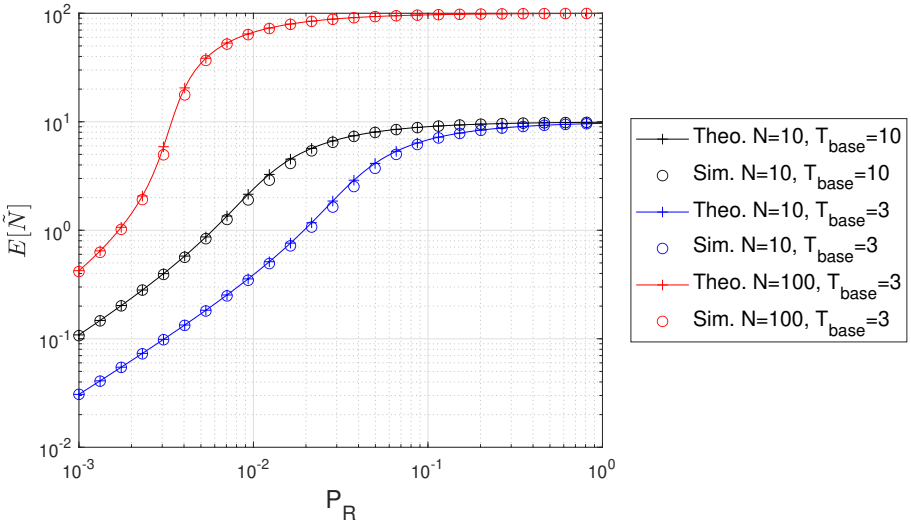
We first evaluate the accuracy of our approximations. An issue that arises is that calculations rely on steady-state analysis. Therefore, the experiment's time horizon, T_{max}, must be selected as high as feasible, leading us to designate it as 10.000. Throughout simulations, we monitor the number of users connected to remote units at each time slot and determine p according to Eq. (3.20). We execute each setup 500 times using MATLAB and obtain the empirical average of the average cost and $\tilde{N}$ values.

From a theoretical standpoint, it is important to note that Eq. (3.21) shows the expected number of users connected to the remote unit and allocated with resources from a user's perspective. The user assumes its availability state is always *Busy*, beneficial while calculating the experienced remote latency parameter. From the overall perspective, i.e., remote server's perspective, the expected value of $\tilde{N}$ is calculated as:

$$\mathbb{E}[\tilde{N}_{remote}] = N \cdot \pi_B. \tag{3.23}$$



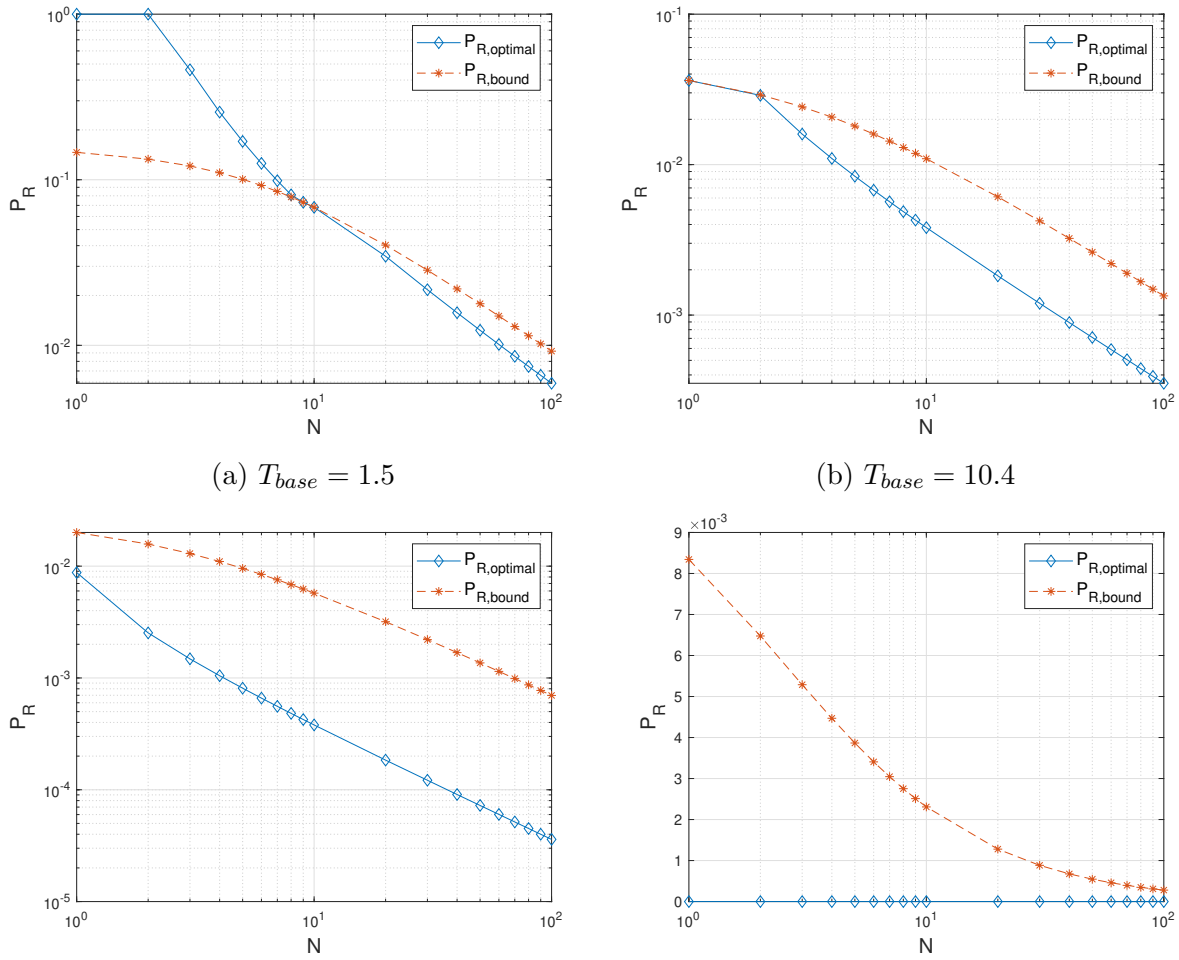
(a) $\overline{\Delta}$ vs P_R



(b) $\tilde{N}$ vs P_R

Figure 3.12: $\overline{\Delta}$ and $\mathbb{E}[\tilde{N}]$ versus P_R graphs ($\alpha = 0.5$, $P_{lim} = 0.1$)

The theoretical p value is approximated from Eq. (3.22), and the theoretical $\mathbb{E}[\tilde{N}]$ and $\overline{\Delta}$ are derived from Eq. (3.23) and Eq. (3.9), respectively. Fig. 3.12 proves that the theoretical $\mathbb{E}[\tilde{N}]$ and $\overline{\Delta}$ are closely aligned with the simulation results. Considering minimizing the average cost function, the optimal P_R values are very close in empirical and theoretical frameworks, and the difference is negligible.



(a) $T_{base} = 1.5$

(b) $T_{base} = 10.4$

(c) $T_{base} = 20$

(d) $T_{base} = 20$

Figure 3.13: $P_{R,optimal}$ versus N graphs ($\alpha = 0.1$ and $P_{lim} = 0.7$)

To examine the impact of the total number of users (N) within the system, one can refer to Fig. 3.13. The closed-form derivations pertaining to $P_{R,optimal}$ in multi-user scenarios are elaborate. Hence, we applied a brute-force method to determine the optimal values, evaluating the average cost function (Eq. 3.9) for each P_R value and finding the one minimizing the cost. In addition, the corresponding p value for the P_R minimizing the cost is used to derive $P_{R,bound}$ based on Eq. (3.10). Both $P_{R,optimal}$ and $P_{R,bound}$ values are depicted in the figure.

Fig 3.13a shows the $P_{R,optimal}$ for a various number of users at an extremely small T_{base} value, in which p parameter is very high when $N = 1$; thereby positioning the system within the 4^{th} region. It states that using **Remote** actions is the wisest move without waiting or utilizing local resources. As N increases, there is a decrease in optimal P_R after a threshold value. Remarkably, an additional region becomes apparent in the multi-user paradigm. For example, when $N = 3$, $P_{R,optimal}$ is between $P_{R,bound}$ and 1, impossible in

single-user case. Let us denominate this region as "Region 3.5" as it is between Region 3, in which $P_{R,optimal} = P_{R,bound}$, and Region 4, where $P_{R,optimal} = 1$. On the other hand, with further increments in the N parameter, the system transitions to Region 3 and, consequently, Region 2. In this scenario, $P_{R,optimal}$ value in Region 2 no longer equals the extreme point as in Eq. (3.13). However, it is still smaller than $P_{R,bound}$ and larger than 0. Another key point is that increasing the N value does not carry the system to Region 1. If the granularity of P_R values is high enough, the optimal P_R value is above 0, resulting in rare server activity with tolerable p value.

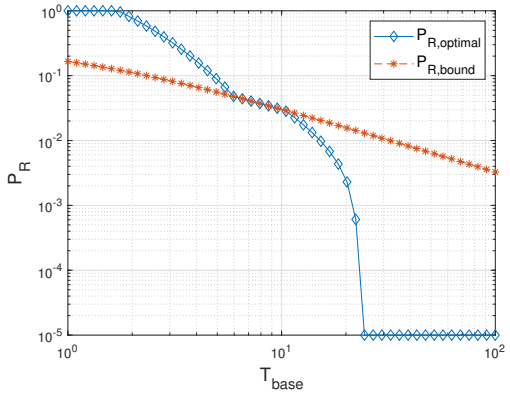
All previous points are valid for the frameworks in 4^{th} region in the single-user paradigm. Given $T_{base} = 10.4$, for instance, the one-user system stays in the third region (see Fig. 3.13b). Akin to the Fig. 3.13a, increasing N decreases p and shifts the system to Region 2. Additionally, if the single-user paradigm is in Region 2 (refer Fig. 3.13c), it stays there while N increases. The optimal P_R value decreases but does not reach 0. Technically, even a minimal remote action probability can benefit all users. The only possible case where $P_{R,optimal} = 0$ is when the system is in Region 1 within a single-user paradigm. In this scenario, employing solely local sources is optimal, even when only one user exists. Increasing the number of users worsens the remote latencies; therefore, the optimal strategy remains to avoid using any **Remote** action. This scenario is depicted in Fig. 3.13d. It can be concluded that the T_{base} parameter is vital to position the single-user system, thereby influencing the behavior in case of a variant number of users.

From a different perspective, given a fixed number of users, the results of the increment on T_{base} is portrayed in Fig. 3.14. Similar to Fig. 3.13, the behavior depends on the region corresponding to the smallest T_{base} value. In Fig. 3.14a, the system is in Region 4 at $T_{base} = 1$: however, the system shifts towards the first region and $P_{R,optimal}$ decreases as T_{base} increases. It is pertinent to underscore that $P_{R,optimal}$ drops to zero[4] at very high T_{base} values unlike the ones in Fig. 3.13. The reason is that T_{base} directly influences the single-user scenario, unlike the N parameter.

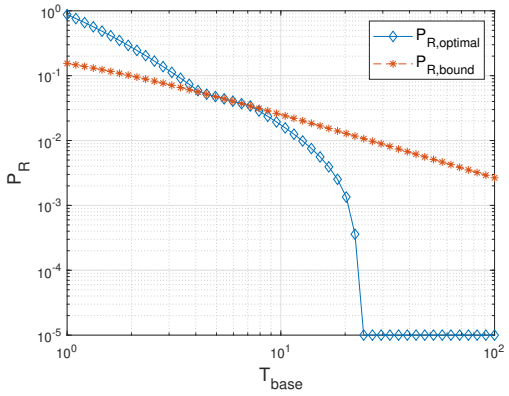
Other figures (Fig. 3.14b, 3.14c, and 3.14d) illustrate P_R versus T_{base} for different N values, placing the systems into Region 3.5, Region 3, and Region 2 when $T_{base} = 1$, respectively. They all manifest analogous characteristics, such as shifting towards the first region and reaching it after one point.

Ultimately, Fig. 3.15 illustrates the average cost function when the derived $P_{R,optimal}$ is selected as the remote action probability, along with fully local and remote cases. As seen, the average cost in the fully local case is independent of the N and T_{base} parameters since these parameters solely affect the remote characteristics of the system. However, the average cost value in the fully remote case increases as N or T_{base} increases. Notably, the average cost function with the optimized action probabilities persistently remains either smaller or equivalent to the average cost when employing a single resource type.

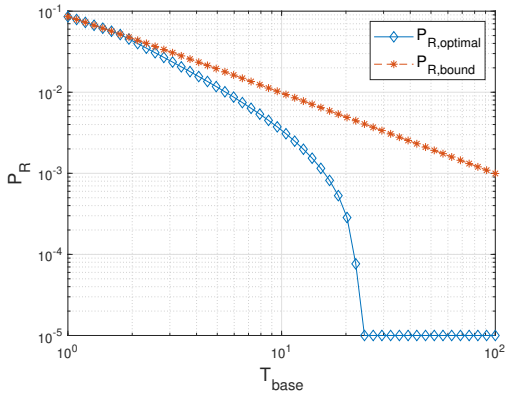
[4]The point dropped is seen as 10^{-5} due to plotting the y-axis on a log scale.



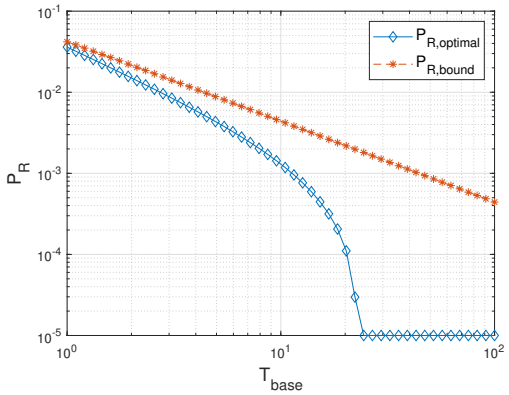
(a) $N = 2$



(b) $N = 3$

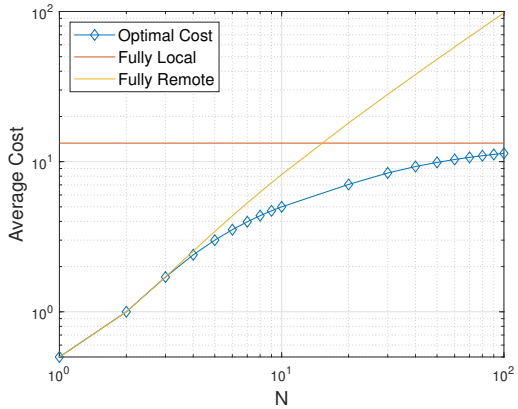


(c) $N = 12$

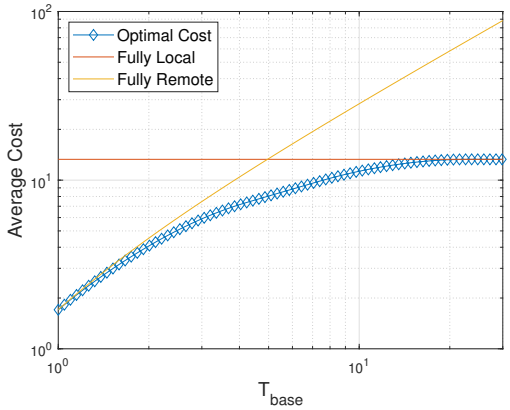


(d) $N = 30$

Figure 3.14: $P_{R,optimal}$ versus T_{base} graphs ($\alpha = 0.1$ and $P_{lim} = 0.7$)



(a) $\overline{\Delta}$ versus N ($T_{base} = 1$)



(b) $\overline{\Delta}$ versus T_{base} ($N = 3$)

Figure 3.15: $\overline{\Delta}$ versus N and T_{base} graphs ($\alpha = 0.1$ and $P_{lim} = 0.7$)

Chapter 4

Informed Decision

In Chapter 3, the concept of the blind decision setting, wherein the user decides on offloading strategies regardless of the current cost function, was explained. However, utilizing the information on the staleness, characterized by the cost function of the model, can help the user improve freshness. This chapter focuses on formulating the problem as a Markov Decision Process [58] and ascertaining the optimal strategy by employing dynamic programming over a finite horizon.

We will proceed by building upon what has been presented in Chapter 2. In contrast to what discussed in Section 3.1, the user's action decisions depend on the present information on the staleness and current time. It is worth noting that while remote latency parameter p and local success probability α values still incorporate an element of randomness, choosing the optimal action is no longer based on chance.

4.1 Dynamic Programming

Dynamic programming (DP) is a mathematical optimization algorithm that decomposes complex problems into smaller, more manageable subproblems and addresses them recursively. Prior to utilizing the DP method, the problem shall be restated as a Markov Decision Process (MDP). The MDP model with a finite horizon T is denoted by a 4-tuple $\{X, U, S, G\}$, where X indicates a set of possible states, U is a set of possible actions, S represents the randomness factors in state transitions, and G is a general cost function. We structure our analysis into two distinct segments, distinguishing between scenarios with and without restrictions on **Local** actions. All components of the MDP, apart from X, remain identical in both situations with or without local limitations. We first elucidate the unconstrained version in subsection 4.1.1, then clarify the differences between the two scenarios and propose a layered state space solution to resolve the problem in subsection 4.1.2.

4.1.1 Unconstrained Local Action Analysis

Definition 10 (Unconstrained State Space, (X_U))**.** X_U denotes the set of all possible states without constraint on **Local** actions. A state of a user at the onset of time slot k is represented by $x_k = (\Delta_k, \varsigma_k), x_k \in X_U$, where Δ_k and ς_k are the cost function value and availability of the user at k^{th} time slot, respectively.

Definition 11 (Action Space, (U))**.** If *Idle*, a user makes an offloading decision at the onset of each time slot, denoted by a vector $u_k = [r, l, w]$, where k is the time slot index. All possible actions, $r, l, w \in \{0, 1\}$, constitute the action space. Notably, the notation comes from the first letter of the actions **Remote**, **Local**, and **Wait**, and $r_k + l_k + w_k = 1$. After a **Remote** action is employed, the availability state of the user reverts to *Busy* in the next time slot. In mathematical terms, $\varsigma_{k+1} = 1$ if $r_k = 1$. When the user is *Busy*, the **Wait** action is selected automatically, i.e., $u_k = [r_k, l_k, w_k] = [0, 0, 1]$ if $\varsigma_k = 1$. Notably, the **Wait** action is the only option in *Busy* availability state, and the state of the next time slot is subject to randomness.

Definition 12 (Random Variables, (S))**.** The dynamic part of the system, termed $s_k \in S$, $\forall k = 1, ..., T$, encompasses two random variables, namely $s_k^L, s_k^R \in \{0, 1\}$. The **Local** action's success and the **Remote** action's latency are both subject to randomness, and the probabilities depend on α and p, respectively. Within the model, $s_k^L = 1$ signifies a **Local** action is taken at k^{th} time slot, and it is successful, whereas $s_k^R = 1$ means the remote serves executes the task with success while the user is waiting for a response from the remote server. The conditional probabilities are given in Eq. (4.1) and (4.2).

$$P(s_k^L = 1 | l_k) = \begin{cases} \alpha & \text{if } l_k = 1 \\ 0 & \text{if } l_k = 0 \end{cases} \tag{4.1}$$

$$P(s_k^R = 1 | \varsigma_k) = \begin{cases} p & \text{if } \varsigma_k = 1 \\ 0 & \text{if } \varsigma_k = 0 \end{cases} \tag{4.2}$$

Definition 13 (State transition function, (f))**.** The system state at time $k + 1$ depends on the previous state, the action taken, and the disturbance, or stochastic, factor. The transition function maps these factor to the subsequent state as:

$$x_{k+1} = f_k(x_k, u_k, s_k) = (\Delta_{k+1}, \varsigma_{k+1})$$
$$x_{k+1} = \begin{cases} (0, 0) & \text{if } (l_k = 1 \text{ and } s_k^L = 1) \text{ or } (\varsigma_k = 1 \text{ and } s_k^R = 1) \\ (\Delta_k + 1, 0) & \text{if } (l_k = 1 \text{ and } s_k^L = 0) \text{ or } (\varsigma_k = 0 \text{ and } w_k = 1) \\ (\Delta_k + 1, 1) & \text{if } r_k = 1 \text{ or } (\varsigma_k = 1 \text{ and } s_k^R = 0). \end{cases} \tag{4.3}$$

The initial point, denoted as x_0, is set to be equal to $(0, 0)$. As we traverse through time, the cost value at time slot k always remains smaller than or equal to k in our setting. This knowledge has been leveraged to streamline complexity, obtaining a transition graph as seen in Fig. 4.1. In the figure, the superscript of nodes denotes the cost value, while the subscript denotes the availability state. Here, the time index progresses towards the right side of the figure, and all possible states of a time slot are lined up vertically.

When the user is *Idle* with a cost function value Δ, it can employ (i) the **Remote** action, depicted with blue lines, and transition to the *Busy* availability state with a cost function value $\Delta+1$ in the next time slot, (ii) the **Local** action, resulting in successful task execution and cost reset by the end of the time slot with a probability of α (illustrated with dashed red lines), or unsuccessful task execution with a probability of $1-\alpha$ (illustrated with solid red lines), or (iii) the **Wait** action, shown with cyan lines, increasing the cost function by one without altering the availability state.

Conversely, the user must wait when in *Busy* availability state. While waiting, the remote units may complete the task, providing a response to the user by the end of the time slot with a probability p (illustrated with dashed green lines), after which the user resets the cost function and becomes *Idle*. Or, the remote units may not execute the task in that time slot (illustrated with solid green lines) with a probability of $1-p$, and the user continues to be *Busy*. Cost resets are expressed with dashed lines, while the solid lines indicate an increase in the cost function.

Definition 14 (General Cost Function Space, (G))**.** The cost function defined in Chapter 2 is utilized to represent staleness. The cost incurred at the k^{th} time slot is denoted by $g_k(x_k, u_k, s_k) \in G$, equivalent to Δ_k. Similarly, the terminal cost is $g_T(x_T)$ and equal to Δ_T.

Definition 15 (Total Cost Function, (J))**.** The total cost, which we want to minimize, is represented by $J_\pi(x_0)$ where x_0 stands for the system's initial state and $\pi = \{u_0, ..., u_{T-1}\}$ is the *policy*, describing the sequence of chosen actions. Considering the presence of a disturbance, the total cost can be formulated as follows:

$$J_\pi(x_0) = \mathop{\mathbb{E}}_{s_k}\left\{ g_T(x_T) + \sum_{k=1}^{T-1} g_k(x_k, u_k, s_k) \right\}. \tag{4.4}$$

The objective of Chapter 4 is to optimize over closed-loop policies, wherein the task involves selecting u_k for each k and potential x_k and finding the optimal policy (π^*). In pursuit of that, the cost-to-go function $J_k(x_k)$ is defined to measure the **minimum** total cost accumulated from time slot k to the terminal slot T when the state of the system at k^{th} time slot is x_k as in [59]:

$$J_k(x_k) = \min_{u_k} \mathop{\mathbb{E}}_{s_k}\left\{ g_k(x_k, u_k, s_k) + J_{k+1}(f_k(x_k, u_k.s_k)) \right\} \tag{4.5}$$

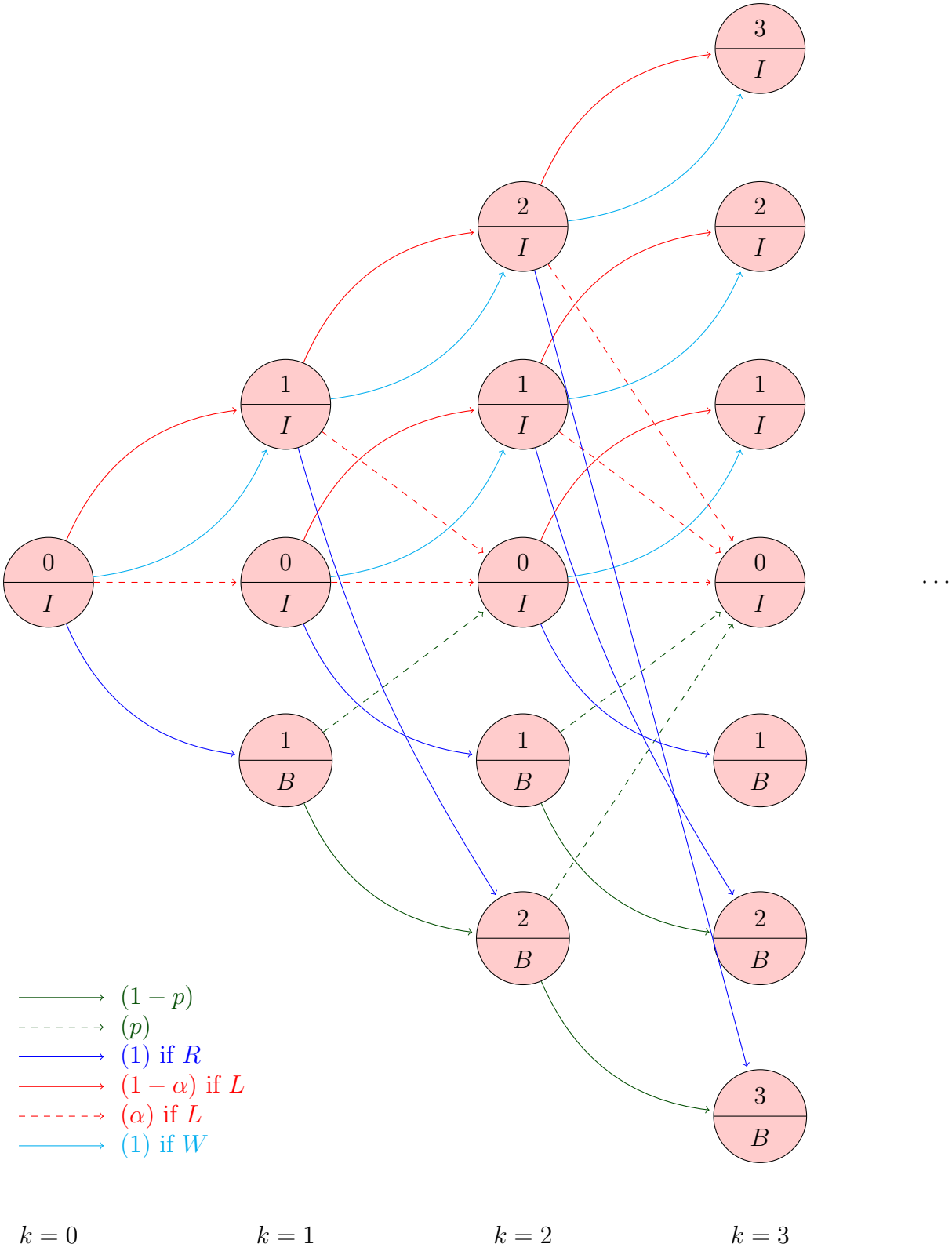


Figure 4.1: Markov decision process

The optimal policy π^* can be obtained by solving Eq. (4.5), iterating the time from $k = T$ to $k = 0$. The optimal cost $J^*(x_0)$ is equivalent to $J_0(x_0)$. The values $\pi^* = [u_0^*, ..., u_{T-1}^*]$ minimizing the right side of the equation, also named Bellman equation, are the optimal policy [[59], Sec. 1.3]. The right side of the Bellman Equation can be constructed as follows:

$$J_k(x_k) = \Delta_k + \min_{u_k} \mathop{\mathbb{E}}_{s_k} \Big\{ J_{k+1}(f_k(x_k, u_k.s_k) \Big\}$$

$$J_k(x_k) = \begin{cases} \Delta_k + \underset{r,l,w \in \{0,1\}}{min} \Big[J_{k+1}^W(x_{k+1}), J_{k+1}^R(x_{k+1}), J_{k+1}^L(x_{k+1}) \Big], & \text{if } \varsigma_k = 0 \\ \Delta_k + p \times J_{k+1}((0,0)) + (1-p) \times J_{k+1}((\Delta_k + 1, 1)), & \text{if } \varsigma_k = 1 \end{cases} \tag{4.6}$$
$$\text{subject to} \quad r + l + w = 1$$

where

$$J_{k+1}^W(x_{k+1}) = J_{k+1}((\Delta_k + 1, 0))$$
$$J_{k+1}^R(x_{k+1}) = J_{k+1}((\Delta_k + 1, 1))$$
$$J_{k+1}^L(x_{k+1}) = \alpha \times J_{k+1}((0,0)) + (1-\alpha) \times J_{k+1}((\Delta_k + 1, 0)).$$

Starting from $k = T - 1$, the time index regresses in reverse until $k = 0$ while computing the cost-to-go values of each state and their associated actions. Utilizing MATLAB, we have derived the optimal actions and generated a look-up table indicating the optimal actions based on the present time and state along with the time horizon. This can be done for any set of p, α, and P_{lim} parameters. Knowledge of these three is sufficient to make an informed decision. Notably, this iteration has a computational complexity $O(T^2)$, and the size of the generated look-up tables is proportional to T^2.

4.1.2 Constrained Local Action Analysis

As stated, subsection 4.1.1 presents the general logic behind our algorithm for the unconstrained local action case. However, the approach is unsuitable for situations where the users' **Local** actions are restrained for various reasons. To clear that obstacle, we employed the same P_{lim} parameter as in the blind decision setting to introduce how many **Local** actions are allowed to the user for the given time horizon.

Definition 16 (Allowed Local Action Number, (Γ))**.** The total number of allowed local actions within a finite horizon is represented by Γ. It is calculated as follows:

$$\Gamma = P_{lim} \cdot T \tag{4.7}$$

The number of allowable local actions until the end of the time horizon at time index k is denoted as Γ_k and computed as shown in Eq. (4.8).

$$\Gamma_k = \Gamma - \sum_{n=0}^{k-1} l_n \tag{4.8}$$

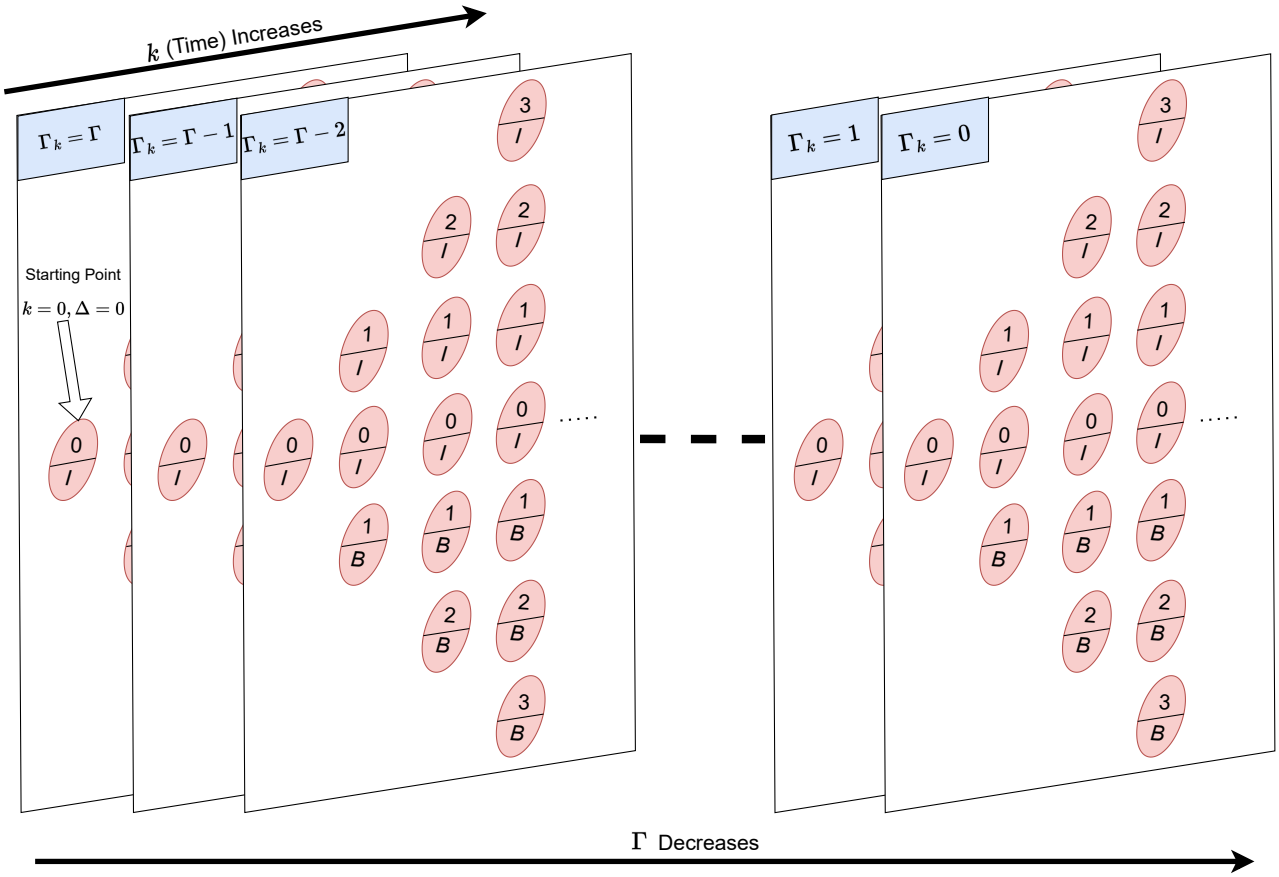


Figure 4.2: Layered Markov decision process for the constraint case

At the initial time slot $k = 0$, Γ_0 is equal to Γ. Whenever a **Local** action is employed, successful or unsuccessful, Γ_k decreases by 1 until no **Local** action rights remain. At that point, the sole option is to utilize remote servers for data computation or gathering tasks. We restrict the use of **Local** action by defining $l_k = 0$ if $\Gamma_k = 0$.

With the inclusion of the $\Gamma_k \in \{0, 1, ..., \Gamma\}$ parameter, a new state space and transition function shall be defined. In brief, a new dimension is added to the previously explained Markov Chain, and the updated 3-dimensional version is generated as illustrated in Fig. 4.2[1]. Utilizing **Local** actions decreases the Γ_k value, and the system goes one layer below. In the last layer, i.e., $\Gamma_k = 0$, the only options are waiting or employing a remote server. Thus, the Γ_k value cannot be smaller than 0.

[1]It must be noted that only the nodes are portrayed in the graph due to high-complexity of the illustration.

Definition 17 (Constrained State Space, (X_C))**.** X_C represents the set of all possible states in the constrained local action situation. At time index k, the state is denoted by $x_k = (\Delta_k, \varsigma_k, \Gamma_k), x_k \in X_C$.

The transition function for the constraint case is seen as follows:

$$x_{k+1} = f_k(x_k, u_k, s_k) = (\Delta_{k+1}, \varsigma_{k+1}, \Gamma_{k+1})$$
$$x_{k+1} = \begin{cases} (0, 0, \Gamma_k - 1) & \text{if } l_k = 1 \text{ and } s_k^L = 1 \\ (0, 0, \Gamma_k) & \text{if } \varsigma_k = 1 \text{ and } s_k^R = 1 \\ (\Delta_k + 1, 0, \Gamma_k - 1) & \text{if } l_k = 1 \text{ and } s_k^L = 0 \\ (\Delta_k + 1, 0, \Gamma_k) & \text{if } \varsigma_k = 0 \text{ and } w_k = 1 \\ (\Delta_k + 1, 1, \Gamma_k) & \text{if } r_k = 1 \text{ or } (\varsigma_k = 1 \text{ and } s_k^R = 0) \end{cases} \tag{4.9}$$

The fundamental Bellman equation (4.5) is still an essential aspect of the constrained scene. However, the construction differs due to the newly added Γ_k dimension. The constrained version takes the form seen in Eq. (4.10).

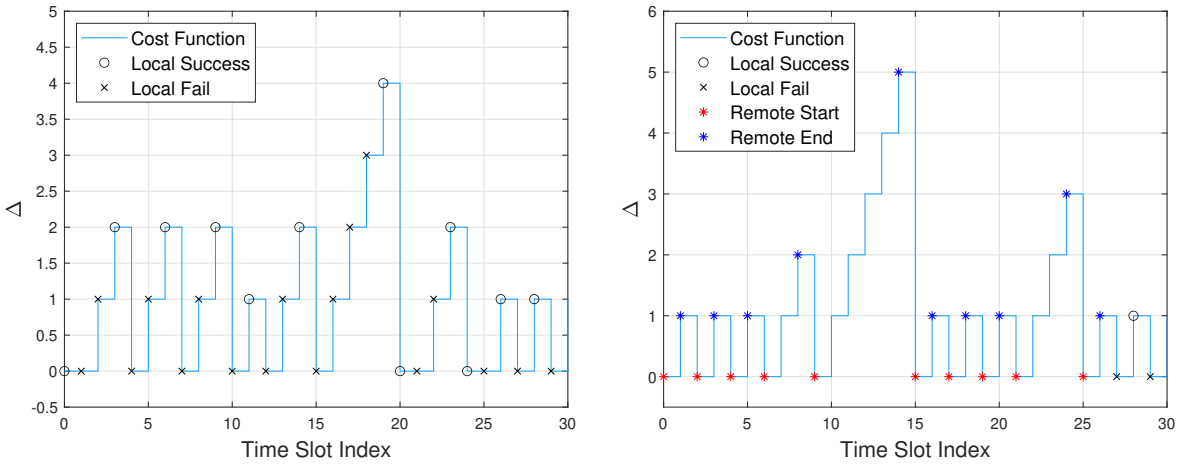
$$J_k(x_k) = \begin{cases} \Delta_k + \underset{r,l,w \in \{0,1\}}{min} \Big[J_{k+1}^W(x_{k+1}), J_{k+1}^R(x_{k+1}), J_{k+1}^L(x_{k+1}) \Big], & \text{if } \varsigma_k = 0 \\ \Delta_k + p \times J_{k+1}((0, 0, \Gamma_k)) + (1 - p) \times J_{k+1}((\Delta_k + 1, 1, \Gamma_k)), & \text{if } \varsigma_k = 1 \end{cases} \tag{4.10}$$
$$\text{subject to} \quad r + l + w = 1$$
$$\Gamma_k \overset{(a)}{\geq} l_k \, , k = 1, ..., T$$

where

$$J_{k+1}^W(x_{k+1}) = J_{k+1}((\Delta_k + 1, 0, \Gamma_k))$$
$$J_{k+1}^R(x_{k+1}) = J_{k+1}((\Delta_k + 1, 1, \Gamma_k))$$
$$J_{k+1}^L(x_{k+1}) \overset{(b)}{=} \alpha \times J_{k+1}((0, 0, \Gamma_k - 1)) + (1 - \alpha) \times J_{k+1}((\Delta_k + 1, 0, \Gamma_k - 1))$$

The constrained local action minimization problem has an additional restriction compared to the unconstrained one. In the problem formulation, (a) is added to ensure that $l_k = 0$ if $\Gamma_k = 0$. As it is restricted, Γ_k cannot diminish to 0; thus, (b) is not computed if $\Gamma_k = 0$.

In the constrained setting, the employed algorithm has a complexity of $O(T^3)$ as another difference from the unconstrained case. If the T value is high, then the iteration time to compute all possible cost-to-go values might grow. Additionally, the size of the look-up table increases as well.



(a) $p = 0.5$
($P_{R,optimal} = 0$ in blind decision-making)

(b) $p = 0.6$
($P_{R,optimal} = 1$ in blind decision-making)

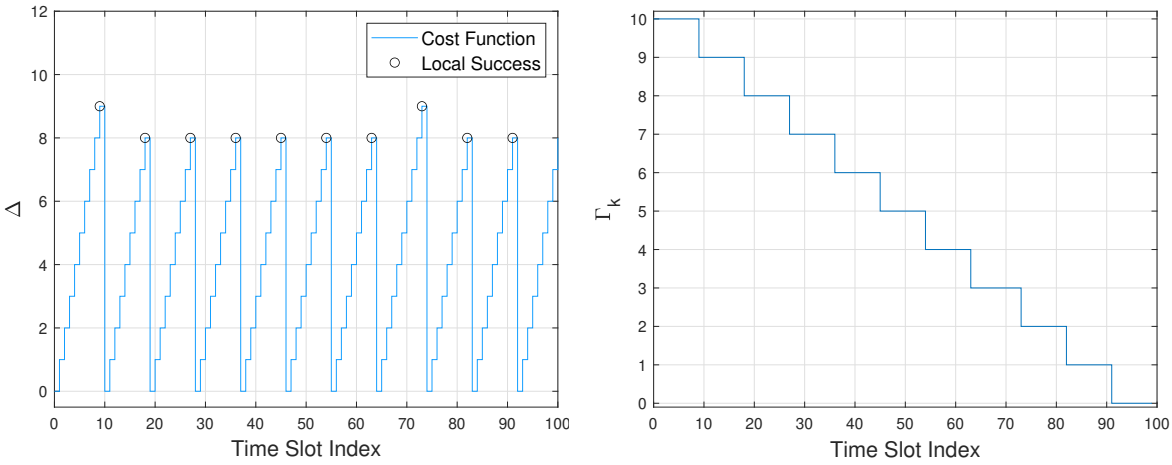
Figure 4.3: Informed decision-making realizations ($\alpha = 0.45$, $P_{lim} = 1$, $T = 30$)

4.2 Results

As stated, informed decision-making differs from blind decision-making in that it utilizes information about the current state and time. Nevertheless, the optimal actions might be identical depending on the parameters, which is much more apparent in the no-limitation circumstance. For instance, in a single-user system, intuitively speaking, one of the **Local** or **Remote** actions is superior, and it should be selected in both settings if there are no constraints. The addition of informed decision-making is the information about when the epoch, denoted for the time from $k = 0$ to $k = T$, ends. In contrast, the blind case assumes the time is infinite and acts accordingly.

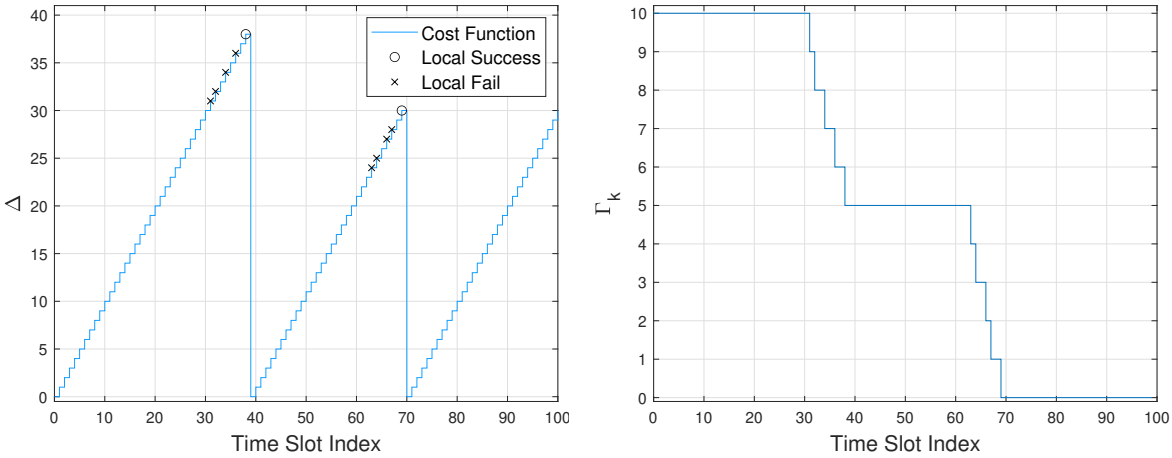
The realizations showing the cost function over time for informed decision-making in two different blind decision regions are illustrated in Fig. 4.3a and 4.3b. The former stands for utilizing sole local sources in both decision-making settings. Conversely, the latter represents the situation in which $P_{R,optimal} = 1$ in the blind decision-making case, suggesting employing the remote server exclusively. The informed decision also verifies this selection; however, it also proposes to utilize **Local** actions as the time index approaches the end of epochs. The explanation is that the reception of the remote server's response might take some time, and the user may only acquire it after the end of the epoch. This phenomenon occurs because of the finite horizon DP implementation.

Blind decision-making entails the utilization of action probabilities at each time slot to determine whether to use a **Local** action. Consequently, the **Local** actions are distributed all over the time axis. Conversely, the power of informed decision-making comes from knowing when to utilize a **Local** action. The user might disconnect from the remote



(a) Δ versus Time (b) Γ_k versus Time

Figure 4.4: A realization ($p \approx 0$, $\alpha = 1$, $P_{lim} = 0.1$, $T = 100$)



(a) Δ versus Time (b) Γ_k versus Time

Figure 4.5: A realization ($p \approx 0$, $\alpha = 0.2$, $P_{lim} = 0.1$, $T = 100$)

server temporarily. Until the conditions improve, the sole viable option is to employ local resources even if there is a possibility of failure and local action restrictions.

An experiment has been devised to monitor a user's behavior in a comparable situation. In the simulation environment, the p value is set to a very small value, implying a very large remote latency, and P_{lim} is selected as 0.1, meaning the user can use the **Local** action 10 times when $T = 100$. The decision mechanism of the user is observed for two different α values, as shown in Fig. 4.4 and 4.5. In the scene of perfect local task execution, that is $\alpha = 1$, the time instants of the **Local** actions are evenly spaced (refer 4.4a, 4.4b).

However, introducing a local task fail probability reduces the trust to the **Local** action and concentrates the actions toward the middle of the epoch (see Fig. 4.5a, 4.5b). The situations where the user employed **Local** actions in the earlier stages make the cost function incur much higher values in the later stages, and vice versa. Knowledge of the present cost value and time, along with the awareness of the time horizon, empowers the user to align itself based on the anticipated outcomes.

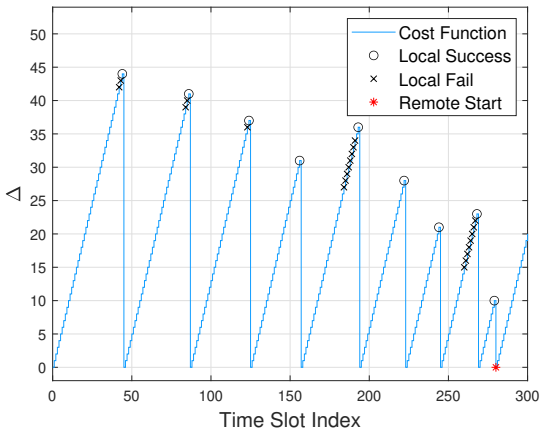
Apart from the extreme conditions, the optimal behavior of a user also depends on the time instants of the **Remote** actions. In informed decision-making, the region concept (See 3.6) is not valid anymore. The effect of a tiny alteration on the parameters does not trigger a significant change. Instead, it shifts the optimal sequence of the actions. Fig. 4.6 shows five distinct realizations corresponding to different values of p. All other parameters, such as P_{lim}, α, and T, are kept constant in the environment settings. When p is small, indicating high remote latency, **Local** actions are employed conservatively, as depicted in Figure 4.6a. The user only uses **Remote** actions after all **Local** action rights are spent. Increasing the p value (refer to Figure 4.6b) induces users to consume the **Local** action rights faster. This shift is attributed to a higher trust level in **Remote** actions compared to the scenario with minimal p values, prompting the user to complete **Local** actions earlier.

When neither local nor remote options dominate, users distribute **Local** actions over time, as illustrated in Figure 4.6c, where the superiority of one execution option over the other is not evident. Conversely, with low remote latency (i.e., high p values), users delay the use of **Local** actions until the end of epochs, exemplified in Figure 4.6d, which is the opposite scenario of in Fig. 4.6b. For very high p values, illustrated in Figure 4.6e, the optimal action characteristics are akin to 4.3b. In this case, the user solely employs remote resources due to significantly low remote latency, and the role of P_{lim} is negligible as the user does not use the **Local** actions[2].

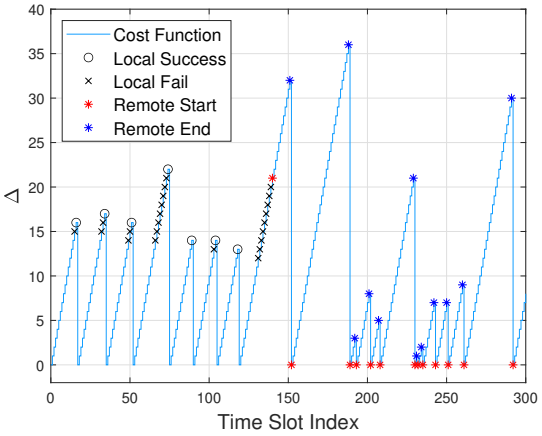
The intricate interplay among P_{lim}, α, and p continues to be evident in informed decision-making, similar to the blind setting. The distinction is that changing the parameters α and p shifts the time instants of the actions rather than changing the action probabilities, while P_{lim} extends or shrinks the interval in which **Local** actions are used. Fig. 4.7 illustrates realizations for different α and P_{lim} values, satisfying that the **Local** actions are superior to remote ones[3]. If it were the other way around, we would observe only **Remote** actions because **Remote** actions would yield superior results compared to **Local** ones, and there is no limitation on the use of **Remote** actions. Higher α values lead users to trust **Local** actions more and prioritize consuming them first, similar to the effect of decreasing p. This influence is apparent while comparing Fig. 4.7a and 4.7b, as well as Fig. 4.7c and 4.7d. On the other hand, increasing P_{lim} extends the interval duration of employing **Local** actions. In instances where $P_{lim} = 1$, as depicted in Fig. 4.7e and 4.7f, the system solely relies on the **Local** actions as it is the better choice.

[2]Except at the final time slots of an epoch as in Fig. 4.3b.

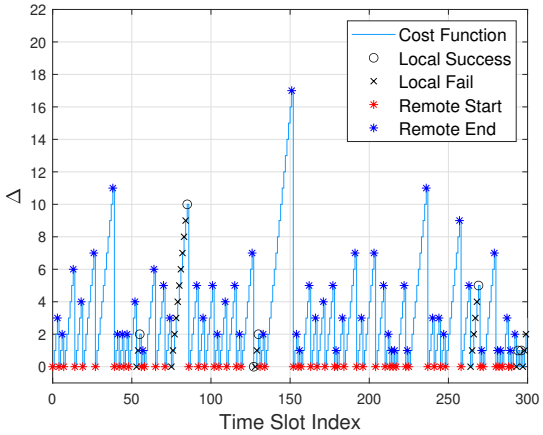
[3]The influence of P_{lim} is trivial in the converse case, as explained in the prior paragraph.



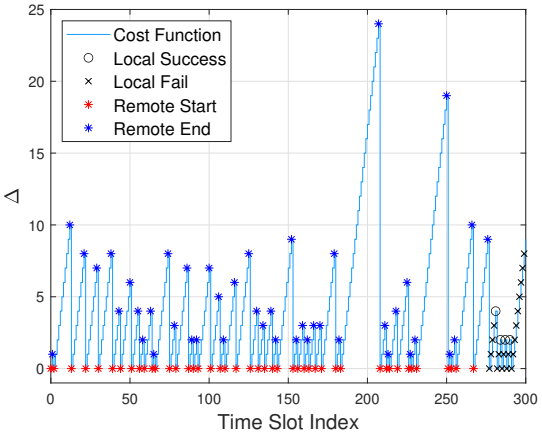
(a) $p = 0.01$



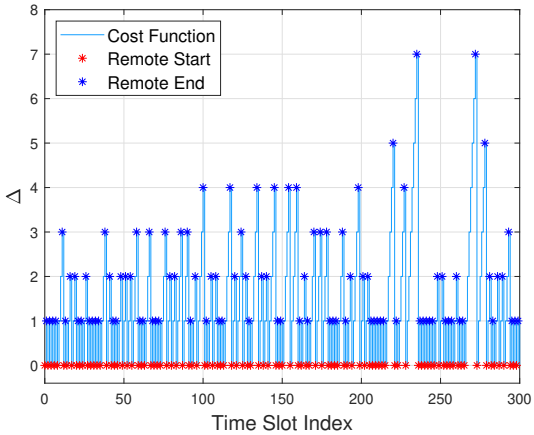
(b) $p = 0.05$



(c) $p = 0.18$

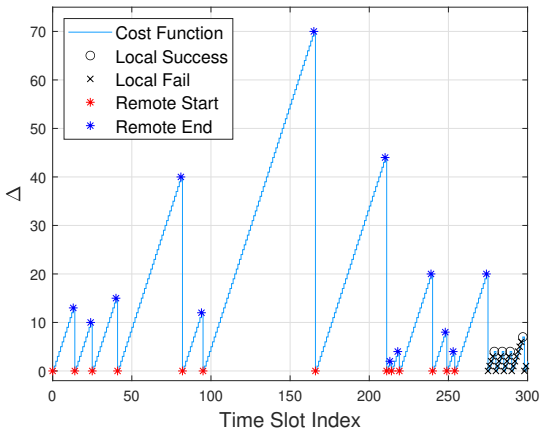


(d) $p = 0.2$

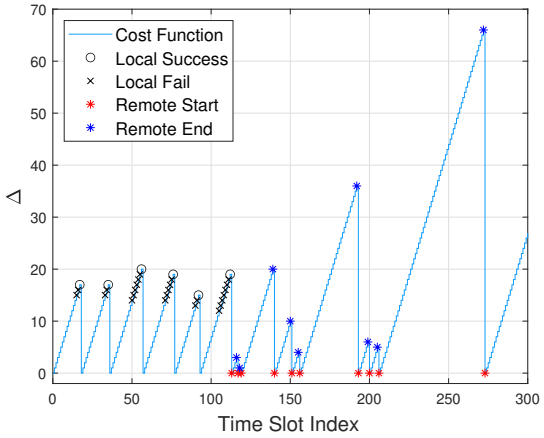


(e) $p = 0.5$

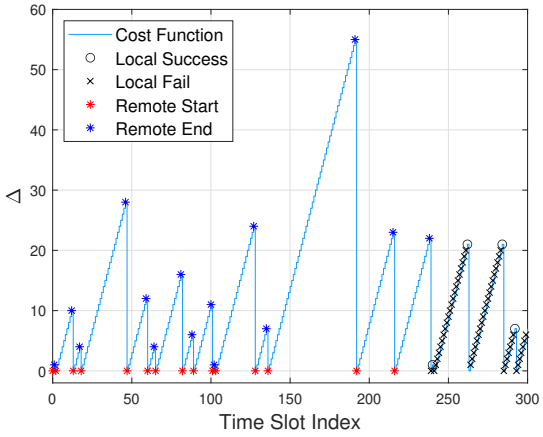
Figure 4.6: Realizations for different p values ($\alpha = 0.2$, $P_{lim} = 0.1$, $T = 300$)



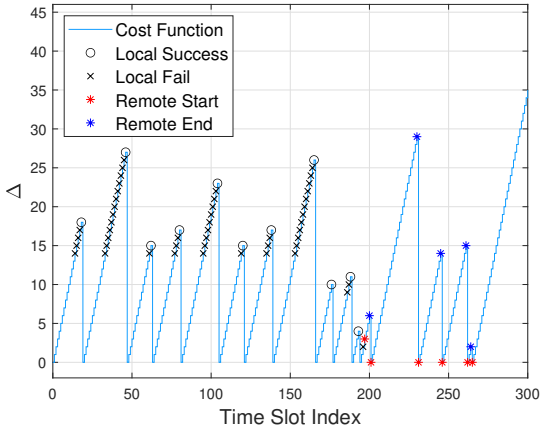
(a) $\alpha = 0.05$ and $P_{lim} = 0.1$



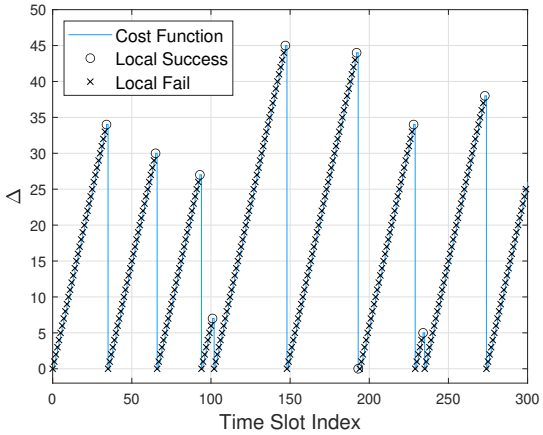
(b) $\alpha = 0.20$ and $P_{lim} = 0.1$



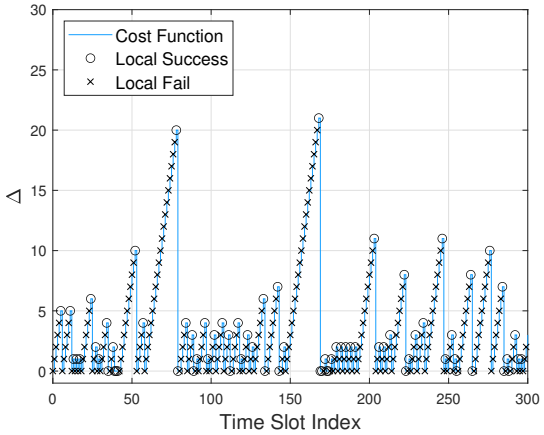
(c) $\alpha = 0.05$ and $P_{lim} = 0.2$



(d) $\alpha = 0.20$ and $P_{lim} = 0.2$



(e) $\alpha = 0.05$ and $P_{lim} = 1$



(f) $\alpha = 0.20$ and $P_{lim} = 1$

Figure 4.7: Realizations for different α and P_{lim} values ($p = 0.05$, $T = 300$)

Chapter 5

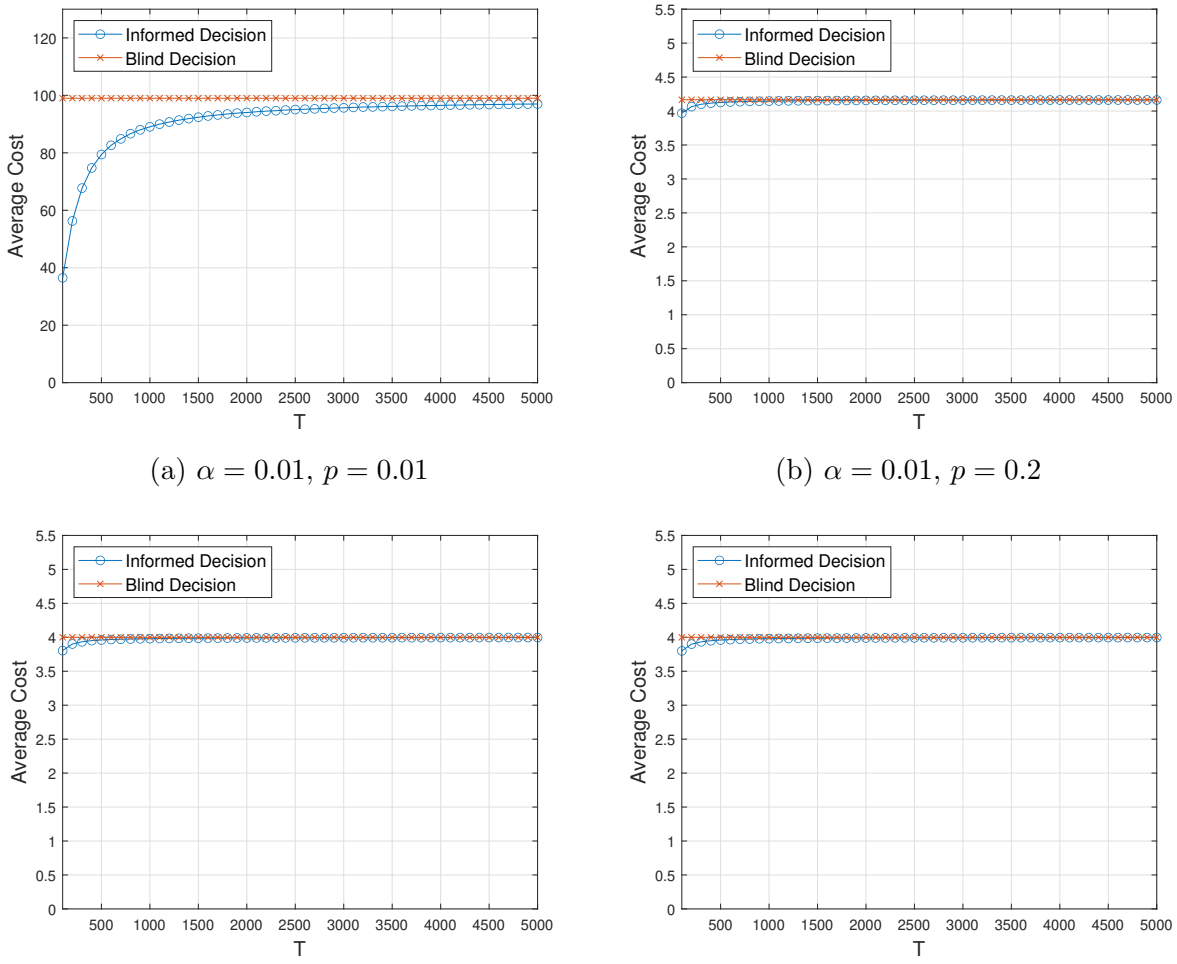
Discussion

This study delved into two decision-making strategies, blind and informed, intending to maximize freshness. In practical applications, storing the optimal actions for different situations can be a problem in memory-constrained systems. The blind setting, in which the user does not utilize the current cost value, addresses this challenge. We formulated the problem as an open-loop minimization problem, which enables choosing all actions at time $k = 0$ based on action probabilities. Subsequently, we performed a steady-state analysis to derive the optimal action probabilities.

In contrast, the informed setting uses the current cost value and time, allowing us to derive a closed-loop minimization solution. This approach involves making decisions at the beginning of each time slot. The optimal solutions and the resulting minimal average cost values may coincide with the open-loop solution depending on the parameters α, P_{lim} and p. The following paragraphs will further explore the dynamics of the two settings, employing graphs to visually stress how the average cost function alters when the parameters and time horizon differ.

Fig. 5.1 shows the average cost functions for various time horizon T values in the unconstrained local action scenario, with $P_{\text{lim}} = 1$. In the figures, the red lines represent the steady-state average cost of the blind decision approach, where optimal P_R and P_L are employed for stochastic decision-making. The value remains constant across the time horizon axis since the impact of the transition phase, i.e., the phase until reaching the steady state, is not included. Conversely, the blue lines illustrate the average cost of the informed decision-making settings for different T values, obtained through Monte Carlo simulations, running 1.000.000 times for each T value and averaging the average cost function results. MATLAB was the tool used for these simulations.

Each run lasts exactly T time slots. For small T values, a single Monte Carlo run ends before reaching the expected cost values. In simpler terms, it spends most of the time in the transient phase, influencing the overall average. Accordingly, the average cost difference



(a) $\alpha = 0.01$, $p = 0.01$

(b) $\alpha = 0.01$, $p = 0.2$

(c) $\alpha = 0.2$, $p = 0.01$

(d) $\alpha = 0.2$, $p = 0.2$

Figure 5.1: Average cost versus T graphs for different α and p values ($P_{lim} = 1$)

between the blind and informed decision-making can be significant, as seen in Fig 5.1a. However, as T increases, the effect of the transition phase decreases. Therefore, we focus on the cost difference at higher T values when comparing the settings.

Fig. 5.1a portrays the average cost for small local success probability (α) and remote latency parameter (p) values. With $P_{lim} = 1$, the user makes a binary selection, utilizing either solely local or remote resources, as one is probably superior to the other. However, both actions have low success probabilities, leading to high average costs. On the other hand, the same settings but with higher p values, illustrated in Fig. 5.1b, have much smaller average cost values in both decision-making strategies. In this setting, utilizing remote resources provides much better results than resorting to the local ones, and both

decision-making strategies use **Remote** actions in all time slots when the node is *Idle*[1]. Additionally, the cost difference in small T values is much smaller than the previously explained $\alpha = 0.01$ and $p = 0.01$ case because the steady state cost is much lower, making the transition phase much shorter.

The results of an unconstrained local action system with high local success probability ($\alpha = 0.2$) and relatively low remote latency parameter ($p = 0.01$) values are depicted in Fig. 5.1c. In this setting, the user employs local resources in each time slot as **Local** actions are superior to **Remote** actions. Increasing the p value to 0.2 does not alter anything since the local success probability is the same even if the p value is different, forcing the user to use **Local** actions[2]. This behavior can be observed by comparing Fig. 5.1c (former case) and Fig. 5.1d (latter case).

All graphs in Fig. 5.1 represent the unconstrained local action case, where the users make binary decisions about offloading. They choose to employ either solely **Local** or **Remote** actions, which is the same in both blind and informed decision-making approaches. Thus, the average cost results of both settings are the same, which can be observed from the fact that the average cost in the informed decision setting converges to the steady state average cost as T increases[3]. Nevertheless, the power of the informed decision is to know when to use **Local** actions.

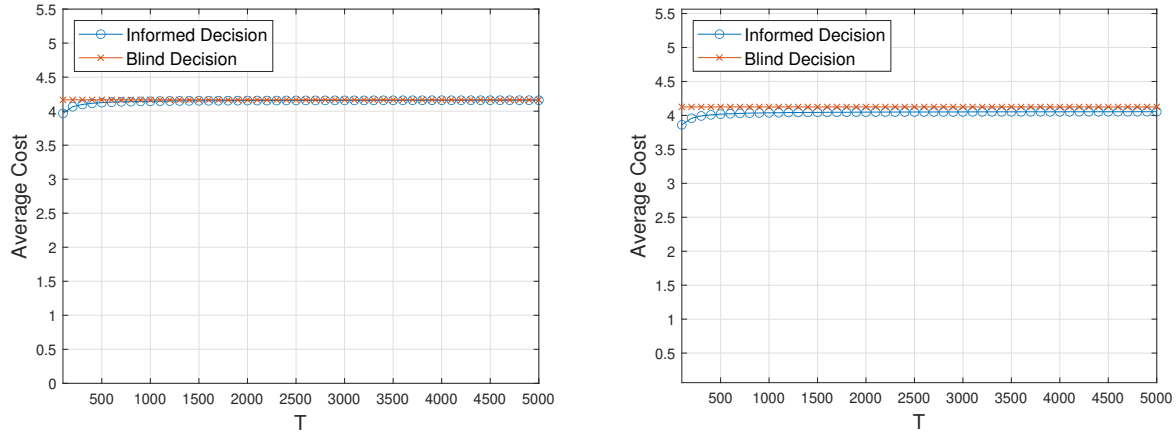
The graphs illustrating the average costs in the constrained limit action case for $\alpha = 0.2$ and $\alpha = 1$ are shown in Fig. 5.2. Notably, these graphs depict systems with a high remote latency parameter p, indicating that the user can quickly execute tasks via remote units, even if it consumes all the local action rights. The value of α is also essential since, 1% of the time, the user employs **Local** actions in this setting, and knowing the optimal time slots to use **Local** actions improves timeliness if the **Local** actions are better than **Remote** ones. The average cost of the informed decision, where $\alpha = 0.2$ (Fig. 5.2a), converges to the steady-state solution as T increases. However, the converged average cost of the informed decision is smaller than the steady-state average cost in the case of $\alpha = 1$, as seen in Fig. 5.2b. However, this difference is insignificant as the p value is high enough to compensate for local task execution limitations.

The advantage of informed decision-making is revealed at small remote latency parameter (p) values. In this case, the user cannot aggressively use the **Remote** actions due to longer response times of the remote task execution. On the other hand, greedy use of the **Local** actions quickly consumes the allowed local action rights. Considering the limitations on the **Local** actions, the timings of the **Local** actions become incredibly vital. The average cost values for both informed and blind decisions, given a small remote latency parameter

[1]Except the last one or two time slots as explained in Sec. 4.2.

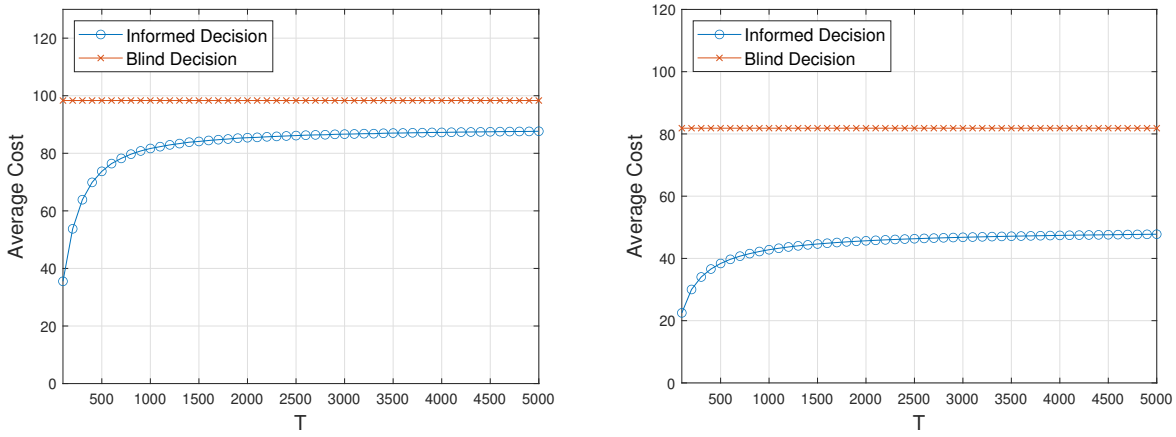
[2]When α and p values equal, the user prefers to employ **Local** actions due to an additional 1 time slot to forward the task to the initial server in **Remote** actions.

[3]It can be hard to observe from Fig. 5.1a since the transient phase still influences the overall average even at $T = 5.000$. However, based on our experiments, we observe that convergence becomes more apparent with increasing values of T.



(a) $\alpha = 0.2$ (b) $\alpha = 1$

Figure 5.2: Average cost versus T graphs for different α values ($p = 0.2$ and $P_{lim} = 0.01$)



(a) $\alpha = 0.2$ (b) $\alpha = 1$

Figure 5.3: Average cost versus T graphs for different α values ($p = 0.01$ and $P_{lim} = 0.01$)

($p = 0.01$), are depicted in Fig. 5.3. The first figure, illustrated in Fig. 5.3a, shows the average cost when $\alpha = 0.2$. Without any local constraints, the user would pick the **Local** action in each time slot. This unconstrained scenario with the same α and p values is depicted in Fig. 5.1c. However, as there is a certain number of allowed local action rights, their distribution is crucial, which is the difference between informed and blind decision-making settings. It is evident from Fig. 5.3a that the average cost gap between informed and blind decisions is more prominent in the constrained case.

From another point of view, a scenario with consistent local action success, where $\alpha = 1$, is illustrated in Fig. 5.3b. Here, the **Local** actions are evenly distributed at equal intervals

since the system does not trust **Remote** actions. A realization example of the informed case under these system parameters resembles Fig. 4.4a, whereas the blind case randomly distributes these local rights over time. In this case, the gap between informed and blind decision cases is particularly evident. The informed decision reduces the cost function by approximately 42%.

As observed from the results, informed decision-making consistently yields superior results to the blind case. Nonetheless, the Dynamic Programming approach is not perfect, and several issues must be addressed.

- **Issue 1: Memory**

Storing the look-up table generated by Dynamic Programming might pose challenges for memory-constrained devices. As stated, the size of the look-up table is proportional to T^2 in the unconstrained local action case and T^3 in the constrained one. Higher T values make informed decision-making unsuitable for such devices due to high complexity. Islam et al. [60] underlined this problem of Dynamic Programming solutions, stating that the stored results of every sub-problem, regardless of their utility, consume an extensive amount of space in the memory, resulting in using more energy and resources. While this concern is not explicitly addressed in the thesis, it must be considered depending on the application.

- **Issue 2: Incorrect Parameter Values**

The second issue concerns potentially incorrect information about the system parameters, such as p, α, and P_{lim}. The current model assumes the user has the perfect system information and acts accordingly. However, acquiring the perfect information is not an easy task. The system parameters may change over time, making their estimate even more challenging. Mancuso et al. [27] emphasized that state-aware methods are highly susceptible to errors and stated that the stateless methods gave almost equal or sometimes better results than the counter-part stateful methods in case of incorrect information. Therefore, the influence of incorrect system information is another issue that must be considered while utilizing informed decision-making.

Overall, the implementation of blind decision-making is straightforward because it does not demand an extra amount of space in the IoT device's memory. Additionally, it does not require additional units to obtain information about the current cost function or state, except for the availability state. However, the results obtained are suboptimal due to the random timing of the actions, and the user is unaware of the current situation. It relies solely on randomness; thereby, each realization significantly differs. As a result, there is no control over the flow. For instance, the user may not use any action for an extended period, even if the cost function increases significantly.

Conversely, the informed decision-making consciously controls the flow. Whenever the cost function surpasses a certain threshold, depending on the current time and system parameters, the user tries to reduce it by taking action until the cost is reset. The user adapts the actions according to the success of the task executions, yielding superior results.

	Blind Decision	Informed Decision
Advantages	• Easy to implement • Does not have to know the current cost value • Requires less space in the memory, thereby faster decision making [60]	• Arranges the actions according to previous outputs • Better flow control • Optimal solution
Drawbacks	• Suboptimal solution • Closely linked to randomness • Hard to control the flow	• Requires the cost information • Takes up a lot of space in the memory [60] • Error-prone in case of incorrect information [27] • Can yield a suboptimal solution for a small time horizon

Table 5.1: The summary of the discussion

However, this approach requires access to information such as the number of local action rights left, current cost, time, and time horizon since it combines all these values and takes a decision using a look-up table. In addition, the size of the look-up tables is another problem since the user may need many different look-up tables for different situations and store many unnecessary data in the memory, some of which may not even be used for a long time. This situation can be a notable challenge for memory-constrained devices and decreases flexibility. Furthermore, a piece of incorrect information about the system parameters can make results worse.

To conclude, neither of the decision-making settings is flawless. Each setting presents its advantages and drawbacks, as outlined in Table 5.1. It is the designer's responsibility to decide the most suitable offloading decision-making strategy.

Chapter 6

Conclusion

Within the scope of the thesis, we formulated a timeliness-based slotted-time data computing/gathering offloading model suitable for time-sensitive IoT applications. The developed framework provides solutions for users experiencing a dilemma between accuracy and latency. This situation was manifested through a dichotomy between "Local" and "Remote". In this context, "Local" represents the less powerful resources in proximity, where both latency and accuracy of the data computing or gathered data is low. On the other hand, "Remote" pertains to the robust units positioned relatively farther from the user, where the gathered/computed data is accurate in exchange for high latency. We solved the offloading problem with two distinct approaches, each harnessing a user-driven decision-making model, and introduced and implemented the local computation limitation issue into both approaches.

First, we employed a stochastic decision-making strategy where the user makes the offloading decisions without any information about how timely the flow is, solely based on predefined probabilities. The objective was to identify the optimal probabilities for employing local and remote sources, along with the probability of waiting. We formulated this problem as an open-loop minimization problem. We derived a closed-form steady-state solution using convex optimization for single-user cases and employed brute-force methods to obtain a user's behavior in the multiple-user scenario.

Next, we treated the problem as a Markov Decision Process (MDP), where the user knows the system's timeliness and other relevant values and employed Dynamic Programming (DP) to determine the optimal action sequence. We analyzed the system over a finite horizon and revealed that the user's awareness of the current freshness of the flow helps the user to make logical decisions. We showed the critical role of knowing when to use local sources in improving freshness, especially when the remote latency is high.

Subsequently, we conducted a comparison between the two approaches and obtained solutions. We demonstrated that the Dynamic Programming solution yields superior results to the stochastic decision-making strategy and elucidated the scenarios where these two approaches follow the same offloading strategy. In addition to this, we discussed potential issues regarding the evaluated offloading approaches and underlined that the ideal strategy hinges on the system requirements, necessitating careful consideration by the designer.

As we bring this thesis to a closure, we anticipate that these findings will contribute to the ongoing efforts to strike a balance between accuracy and latency. In future studies, we plan to extend the model by employing a game theoretic approach to investigate the behavior of users. Additionally, we exclusively focused on the single-user scenario in the informed decision-making setting, leaving multiple-user scenarios as another potential area for future research. We believe that offloading is a milestone in IoT applications and will continue to play a crucial role, and we hope that our research will pave the way for numerous studies.

Appendix A

Proofs and Derivations

A.1 Derivation of Availability State Probabilities

$$\pi_I = \frac{p}{P_R + p} \text{ and } \pi_B = \frac{P_R}{P_R + p}$$

Proof. We conduct steady-state analysis to derive stationary *Idle* and *Busy* availability state probabilities. Note that $\pi_I + \pi_B = 1$.

$$\pi_I = \pi_I * (P_L + P_W) + \pi_B * p$$
$$\pi_I * (1 - P_L - P_W + p) = p$$
$$\pi_I * (P_R + p) = p$$
$$\pi_I = \frac{p}{P_R + p} \text{ and } \pi_B = 1 - \pi_I = \frac{P_R}{P_R + p}$$

□

A.2 Derivation of Closed Form $P_\chi(x)$

We derive the PMF of the inter-reset time, denoted as $P_\chi(x)$, including the extreme cases.

When $p \neq 1, \alpha P_L + P_R \neq 1$ and $\alpha P_L + P_R \neq p$:

$$P_\chi(x)_{x \in \mathbb{N}^+} = \frac{(\alpha P_L + P_R)(\alpha P_L - p)}{\alpha P_L + P_R - p}(1 - \alpha P_L - P_R)^{x-1} + \frac{pP_R}{\alpha P_L + P_R - p}(1-p)^{x-1}$$

$$P_\chi(x) = \begin{cases} \alpha P_L(1-p)^{x-1}h[x-1] + pP_R(x-1)(1-p)^{x-2}h[x-2] & \text{if } p = \alpha P_L + P_R \neq 1 \\ pP_R(1-p)^{x-2}h[x-2] + \alpha P_L\delta[x-1] & \text{if } p \neq 1, \alpha P_L + P_R = 1 \\ (\alpha P_L + P_R)(1-\alpha P_L)(1-\alpha P_L - P_R)^{x-2}h[x-2] & \text{if } p = 1, \alpha P_L + P_R \neq 1 \\ \quad +\alpha P_L\delta[x-1] & \\ \alpha P_L\delta[x-1] + P_R\delta[x-2] & \text{if } p = \alpha P_L + P_R = 1 \end{cases}$$

Proof. For the sake of simplicity, let us name each state seen in Fig. 3.2 as $\pi_{\Delta,\varsigma}$ where Δ and ς are the cost and availability state. It is trivial to write the first terms of the PMF. The only possible way to have $\chi = 1$ is the successful local task execution. In order for χ to be equal to 2, the user needs to follow $[\pi_{0,I} \to \pi_{1,I} \to \pi_{0,I}]$ or $[\pi_{0,I} \to \pi_{1,B} \to \pi_{0,I}]$ paths. The user follows one of the $[\pi_{0,I} \to \pi_{1,I} \to \pi_{2,I} \to \pi_{0,I}]$, $[\pi_{0,I} \to \pi_{1,I} \to \pi_{2,B} \to \pi_{0,I}]$, or $[\pi_{0,I} \to \pi_{1,B} \to \pi_{2,B} \to \pi_{0,I}]$ paths for $\chi = 3$. The pattern can be seen below.

$P_\chi(1) = \alpha P_L$

$P_\chi(2) = \alpha P_L(1 - \alpha P_L - P_R) + pP_R$

$P_\chi(3) = \alpha P_L(1 - \alpha P_L - P_R)^2 + pP_R(1 - \alpha P_L - P_R) + pP_R(1 - p)$

$P_\chi(4) = \alpha P_L(1 - \alpha P_L - P_R)^3 + pP_R[(1 - \alpha P_L - P_R)^2 + (1 - p)(1 - \alpha P_L - P_R) + (1 - p)^2]$

$\vdots$

- For the general case when $p \neq 1$,$\alpha P_L + P_R \neq 1$, and $\alpha P_L + P_R \neq p$:

$$P_\chi(x) = \alpha P_L(1 - \alpha P_L - P_R)^{x-1} + pP_R \sum_{n=0}^{x-2} (1 - \alpha P_L - P_R)^n (1 - p)^{x-2-n}$$

$$= \alpha P_L(1 - \alpha P_L - P_R)^{x-1} + pP_R(1 - p)^{x-2} \sum_{n=0}^{x-2} \left(\frac{1 - \alpha P_L - P_R}{1 - p}\right)^n$$

Let $\left(\frac{1 - \alpha P_L - P_R}{1 - p}\right) = \kappa$, then:

$$= \alpha P_L(1 - \alpha P_L - P_R)^{x-1} + pP_R(1 - p)^{x-2}\left(\frac{1 - \kappa^{x-1}}{1 - \kappa}\right)$$

$$= \alpha P_L(1 - \alpha P_L - P_R)^{x-1} + \frac{pP_R}{\alpha P_L + P_R - p}[(1 - p)^{x-1} - (1 - \alpha P_L - P_R)^{x-1}]$$

$$= \frac{(\alpha P_L + P_R)(\alpha P_L - p)}{\alpha P_L + P_R - p}(1 - \alpha P_L - P_R)^{x-1} + \frac{pP_R}{\alpha P_L + P_R - p}(1 - p)^{x-1}$$

- If $p = \alpha P_L + P_R = 1$:
  $P_\chi(x) = \alpha P_L \delta[x - 1] + P_R \delta[x - 2]$

- If $p = 1$ and $\alpha P_L + P_R \neq 1$:
  $P_\chi(x) = \alpha P_L(1 - \alpha P_L - P_R)^{x-1} h[x - 1] + P_R(1 - \alpha P_L - P_R)^{x-2} h[x - 2]$
  $= \alpha P_L \delta[x - 1] + (\alpha P_L + P_R)(1 - \alpha P_L)(1 - \alpha P_L - P_R)^{x-2} h[x - 2]$

- If $p \neq 1$ and $\alpha P_L + P_R = 1$:
  $P_\chi(x) = pP_R(1 - p)^{x-2} h[x - 2] + \alpha P_L \delta[x - 1]$

- If $p = \alpha P_L + P_R \neq 1$:
  $P_\chi(x) = \alpha P_L(1 - p)^{x-1} h[x - 1] + pP_R(x - 1)(1 - p)^{x-2} h[x - 2]$

□

A.3 Proof of The Cost Function Ensemble Average

We prove the average cost equation for $\forall \alpha, P_L, P_R \in [0,1]$, and $\forall p \in (0,1]$.

$$\overline{\Delta} = -1 + \frac{1}{p} + \frac{1}{\alpha P_L + P_R} - \frac{1}{p + P_R}$$

Proof. The general PMF equation (Eq. 3.7) is used for the proof. The other cases are trivial and does not change the result.

$$P_\chi(x)_{x\in\mathbb{N}^+} = \frac{(\alpha P_L + P_R)(\alpha P_L - p)}{\alpha P_L + P_R - p}(1 - \alpha P_L - P_R)^{x-1} + \frac{pP_R}{\alpha P_L + P_R - p}(1-p)^{x-1}$$

For the sake of simplicity, let us denote $\psi = \frac{(P_R+\alpha P_L)(\alpha P_L - p)}{P_R + \alpha P_L - p}$. Then:

$$P_\chi(x) = \psi(1 - \alpha P_L - P_R)^{x-1} + (\alpha P_L - \psi)(1-p)^{x-1}$$

Derive $\mathbb{E}[\chi]$:

$$\begin{aligned}
\mathbb{E}[\chi] &= \lim_{n\to\infty}\sum_{x=1}^{n} x\psi(1-\alpha P_L - P_R)^{x-1} + \lim_{n\to\infty}\sum_{x=1}^{n} x(\alpha P_L - \psi)(1-p)^{x-1} \\
&= \psi \lim_{n\to\infty}\sum_{x=1}^{n} x(1-\alpha P_L - P_R)^{x-1} + (\alpha P_L - \psi)\lim_{n\to\infty}\sum_{x=1}^{n} x(1-p)^{x-1} \\
&= \psi\frac{1}{(\alpha P_L + P_R)^2} + (\alpha P_L - \psi)(\frac{1}{p^2}) = \psi(\frac{1}{(\alpha P_L + P_R)^2} - \frac{1}{p^2}) + \frac{\alpha P_L}{p^2} \\
&= \psi(\frac{1}{\alpha P_L + P_R} - \frac{1}{p})(\frac{1}{\alpha P_L + P_R} + \frac{1}{p}) + \frac{\alpha P_L}{p^2} = \frac{p + P_R}{p(\alpha P_L + P_R)}
\end{aligned}$$

Derive $\mathbb{E}[\chi^2]$:

$$\begin{aligned}
\mathbb{E}[\chi^2] &= \lim_{n\to\infty}\sum_{x=1}^{n} x^2\psi(1-\alpha P_L - P_R)^{x-1} + \lim_{n\to\infty}\sum_{x=1}^{n} x^2(\alpha P_L - \psi)(1-p)^{x-1} \\
&= \psi \lim_{n\to\infty}\sum_{x=1}^{n} x^2(1-\alpha P_L - P_R)^{x-1} + (\alpha P_L - \psi)\lim_{n\to\infty}\sum_{x=1}^{n} x^2(1-p)^{x-1} \\
&= \psi[\frac{2}{(\alpha P_L + P_R)^3} - \frac{1}{(\alpha P_L + P_R)^2}] + (\alpha P_L - \psi)[\frac{2}{p^3} - \frac{1}{p^2}] \\
&= \psi[\frac{2}{(\alpha P_L + P_R)^3} - \frac{2}{p^3} + \frac{1}{p^2} - \frac{1}{(\alpha P_L + P_R)^2}] + \frac{2\alpha P_L}{p^3} - \frac{\alpha P_L}{p^2} \\
&= \psi[\frac{2}{(\alpha P_L + P_R)^3} - \frac{2}{p^3}] + \frac{2\alpha P_L}{p^3} + \psi(\frac{1}{p^2} - \frac{1}{(\alpha P_L + P_R)^2}) - \frac{\alpha P_L}{p^2} \\
&= \psi[\frac{2}{(\alpha P_L + P_R)^3} - \frac{2}{p^3}] + \frac{2\alpha P_L}{p^3} - \frac{p + P_R}{p(\alpha P_L + P_R)}
\end{aligned}$$

Calculate $\frac{\mathbb{E}[\chi^2]}{2\mathbb{E}[\chi]}$.

$$
\begin{aligned}
\frac{\mathbb{E}[\chi^2]}{2\mathbb{E}[\chi]} &= \frac{\psi[\frac{2}{(\alpha P_L+P_R)^3}-\frac{2}{p^3}]+\frac{2\alpha P_L}{p^3}-\frac{p+P_R}{p(\alpha P_L+P_R)}}{2(\frac{p+P_R}{p(\alpha P_L+P_R)})} \\
&= -\frac{1}{2}+\frac{\psi[(\frac{1}{\alpha P_L+P_R}-\frac{1}{p})(\frac{1}{(\alpha P_L+P_R)^2}+\frac{1}{p(\alpha P_L+P_R)}+\frac{1}{p^2})]+\frac{\alpha P_L}{p^3}}{\frac{p+P_R}{p(\alpha P_L+P_R)}} \\
&= -\frac{1}{2}+\frac{\frac{(p-\alpha P_L)}{p}[\frac{1}{(\alpha P_L+P_R)^2}+\frac{1}{p(\alpha P_L+P_R)}+\frac{1}{p^2}]+\frac{\alpha P_L}{p^3}}{\frac{p+P_R}{p(\alpha P_L+P_R)}} \\
&= -\frac{1}{2}+\frac{[\frac{1}{(\alpha P_L+P_R)^2}+\frac{1}{p(\alpha P_L+P_R)}+\frac{1}{p^2}]-\frac{\alpha P_L}{p(\alpha P_L+P_R)}[\frac{1}{p}+\frac{1}{\alpha P_L+P_R}]}{\frac{p+P_R}{p(\alpha P_L+P_R)}} \\
&= -\frac{1}{2}+\frac{[(\frac{1}{p}+\frac{1}{\alpha P_L+P_R})^2-\frac{1}{p(\alpha P_L+P_R)}]-\frac{\alpha P_L}{p(\alpha P_L+P_R)}[\frac{1}{p}+\frac{1}{\alpha P_L+P_R}]}{\frac{p+P_R}{p(\alpha P_L+P_R)}} \\
&= -\frac{1}{2}+\frac{(\frac{1}{p}+\frac{1}{\alpha P_L+P_R})[\frac{1}{p}+\frac{1}{\alpha P_L+P_R}-\frac{\alpha P_L}{p(\alpha P_L+P_R)}]-\frac{1}{p(\alpha P_L+P_R)}}{\frac{p+P_R}{p(\alpha P_L+P_R)}} \\
&= -\frac{1}{2}+\frac{1}{p}+\frac{1}{\alpha P_L+P_R}-\frac{1}{p+P_R} \\
\therefore \overline{\Delta} &= \frac{\mathbb{E}[\chi^2]}{2\mathbb{E}[\chi]}-\frac{1}{2} = -1+\frac{1}{p}+\frac{1}{\alpha P_L+P_R}-\frac{1}{p+P_R}
\end{aligned}
$$

□

A.4 Proof of The First Extreme Point

We prove $P_{R,extrema,1}$ is bigger than $-\frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}$, and this extreme point is a local minimum.

Proof. Proving $P_{R,extrema,1} > -\frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}$:

$$p^2 > 0$$
$$\frac{p^2}{p+\alpha P_{lim}} > 0$$
$$p - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}} > 0$$
$$(p - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}})\frac{\sqrt{\frac{p}{p+\alpha P_{lim}}}}{1-\sqrt{\frac{p}{p+\alpha P_{lim}}}} > 0$$
$$\frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}\sqrt{\frac{p}{p+\alpha P_{lim}}}}{1-\sqrt{\frac{p}{p+\alpha P_{lim}}}} > 0$$
$$\frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}{1-\sqrt{\frac{p}{p+\alpha P_{lim}}}} + \frac{p\alpha P_{lim}}{p+\alpha P_{lim}} > 0$$
$$P_{R,extrema,1} = \frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}}{1-\sqrt{\frac{p}{p+\alpha P_{lim}}}} > -\frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}$$

Applying second derivative test to prove that it is a local minimum:

$$\begin{aligned}\frac{\partial^2\overline{\Delta}_{LLZ}}{\partial P_R^2} &= \frac{\partial[\frac{1}{(p+P_R)^2} - (\frac{p}{p+\alpha P_{lim}})(\frac{1}{P_R+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^2]}{\partial P_R} \\ &= \frac{-2}{(p+P_R)^3} - (\frac{-2p}{p+\alpha P_{lim}})(\frac{1}{P_R+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^3 \\ &= (\frac{2p}{p+\alpha P_{lim}})(\frac{1}{P_R+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^3 - \frac{2}{(p+P_R)^3}\end{aligned}$$

$$
\begin{aligned}
\frac{\partial^2 \overline{\Delta}_{LLZ}}{\partial P_R^2}|_{P_R=P_{R,extrema,1}} &= (\frac{2p}{p+\alpha P_{lim}})(\frac{1}{P_{R,extrema,1}+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^3 - \frac{2}{(p+P_{R,extrema,1})^3} \\
&\overset{(a)}{=} \frac{2}{(p+P_{R,extrema,1})^2(P_{R,extrema,1}+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}})} - \frac{2}{(p+P_{R,extrema,1})^3} \\
&= \frac{2}{(p+P_{R,extrema,1})^2}[\frac{1}{P_{R,extrema,1}+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}}} - \frac{1}{p+P_{R,extrema,1}}] \\
&= \frac{2}{(p+P_{R,extrema,1})^2}[\frac{p(1-\frac{\alpha P_{lim}}{p+\alpha P_{lim}})}{(p+P_{R,extrema,1})(P_{R,extrema,1}+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}})}] \\
&= \frac{2}{(p+P_{R,extrema,1})^2}[\frac{p(\frac{p}{p+\alpha P_{lim}})}{(p+P_{R,extrema,1})(P_{R,extrema,1}+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}})}] \\
&> 0
\end{aligned}
$$

(a) $\frac{\partial \overline{\Delta}_{LLZ}}{\partial P_R} = 0$ at extrema : $\frac{1}{(p+P_{R,extrema,1})^2} = (\frac{p}{p+\alpha P_{lim}})(\frac{1}{P_{R,extrema,1}+\frac{p\alpha P_{lim}}{p+\alpha P_{lim}}})^2$

To sum up, $P_{R,extrema,1}$ is bigger than $-\frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}$, and it is a local minimum. □

A.5 Proof of The Second Extreme Point

We prove $\forall p, \alpha \in (0,1]\ P_{R,extrema,2} < \aleph = -\frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}$.

Proof.

$$
\begin{aligned}
\frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} + \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}}{-1-\sqrt{\frac{p}{p+\alpha P_{lim}}}} &< -\frac{p\alpha P_{lim}}{(p+\alpha P_{lim})} \\
\frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} + \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}}{1+\sqrt{\frac{p}{p+\alpha P_{lim}}}} &> \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})} \\
p\sqrt{\frac{p}{p+\alpha P_{lim}}} + \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})} &> \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}(1+\sqrt{\frac{p}{p+\alpha P_{lim}}}) \\
p\sqrt{\frac{p}{p+\alpha P_{lim}}} &> \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})}\sqrt{\frac{p}{p+\alpha P_{lim}}} \\
p &> \frac{p\alpha P_{lim}}{(p+\alpha P_{lim})} \\
p(p+\alpha P_{lim}) &> p\alpha P_{lim} \\
p &> 0
\end{aligned}
$$

□

A.6 Derivation of p_{min}

We derive the condition for $P_{R,extrema,1} \leq 0$.

Proof.

$$
\begin{aligned}
P_{R,extreme} &\leq 0 \\
\frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}{1-\sqrt{\frac{p}{p+\alpha P_{lim}}}} &\leq 0 \\
p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}} &\leq 0 \\
\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{\alpha P_{lim}}{p+\alpha P_{lim}} &\leq 0 \\
\sqrt{\frac{p}{p+\alpha P_{lim}}} &\leq \frac{\alpha P_{lim}}{p+\alpha P_{lim}} \\
\frac{p}{p+\alpha P_{lim}} &\leq \frac{(\alpha P_{lim})^2}{(p+\alpha P_{lim})^2} \\
p(p+\alpha P_{lim}) &\leq (\alpha P_{lim})^2 \\
p^2 + \alpha P_{lim} p - (\alpha P_{lim})^2 &\leq 0 \\
\big(p+\alpha P_{lim}(\frac{1-\sqrt{5}}{2})\big)\big(p+\alpha P_{lim}(\frac{1+\sqrt{5}}{2})\big) &\overset{(a)}{\leq} 0 \\
\big(p+\alpha P_{lim}(\frac{1-\sqrt{5}}{2})\big) &\leq 0 \\
p &\leq \alpha P_{lim}\frac{(\sqrt{5}-1)}{2}
\end{aligned}
$$

(a) $\forall p, \alpha, P_{lim} \in (0,1] \quad p + \alpha P_{lim}(\frac{1+\sqrt{5}}{2}) > 0$

□

A.7 Derivation of $p_{max,LLZ}$

We derive the condition for $P_{R,extrema,1} \leq P_{R,bound}$. Let's denote $m = \sqrt{\frac{p}{p+\alpha P_{lim}}}$. Note that $0 < m \leq 1$, $\forall p, \alpha, P_{lim} \in (0, 1]$.

Proof.

$$P_{R,extrema,1} \leq P_{R,bound}$$

$$\frac{p\sqrt{\frac{p}{p+\alpha P_{lim}}} - \frac{p\alpha P_{lim}}{p+\alpha P_{lim}}}{1 - \sqrt{\frac{p}{p+\alpha P_{lim}}}} \leq \frac{p(1-P_{lim})}{p+P_{lim}}$$

$$\frac{mp - m^2 P_{lim}\alpha}{1-m} \leq \frac{p(1-P_{lim})}{p+P_{lim}}$$

$$\frac{mp - m^2 P_{lim}\alpha}{1-m}(\frac{p+P_{lim}}{p}) \leq 1 - P_{lim}$$

$$(mp - m^2 P_{lim}\alpha)(p+P_{lim}) \leq (p - pP_{lim})(1-m)$$

$$(mp - \frac{P_{lim}\alpha p}{p+\alpha P_{lim}})(p+P_{lim}) \leq p - pP_{lim} + mpP_{lim} - mp$$

$$mp^2 + pP_{lim}m - \frac{P_{lim}\alpha p(p+P_{lim})}{p+\alpha P_{lim}} \leq p - pP_{lim} + mpP_{lim} - mp$$

$$m(p^2 + pP_{lim} - pP_{lim} + p) \leq \frac{P_{lim}\alpha p(p+P_{lim})}{p+\alpha P_{lim}} + p - pP_{lim}$$

$$m(p+1) \leq \frac{P_{lim}\alpha(p+P_{lim})}{p+\alpha P_{lim}} + 1 - P_{lim}$$

$$m(p+1) \leq \frac{P_{lim}\alpha p + \alpha P_{lim}^2 + p + \alpha P_{lim} - pP_{lim} - \alpha P_{lim}^2}{p+\alpha P_{lim}}$$

$$m(p+1) \leq \frac{P_{lim}\alpha p + p + \alpha P_{lim} - pP_{lim}}{p+\alpha P_{lim}}$$

$$m \leq \frac{P_{lim}\alpha p + p + \alpha P_{lim} - pP_{lim}}{(p+\alpha P_{lim})(p+1)}$$

$$m^2 \leq (\frac{P_{lim}\alpha p + p + \alpha P_{lim} - pP_{lim}}{(p+\alpha P_{lim})(p+1)})^2$$

$$\frac{p}{p+\alpha P_{lim}} \leq (\frac{P_{lim}\alpha p + p + \alpha P_{lim} - pP_{lim}}{(p+\alpha P_{lim})(p+1)})^2$$

$$p(p+\alpha P_{lim})(p+1)^2 \leq [P_{lim}\alpha(p+1) + p(1-P_{lim})]^2$$

$$
\begin{aligned}
(p^2+\alpha p P_{lim})(p+1)^2 &\leq \Big[P_{lim}^2\alpha^2(p+1)^2 + p^2(1-P_{lim})^2 \\
&\quad + 2P_{lim}\alpha(p+1)p(1-P_{lim})\Big] \\
(p^2+\alpha p P_{lim})(p^2+2p+1) &\leq \Big[P_{lim}^2\alpha^2(p^2+2p+1) \\
&\quad + p^2(1-2P_{lim}+P_{lim}^2) \\
&\quad + 2P_{lim}\alpha(p^2(1-P_{lim})+p(1-P_{lim})\Big] \\
\Big[p^4+p^3(2+\alpha P_{lim})+p^2(1+2\alpha P_{lim})+p(\alpha P_{lim})\Big] &\leq p^2\Big[\alpha^2P_{lim}^2+1-2P_{lim}+P_{lim}^2 \\
&\quad + 2\alpha P_{lim} - 2\alpha P_{lim}^2\Big] + p\Big[2\alpha^2P_{lim}^2 \\
&\quad + 2\alpha P_{lim} - 2\alpha P_{lim}^2\Big] + \Big[\alpha^2P_{lim}^2\Big] \\
p^4 + p^3\Big[2+\alpha P_{lim}\Big] + p^2\Big[-\alpha^2P_{lim}^2 &\leq 0 \\
+2P_{lim} - P_{lim}^2 + 2\alpha P_{lim}^2\Big] & \\
+p\Big[-2\alpha^2P_{lim}^2 - \alpha P_{lim} + 2\alpha P_{lim}^2) - \Big[\alpha^2P_{lim}^2\Big] & \\
p^4+p^3(2+\alpha P_{lim})+p^2[2P_{lim}-P_{lim}^2(1-\alpha)^2]+p(2\alpha P_{lim}^2(1-\alpha)-\alpha P_{lim})-(\alpha^2P_{lim}^2) &\leq 0
\end{aligned}
$$

□

A.8 Proof of The Positive Root of $f(\cdot)$ Function

We prove that there is only one positive root of the $f(\cdot)$ and it is between 0 and 1.

$$f(p) = p^4+p^3(2+\alpha P_{lim})+p^2[2P_{lim}-P_{lim}^2(1-\alpha)^2]+p(2\alpha P_{lim}^2(1-\alpha)-\alpha P_{lim})-(\alpha^2P_{lim}^2)$$

Proof. Considering the coefficients of $f(p)$, the one with the lowest degree $(-\alpha^2P_{lim}^2)$ is always negative, while $(2\alpha P_{lim}^2(1-\alpha)-\alpha P_{lim})$ can be either positive or negative depending on the parameters α and P_{lim}. The other three are always positive. Hence, there is only one sign change. In this condition, Descartes's Rule of Signs states there is a single positive root, denoted as $p_{max,LLZ}$. To show $p_{max,LLZ} \in (0,1)$:

$$
\begin{aligned}
f(0^+) &= -\alpha^2P_{lim}^2 < 0 \\
f(1) &= 1+(2+\alpha P_{lim})+(2P_{lim}-P_{lim}^2(1-\alpha)^2)+(2\alpha P_{lim}^2(1-\alpha)-\alpha P_{lim})-(\alpha^2P_{lim}^2) \\
&= 3+2P_{lim}+P_{lim}^2(-1+4\alpha-4\alpha^2) > 0
\end{aligned}
$$

Via Intermediate Value Theorem, $\exists p_{max,LLZ} \in (0,1]$ such that $f(p_{max,LLZ}) = 0$.

$\therefore$ There is solely one positive root and it is between 0 and 1.

□

A.9 Proof of The Third Extreme Point

We prove that $P_{R,extrema,3}$ is a local maximum via the second derivative test.

Proof.

$$\begin{aligned}\frac{\partial^2\overline{\Delta}_{TLZ}}{\partial P_R^2} &= \frac{\partial[\frac{1}{(p+P_R)^2} - (\frac{1}{1-\alpha})\frac{1}{(P_R+\frac{\alpha}{1-\alpha})^2}]}{\partial P_R} \\ &= \frac{-2}{(p+P_R)^3} - (\frac{-2}{1-\alpha})(\frac{1}{P_R+\frac{\alpha}{1-\alpha}})^3 \\ &= (\frac{2}{1-\alpha})(\frac{1}{P_R+\frac{\alpha}{1-\alpha}})^3 - \frac{2}{(p+P_R)^3}\end{aligned}$$

Employing second derivative test:

$$\begin{aligned}\frac{\partial^2\overline{\Delta}_{TLZ}}{\partial P_R^2}|_{P_R=P_{R,extrema,3}} &= (\frac{2}{1-\alpha})(\frac{1}{P_{R,extrema,3}+\frac{\alpha}{1-\alpha}})^3 - \frac{2}{(p+P_{R,extrema,3})^3} \\ &\overset{(a)}{=} \frac{2}{(p+P_{R,extrema,3})^2(P_{R,extrema,3}+\frac{\alpha}{1-\alpha})} - \frac{2}{(p+P_{R,extrema,3})^3} \\ &= \frac{2}{(p+P_{R,extrema,3})^2}[\frac{1}{P_{R,extrema,3}+\frac{\alpha}{1-\alpha}} - \frac{1}{p+P_{R,extrema,3}}] \\ &= \frac{2}{(p+P_{R,extrema,3})^2}[\frac{p-\frac{\alpha}{1-\alpha}}{(p+P_{R,extrema,3})(P_{R,extrema,3}+\frac{\alpha}{1-\alpha})}] \\ &\overset{(b)}{<} 0\end{aligned}$$

(a) $\frac{\partial\overline{\Delta}_{TLZ}}{\partial P_R} = 0$ at extrema : $\frac{1}{(p+P_{R,extrema,3})^2} = (\frac{1}{1-\alpha})(\frac{1}{P_{R,extrema,3}+\frac{\alpha}{1-\alpha}})^2$

(b) $p < \frac{\alpha}{1-\alpha}$ in the area of interest

To sum up, $P_{R,extrema,3}$ is a local maxima.

□

Appendix B

List of Abbreviations

AoI	Age of Information
AoP	Age of Processing
CDF	Cumulative Distribution Function
CMOP	Constrained Multi-objective Optimization Problem
DP	Dynamic Programming
EH	Energy Harvesting
IoMT	Internet of Medical Things
IoT	Internet of Things
LLZ	Local Limited Zone
MCC	Mobile Cloud Computing
MDP	Markov Decision Process
PMF	Probability Mass Function
RAT	Radio Access Technology
TLZ	Total Limited Zone
QoE	Quality of Experience
QoS	Quality of Service

Bibliography

[1] M. Rothmuller and S. Barker, "IoT–The Internet of Transformation 2020," Jupiter Research, White Paper, April 2020.

[2] Z. Ling, F. Hu, H. Zhang, and Z. Han, "Age-of-information minimization in healthcare iot using distributionally robust optimization," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 154–16 167, 2022.

[3] G. Dimitrakopoulos and P. Demestichas, "Intelligent transportation systems," *IEEE Vehicular Technology Magazine*, vol. 5, no. 1, pp. 77–84, 2010.

[4] Q. Abbas, S. Zeb, S. A. Hassan, R. Mumtaz, and S. A. R. Zaidi, "Joint optimization of age of information and energy efficiency in iot networks," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020.

[5] Q. Wang, H. Chen, Y. Li, Z. Pang, and B. Vucetic, "Minimizing age of information for real-time monitoring in resource-constrained industrial iot networks," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019.

[6] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.

[7] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, Feb. 2019.

[8] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[9] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[10] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, 2019.

[11] Z. Ning, P. Dong, X. Wang, X. Hu, L. Guo, B. Hu, Y. Guo, T. Qiu, and R. Y. K. Kwok, "Mobile edge computing enabled 5g health monitoring for internet of medical things: A decentralized game theoretic approach," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 463–478, 2021.

[12] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. K. Kwok, "Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2212–2225, 2021.

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *CoRR*, vol. abs/1701.01090, 2017.

[14] T. Zheng, J. Wan, J. Zhang, C. Jiang, and G. Jia, "A survey of computation offloading in edge computing," in *2020 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2020.

[15] S. Zeadally, J. Guerrero, and J. Contreras, "A tutorial survey on vehicle-to-vehicle communications," *Telecommunication Systems*, vol. 73, no. 3, p. 469–489, Dec. 2019.

[16] A. Alalewi, I. Dayoub, and S. Cherkaoui, "On 5g-v2x use cases and enabling technologies: A comprehensive survey," *IEEE Access*, vol. 9, pp. 107 710–107 737, 2021.

[17] A. Shahraki, A. Taherkordi, O. Haugen, and F. Eliassen, "A survey and future directions on clustering: From wsns to iot and modern networking paradigms," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, p. 2242–2274, Jun. 2021.

[18] J. M. Juran and A. B. Godfrey, *Juran's quality handbook*, ser. Juran's quality handbook, 5th Edition. McGraw Hill, 1999.

[19] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2525–2553, 2019.

[20] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5g networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.

[21] C. Li, Y. Zhang, Q. Sun, and Y. Luo, "Collaborative caching strategy based on optimization of latency and energy consumption in mec," *Knowledge-Based Systems*, vol. 233, p. 107523, 2021.

[22] S. Zhang, L. Wang, H. Luo, X. Ma, and S. Zhou, "Aoi-delay tradeoff in mobile edge caching with freshness-aware content refreshing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5329–5342, 2021.

[23] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh, "Fresh caching for dynamic content," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021.

[24] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[25] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput*, vol. 6, pp. 4–23, 01 2009.

[26] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[27] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Stateful versus stateless selection of edge or cloud servers under latency constraints," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2022.

[28] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[29] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 2992–3005, 2021.

[30] X. Zhao, M. Hosseinzadeh, N. Hudson, H. Khamfroush, and D. E. Lucani, "Improving the accuracy-latency trade-off of edge-cloud computation offloading for deep learning services," in *2020 IEEE Globecom Workshops*, 2020.

[31] A. Acheampong, Y. Zhang, X. Xu, and D. Kumah, "A review of the current task offloading algorithms, strategies and approach in edge computing systems," *Computer Modeling in Engineering & Sciences*, vol. 134, pp. 1–54, 01 2022.

[32] N. Li, S. Yang, Z. Wang, W. Hao, and Y. Zhu, "Multi-tier mec offloading strategy based on dynamic channel characteristics," *IET Communications*, vol. 14, no. 22, p. 4029–4037, Dec. 2020.

[33] Q. Tang, H. Lyu, G. Han, J. Wang, and K. Wang, "Partial offloading strategy for mobile edge computing considering mixed overhead of time and energy," *Neural Computing and Applications*, vol. 32, no. 19, p. 15383–15397, Aug. 2019.

[34] A. Ndikumana, K. K. Nguyen, and M. Cheriet, "Age of processing-based data offloading for autonomous vehicles in multirats open ran," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 450–21 464, 2022.

[35] L. Badia, M. Miozzo, M. Rossi, and M. Zorzi, "Routing schemes in heterogeneous wireless networks based on access advertisement and backward utilities for qos support [quality of service based routing algorithms for heterogeneous networks]," *IEEE Communications Magazine*, vol. 45, no. 2, pp. 67–73, 2007.

[36] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[37] Y. Liao, L. Shou, Q. Yu, Q. Ai, and Q. Liu, "Joint offloading decision and resource allocation for mobile edge computing enabled networks," *Computer Communications*, vol. 154, pp. 361–369, 2020.

[38] M. W. Baidas, "Resource allocation for offloading-efficiency maximization in clustered noma-enabled mobile edge computing networks," *Computer Networks*, vol. 189, p. 107919, 2021.

[39] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[40] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015.

[41] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, 2012.

[42] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[43] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," 2017.

[44] R. Li, Q. Ma, J. Gong, Z. Zhou, and X. Chen, "Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time iot applications," *IEEE Internet of Things Journal*, vol. 8, no. 19, pp. 14 471–14 484, 2021.

[45] Z. Tang, Z. Sun, N. Yang, and X. Zhou, "Age of information analysis of multi-user mobile edge computing systems," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021.

[46] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Analysis on computation-intensive status update in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4353–4366, 2020.

[47] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Data quality-aware task offloading in mobile edge computing: An optimal stopping theory approach," *Future Generation Computer Systems*, vol. 117, pp. 462–479, 2021.

[48] X. Huang, R. Yu, J. Kang, and Y. Zhang, "Distributed reputation management for secure and efficient vehicular edge computing and networks," *IEEE Access*, vol. 5, pp. 25 408–25 420, 2017.

[49] R. Talak, I. Kadota, S. Karaman, and E. Modiano, "Scheduling policies for age minimization in wireless networks with unknown channel state," in *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018.

[50] X. Wu, X. Li, J. Li, P. Ching, and H. V. Poor, "Deep reinforcement learning for lot networks: Age of information and energy cost tradeoff," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020.

[51] Y. Zhu, W. Zhang, Y. Lin, Y.-H. Lo, and Y. Zhang, "Improving age of information in large-scale energy harvesting networks," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2023.

[52] J. Sun, L. Wang, Z. Jiang, S. Zhou, and Z. Niu, "Age-optimal scheduling for heterogeneous traffic with timely throughput constraints," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1485–1498, 2021.

[53] A. Munari, T. De Cola, and L. Badia, "Local or edge/cloud processing for data freshness," in *IEEE Globecom*, December 2023.

[54] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & Systems (2nd Ed.)*. USA: Prentice-Hall, Inc., 1996.

[55] J.-B. Seo and J. Choi, "On the outage probability of peak age-of-information for d/g/1 queuing systems," *IEEE Communications Letters*, vol. 23, no. 6, pp. 1021–1024, 2019.

[56] T. Z. H. Ernest and A. S. Madhukumar, "Age of information outage probability analysis for computation offloading in iiot networks," *IEEE Communications Letters*, vol. 27, no. 9, pp. 2471–2475, 2023.

[57] G.-Y. Lin, Y.-C. Huang, and Y.-P. Hsu, "Outage analysis of age-of-information for multi-source systems," 2022.

[58] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[59] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2005, vol. I.

[60] A. Islam, "A survey on task offloading in multi-access edge computing," *Journal of Systems Architecture*, vol. 118, 06 2021.