UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Strategic insights and simulation analysis in coalitional game theory for edge computing coinvestment

Informe de Proyecto de Grado presentado por

Santiago Tabarez Rama

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Eduardo Grampin

Montevideo, July 29, 2025

Agradecimientos

A mi familia, que siempre me apoyó de todas las formas posibles para terminar la carrera, que fue comprensiva y me dio ánimos en momentos de frustración, y especialmente a mi padre por su consejo: "dale una oportunidad al mundo académico, que puede ser lo tuyo".

A mis amigos, que bancaron mis ausencias en estos últimos años porque "está con cosas de la facultad".

A mi tutor Eduardo, que me dio total libertad para desarrollar esta tesis, tuvo mucha paciencia con mis prórrogas y me presentó oportunidades para seguir formándome profesionalmente.

Al equipo que desarrolló el artículo académico en el que se basa esta tesis, por su disposición absoluta a tener todas las reuniones necesarias para aclarar mis dudas, escuchar mis ideas, y por ofrecerme la oportunidad de continuar mi formación profesional con ellos.

Abstract

This thesis explores the application of cooperative game theory to optimize resource allocation by conducting an in-depth analysis of the academic article "Coalitional Game Theoretical Approach to Coinvestment with Application to Edge Computing," focusing on identifying and validating the theoretical properties of the proposed model, exploring implications for software implementation, and proposing extensions to improve its applicability and effectiveness. The referenced article studies how different stakeholders, specifically a Network Owner (who controls the infrastructure) and multiple Service Providers (who use this infrastructure to deliver services), can jointly invest in shared resources, such as computational capacity in Edge Computing. The article proposes a cooperative game theoretical model that determines how stakeholders should allocate resources optimally, share investment costs, and fairly distribute revenues among themselves based on each stakeholder's contribution. The main analysis is structured in two stages. In the first stage, we examine the original model and identify possible simplifications that significantly reduce its computational complexity, transforming it from a non-deterministic exponential time problem into one deterministic and solvable in linear time. These simplifications will preserve the exact results. Additionally, we highlight constraints that were overlooked in the original formulation, enhancing the theoretical accuracy of the model. Furthermore, we study the risks arising from inaccuracies or incorrect estimates in critical parameters, offering simplified analytical equations that stakeholders can use to quantify and evaluate the potential impact of these errors on their expected outcomes. In the second stage, we propose alternative utility functions specifically designed to address the limitations identified in the original model, particularly its reduced applicability to realistic scenarios. These alternative functions are systematically studied and compared to evaluate their implications for resource allocation and incentive compatibility. Additionally, we introduce model extensions that optimize resource allocation under more realistic conditions; however, these enhancements result in increased computational complexity. Both main stages of this thesis follow a structured approach composed of two complementary parts. The first part is theoretical, involving an analytical study of the model, its parameters, and the derived equations. The second part is practical, providing empirical evidence through numerical simulations and sensitivity analyses. These simulations illustrate theoretical insights and help to evaluate the effects of varying utility function parameters. For the purpose of conducting a systematic study, the model is implemented and the results of the executions are stored and analyzed with standard tools; the programming language is Python, the database is MySQL, and the business intelligence tool is Metabase. This combination of theory and empirical validation clearly demonstrates the practical relevance of game theoretical models and provides a robust framework for assessing parameter impacts in real-world scenarios.

Key words: Edge Computing, Coinvestment, Cooperative (Coalitional) Game Theory, Grand Coalition, Core of The Game, Core Stability, Shapley Value, Convexity, Marginal Contribution, Strategy-Proofness, Risk Analysis, Simulation, Sensitivity Analysis, Prosumers.

Contents

Chapter 1

Introduction

The aim of this thesis is to study, validate and extend the model proposed in the academic article (Rosario Patanè, Chahed, Kiedanski, & Kofman, 2023). This research is motivated by the significant financial challenges associated with deploying computational infrastructure, particularly in contexts such as Edge Computing (EC). Deploying computational resources at edge locations is typically costly and often exceeds the capacity of a single entity. To address this, the article proposes a cooperative game theoretical model in which multiple actors collaborate to invest in shared infrastructure. Specifically, the scenario involves two distinct types of stakeholders: a single entity that owns and operates the physical resources, called the Network Owner (NO) and multiple Service Providers (SPs) that use this infrastructure to deliver services. Since we are adopting a game-theory approach, in future sections, we may refer to the previously named actors as players and the process of assigning resources and defining payments and revenues as a game. As a general principle throughout this thesis, any modifications introduced to address the limitations of the original model will be as minimal as possible, preserving the integrity of the core framework.

1.1 Problem Definition

The Network Owner is responsible for deploying, managing, and allocating the shared computational resources. Service Providers utilize these resources to generate revenue by offering services to their customers. SPs may deliver different types of services: some may provide real-time services that require strict timing constraints and thus drive investment towards high-quality resources, while others offer regular (non-real-time) services. SPs can also be classified on the basis of their load profile. For example, a business collaboration platform provider would experience the highest traffic during office hours, as it primarily serves workplaces and professionals. In contrast, a real-time gaming service provider would exhibit a more regular traffic pattern throughout the day, with a peak in the evening. These variations in traffic profiles influence how resources are allocated and optimized within the shared infrastructure.

The model proposed a set of time-slots with a default value of 96 units to discretize the load and utility functions. Traffic generation for Service Providers follows realistic daily patterns, as specified in the article (A. P. Vela A. Vía & Velasco, 2016). Specifically, the model uses a sinusoidal function to represent typical daily fluctuations in user demand, accurately reflecting real-world scenarios. By adjusting the function's parameters (such as average load and a set of amplitude and offset values), different types of Service Providers can be modeled based on how their traffic load fluctuates throughout the day. It is important to note that the model does not account for variations throughout the investment period or interday fluctuations beyond the predefined daily pattern. This load for a given player i in a time-slot t is defined as:

$$l_i^t = a_0 + \sum_{k=1}^{K} a_k \sin\left(2k\pi \frac{t - t_k}{T}\right)$$

where the variables are defined as follows:

- a_0: Average load .
- a_k: Set of k values representing the amplitude of each sinusoidal component, indicating the magnitude of the corresponding harmonic in the overall traffic pattern.
- t_k Set of k values representing the phase offsets, which determine the timing (or shift) of each sine wave.
- T is the number of daily time-slots.

The load is used as input within two other parameters; β indicates the benefit factor, which quantifies the revenue generated per unit of load, while ξ indicates the diminishing return effect, measuring the rate at which additional resource allocation yields progressively smaller utility improvements. The utility function for a given Service Provider i in the time-slot t is modeled as follows:

$$u_i(l_i^t, h^i) = \beta_i \cdot l_i^t \cdot (1 - e^{-\xi h^i})$$

where the variables are defined as follows:

- u_i : The utility produced by player i in time-slot t given the load l_i^t in that time-slot and the fixed allocated resources h^i.
- l_i^t : The number of requests expected for SP i in time-slot t.
- h^i : Denotes the amount of resources allocated to the SP i, measured in millicores.
- β_i : The benefit factor, indicating how each unit of load is converted into monetary units.
- ξ : Models the diminishing returns effect. Quantifies how quickly the benefits of extra resources taper off; a higher value of ξ implies that a small increase in resources produces a significant initial improvement, after which performance saturates rapidly.

Note that when allocation is zero, the utility is also zero, and increasing the allocation increases the utility, but in a sub-linear manner.

The simulation of a game under this model involves three critical steps:

- Step 1: Maximizing the grand coalition's total value: Determining optimal resource allocation among all SPs to maximize combined benefits, accounting for both the revenues from user demand and the associated infrastructure costs. This involves the optimization problem of maximizing the value of the Grand Coalition that can be expressed by the following equations:

$$v(S) = \max_{\vec{h},C} v^{\vec{h},C,S} \triangleq \max_{\vec{h},C} D \sum_{i \in S} \sum_{t=1}^{T} u^i(l_t^i, h^i) - d \cdot C \tag{1.1}$$

$$\text{s.t.} \sum_{i \in S \setminus \{\text{NO}\}} h^i = C; \quad h^{\text{NO}} = 0. \tag{1.2}$$

$$C,\, h^i \geq 0, \quad \forall t \in [T], \quad \forall i \in S. \tag{1.3}$$

Where the variables are defined as follows:

- $\vec{h}$: Allocation vector, defining the optimal allocation for each SP.
- D : The duration of the investment in days; the default value is 3 years.
- d : The CPU allocation price, measured in dollars per millicore.
- C : The sum of all the service provider's allocation h.
- N : The coalition with all players, also known as the Grand Coalition.

- Step 2. Calculating each player's payoff. To ensure a fair distribution of the total benefits generated by a coalition, each player's payoff is determined using their Shapley value. This value assigns a payoff based on the player's marginal contribution to all possible coalitions. While a detailed explanation of the concept will be provided in future sections, for now, we present the equation used to compute its value:

$$\phi_i = \frac{1}{|N|!} \sum_{S \subseteq N \setminus \{i\}} |S|! \cdot (|N| - |S| - 1)! \cdot \Delta_i(S)$$

where the variables are defined as follows:

- ϕ_i: The Shapley value for the player i, representing the fair payoff assigned to the player i from the total value generated by the coalition.
- N: The set of all players in the game.
- $S \subseteq N \setminus \{i\}$: A subset of players that does not include the player i. The sum is taken over all such subsets.
- $(|N|-|S|-1)!$: The factorial of the number of players not in S and excluding i, representing the number of ways to order the remaining players.
- $\Delta_i(S)$: The marginal contribution of the player i to coalition S, defined by

$$\Delta_i(S) = v(S \cup \{i\}) - v(S),$$

where $v(\cdot)$ is the value function of a coalition.

Note that this approach requires evaluating the value of every possible coalition as defined in Step 1. Since coalitions without the Network Owner have a value of zero because NO is needed for hardware deployment, the maximization problem in Step 1 must be executed only for those coalitions that include the NO. However, for a set of n players, this still requires performing 2^{n-1} evaluations, making this step computationally intensive and infeasible as n grows large.

- Step 3. Calculating each player's payments and revenues. This step consists in establishing fair financial exchanges between stakeholders by determining how much each player should pay or receive from others. The goal is to ensure that each player's payoff, defined as revenues minus payments, coincides with their Shapley value. This guarantees a fair and stable distribution of benefits. In mathematical terms, this corresponds to solving the following system:

$$\begin{aligned} & r_i - p_i = x_i, \quad \forall i \in N \\ \text{subject to} \quad & \sum_{i \in N} r_i = D \cdot \sum_{i \in N} \sum_{t=1}^{T} u_i(l_i^t, h^{i*}) \\ \text{where:} \quad & h^*, C^* = \arg\max_{\hat{h}, C} v^i(\hat{h}, C, N) \end{aligned}$$

where the variables are defined as follows:

- x_i: The payoff of the player i, which is equal to its Shapley value (ϕ_i).
- p_i: The payment of the player i. A positive p_i indicates that the player pays to others, while a negative p_i indicates that the player receives funds from others.
- r_i: The revenue collected by the player i for the services rendered.

Note that this framework leads to a system of $n + 1$ equations with $2n$ variables, which admits multiple possible solutions. This is not a limitation, as all solutions yield the same payoff for each player, and we can choose the one that minimizes the required payments to achieve these payoffs.

In summary, the proposed solution effectively addresses the co-investment challenge by combining realistic load modeling, an economically grounded utility function, and a structured game-theoretical framework. It optimizes the total value of the grand coalition, assigns individual contributions via the Shapley Value, and defines fair mechanisms for payments and revenues. This ensures that one Network Owner and multiple Service Providers share costs and benefits equitably. The model captures variability in traffic while offering a scalable and stable approach to fair resource allocation as the number of stakeholders grows.

1.2 Research Aims and Expected Outcomes

This research aims to analyze and validate the proposed model for addressing the Edge Computing problem, identify its limitations, and propose modifications to overcome them while preserving the core principles of the original model.

In a first stage, we examine the model from two different perspectives. First, we analyze the mathematical equations underlying each one of the previously defined steps, employing both analytical methods and visualizations to evaluate their behavior. Second, we assess the strategic implications for stakeholders, for example, investigating whether they have incentives to misreport their parameters and also evaluating the risks associated with their parameters' misestimations. The primary aim of this stage is to validate the original model and identify its weaknesses. In a second stage, we build upon these insights by introducing enhancements that increase the model's complexity in order to better adapt it to real-world scenarios.

The Edge Computing problem belongs to a broader category of academic literature that explores mechanisms for resource exchange among actors who simultaneously produce and consume a specific good or service, often referred to as prosumers. Most of these related studies focus on local and decentralized energy markets. For example, how households equipped with batteries and/or electricity generation systems (typically solar panels) can trade energy locally, thereby reducing their overall energy costs compared to relying solely on their own production and storage capacities.

Although the Edge Computing problem differs fundamentally in its characteristics, SPs can also be considered prosumers, as the amount of hardware allocated could be seen as the resources they produce and the hardware they need could be seen as the resources they consume. To maximize profit, SPs could exchange resource allocations when actual usage parameters deviate from those used in the initial allocation. Since the original allocation remains fixed throughout the investment period and across all time-slots, a natural improvement is to enable dynamic exchanges of resources. In the second stage, where per time-slot allocation is introduced, such exchanges could be beneficial when SPs face parameter values different from those assumed initially. For instance, in a given time-slot, one SP might anticipate underutilization of part of its allocated resources based on historical load data, while another expects higher demand or increased benefit. In this case, the first SP can sell part of its unused allocation to the second, generating mutual gains.

The expected outcome of this thesis is to develop a deeper understanding of the proposed model by validating it, but at the same time identifying simplifications and restrictions that were originally overlooked, as well as aspects where the model shows limited alignment with real-world scenarios. Building on these insights, we aim to introduce the minimal set of modifications required to improve its practical relevance, while preserving the core principles and structure of the original formulation. Additionally, we aim to develop a software tool enabling us to run simulations, store results in a database, visualize data, and execute both the original model as stated in the reference academic article and the alternative proposed modifications.

Overview of Conclusions

- We identified several simplifications of the original model, significantly reducing analytical complexity and computational requirements. These simplifications enabled simulations to run in linear rather than exponential time, greatly enhancing practical feasibility. They also allowed for a more comprehensive and straightforward analysis of the model's core equations.

- Constraints such as the requirement for positive allocation, along with limitations of the original model, were identified, especially conditions under which its assumptions did not hold or resulted in unrealistic outcomes. To overcome these limitations, we proposed alternative utility function formulations and systematically compared their implications for resource allocation and incentive compatibility.

- We gained insights into the internal dynamics of the model, particularly regarding the role and interpretation of each variable. The diminishing return parameter was notably clarified, and we showed that it can be related to measurable real-world parameters, helping to explain its influence on optimal resource allocation and utility.

- We conducted a risk analysis focusing on the consequences of misestimating key parameters, providing equations that allow service providers to evaluate potential losses and support more informal strategic decisions.

- By introducing additional complexity into the model, particularly allowing service providers to have varying allocations across different time-slots, we achieved a more realistic representation of edge computing scenarios. Although these enhancements invalidated the initial simplifications, the simplified model remains valuable as a clearly defined worst-case scenario for comparative analysis, providing a conservative lower bound on service providers' utility under more complex scenarios.

- We proposed three alternative utility functions based on modifications of the exponential saturation structure of the original utility function. By varying parameters within the exponential term, we obtained distinct utility equations exhibiting different behaviors. We then analyzed these variations and compared their trade-offs in terms of economic interpretation, resource allocation, and strategic implications.

- We developed a software tool to support simulation analyses, including scenario configuration, execution, data storage, and visualization. This tool facilitated the comparison of various model formulations and provided deeper insight into their behaviors.

- We found that both the original model and our modified versions are not strategy-proof, identifying a single parameter, the benefit factor, as the source of this limitation. To mitigate this, we proposed mechanisms such as a secondary market to allow service providers to trade allocations, and introduced priority based differentiation in internet services linked to declared values. These mechanisms create strategic trade-offs that encourage truthful declarations.

- We reviewed academic articles provided within the thesis to identify concepts that could inspire future iterations of our model.

- We found that the original diminishing return parameter implicitly combined two distinct real-world aspects: the fraction of requests served at the edge, and the hardware resources required per request. Because coupling these two characteristics restricts the diversity of achievable SPs profiles, we proposed using two separate parameters instead.

- We outlined potential directions for future work. As discussed in the previous point, one possibility is to redefine the utility function using two separate parameters to decouple the fraction of requests served at the edge from the hardware required to process them. Additionally, some of the ideas and mechanisms presented in the reviewed academic articles, such as secondary markets, ramp constraints, or dynamic coordination mechanisms, can also be adapted and incorporated into future versions of the model. Whether to implement the proposed separation of parameters or adopt one of the other modified utility functions presented in this work is a design decision

that should be made in coordination with the rest of the research team. Once this decision is made, the next step would be to proceed with the formal definition and analysis of a selected new utility function.

1.3 Document Structure

This document is structured into seven chapters and an Appendix with five extra chapters.

- Chapter 1: Introduction
  - We motivate the work, state and define the problem, set the main objectives of the investigation, establish the expected outcomes, and give a summary of the conclusions.
- Chapter 2: Background Review and Related Work
  - Background Review: We identify existing research and established real-world cases related to cooperative resource sharing, particularly within edge computing contexts. We examine previous academic research, highlighting its parallels and implications for Edge Computing scenarios. 2.2.1.
  - Prosumers Academic Articles: This thesis proposal included several relevant academic articles specifically addressing various aspects of the prosumer problem. In this section, we summarize key insights and ideas from these sources that have potential relevance for our research. Here, we present a concise overview of these insights without explicitly mentioning each individual article 2.2.3. For a comprehensive review of each of these articles and detailed commentary on their connection to Edge Computing, please refer to Appendix Chapter A.
- Chapter 3: Analytical Investigation of the Originally Proposed Edge-Computing Model
  - As a consequence of the independent contribution property, we identify simplifications that significantly reduce both the model complexity and computational demands 3.
  - Once this simplified yet equivalent version of the model is obtained, we perform a detailed analysis of its fundamental equations 3.8.
  - We explore the strategic implications of the model, particularly focusing on risk assessment and strategy-proofness 3.11.
  - We outline the limitations of the previously mentioned simplifications by identifying the conditions under which these simplifications no longer hold. We also discuss scenarios in which our simplified model represents a worst-case scenario 3.16.
- Chapter 4: Empirical Illustration and Results Analysis of the Originally Proposed Edge-Computing Model
  - We review and discuss the simulation results presented in the referenced academic article 4.2.
  - We perform our own simulations, focusing on performing a sensitivity analysis of each parameter involved in the utility and resource allocation functions 4.3.
- Chapter 5: Analytical Investigation of the Proposed Modifications to the Edge-Computing Model
  - We propose modifications to the net utility function to address the previously detected limitations and analyze its consequences 5.2.
  - We study strategy-proofness for the proposed modified utility functions 5.8.
  - We extend the model by adding synergies and externalities that better reflect real-world scenarios and disrupt the independent contribution among Service Providers 5.9.
  - We study how the Shapley value changes when more than one Network Owner is considered in the coinvestment 5.10.

- Chapter 6: Empirical Illustration and Results Analysis of the Proposed Modifications to the Edge-Computing Model
  - We provide some numerical examples regarding strategy-proofness 6.4
  - We perform a sensitivity analysis for each variable of each one of the proposed modified models and compare them 6.3.
  - We perform more general simulations focusing on the effects of the interdependent contributions among Service Providers 6.5
- Chapter 7: Developed Software Description
  - We describe the software developed to run simulations and provide a guide on how to execute it.
- Chapter 8: Conclusions and Future Work
  - Describe our conclusions and suggest future interaction of the model.
- Appendix A: Related Academic Articles
  - This chapter reviews the academic articles provided as part of the thesis proposal. Although these works address diverse problems, they share a common conceptual foundation: the idea of prosumers, entities that both produce and consume resources. Most of these articles focus on peer-to-peer (P2P) energy markets, where households equipped with energy storage and generation capabilities trade electricity locally. We briefly summarize each article, highlighting ideas or mechanisms relevant to the Edge Computing coinvestment model, thereby placing it within the broader context of prosumer-based cooperative systems.
- Appendix B: Demonstrations
  - In this chapter, we include the detailed mathematical demonstrations that were excluded from the main body of the thesis to maintain a smooth and uninterrupted flow of the core arguments. When a demonstration is particularly short or directly relevant to the understanding of a section, it is kept within the main text. However, more technical or less central derivations are placed in the appendix to make the document easier to read without omitting the necessary rigor.
- Appendix C: Additional Considerations
  - This chapter expands on several ideas that, while relevant to our study, do not belong to the core of the proposed model. Rather, they represent general considerations, extensions, observations, and consequences that emerge from it. We explore how some of the identified properties may influence the model's behavior in real-world settings and discuss the extent to which the model is suited to practical applications.
- Appendix D: Software Manual
  - This chapter is dedicated to the user and developer manual of the simulation software developed during this thesis. The manual explains how to install, configure, and use the tool, how to modify it if needed, and how to interpret the outputs. This allows other researchers or stakeholders to replicate the results, explore new scenarios, and extend the analysis in a consistent and efficient way.
- Appendix E: Variable definitions
  - We provide a brief definition for each variable used throughout this thesis.

Chapter 2

Background Review and Related Work

2.1 Background Review

Although we have previously defined the problem addressed by the referenced academic article, we assume that the reader has read and understood its main ideas and outcomes. Furthermore, throughout this document, each core concept or theoretical notion is explicitly defined at the point where it is introduced and applied. This choice aims to facilitate readability and ensure that definitions remain closely connected to their practical implementation.

2.2 Related Work

The referenced article explicitly states that:

> to the best of our knowledge we are the first to apply it for co-investment in EC, and thus there are no other works with which we can compare our proposal

The main difference in the approach is the application of cooperative game theory, specifically coalition formation and Shapley value payoff allocation, to analyze joint investment decisions among independent entities in Edge Computing environments. Prior to this work, such cooperative game theoretical analysis had not been applied in the specific context of resource co-investment for Edge Computing infrastructures, making their proposal fundamentally different from other existing research.

2.2.1 Real-World Cases and Examples

Several real-world cases provide practical insights into coinvestment strategies in edge computing, illustrating both opportunities and challenges in joint investments.

In telecom cloud collaborations, telecom operators such as AT&T, Verizon, and SK Telecom have partnered with cloud providers like AWS, through its Wavelength initiative, and Microsoft with Azure Edge Zones, to jointly deploy edge computing infrastructure. Such initiatives distribute the significant capital expenditures associated with deploying edge nodes, aligning with the coinvestment framework examined in this thesis.

Additionally, recent deployments of Multi-access Edge Computing (MEC) integrated with small cell networks have been driven by joint investments from telecom operators. These collaborative efforts

are primarily targeted at supporting low-latency applications, including augmented reality, real-time gaming, and autonomous driving assistance, which require extensive distributed computing resources.

In the automotive industry, manufacturers such as Tesla and Renault, alongside telecom providers, are exploring shared investments in roadside edge infrastructure. These nodes play a critical role in facilitating vehicle-to-everything (V2X) communications, enhancing safety, and reducing latency for real-time vehicular applications.

Smart city initiatives have similarly benefited from joint investments between municipalities, telecom operators, and technology companies. These collaborations help develop urban edge computing facilities that support various applications, including IoT deployments, surveillance, traffic management, and public safety systems.

Moreover, the analogy of coinvestment is also found in energy and storage systems. For instance, cooperative investments in battery storage within smart grid projects provide useful parallels, illustrating the broader relevance of game-theoretical coinvestment models beyond the context of edge computing.

2.2.2 Related Literature

Given the originality and the limited availability of directly comparable research, some articles from closely related domains can provide valuable context.

A survey by Moura and Hutchison (2019) presents an overview of game theory applications in multiaccess edge computing, discussing resource allocation and computation offloading scenarios (Moura & Hutchison, 2019). Computation offloading refers to the process of transferring resource intensive computational tasks from a device to a more powerful remote processor, such as a server or cloud platform. Although it does not explicitly address coinvestment, its insights into the broader use of game theory within edge computing provide relevant background knowledge.

In their work, Zhang et al. (2019) apply coalitional game theory to computation offloading in Nonorthogonal Multiple Access (NOMA)-enabled MEC scenarios (Zhang et al., 2019). While their primary focus is offloading rather than coinvestment, their methodologies and insights remain highly relevant, particularly for understanding the applicability of cooperative game theory to resource sharing problems in edge computing environments.

Industry insights from the ETSI White Paper (2019) provide valuable perspectives on transitioning from theoretical frameworks to actual infrastructure deployments and software developments within edge computing (European Telecommunications Standards Institute (ETSI), 2019). This source underscores the practical challenges and collaborative opportunities that directly align with joint investment initiatives.

Finally, a practical empirical study conducted by Das and Mukherjee (2021) explores edge computing deployments using AWS, emphasizing performance optimization and cost-sharing issues (Das & Mukherjee, 2021). Their findings validate the practical necessity of structured cost allocation methodologies, reinforcing the importance of coinvestment frameworks.

Using This Background

The real world cases and academic literature reviewed provide a useful context for understanding the practical implications of coinvestment frameworks. Joint investments in telecom, automotive infrastructure, and urban computing underscore the practical need for cooperative financial models that distribute risks and reduce capital barriers for individual stakeholders.

Moreover, by connecting theoretical concepts, such as coalitional game theory and the Shapley value, with concrete industry examples such as AWS Wavelength or MEC deployments, this thesis highlights the practical applicability and relevance of its theoretical contributions.

Additionally, positioning this work within the broader literature that explores game theory approaches

to resource allocation and economics in edge computing helps underline both the theoretical novelty and practical significance of the proposed approach.

2.2.3 Prosumers Academic Articles

This section summarizes insights from related academic articles provided alongside the primary reference (Rosario Patanè et al., 2023), which we extensively analyze throughout this thesis. Most of these articles focus on local energy markets and cooperative game theory approaches to resource management.

While we have not directly integrated these findings into our current work, we identify meaningful parallels between local energy management scenarios and edge computing environments. Therefore, we recommend the reader to first fully review and understand the outcomes, limitations, and proposed improvements presented in the main body of this thesis before engaging with this section. Doing so will provide the necessary context to better appreciate how these related works might contribute to addressing unresolved challenges or extending the current model in future research.

Summary of Ideas Related to the Edge Computing Problem

The articles analyzed in the appendix A covering coalitional game theory approaches for local energy trading, flexible prosumer models with storage, auction mechanisms for local markets, and misalignments in demand response objectives primarily examine resource sharing coordination at the local level. Despite their focus on local energy markets, several conceptual parallels can be drawn to edge computing scenarios. In both domains, resources are distinguished as either local (battery capacity or edge server capacity) or global (central grid or cloud resources). This section summarizes how each article's insights could potentially inform the edge computing model and identifies necessary modifications for practical implementation.

Coalitional Game-Theoretical Approaches for Edge Resource Coinvestment

Articles examining joint investments in battery storage propose scenarios analogous to edge computing stakeholders coinvesting in shared micro data centers.

- Major Adjustments: The concept of ramp constraints translates directly into the challenges faced when reallocating resources dynamically at each time-slot in edge computing scenarios. While reallocating computational resources such as CPU or memory might be nearly instantaneous, reassignments involving significant data migration, storage operations, or the duplication of entire virtual environments impose substantial ramp constraints. Thus, analogous ramp constraints must be explicitly modeled in dynamic allocation contexts to realistically reflect resource reallocation limitations.

- Applicability: Core concepts such as stable coalitions and Shapley value distributions remain potentially applicable. However, adjustments are necessary due to the difference between linear programming methods and the nonlinear diminishing returnutility functions present in the edge computing model. Therefore, suitable modifications in mathematical modeling are essential to accurately capture the resource sharing dynamics specific to computational workloads.

Discrete and Stochastic Storage vs. Shared Edge Servers

The notion of ramp constraints and discrete battery sizes in energy storage can parallel discrete CPU units (millicores) in edge computing, with analogous ramp constraints modeled as the maximum rate of task offloading or resource allocation.

- Major Adjustments: Edge computing resource allocations typically occur at discrete intervals (e.g., 15-minute time slots), which parallel the discrete intervals used in storage scheduling within energy systems. The main adjustment required involves accommodating discrete resource blocks and handling resource adjustments at predefined intervals. Additionally, the latency constraints in edge computing differ significantly from energy market pricing signals.

- Applicability: Insights into multi-stage resource scheduling from storage games can inform the structuring of discrete resource blocks among edge computing participants.

Auction-Based Approaches for Local Energy and Edge Computing

Articles proposing combinatorial double auctions (e.g., CombFlex) for energy trading can be adapted to allocate computing resources over multiple time slots.

- Major Adjustments: Auctions would focus on computational task requirements within specific time windows, incorporating edge computing's different cost structures.

- Applicability: Multi-interval bidding structures from energy auctions can effectively translate to edge computing resource allocation scenarios involving correlated resource usage over multiple time-slots.

Forecasting, Benchmarks, and Flexibility for Edge Computing

Standardized benchmarks proposed for grid flexibility can similarly apply to edge computing, where benchmarks would evaluate the efficiency of allocation and scheduling mechanisms against typical usage patterns.

- Major Adjustments: Metrics would shift from energy load curves to computational loads, latency violations, and resource utilization.

- Applicability: Benchmarking methodologies promote systematic comparisons across resource scheduling strategies, fostering the use of machine learning techniques such as reinforcement learning to optimize dynamic edge resource management.

Misalignments in Objectives: Energy vs. Edge Computing

Articles highlighting misaligned objectives in local energy markets illustrate how simultaneous resource scheduling can inadvertently create peaks or imbalances.

- Major Adjustments: In edge computing, peaks correspond to periods of high concurrency, potentially causing latency and throughput issues, particularly during resource reallocation at discrete time-slot transitions. Similar to energy markets, where pricing changes or industry operations commencing at specific times (e.g., at the start of business hours) create consumption imbalances, simultaneous resource adjustments at the boundaries of discrete time-slots in edge computing could lead to significant delays or bottlenecks.

- Applicability: The concept of unintended resource allocation peaks highlights the necessity of carefully designed scheduling mechanisms. These mechanisms must account for simultaneous reallocation at predefined intervals, aligning user incentives with system constraints to maintain stable and efficient edge computing operations.

Concluding Remarks

In summary, although significant differences exist between local energy management and edge computing resource allocation, the reviewed literature reveals fundamental principles and methodologies that can inform the development of cooperative edge computing infrastructures. Adapting these methodologies to edge computing requires redefining physical constraints, adjusting time granularities, and revising cost and utility structures. Nevertheless, the core principles of coalition formation, combinatorial auctions, and multi-stage strategic scheduling remain highly relevant and adaptable, provided the necessary modifications are thoroughly addressed.

Chapter 3

Analytical Investigation of the Originally Proposed Edge-Computing Model

In the first part of this theoretical chapter, we show that, without changing the existing model, the optimal allocation for each Service Provider can be calculated in a deterministic manner and computed in constant time $\mathcal{O}(1)$. This contrasts with the current approach, which relies on optimization methods that maximize the value of the Grand Coalition. Furthermore, we demonstrate that the full load function can be omitted, as the same results are obtained by using only the average load.

$$\mathcal{O}(2^{n-1}) \longrightarrow \mathcal{O}(n)$$

Furthermore, we prove that, within the proposed model, the Shapley value can be calculated in a straightforward manner, significantly reducing the computational complexity. Instead of evaluating every possible coalition containing the NO, which requires exponential time, the calculation can be performed in constant time $\mathcal{O}(1)$. Finally, we demonstrate that both the model and its software implementation can be solved in linear time $\mathcal{O}(n)$, where n represents the number of players. This marks a substantial improvement compared to the original formulation and implementation, where scaling beyond a few dozen players is computationally infeasible.

In the second part, we analyze the strategic implications of the proposed model. We begin by deriving a simplified, yet equivalent, version of the net utility function and use it to demonstrate that the model is not strategy-proof. This finding implies that players have incentives to misrepresent the parameters of their utility functions. We then propose potential approaches to address the lack of strategy-proofness. Next, we examine the economic impact of inaccurate estimations of utility function parameters by SPs, comparing the effects of both underestimations and overestimations.

Finally, we define the scope of the proposed simplifications by identifying changes to the model, such as sublinearity in allocation pricing or dynamic allocation per time-slot, that disrupt the independent contribution of SPs and thereby invalidate most of the proposed simplifications. If these changes arise due to positive externalities or synergies among players (such as the previously mentioned), then the original model may be interpreted as a worst-case scenario in terms of players' revenues and payoffs. In the second stage, we incorporate some of these changes into the model and compare the resulting outcomes in terms of both allocation and revenue.

3.1 SPs Independent Contribution and Game Convexity

The Theorem 1 of the referenced article demonstrates the game is convex; for that, it uses the definitions of the value function as expressed in equations 1.1, 1.2 and 1.3. In cooperative game theory, a game is said to be convex when the gains from forming coalitions grow at least as fast as the sum of individual contributions, reflecting increasing or additive returns.

Formally, if we let $v : 2^N \to \mathbb{R}$ to be the value function of the game, where N is the set of players and $v(S)$ denotes the value of the coalition $S, T \subseteq N$. The game is convex, if for all subsets S and T of N, the following inequality holds:

$$v(S) + v(T) \leq v(S \cup T) + v(S \cap T).$$

An equivalent and perhaps more intuitive convexity definition is that for any player $i \in N$ and for any pair of coalitions $S \subseteq T \subseteq N \setminus \{i\}$:

$$v(S \cup \{i\}) - v(S) \leq v(T \cup \{i\}) - v(T).$$

This means that the marginal contribution of player i is non-decreasing with respect to the size of the coalition: typically, the larger the coalition, the higher the additional value generated by player i joining it.

From an economic perspective, typically convexity implies:

- Players have incentives to cooperate in larger coalitions, as their marginal contribution is better valued.
- Larger coalitions tend to be stable, since players prefer to stay within them.

Now let's evaluate how these definitions apply to our game, which has some particularities that are worth mentioning. If we take the second definition of convexity, we can identify three different cases:

- $NO \notin T$ and $NO \notin S$: The expression is equal and zero in both sides.
- $NO \in T$ and $NO \notin S$: Only case when the strict inequality is meet.
- $NO \in T$ and $NO \in S$: The expression is equal on both sides, indicating the marginal contribution is equal for any coalitions with the NO in it.

Therefore, although the game does not exhibit strictly increasing marginal contributions in the sense of complementarity between the SPs, it still satisfies the formal conditions for convexity. However, the typical implications of convexity are not met here; SPs don't have incentives to cooperate in larger coalitions as their marginal contribution remains constant. As a consequence, larger coalitions are not stable since SPs are indifferent about other SPs joining the coalition. This characteristic is known as independent contribution or additive value function.

The referenced article identifies this independent contribution between SPs. However, it overlooks some of its implications, leading to a solution that, while correct, could be streamlined further. In the next sections, we will explore how this property influences other relevant aspects of the game and discuss potential modifications to the model that would affect the independent contribution property.

3.2 Analyzing Service Providers' Net Utility

In this analysis, we introduce a new equation crucial for future demonstrations; a Service Provider's net utility is defined as the difference between its revenues or utility (from now on, we are going to

call it gross utility), given by the utility function, and the cost of its resource allocation. Under the independent contribution property, the net utility represents the SP's marginal contribution to any coalition that includes the NO. Before we study the net utility function and propose simplifications of its calculation, we review the meaning of ξ in the proposed model.

3.2.1 Interpretation of the Diminishing Returns Parameter

The instantaneous utility of service provider i at time-slot t is defined as:

$$u_i\big(l_i^t, h_i\big) = \beta_i \, l_i^t \big(1 - e^{-\xi \, h_i}\big).$$

In the referenced article, the parameter ξ is an exogenous shape parameter that models how quickly the term $1 - e^{-\xi \, h^i}$ approaches its saturation limit of 1. A larger value of ξ implies quicker saturation, reflecting stronger diminishing returns, whereas a smaller ξ corresponds to a more gradual increase in utility as resource allocation h^i grows. Originally, ξ is treated as a fixed, uniform parameter common to all SPs and time-slots within the model, independent of individual SP characteristics.

After evaluating the implications of this assumption and consulting with the authors of the article, we choose to redefine ξ as a service provider-specific variable, denoted as ξ^i. This decision arises from the recognition that treating ξ uniformly would imply all SPs share an identical relationship between monetized load ($l_{ts}^i \cdot \beta^i$) and the hardware allocation h^i. Such uniformity restricts the model's capacity to accurately represent diverse SP behaviors and characteristics.

Among the model's variables, ξ^i poses the greatest challenge when it comes to interpreting its physical meaning. In later sections, we will explore this parameter thoroughly and reinterpret it from a practical standpoint. For the present discussion, however, it suffices to acknowledge that ξ^i is now regarded as an intrinsic variable of each service provider, rather than a universal parameter of the model.

3.2.2 Calculating the Net Utility

Calculating a Service Provider's net utility involves two key equations from the referenced article: the coalitional value function, $v(S)$, and the utility function, $u_i(l_i^t, h_i)$. For any player i and a time-slot t, these equations are defined as follows:

$$v(S) = \left(D \cdot \sum_{i \in S} \sum_{t=1}^{T} u^i(l_t^i, h^i) \right) - d \cdot C \quad \text{subject to} \quad \text{NO} \in S$$

$$u^i(l_t^i, h^i) = \beta^i \cdot l_t^i \cdot (1 - e^{-\xi^i \cdot h^i}) \qquad \text{subject to} \quad h^i = 0 \text{ for NO}$$

By applying the independent contribution property to the first equation, we can isolate the portion of the total cost attributable to a specific player. This allows us to express the net utility function for player i at time-slot t as:

$$u_{ts\_net}^i(l_t^i, h^i) = \beta^i \cdot l_t^i \cdot (1 - e^{-\xi^i \cdot h^i}) - \frac{d \cdot h^i}{T \cdot D} \tag{3.1}$$

Consequently, the value function of a coalition can be redefined as

$$v(S) = \sum_{i \in S} u_{net}^i(l_t^i, h^i) \tag{3.2}$$

3.2.3 Net Utility Calculation Equivalence

In the proposed model, the average load was used within a sinusoidal equation and a set of hyper-parameters to reflect changes in traffic load throughout the day. In this section, we prove that, as a consequence of having a fixed allocation at all time-slots and the player's independent contribution, we can use the average load and avoid calculating each time-slot's specific load and utility, thereby simplifying the model.

Starting with the net utility function for the player i in the time-slot t, as defined in Equation 3.1, we calculate the total net utility for that player over the entire investment period T as follows:

$$U^i_{\text{tot\_net}} = D \cdot \left(\sum_{t=1}^{T} \beta^i \cdot l^i_t \cdot (1 - e^{-\xi^i \cdot h^i}) \right) - d \cdot h^i$$

Considering l^i_{avg} as the average load. By definition:

$$\sum_{t=1}^{T} l^i_t = T \cdot l^i_{avg}$$

We can then express the total net utility for the player i using the average load.

$$U^i_{tot\_net} = D \cdot T \cdot \left(\beta^i \cdot l^i_{avg} \cdot (1 - e^{-\xi^i \cdot h^i}) \right) - d \cdot h^i \tag{3.3}$$

Accordingly, the value function of a coalition, accounting for the average load, is defined as:

$$v(S) = D \cdot T \cdot \sum_{i \in S} u^i_{net}(l^i_{\text{avg}}, h^i) \tag{3.4}$$

3.2.4 Constraint for Positive Net Utlity

To ensure that the total net utility is positive, we require:

$$U^i_{tot\_net} > 0$$

This condition is met under the following constraint:

$$d \cdot h^i < D \cdot T \cdot \beta^i \cdot l^i_{avg}$$

A detailed demonstration is provided in the Section B of the demonstrations Appendix.

3.3 Analyzing Service Providers' Optimal Allocation

In the proposed model, each player's optimal allocation was calculated to maximize the total global net utility; this is the grand coalition value. This implies a maximization algorithm considering the summation of all Service Providers' utility functions and the subtraction of the total allocation price. In this section, we prove that we can achieve the same result in a more straightforward and deterministic manner.

Having previously isolated each Service Provider's net utility function as expressed in equation 3.3, we analyze this function's critical points to identify a unique maximum representing the optimal allocation for player i at:

$$h^i = \frac{1}{\xi^i} \cdot \ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi^i}{d}\right) \tag{3.5}$$

Additionally, we identify a condition on the SP parameters that must be met for the allocation to be positive:

$$d < D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi^i \tag{3.6}$$

This condition represents the minimum requirement for an SP to have any incentive to participate in the co-investment. The complete demonstration is in Section B.1 of the demonstrations chapter of the Appendix.

Finally, we prove that $U^i_{\text{tot\_net}}$ is always non-negative for the optimal allocation h^i and is equal to zero only when the optimal allocation is zero, showing that only constraint 3.6 should be added to our model.The complete demonstration is in Section B.2 of the demonstrations chapter of the Appendix.

3.4 Service Providers's Marginal Contribution

As a consequence of the independent contribution among SPs and the NO being a veto player, for an SP in any coalition within the NO, this contribution remains consistent and equals the player's net utility $U^i_{tot\_net}$. Conversely, for coalitions excluding the NO, the contribution is zero. In the next section, we will find some implications of this marginal contribution characteristic when calculating the Shapley value.

3.5 Shapley Value Calculation

The Shapley Value is a fundamental concept in cooperative game theory. It uniquely determines the payoff for each player within a coalition by quantifying their individual contribution to each possible group's overall success. This value is calculated as the weighted average of the marginal contributions that a player makes to all possible coalitions to which they can join. This approach ensures a fair distribution of payoffs, accurately reflecting each player's contributions and the significance of the coalitions they impact.

In the referenced article, the Shapley value is used to define the payoff vector. where the payoff of the player i is denoted by x^i. In this section, we show that previous calculations of the Shapley value have underestimated the impact of the independent contribution property. By fully integrating it, we can significantly simplify the computation, reducing its complexity to a constant time $\mathcal{O}(1)$. This enhancement dramatically increases the scalability of simulations, extending from a small group of players to scenarios that include hundreds or even thousands of players.

Intuitively, we can observe that as a consequence of having the same marginal contribution for coalitions within the NO, we should be able to calculate the Shapley value without considering all possible coalitional values. Our next step is to work with the Shapley value equation, demonstrate that this simplification is possible, and give an equivalent formula for finding the Shapley value for a Service Provider and for the Network Owner.

The Shapley value for the player i is expressed by the following formula:

$$\phi^i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v(S \cup \{i\}) - v(S))$$

where $v(S)$ is the value of the coalition S and $|S|$ is its size. This formula can be dissected into the weighting factor and the marginal contribution.

3.5.1 The Weighting Factor

The weighting factor, represented by $\frac{|S|!(|N|-|S|-1)!}{|N|!}$ reflects how different coalition sizes are considered in the overall computation by accounting for all permutations of coalitions that can be formed, with and without the player i. The sum of this factor totals 1, symbolizing the balanced probability of coalition formations.

For each coalition S that contains player i, this factor can be seen as the probability that coalition S is formed first, followed by the joining of player i and then the remaining players joining in any order. It similarly represents the probability for the reverse formation sequence.

We show that for any player i, there is an equal probability $\frac{1}{2}$ of forming a coalition with or without any other specific player k. Considering a set of players N, with $|N| = n$, and a pair of different players i and k. The probability calculations involve:

- The number of subsets of $N \setminus \{k\}$ that include i but exclude k, totaling 2^{n-2}.
- The number of subsets where both i and k are included is also 2^{n-2}.

Since the total number of coalitions that can include i is 2^{n-1}, the probability that i is in a coalition with or without k is given by:

$$P(\text{Coalition with } k) = P(\text{Coalition without } k) = \frac{2^{n-2}}{2^{n-1}} = \frac{1}{2}$$

This shows that there is an equal probability that k is included or excluded from any subset containing i. Finally, setting $k = NO$ and $i = SP_i$, the sum of the weighting factor equals $\frac{1}{2}$ when $NO \in S$ and $\frac{1}{2}$ when $NO \notin S$.

3.5.2 Marginal Contribution Factor

The second term in the Shapley value equation represents the player's marginal contribution to any potential coalition. As previously established, this marginal contribution for a given player remains constant for coalitions that include the NO and is zero for those that do not include the NO.

3.5.3 Simplifying Shapley Value Calculation

By combining the weighting factor and marginal contribution parts of the formula, we can conclude that for any SP, the Shapley value is equal to half of the value it brings to any coalition, which in turn is equal to half of its net utility. For the NO, it is equal to half of the summation of all SP's net utility.

$$x^{SP_i} = \phi^{SP_i} = \frac{v(\{SP_i\})}{2} = \frac{U^i_{tot\_net}}{2}, \forall SP_i \in N \tag{3.7}$$

$$x^{NO} = \phi^{NO} = \frac{1}{2}\sum_{i=1}^{n} v(\{SP_i\}) = \frac{1}{2}\sum_{i=1}^{n} U^i_{tot\_net} \tag{3.8}$$

3.6 Analyzing the Core of the Game

In this section, we analyze the Core of the game. Although the referenced article correctly establishes the nonemptiness of the Core from a mathematical perspective, we showed in section B that achieving a positive net utility might require a negative allocation. This highlights the practical necessity of

enforcing the positive allocation constraint expressed in (3.6). As a result, if a player's utility parameters do not satisfy this condition, the player will be excluded from the Core and have no incentive to participate in the co-investment. Furthermore, we show that, due to the independent contribution property, the Core reduces to a unique point.

3.6.1 Defining the Core of the Game

The Core represents the set of feasible payoff allocations among players such that no subgroup, or coalition, has an incentive to deviate from the grand coalition. It is a central concept in cooperative game theory, as it ensures stability in the distribution of benefits. If a payoff vector belongs to the Core, no coalition can improve upon it by breaking away.

Formally, the Core is defined as:

$$C = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \sum_{i \in N} x_i = v(N), \quad \sum_{i \in S} x_i \geq v(S), \quad \forall S \subseteq N \right\}$$

Where:

- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the payoff vector.
- $v(S)$ denotes the value of coalition S.
- The first condition ensures efficiency, meaning the total distributed payoff equals the value of the grand coalition.
- The second condition ensures coalitional rationality, meaning no subgroup receives less than what it could earn on its own.

Thus, the Core contains all payoff allocations where all players prefer remaining in the grand coalition. A non-empty Core ensures stability and prevents deviation.

3.6.2 Uniqueness of the Core under Independent Contribution with a Veto Player

Our game is characterized by independent contributions and the presence of a veto player (the NO). In this setting, the value of any coalition depends entirely on the SPs it includes, but only if the NO is present. That is, the NO enables value creation but does not contribute value on its own. The characteristic function is defined as:

$$v(S) = \begin{cases} \sum_{i \in S \setminus \{\text{NO}\}} v(\{i\}) & \text{if NO} \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Under these assumptions, coalitional rationality simplifies to:

$$\sum_{i \in S} x_i = \sum_{i \in S \setminus \{\text{NO}\}} v(\{i\}), \quad \forall S \subseteq N \text{ with NO} \in S,$$

and efficiency becomes:

$$\sum_{i \in N} x_i = \sum_{i \in N \setminus \{\text{NO}\}} v(\{i\}).$$

These two conditions uniquely determine the payoff vector. Specifically, the only vector satisfying both conditions is the Shapley value, which in this case is given by:

$$x_{SP_i} = \frac{1}{2} v(\{SP_i\}), \quad \forall SP_i \in N,$$
$$x_{\text{NO}} = \frac{1}{2} \sum_{SP_i \in N} v(\{SP_i\}).$$

Any deviation from this payoff vector would violate either coalitional rationality or efficiency. For instance, if an SP is assigned less than half of its marginal contribution, it would be strictly better off by forming a coalition with the NO alone, excluding all other players. On the other hand, assigning the NO less than half of the total contributions from all SPs would result in a total payoff that falls short of the coalition's full value, thus violating efficiency.

Therefore, under the assumptions of independent contributions and a veto player, the Core reduces to a unique point. This point corresponds exactly to the Shapley value, indicating that cooperation does not generate additional surplus beyond the sum of individual contributions.

If the assumption of independent contributions no longer holds in future extensions of the model, for example, if coalition synergies or complementarities arise, then this payoff vector can still be used as a reference or estimation. However, unlike in earlier approximations, it would not represent a worst-case scenario, since it may underestimate or overestimate players' actual incentives depending on the specific form of interdependence.

3.6.3 Calculating Payments and Revenues

In the referenced article, the definition of the aim of payments is ambiguous. To clearly illustrate this, we present direct extracts from different sections of the article.

In the "COINVESTMENT MODEL" section, the article states:

> The payoff of any player $i \in \mathcal{N}$ is $x^i = r^i - p^i$, where r^i and p^i are the revenues and the payment, i.e. the capital cost, respectively.

This statement implies that payments serve to redistribute payoffs among players, aligning each player's payoff with their Shapley value.

However, in the subsequent paragraph of the same section, the article presents another definition:

> ... the sum of all players' payments should be such that $\sum_{i \in S} p^i = d \cdot \mathcal{C}$,

where $\mathcal{C}$ represents the total capacity allocated and d is the price per millicore. This second statement suggests that payments are instead intended specifically to cover the deployment costs.

Also in the "ANALYSIS" section, subsection "Initial investment of players," which addresses how revenues and payments are calculated. The article states that the calculations aim to determine how much each player must pay at the beginning of the investment, defining the relationships between the players' revenues, payments, and their respective payoffs as follows:

$$r^i - p^i = x^i, \quad \forall i \in N \tag{3.9}$$

$$\text{s.t.} \quad \sum_{i \in N} r^i = D \cdot \sum_{i \in N} \sum_{t=1}^{T} u^i(l_t^i, h^{i*}) \tag{3.10}$$

$$\text{where: } h^*, C^* = \arg\max_{\hat{h}, C} v^i(\hat{h}, C, N) \tag{3.11}$$

Thus, the article simultaneously presents two different interpretations of the payment variable p^i: one aimed at redistributing payoffs to achieve fairness (through the Shapley value), and another exclusively designed to distribute the initial infrastructure deployment costs. It is important to note that requiring payments to achieve fairness entirely at the beginning of the investment period may not be attractive for SPs, as it could imply significant upfront payments. A more realistic approach might involve covering only the deployment costs upfront, while distributing payments related to fairness (aligned with Shapley values) periodically over the investment duration.

To clarify the previous ambiguity, we define:

- p^i_{alloc} As the payment made by SP i with the sole purpose of paying for the initial hardware deployement such that $\sum_{i \in S} p^i_{alloc} = d \cdot \mathcal{C}$
- p^i_{fair} As the payment made (or received if negative) by player i with the sole pourpose of redistributing payoffs to achieve fairness

Then we take $p^i = p^i_{alloc} + p^i_{fair}$ which aligns with the framework described in equations from 3.9 to 3.11. This framework results in a system of $n+1$ equations and $2n$ variables, allowing for multiple possible solutions. Although these solutions will lead to the same payoff for each player, the specific payments may vary. By applying the independent contribution property, we propose that each Service Provider can calculate its own payments and revenues.

As we have previously demonstrated, each SP can independently calculate its net utility using equation 3.3, which we rewrite here.

$$U^i_{tot\_net} = D \cdot T \cdot \left(\beta^i \cdot l^i_{avg} \cdot (1 - e^{-\xi^i \cdot h^i})\right) - d \cdot h^i$$

The first term of this equation corresponds to the SP revenue, the second term corresponds to the SP payment for allocation p^i_{alloc}. Also, from equation 3.7 we know that an SP payoff is half of its total net utility value, and from equation 3.8 we know that the other half should be given to the NO corresponding to p^i_{fair}.

Formally, for SP i:

$$p^{\mathrm{SP}_i} = p^i_{alloc} + p^i_{fair} = d \cdot h_i + \frac{u^i_{\mathrm{tot\_net}}}{2} \tag{3.12}$$

$$x^{SP_i} = \frac{U^i_{tot\_net}}{2}, \forall SP_i \in N \tag{3.13}$$

$$r^{SP_i} = D \cdot T \cdot \left(\beta^i \cdot l^i_{avg} \cdot (1 - e^{-\xi^i \cdot h^i})\right) \tag{3.14}$$

In the case of the NO, it does not allocate any resources and does not generate any revenue. As we have seen in equation 3.8, its payoff corresponds to the sum of half of the net utility of each SP.

Formally, for the NO, being n the amount of service providers:

$$p^{NO} = -\frac{1}{2} \sum_{i=1}^{n} U^i_{tot\_net} \tag{3.15}$$

where the negative sign indicates that it is receiving the payment from other players, contrary to the SPs that pay to the NO.

To complete this section, we analyze the following claims of the referenced article that are under the section "ANALYSIS", subsection "Relevant properties of our game":

> If player i does not produce revenues and makes no payments, then it is a null player, i.e. $v(S \cup \{i\}) = v(S)$ Note that there can be players that do not pay or are even paid ($p^i \leq 0$), which still positively contribute to the coalition. For instance, any SP i can positively contribute to the coalition collecting large revenues r^i.

These statements may be generally valid in cooperative game theory. However, in our specific model, they do not apply.

Regarding the null player statement; revenues in our model are defined as the gross utility generated by each SP, reflecting the income derived from serving their customers. The motivation for participating in the co-investment is to achieve a positive marginal contribution, which is equivalent to the net utility. If an SP does not generate any revenue, it will fail to produce a positive net utility and consequently has no incentive to allocate resources; indeed, allocating resources in such a case would lead to economic losses. Moreover, if a player's revenues merely cover the deployment costs, then under the independent contribution property, the player neither gains nor loses money. In that scenario, the incentive to participate would rely solely on anticipated future benefits (for example, an increase in the β parameter) that are not yet incorporated into the SP utility function. In summary, null players (those that do not generate a positive marginal contribution) are not allowed in our model.

Regarding the claim that some players do not pay or are even paid, our analysis shows that in our model p^i is never zero for any player and is negative only for the NO. This indicates that SPs always make positive payments and never receive funds from other players, a result that follows directly from the independent contribution.

Note that we could consider the case where SP's net utility is negative, indicating that they are subsidizing a service, but as shown in the demonstrations Section B.2 positive allocations always lead to positive net utility.

3.7 Computational Complexity of the Game

The independent contribution has significant implications for the computational model of our cooperative game. So far, we have established some key aspects:

- Objective function: Maximizing the coalitional value can be done in linear time and in a deterministic manner. This eliminates the need for approximation-based numerical methods, which are computationally intensive.
- Marginal contribution consistency: For coalitions that include the Network Owner, each SP's marginal contribution remains constant, and it is zero for coalitions without the NO. This consistency allows the Shapley Value and thus the payoff to be computed directly, avoiding the need to evaluate all possible coalition formations.
- Allocation payment calculation: Since the per-unit price of the allocation is fixed and the allocation remains constant across all time-slots, calculating each player's share of the total cost p^i_{alloc} becomes straightforward.
- Fairness payment calculation: All SPs should pay half of their total net utility to the NO this is defined as p^i_{fair}.
- The two previous points simplifies the determination of payments and revenues, leading to a unique solution.

These characteristics reveal the game's inherent linear time complexity, allowing SPs to independently compute their optimal allocations, revenues, payments, and payoffs. Additionally, the Network Owner's payoff can be derived directly from the total payments made by all SPs. As a result, for n players, the computational model can be solved in linear time $\mathcal{O}(n)$, which simplifies implementation and improves both the efficiency and transparency of the solution process.

Further and more general observations on the individual contribution property are provided in Section C of the Appendix.

3.8 Model Simplification and Critical Functions Analysis

In this section, we derive compact yet equivalent expressions for the net utility and optimal allocation functions by consolidating all parameters into their most concise form. This streamlined representation retains full equivalence with the original model while making it easier to analyze how variations in model parameters affect an SP's net utility and optimal allocation.

3.8.1 Simplified Net Utility

To simplify the net utility equation (3.3), we begin by expressing it in its original form:

$$U^i_{\text{tot\_net}} = D \cdot T \cdot \left(\beta^i \cdot l^i_{\text{avg}} \cdot (1 - e^{-\xi^i \cdot h^i})\right) - d \cdot h^i$$

Observe that the terms β^i and l^i_{avg} appear as a product. This allows us to introduce a new variable, denoted as ρ^i, representing potential monetization.

$$\rho^i = \beta^i \cdot l^i_{\text{avg}} \tag{3.16}$$

This variable represents the maximum potential utility, which remains unattainable in practice.

Formally:

$$\lim_{\substack{h \to \infty \\ d \to 0}} U^i_{\text{tot\_net}} = \rho$$

Next, we define the amortized unit price d', which represents the cost associated with a single time-slot:

$$d' = \frac{d}{D \cdot T} \tag{3.17}$$

For simplicity, from now on, we omit the superscript i in the following analysis, as it is not essential for our purposes. The resulting simplified net utility equation for a single time-slot is:

$$U_{\text{ts\_net}} = \rho \left(1 - e^{-\xi h}\right) - d'h \tag{3.18}$$

To obtain the total net utility over the entire investment period, we multiply the single time-slot utility by the total number of time-slots: $D \cdot T$.

3.8.2 Simplified Optimal Allocation

Using the simplified net utility function we just derived (3.18), we determine the optimal allocation h^* by maximizing the net utility with respect to h. This leads to the following simplified optimal allocation equation:

$$h^* = \frac{1}{\xi} \ln\left(\frac{\rho\xi}{d'}\right) \tag{3.19}$$

The condition that ensures $h^* > 0$, corresponding to equation (3.6), simplifies the following:

$$d' < \rho \cdot \xi \tag{3.20}$$

3.9 Derivation of Different Expressions for Allocation and Optimal Net Utility Functions

Having derived simplified versions of the allocation and net utility equations, we now obtain different expressions for the optimal allocation condition. By substituting these expressions into the net utility function, we derive alternative forms in which one variable is eliminated.

Introducing Notation

Since the simplified utility functions involve different variables, we explicitly incorporate these variables into the equation notation. This allows the net utility function, as defined in (3.18), to be written as:

$$U_{\text{ts\_net}}(\xi, \rho, d', h) = \rho \left(1 - e^{-\xi h}\right) - d'h \tag{3.21}$$

3.9.1 Derivation of Different Expressions for the Allocation Function

The optimal allocation equation (3.19) defines a relationship between the parameters of the utility function. By manipulating this equation, we can solve for different parameters, obtaining expressions that will be used in later analyses.

Solving for ρ:

$$\rho = \frac{d' \cdot e^{h \cdot \xi}}{\xi} \tag{3.22}$$

Solving for d':

$$d' = \frac{\xi \cdot \rho}{e^{h \cdot \xi}} \tag{3.23}$$

Solving for ξ:

$$\xi = \frac{d' \cdot e^{h \cdot \xi}}{\rho} \tag{3.24}$$

The last equation is a transcendental equation, because the variable appears inside and outside an exponential function. Such equations cannot be solved algebraically in a straightforward manner.

3.9.2 Derivating Different Expressions for the Utility Function

The net utility function for a single time-slot, given by (3.18), depends on four variables: ξ, ρ, d', and h, which must satisfy the optimal allocation condition presented in (3.19). By substituting relationships derived from this condition into the utility function, we obtain alternative expressions in which one of these variables is eliminated. However, since no explicit algebraic solution exists for ξ, it cannot be directly substituted.

In Section B.3 of the Demonstrations chapter in the Appendix, we present detailed step-by-step derivations of these alternative expressions for the net utility function, along with interpretations of the resulting equations after substituting d' and ρ. In the following subsection, we specifically analyze the expression obtained by substituting h^*, as it is the most significant and useful form.

3.9.3 Utility Function Subject to the Optimal Allocation Condition

$$U_{\text{ts\_net}}(\xi, \rho, d') = \rho \left(1 - \frac{d'}{\rho\xi}\right) - \cdot \frac{d'}{\xi} \ln\left(\frac{\rho\xi}{d'}\right) \tag{3.25}$$

This expression is valuable because it eliminates the need to explicitly compute the allocation, simplifying both analytical derivations and numerical computations. By removing h from the equation, sensitivity analyses become more direct, clearly highlighting how the fundamental parameters (ρ, ξ, and d') influence net utility.

The first term, $\rho\big(1 - \frac{d'}{\rho\xi}\big)$, represents the fraction of the potential monetization that is converted into gross utilities.

The second term, $-\frac{d'}{\xi}\ln\big(\frac{\rho\xi}{d'}\big)$, represents the amount of allocated resources at h^* multiplied by the amortized per-millicore price.

Taken together, these two parts show how revenue generation, expressed through ρ, and the corresponding cost penalty interact under varying parameter values. They show the balance between higher benefits and increased costs.

We also provide an equivalent expression of this function that will be used in future sections:

$$U(\rho) = \rho \; - \; \frac{d'}{\xi}\Big[1 + \ln\big(\tfrac{\rho\xi}{d'}\big)\Big]$$

Although this expression is less intuitive, it clearly separates the maximum potential monetization ρ in the first term from the second term, which combines the unmonetized requests with the costs associated with monetized ones. Note that this function is strictly convex in ρ and asymptotically linear ($\lim_{\rho\to\infty} U(\rho)/\rho = 1$), implying increasing marginal returns that converge to one. These considerations suggest that, although the logarithmic form simplifies the analysis, it may not fully reflect hardware demands as load grows. In Section 5.3 we propose net utility function modifications to address these limitations.

In table 3.1 we present a summary of the variables influenced by analyzing their derivatives.

Parameter	$\frac{\partial U(h^*)}{\partial(\cdot)}$	Interpretation
ρ	Positive	Greater monetization increases net utility.
ξ	Positive	Higher ξ improves resource efficiency, increasing net utility.
h	$\begin{cases} > 0, & h < h^* \\ < 0, & h > h^* \end{cases}$	Net utility follows diminishing returns and is maximized at h^*.
d'	Negative	Higher marginal cost reduces net utility.

Table 3.1: Effects of parameters on net utility at the optimal allocation.

3.10 Allocation Function Analysis

From Equation 3.19,

$$h^* = \frac{1}{\xi}\ln\big(\tfrac{\rho\xi}{d'}\big),$$

we see two clear parts:

- $\frac{1}{\xi}$: the inverse of the sensitivity. A larger ξ cuts the allocation.
- $\ln\left(\frac{\rho\xi}{d'}\right)$: a logarithm that rises with $\rho\xi$ and falls with d'.

Together, a bigger ξ lowers h^* directly and also shapes how ρ and d' push on it.

Effect of ρ

$$\frac{\partial h^*}{\partial \rho} = \frac{1}{\xi\rho}.$$

Because the equation for h^* contains $\rho = \beta\, l_{\text{avg}}$ only inside a logarithm, raising β increases the allocation slowly. This is reasonable: a larger benefit factor alone does not call for a proportional rise in capacity, so a sub-linear (logarithmic) response may accurately reflect real-world scenarios. Load, however, should scale differently. When l_{avg} doubles, the optimal allocation needs to grow by roughly the same amount, and often even more. A purely logarithmic response can therefore underestimate the capacity that real systems must deploy as load rises.

Effect of d'

$$\frac{\partial h^*}{\partial d'} = -\frac{1}{\xi d'}.$$

A higher marginal cost d' lowers the allocation. The drop is steep when d' is small and flattens as d' rises, matching the shape of a logarithm.

Effect of ξ

$$\frac{\partial h^*}{\partial \xi} = -\frac{1}{\xi^2}\left[\ln\left(\frac{\rho\xi}{d'}\right) - 1\right].$$

ξ changes both the scale and the inside of the logarithm, so its impact is mixed. If $\ln(\frac{\rho\xi}{d'}) < 1$,raising ξ increases h^*; after $\ln(\frac{\rho\xi}{d'}) > 1$, raising ξ lowers h^*. We name this maximum as ξ_{peak}.

Since the effect of ξ is the more complex one, we dedicate the next section to analyze its effects in more depth. Before that, we present the table 3.2 to summarize the results of this section.

Parameter	**Sign of Derivative**	**Interpretation**	**Mathematical relationship**
ρ	Positive	Larger potential monetization increases allocation.	Logarithmic
ξ	Positive until ξ_{peak}, negative after ξ_{peak}	Benefit-driven rise followed by resource efficiency-driven fall.	Linear and logarithmic
d'	Negative	Increased marginal cost reduces allocation.	Logarithmic

Table 3.2: Summary of parameter influences on optimal allocation based on derivatives.

3.10.1 Influence of the Diminishing Returns Parameter on Allocation

We identified a specific value of ξ that maximizes the allocation, denoted as ξ_{peak}. The allocation at this value is $h(\xi_{\text{peak}})$, and the corresponding net utility is $U_{\text{ts\_net}}(\xi_{\text{peak}})$.

This maximum arises from the interaction between the inverse factor $\frac{1}{\xi}$ and the logarithmic term $\ln\left(\frac{\rho\xi}{d'}\right)$. For smaller values of ξ, the logarithmic term grows rapidly, causing a quick rise in allocation. However, beyond ξ_{peak}, the inverse term $\frac{1}{\xi}$ becomes dominant, outweighing the logarithmic growth, and the allocation begins to decline.

Formally, we can define ξ_{peak} as the solution to the following maximization problem:

$$\xi_{\text{peak}} = \arg\max_{\xi>0} \left\{ h^*(\xi) = \frac{1}{\xi} \ln\left(\frac{\rho\xi}{d'}\right) \right\} \tag{3.26}$$

We analyze $h^*(\xi)$ as a function of ξ, holding ρ and d' constant. By studying its derivatives, we find a unique critical point ξ_{peak} that maximizes the allocation. We then compute the allocation at this peak, $h^*(\xi_{\text{peak}})$, and the corresponding time-slot net utility $U_{\text{ts\_net}}(\xi_{\text{peak}})$. The detailed derivation is presented in Section B.4 of the Demonstrations chapter of the Appendix.

We find that $h^*(\xi)$ has a unique maximum at:

$$\xi_{\text{peak}} = \frac{d' \cdot e}{\rho} \tag{3.27}$$

The corresponding allocation at this point is:

$$h^*(\xi_{\text{peak}}) = \frac{\rho}{d' \cdot e} \tag{3.28}$$

At ξ_{peak} the value of the exponential term $(-h \cdot \xi\ )$ is exactly 1, implying that the fraction of requests served at the Edge is $\approx 63\%$, and the corresponding net utility is:

$$U_{\text{ts\_net}(h^*,\xi_{\text{peak}})} = \rho \cdot \frac{e-2}{e} \approx \rho \cdot 0.264 \tag{3.29}$$

Interpreting ξ_{peak}

At ξ_{peak}, the allocation reaches its maximum with respect to ξ. Beyond this point, any increase or decrease in ξ results in a reduction of the allocation. While SPs can technically operate with values of ξ below ξ_{peak}, doing so places the system in a sensitive region where small parameter changes can cause significant variations in allocation, due to the steep increase before reaching ξ_{peak}. Moreover, in this range, the service provider receives at most about 26% of the potential monetization. As demonstrated in Section B.5, as ξ decreases, the net utility also decreases.

It is also important to note that if ρ decreases to $\frac{e}{e}$, the positive allocation constraint (Equation 3.20) is no longer satisfied, resulting in a non-positive allocation and effectively preventing the service provider from participating in the co-investment.

To mitigate this issue, we could introduce the following soft constraint:

$$\xi > \frac{d'e}{\rho} \tag{3.30}$$

We will not enforce this constraint in our model since it is not required and we don't want to limit the profiles of the SPs that take part in the coinvestment to only the ones that process a high percentage of

the total requests at the edge. However, this constraint can provide additional stability. In Section C.1 of Appendix Chapter C, we discuss the potential advantages of enforcing this constraint.

3.10.2 Pseudocode for Solving the Model

Before we get into the strategic implications of our model, we provide a pseudocode to illustrate the practical application of all the simplifications we have found so far.

```
function resolve_game(game: Game):

    initialize players_total_net_utility, total_allocation to 0
    for sp in game.service_providers

        # First derivative of the net utility to calculate the optimal allocation
        allocation = 1/sp.xi * np.log((D * T * sp.benefit_factor * sp.avg_load * sp.xi
    ) / d)

        # positive allocation constraint
        if (allocation < 0)
            return
        else
            sp.allocation = allocation

        # Service Provider net utility
        net_utility = sp.benefit_factor * sp.avg_load * (1 - np.exp(-sp.xi * sp.
    allocation)) *  D * T - d * sp.allocation


        # Shapley Value (payoff) for an SP is half its net utility
        sp.payoff = net_utility / 2
        sp.allocation_payment = d * sp.allocation
        sp.fairness_payment = net_utility / 2
        sp.revenue = net_utility + d * sp.allocation

        players_total_net_utility += net_utility
        total_allocation += sp.allocation

    end for

    no = game.network_owner
    no.payoff = players_total_net_utility / 2
    no.payment = -1 * players_total_net_utility / 2
    no.revenue = 0
```

3.11 Strategic Implications

In this second part of the analytical investigation, we explore two main strategic aspects of the model. First, we assess the risks associated with parameter misestimations by quantifying how much the parameters can deviate before the investment becomes unviable and by evaluating the corresponding losses incurred by the Service Providers. Second, we investigate whether the model is strategy-proof and examine how SPs might benefit from misrepresenting their parameters. This analysis directly addresses a point raised in the "CONCLUSION" section of the referenced article, which states:

> For future work, we will consider adding a strategy-proof enforcement feature to ensure that players are truthful.

After demonstrating that the model is not strategy-proof, we comment on mechanisms or model changes that could potentially enhance strategy-proofness. Finally, we enumerate the modifications to the model that would disrupt the independent contribution property and, as a result, invalidate most of the simplifications identified so far. In later sections, we incorporate some of these changes into our model.

3.12 Evaluating the Impact of Parameters Misestimation

The net utility depends on three parameters ξ, β, and l_{avg}. In this section, we focus on the misestimation of l_{avg} and β and its effect on an SP's payoff.

We treat ξ as fixed, since it reflects the SP's business model. In contrast, l_{avg} (requests per time-slot) and β (the benefit factor) can vary or be estimated incorrectly over the investment horizon. Recall $\rho = l_{\text{avg}} \cdot \beta$; let ρ_{exp} be the expected value at investment time and ρ_{real} its realized value. Once hardware is provisioned, allocations remain fixed, so any gap between ρ_{exp} and ρ_{real} directly alters net utility.

SPs face two risk stages: before declaring parameters (and receiving an allocation) and after allocation is defined but $\rho_{\text{real}} \neq \rho_{\text{exp}}$. We begin by analysing the latter, which is more straightforward.

3.12.1 Effects of Misestimation After the Initial Investment

The current Edge Computing model does not account for deviations from the expected utility function values; therefore, risk management and the strategic implications of misestimating these values were not previously considered. In this section, we will analyze the impact of deviations in ρ_{real}.

If this deviation is downward, the utility function can still be used as originally defined to calculate the SP's net utility. However, if the deviation is upward, we must adjust the interpretation of the utility function to reflect the insufficient resource allocation needed to process the additional requests.

To address this, we assume that once the average load is exceeded, no additional utility will be generated. This can be understood either as a lack of sufficient hardware resources to handle the extra requests, leading to an overflow, or as a decline in overall SP's virtual environment performance caused by the increased load, which counteracts any additional utility from the extra requests. In practical terms, this can be seen as requests that cannot be processed at the Edge Computing infrastructure due to resource limitations.

It is important to note that this assumption may not hold in all scenarios, especially during time-slots with low load. For a more accurate analysis, we could model the extra requests as generating additional utility, but with diminishing returns once the maximum expected load is reached. However, such an analysis would require considering the per time-slot load, which is beyond the scope of this section, since we are analyzing the model under the independent contribution property. That said, if per time-slot allocation were adopted, this type of analysis would become more precise, with the only difference being that the effects of misestimation would need to be computed individually for each time-slot. In our current framework, we assume that no additional requests will be processed and the results should be interpreted as a worst-case scenario.

Considering this, we redefine the utility function as the effective utility function, which considers only the requests that generate utility. This adjustment reflects the fact that once the system exceeds its capacity, any unprocessed requests will no longer contribute to the Service Provider's utility.

$$U_{\text{ts\_net\_ef}}(\rho) \;=\; \begin{cases} \rho_{\text{real}}\big(1 - e^{-\xi h}\big) \;-\; d'\,h, & \rho_{\text{real}} \leq \rho_{\text{exp}}, \\ \rho_{\text{exp}}\big(1 - e^{-\xi h}\big) \;-\; d'\,h, & \rho_{\text{real}} > \rho_{\text{exp}}. \end{cases} \tag{3.31}$$

Underestimation or Opportunity Cost

The equation above demonstrates that in cases of underestimation ($\rho_{\text{real}} > \rho_{\text{exp}}$), recalculating the allocation and its associated cost is necessary to account for the missed potential gains due to insufficient resources. Since in this section we are considering only the economic risks after the allocation has been decided, we can't apply it here, but we want to give the definition for the next subsection.

This concept, known as opportunity cost, represents the additional utility that could have been ob-

tained if the allocation had been optimized for ρ_{real} instead of ρ_{exp}. We denote h_{exp} as the allocation corresponding to ρ_{exp} and, consequently, the deployed one, and h_{real} as the allocation that would optimize the utility for ρ_{real}.

To state all combinations of ρ_{real} and ρ_{exp} with h_{real} and h_{exp}, we introduce a new notation to the net utility function where we assume ξ as fixed and take ρ and h as parameters. This will allow us to express the impact of mis-estimation on the net utility equation.

$$U_{\text{ts\_net}}(\rho, h)$$

In this way, the opportunity cost is defined as:

$$\text{Opportunity Cost} = U_{\text{ts\_net}}(\rho_{\text{real}}, h_{\text{real}}) - U_{\text{ts\_net}}(\rho_{\text{exp}}, h_{\text{exp}})$$

To make this explicit, we can expand on the previous equation.

$$\text{Opportunity Cost} = \left(\rho_{\text{real}} \cdot (1 - e^{-\xi h_{\text{real}}}) - d' \cdot h_{\text{real}}\right) - \left(\rho_{\text{exp}} \cdot (1 - e^{-\xi h_{\text{exp}}}) - d' \cdot h_{\text{exp}}\right)$$

The opportunity cost will be further explored in the subsequent section, where the Service Provider has not yet determined its allocation.

Overestimation

Once the allocation has been defined, it is important for an SP to assess the risk of not covering the initial investment and to quantify how much of the expected revenue could be lost due to a lower than expected ρ_{real}. Since we do not recalculate the allocation, the loss due to the overestimation is directly proportional to the magnitude of the overestimation in ρ.

$$\text{Overestimation of } \rho_{\text{real}} = (\rho_{\text{exp}} - \rho_{\text{real}}) \cdot (1 - e^{-\xi \cdot h_{\text{exp}}}) \tag{3.32}$$

The overestimation is calculated with respect to the expected net utility in response to the lower than anticipated ρ_{real}.

Determining the Breaking Point Where the Investment Becomes Unprofitable

In this section, our aim is to find the breaking point where the overestimation equals the expected payoff, making the investment unprofitable. Furthermore, we will determine the maximum allowable decrease in ρ_{exp} that still keeps the investment viable. Knowing this threshold enables SPs to better assess risks and define their strategies. The following equations demonstrate how, starting from the net utility function, we can derive the condition that ensures that the investment remains profitable, that is, the net utility must be greater than zero.

$$U_{\text{ts\_net}} = \rho_{\text{exp}} \cdot (1 - e^{-\xi \cdot h}) - d' \cdot h$$

$$U_{\text{ts\_net}} > 0 \iff \rho_{\text{exp}} > \frac{d' \cdot h}{1 - e^{-\xi \cdot h}}$$

We now introduce ρ_{zero} as the break-even point where the net utility is equal to zero:

$$\rho_{\text{zero}} = \frac{d' \cdot h}{1 - e^{-\xi \cdot h}} \tag{3.33}$$

Let $p_{100} \in [0, 1]$ be the decimal that represents the maximum percentage decrease from ρ_{exp} that keeps the investment non-deficient, the sub-index 100 indicates a 100% decrease in the expected net utility.

$$\begin{aligned} \rho_{\text{zero}} &= \rho_{\text{exp}} \cdot (1 - p_{100}) \\ \rho_{\text{exp}} \cdot (1 - p_{100}) &= \frac{d' \cdot h}{1 - e^{-\xi \cdot h}} \\ p_{100} &= 1 - \frac{d' \cdot h}{\rho_{\text{exp}} \cdot (1 - e^{-\xi \cdot h})} \end{aligned} \tag{3.34}$$

This last equation quantifies the viability threshold, providing a useful decision-making tool for Service Providers to evaluate risks.

Generalizing the Breaking Point to Any Percentage of Loss

We defined p_{100} as the percentage decrease that results in zero net utility. Similarly, we can define $p_\alpha = p_{100} \cdot \alpha$, for any desired percentage decrease α. This linear relationship allows Service Providers to assess the impact of various levels of misestimation on profitability.

Using p_{100}, we formulate:

$$p_\alpha = p_{100} \cdot \alpha \tag{3.35}$$

Alternatively, by substituting directly into equation 3.34, we derive:

$$p_\alpha = \left(1 - \frac{d' \cdot h}{\rho_{\text{exp}} \cdot (1 - e^{-\xi \cdot h})}\right) \cdot \alpha \tag{3.36}$$

Here, α is a decimal representing the desired percentage of decrease. This formulation provides flexibility in assessing the impact of various levels of misestimation on profitability.

3.12.2 Effects of Misestimation Before the Initial Investment

Now we will evaluate the scenario when the allocation has not yet been defined, so we will have to recalculate the optimal allocation based on ρ_{real} instead of ρ_{exp}. Once again, we will work with two base scenarios: one where the actual value ρ_{real} exceeds the expected value, and another where it is less than expected.

Underestimation or Opportunity Cost ($\rho_{\text{real}} > \rho_{\text{exp}}$)

We have already defined the opportunity cost with equation 3.12.1, it is important to note that this amount does not represent a real cost or an effective loss; rather, it defines the amount of monetary units that could have been obtained. In future analysis, we will equalize opportunity cost to the loss due to overestimation; note that in real-world scenarios a Service Provider may not consider them equal in their business model.

Overestimation ($\rho_{\text{real}} < \rho_{\text{exp}}$)

Symmetrically to how we defined the opportunity cost, we can define the overestimation considering the optimal allocation.

$$\text{Allocation Overestimation} = U_{\text{net}}(\rho_{\text{exp}}, h_{\text{exp}}) - U_{\text{net}}(\rho_{\text{real}}, h_{\text{real}})$$

If we wanted to only account for the overestimation of the allocation but without considering the overestimation of ρ, we can calculate it as follows:

$$\text{Allocation Overcost} = U_{\text{net}}(\rho_{\text{real}}, h_{\text{real}}) - U_{\text{net}}(\rho_{\text{real}}, h_{\text{exp}})$$

So far, we have provided a set of useful equations for SPs to acknowledge the effects of misestimating their parameters. In the next section, we will consider the effects of misestimating h_{real} by a small percentage and then for any arbitrary percentage number. We will consider the case of misestimation in allocation and the case of misestimation in parameter ρ.

If we assume that mechanisms for dynamic allocation are available or that SPs can precisely predict their parameter values, it becomes relevant to study scenarios where deviations in ρ are small, and consequently, the deviations in h are also small. We will generalize this to any percentage of estimated deviation to account for scenarios where the Service Provider cannot precisely predict its parameter values. In both cases, we will find the equilibrium point where underestimation and overestimation have equal impacts.

Considering that the model is not strategy-proof and we aim to introduce a mechanism to enforce it, it is relevant to our study to evaluate the misallocation of h instead of ρ, since the Service Providers can calculate their declared parameters to get the allocation they want.

3.12.3 Allocation Misestimation

Given the diminishing returneffect of the allocation on net utility, we hypothesize that underestimating and overestimating h by the same percentage ϵ will not result in equivalent changes in $U_{\text{ts\_net}}$. This suggests an asymmetry in the impact of misestimating h. To verify this, it is necessary to compare the net utility outcomes of overestimating and underestimating h to determine if similar magnitudes of error in either direction affect $U_{\text{ts\_net}}$ equally.

Considering any percentage deviation expressed in its decimal form ϵ, we can evaluate the effect of both overestimating and underestimating h on net utility $U_{\text{ts\_net}}$, and derive a generalized equilibrium point for any percentage of deviation.

For any percentage deviation ϵ, the overestimation and underestimation of h can be represented as follows:

Overestimating h:

$$U_{\text{ts\_net}}(h(1+\epsilon)) = \rho\left(1 - e^{-\xi h(1+\epsilon)}\right) - d'h(1+\epsilon)$$

Underestimating h:

$$U_{\text{ts\_net}}(h(1-\epsilon)) = \rho\left(1 - e^{-\xi h(1-\epsilon)}\right) - d'h(1-\epsilon)$$

Equating Overestimation and Underestimation

To find the point where the impact of overestimating and underestimating h by ϵ is equal, we set the two previous equations equal:

$$\rho\left(1 - e^{-\xi h(1+\epsilon)}\right) - d'h(1+\epsilon) = \rho\left(1 - e^{-\xi h(1-\epsilon)}\right) - d'h(1-\epsilon)$$

After simplifications, we find the expression for the equilibrium point ρ_{eq}:

$$\rho_{eq} = \frac{d' h\epsilon}{\sinh(\xi h\epsilon)e^{-\xi h}} \tag{3.37}$$

This is the generalized equilibrium point for any percentage deviation ϵ. It provides a reference point at which the impacts of overestimation and underestimation of h on net utility are equal.

Moreover, since $\sinh(\xi\, h\, \epsilon) > \xi\, h\, \epsilon$ for any $\epsilon > 0$, we have

$$\rho_{eq} = \frac{d'\, h\, \epsilon}{\sinh(\xi\, h\, \epsilon)\, e^{-\xi\, h}} < \frac{d'\, h\, \epsilon}{(\xi\, h\, \epsilon)\, e^{-\xi\, h}} = \frac{d'}{\xi}\, e^{\xi\, h} = \rho_{\text{exp}}.$$

Note that the expression at the right is the same as the one we get by solving the optimal allocation equation for ρ. Hence ρ_{eq} is always strictly less than the real expected value ρ_{exp}. Furthermore, the function

$$f(\epsilon) \;=\; \frac{\epsilon}{\sinh(\xi\, h\, \epsilon)}$$

is strictly decreasing for $\epsilon > 0$, which implies that ρ_{eq} decreases as ϵ increases.

A detailed, step-by-step derivation of this result is provided in the demonstrations chapter of the appendix (see Appendix B.6).

3.12.4 Practical Implications for Service Providers

The previous analysis yields a single equilibrium expression ρ_{eq} for any percentage deviation ϵ, as given by equation 3.37. Since $\rho_{eq} < \rho_{\text{exp}}$ for all $\epsilon > 0$, any actual ρ_{exp} will lie above this equilibrium. As a result, modest overestimation of h carries a larger utility penalty than a comparable underestimation, and vice versa.

These observations about the asymmetry in misestimating h provide useful insight when we study strategy-proofness. In particular, knowing that ρ_{eq} always falls below ρ_{exp} helps frame how an SP might combine this risk profile with transferring half of its declared net utility: the balance between real revenue and declared transfers will ultimately determine whether underreporting yields a higher retained payoff.

The key reason ρ_{eq} always falls below ρ_{exp} is that utility depends on ρ almost linearly (through the factor $1 - e^{-\xi h}$), whereas the allocation h itself is only a logarithmic function of ρ. When we perturb h by $\pm\epsilon$, the term $e^{-\xi h}$ changes by a factor $e^{\pm\xi\, h\, \epsilon}$, which grows or shrinks exponentially in ϵ, while the cost term $d'\, h$ changes only linearly in ϵ. Because $\sinh(\xi\, h\, \epsilon)$ captures the difference between $e^{\xi\, h\, \epsilon}$ and $e^{-\xi\, h\, \epsilon}$, the equation that balances "overestimating" versus "underestimating" h forces ρ to be smaller than the nominal $\rho_{\text{exp}} = \frac{d'}{\xi} e^{\xi\, h}$. In other words, the exponential sensitivity of revenue to changes in h always overpowers the linear sensitivity of cost, so the unique ρ that makes those two one-sided errors cancel must lie below the value that exactly maximizes the unperturbed allocation.

3.13 Exploring Strategy Proofness

From a game-theory perspective, a dominant strategy is one that yields a payoff for a player that is at least as high as the payoff from any other strategy, regardless of what strategies the other players choose.

A model is considered strategy-proof if truthfulness is a dominant strategy for every participant. In other words, no participant can improve their outcome by misrepresenting their true preferences or information.

In our model, individual contribution guarantees ensure that one SP's misrepresentation does not affect another SP's outcome. However, the NO, who collects half of each SP's net utility, relies on accurate reports and would be affected by any misrepresentation.

Parameters in the utility function quantify real-world interactions, but differ in their susceptibility to misreporting. Specifically, β is more prone to being misrepresented than l_{avg}, since the NO can directly measure the SP's load. The parameter ξ lies in between: it is not defined how it could be directly observed but can sometimes be inferred from an SP's business model or by measuring the need for allocated hardware each request creates. For the purpose of this analysis, we assume that the NO only has precise data for l_{avg}, so any number submitted for β and ξ must be taken at face value. Because β partly defines the parameter $\rho = \beta \cdot l_{\text{avg}}$, this effectively makes ρ subject to misrepresentation.

3.13.1 Getting the Optimal Allocation with Two Misrepresented Parameters

In this Section we prove the model is not strategy-proof by presenting a procedure that enables an SP to achieve the optimal allocation while declaring any desired net utility. Additionally, we provide a more detailed analytical demonstration to gain deeper insights into the model's behavior.

Constructive Method

In this demonstration, we define ξ_{dec} and ρ_{dec} as declared parameters and $U_{\text{ts\_net}}^{declared}$ as declared net utility. The core idea is to use the allocation optimization equation 3.19 to first calculate the optimal allocation h. Then substitute its expression for the optimal allocation solved for ρ 3.22 into the declared net utility function. This gives us an expression for the net utility declared with ξ_{dec} as a variable, $U_{ts\_net}^{declared}(\xi_{dec})$. Finally, by using the declared net utility equation again, we can determine the value of ρ_{dec} corresponding to the value of ξ_{dec} that has been selected.

First, we calculate the allocation h^* according to 3.19

$$h^* = \frac{1}{\xi} \ln\left(\frac{\rho\xi}{d'}\right)$$

Define the declared net utility function:

$$U_{ts\_net}^{declared}(\xi_{dec}, \rho_{dec}) = \rho_{dec}\left(1 - e^{-\xi_{dec}h}\right) - d'h \tag{3.38}$$

Once we find $U_{ts\_net}^{declared}(\xi_{dec})$ using 3.22, we rearrange this equation to solve for ρ_{dec}:

$$\rho_{dec} = \frac{U_{ts\_net}^{declared}(\xi_{dec}) + d'h}{1 - e^{-\xi_{dec}h}}$$

After substitutions and simplification, the expression for ρ_{dec} is as follows:

$$\rho_{dec} = \frac{d'e^{\xi_{dec}h}}{\xi_{dec}} \tag{3.39}$$

Using this equation, SPs can manipulate their declared utility by selecting a desired value for ξ_{dec}, and then calculating the corresponding value of ρ_{dec}.

This method allows SPs to declare parameters ξ_{dec} and ρ_{dec} that maintain the optimal allocation for the real values of ξ and ρ, while also achieving the desired declared net utility $U_{\text{net}}^{declared}$.

A more detailed step-by-step demonstration of this constructive method along with an analytical demonstration is provided in the section B.8. Additionally, in the practical chapter, we provide a numerical example in section 4.4.2

3.13.2 Strategic Misreporting of One Parameter

We previously considered the case where both ρ (through β) and ξ were declared by SP and unknown to NO. We showed SP can obtain the optimal allocation while declaring any desired net utility, proving the model is not strategy-proof and that SPs can arbitrarily profit from misreporting . In the next two subsections, we examine the case where only one parameter is private.

Misreporting ρ

Recall that the SP transfers half of its declared net utility to the NO. In this scenario, the NO can measure ξ but cannot verify ρ, so the SP may declare a value $\rho_{\text{dec}} \neq \rho_{\text{real}}$.

We begin by writing:

- Declared utility (based on ρ_{dec} and the induced allocation h_{dec}):

$$U_{\text{dec}} = \rho_{\text{dec}}\big(1 - e^{-\xi\, h_{\text{dec}}}\big) \;-\; d'\, h_{\text{dec}}$$

- Actual utility (using the true ρ_{real} but the same allocation h_{dec}):

$$U_{\text{real}} = \rho_{\text{real}}\big(1 - e^{-\xi\, h_{\text{dec}}}\big) \;-\; d'\, h_{\text{dec}}$$

- Utility retained by the SP (it keeps all of U_{real} minus half of U_{dec}):

$$U_{\text{SP}}(\rho_{\text{dec}}) = U_{\text{real}} - \tfrac{1}{2}\, U_{\text{dec}}$$

Next, recall that, even when the SP misreports, the mechanism chooses the allocation

$$h_{\text{dec}} \;=\; \frac{1}{\xi}\,\ln\!\Big(\frac{\rho_{\text{dec}}\,\xi}{d'}\Big)$$

We now substitute this into U_{dec} and U_{real} and compute U_{dec} in closed form. Since $e^{-\xi\, h_{\text{dec}}} = d'/(\rho_{\text{dec}}\,\xi)$, we have:

$$U_{\text{dec}} = \rho_{\text{dec}}\Big(1 - \tfrac{d'}{\rho_{\text{dec}}\,\xi}\Big) \;-\; d'\,\frac{1}{\xi}\,\ln\!\Big(\frac{\rho_{\text{dec}}\,\xi}{d'}\Big)$$

This simplifies to:

$$U_{\text{dec}} = \rho_{\text{dec}} - \frac{d'}{\xi} \;-\; \frac{d'}{\xi}\,\ln\!\Big(\frac{\rho_{\text{dec}}\,\xi}{d'}\Big)$$

Now we compute U_{real} in closed form, similarly to what we just did:

$$U_{\text{real}} = \rho_{\text{real}}\Big(1 - \tfrac{d'}{\rho_{\text{dec}}\,\xi}\Big) \;-\; d'\,\frac{1}{\xi}\,\ln\!\Big(\frac{\rho_{\text{dec}}\,\xi}{d'}\Big)$$

Which simplifies to:

$$U_{\text{real}} = \rho_{\text{real}} - \frac{\rho_{\text{real}}\,d'}{\rho_{\text{dec}}\,\xi} \;-\; \frac{d'}{\xi}\,\ln\!\Big(\frac{\rho_{\text{dec}}\,\xi}{d'}\Big)$$

By definition, the retained net utility is:

$$U_{\mathrm{SP}}(\rho_{\mathrm{dec}}) = U_{\mathrm{real}} \;-\; \tfrac{1}{2}\,U_{\mathrm{dec}}$$

Now we substitute the expression we just derived:

$$U_{\mathrm{SP}}(\rho_{\mathrm{dec}}) = \Big[\rho_{\mathrm{real}} - \tfrac{\rho_{\mathrm{real}}\,d'}{\rho_{\mathrm{dec}}\,\xi} - \tfrac{d'}{\xi}\,\ln\!\Big(\tfrac{\rho_{\mathrm{dec}}\,\xi}{d'}\Big)\Big] \;-\; \tfrac{1}{2}\Big[\rho_{\mathrm{dec}} - \tfrac{d'}{\xi} - \tfrac{d'}{\xi}\,\ln\!\Big(\tfrac{\rho_{\mathrm{dec}}\,\xi}{d'}\Big)\Big]$$

After simplifications, we arrive at:

$$U_{\mathrm{SP}}(\rho_{\mathrm{dec}}) = \rho_{\mathrm{real}} - \frac{\rho_{\mathrm{real}}\,d'}{\rho_{\mathrm{dec}}\,\xi} \;-\; \frac{d'}{2\,\xi}\,\ln\!\Big(\frac{\rho_{\mathrm{dec}}\,\xi}{d'}\Big) \;+\; \frac{d'}{2\,\xi} \;-\; \frac{\rho_{\mathrm{dec}}}{2}$$

This single algebraic expression summarizes the SP's retained utility as a direct function of its declared parameter ρ_{dec}.

Finally, we want to find the optimal misreport. To maximize $U_{\mathrm{SP}}(\rho_{\mathrm{dec}})$ over all feasible ρ_{dec} (subject to $\rho_{\mathrm{dec}}\,\xi > d'$), we compute the derivative with respect to ρ_{dec} and set it equal to zero:

$$\frac{d\,U_{\mathrm{SP}}}{d\,\rho_{\mathrm{dec}}} = \frac{\rho_{\mathrm{real}}\,d'}{\rho_{\mathrm{dec}}^2\,\xi} - \frac{d'}{2\,\rho_{\mathrm{dec}}\,\xi} - \frac{1}{2} \;=\; 0$$

Multiply through by $2\,\rho_{\mathrm{dec}}^2\,\xi$ to clear denominators:

$$2\,\rho_{\mathrm{real}}\,d' \;-\; d'\,\rho_{\mathrm{dec}} \;-\; \rho_{\mathrm{dec}}^2\,\xi \;=\; 0$$

or equivalently

$$\rho_{\mathrm{dec}}^2 + \frac{d'}{\xi}\,\rho_{\mathrm{dec}} - \frac{2\,d'\,\rho_{\mathrm{real}}}{\xi} = 0$$

Solving this quadratic for ρ_{dec} and taking the positive root yields the unique maximizer:

$$\rho_{\mathrm{dec}}^{*} = \frac{\frac{d'}{\xi}\left[-1 + \sqrt{1 + \frac{8\,\rho_{\mathrm{real}}\,\xi}{d'}}\right]}{2} \tag{3.40}$$

By construction, ρ_{dec}^{*} satisfies $\rho_{\mathrm{dec}}^{*}\,\xi > d'$, meaning that the allocation is positive. Now that we got the formula for the optimal misreporting of (ρ_{dec}^{*}) we want to compare the retained net utility an SP can get by misreporting this value. For that, we define the retained net utility obtained through a truthful declaration and the retained net utility obtained through an untruthful declaration:

$$U_{\mathrm{truth}} \;=\; U_{\mathrm{SP}}(\rho_{\mathrm{real}}) \quad \text{and} \quad U_{\mathrm{opt}} \;=\; U_{\mathrm{SP}}(\rho_{\mathrm{dec}}^{*})$$

Since we already have the general formula:

$$U_{\mathrm{SP}}(\rho_{\mathrm{dec}}) = \rho_{\mathrm{real}} - \frac{\rho_{\mathrm{real}}\,d'}{\rho_{\mathrm{dec}}\,\xi} - \frac{d'}{2\,\xi}\,\ln\!\Big(\frac{\rho_{\mathrm{dec}}\,\xi}{d'}\Big) + \frac{d'}{2\,\xi} - \frac{\rho_{\mathrm{dec}}}{2}$$

we substitute $\rho_{\mathrm{dec}} = \rho_{\mathrm{real}}$ and $\rho_{\mathrm{dec}} = \rho_{\mathrm{dec}}^{*}$ separately:

$$U_{\mathrm{truth}} = \rho_{\mathrm{real}} - \frac{\rho_{\mathrm{real}}\,d'}{\rho_{\mathrm{real}}\,\xi} - \frac{d'}{2\,\xi}\,\ln\!\Big(\frac{\rho_{\mathrm{real}}\,\xi}{d'}\Big) + \frac{d'}{2\,\xi} - \frac{\rho_{\mathrm{real}}}{2} = \frac{\rho_{\mathrm{real}}}{2} \;-\; \frac{d'}{2\,\xi} \;-\; \frac{d'}{2\,\xi}\,\ln\!\Big(\frac{\rho_{\mathrm{real}}\,\xi}{d'}\Big)$$

$$U_{\mathrm{opt}} = \rho_{\mathrm{real}} - \frac{\rho_{\mathrm{real}}\,d'}{\rho_{\mathrm{dec}}^{*}\,\xi} - \frac{d'}{2\,\xi}\,\ln\!\Big(\frac{\rho_{\mathrm{dec}}^{*}\,\xi}{d'}\Big) + \frac{d'}{2\,\xi} - \frac{\rho_{\mathrm{dec}}^{*}}{2}$$

with ρ^*_{dec} as defined in equation 3.40. Finally, define the retention-ratio and percentage gain as:

$$R \;=\; \frac{U_{\text{opt}}}{U_{\text{truth}}} \qquad \text{Gain} \;=\; (R-1)\times 100\%.$$

Computing R for any fixed triple $(\rho_{\text{real}},\, \xi,\, d')$ shows exactly how much retained utility the SP gains by lying optimally instead of telling the truth. In particular, as $\rho_{\text{real}} \to \infty$

$$\rho^*_{\text{dec}} \;\approx\; \sqrt{\frac{2\,d'\,\rho_{\text{real}}}{\xi}}, \quad U_{\text{opt}} \;\approx\; \rho_{\text{real}}, \quad U_{\text{truth}} \;\approx\; \frac{\rho_{\text{real}}}{2}, \quad R \;\approx\; 2, \quad \text{Gain} \;\approx\; 100\%$$

Thus in the large ρ limit, the SP can roughly double its retained net utility by optimally understating ρ.

Finally we use equation (3.40) to give a declared value of ρ and supposing that this value is ρ^*_{dec} find the corresponding value of ρ_{real}.

$$\rho_{\text{real}} = \frac{\rho^*_{\text{dec}}}{2} \;+\; \frac{\xi\,(\rho^*_{\text{dec}})^2}{2\,d'}$$

Misreporting ξ

Assume ρ is public and ξ is private. SP declares ξ_{dec}

$$h_{\text{dec}} = \frac{1}{\xi_{\text{dec}}}\,\ln\!\Big(\frac{\rho\,\xi_{\text{dec}}}{d'}\Big),$$

valid when $\rho\,\xi_{\text{dec}} > d'$.

Utilities

- Declared net utility:

$$U_{\text{dec}} = \rho\Big[1 - \frac{1+\ln\!\big(\frac{\rho\,\xi_{\text{dec}}}{d'}\big)}{\frac{\rho\,\xi_{\text{dec}}}{d'}}\Big] = \rho\Big[1 - \frac{d'}{\rho\,\xi_{\text{dec}}}\big(1+\ln(\tfrac{\rho\,\xi_{\text{dec}}}{d'})\big)\Big].$$

- Real net utility:

$$U_{\text{real}} = \rho\Big[1 - \big(\tfrac{\rho\,\xi_{\text{dec}}}{d'}\big)^{-\xi_{\text{real}}/\xi_{\text{dec}}} - \frac{\ln\!\big(\frac{\rho\,\xi_{\text{dec}}}{d'}\big)}{\frac{\rho\,\xi_{\text{dec}}}{d'}}\Big].$$

- Utility retained by SP:

$$U_{\text{SP}} = U_{\text{real}} - \tfrac{1}{2}\,U_{\text{dec}}.$$

Solving $\partial U_{\text{SP}}/\partial \xi_{\text{dec}} = 0$ yields a unique solution to the optimal declaration condition.

$$\xi^*_{\text{dec}} < \xi_{\text{real}} \quad \text{whenever} \quad \rho\,\xi_{\text{real}} > d'.$$

Note that the function

$$h_{\text{dec}}(\xi_{\text{dec}}) = \frac{1}{\xi_{\text{dec}}}\,\ln\!\big(\tfrac{\rho\,\xi_{\text{dec}}}{d'}\big)$$

attains its maximum at

$$\xi_{\text{peak}} = \frac{d'\,e}{\rho}, \quad h_{\max} = \frac{\rho}{d'\,e},$$

and for any $h < h_{\max}$ there are two values of ξ_{dec} one below and one above ξ_{peak} that yield the same allocation h_{dec}. SP can therefore choose the smaller of these two ξ_{dec} to reduce its declared utility without changing h_{dec}.

Understating ξ reduces the transfer more than the loss from the smaller h, so SP's utility strictly increases. To understand how much a SP can profit from this misrepresentation, we compare:

$$U_{\text{truth}} = U_{\text{SP}}(\xi_{\text{real}}) \quad \text{and} \quad U_{\text{opt}} = U_{\text{SP}}(\xi^*_{\text{dec}}).$$

Define the gain ratio and the gain in percentage:

$$R = \frac{U_{\text{opt}}}{U_{\text{truth}}}, \qquad \text{Gain} = (R-1) \times 100\%.$$

For fixed ρ, ξ_{real}, and d', we compute R and

$$\text{Gain}(\xi_{\text{real}}) = (R-1) \times 100\%$$

to see exactly how much an SP can increase its retained utility by optimally understating ξ.

In the practical Chapter, we provide a numerical example of this misreporting 4.6

3.14 Conclusions on Strategy-Proofness

The previous analysis confirms that the model is not strategy-proof:

- When both ρ and ξ are private, an SP can choose any ξ_{dec} and compute a matching ρ_{dec} that preserves the optimal allocation h^* while declaring any desired net utility.
- When only one parameter is private:
  - Under-declaring ρ trades off a slightly smaller h^* against lower transfers, strictly increasing the SP's retained utility.
  - Under-declaring ξ similarly reduces transfers by more than the loss in allocation, again raising retained utility.
- In every scenario, the dominant strategy for an SP is to under-declare its private parameter(s) rather than over-declare.
- As ρ_{real} or ξ_{real} tend to infinity, the maximum percentage gain from under-declaring approaches 100%.
- For realistic, finite parameter values, a finite upper bound exists on the percentage gain achievable by under-declaring.

3.15 Proposing Mechanisms to Enhance Strategy-Proofness

The previous section showed that the current Edge Computing model is not strategy-proof. To achieve fair and incentive-compatible outcomes, we can either extend the model with external mechanisms or modify it.

In mechanism design and cooperative game theory, the Vickrey-Clarke-Groves (VCG) mechanism is a standard tool for enforcing truthfulness. We describe below why VCG is infeasible in our setting. We then consider post-initial allocation market mechanisms, which introduce a secondary trading stage to discourage misrepresentation.

3.15.1 Considering the Application of the Vickrey-Clarke-Groves (VCG) Mechanism

The VCG mechanism maximizes the reported social welfare, which in our model corresponds to the grand coalition's total net utility under the optimal allocation. Each SP is charged a transfer equal to the externality it imposes on the rest of the coalition, ensuring that truthful reporting becomes a dominant strategy. This mechanism is feasible when the total value of the grand coalition is known, typically because payments flow into a common account, even if individual contributions remain unknown. In our scenario, however, each SP's revenue remains private, preventing verification of the coalition's total net utility. As a result, the VCG mechanism cannot be effectively enforced.

3.15.2 Considering Post-Allocation Market Mechanisms

This approach preserves the original model but adds a secondary market after the initial resource allocation. SPs may buy or sell allocation shares through auctions or bilateral trades. For example, a new SP entering the coalition or an existing SP that underestimated its needs can purchase additional allocation at a price equal to the seller's expected remaining revenue, based on the seller's declared utility. Sellers are obliged to complete the sale.

Under this scheme, SPs that declared truthfully receive their anticipated profits sooner when they sell part of their allocation. Those that misrepresented their utility risk getting purchased their allocation at a lower value than the one they would get. By acquiring allocations from SPs with lower per-unit net utility, honest SPs are rewarded, and misreporting is partially deterred.

However, an SP aware of having high per-allocated unit net utility can still benefit by slightly understating its value to reduce the declared net utility while most likely retaining all the allocation. Conversely, an SP with relatively low per-allocation unit net utility might choose to hold its allocation rather than selling, preferring continued low-latency service delivery over immediate revenue. Such a scenario could discourage some truthful SPs from wanting to sell their allocation. However, this scenario is unlikely, as low-latency SPs typically have a higher benefit factor.

Different approaches to these market-based mechanisms are explored in the related prosumer academic articles. In Chapter A of the Appendix, we explore and summarize each of these articles. These mechanisms can improve incentive compatibility and overall efficiency, and extending trades to per-time-slot units further enhances flexibility and risk management. Nevertheless, because SPs can still find profitable misreporting strategies, post-allocation markets alone cannot fully guarantee strategy-proofness.

3.15.3 Externally Inferring the Diminishing Returns Parameter

The specific method used to share hardware infrastructure among SPs is beyond our scope. But in any scenario of virtualization, the parameter ξ can be inferred by analyzing the relation between served requests and required hardware resources. Specifically, ξ quantifies the amount of computing resources needed to attend to a certain amount of requests in a certain amount of time.

A higher ξ implies that minimal extra resources are enough as potential monetization increases, while a lower ξ indicates more gradual scaling, typical of background-processing services. Section 4.2.2 presents empirical insights of this behavior in our current model and Section 5.2 formalizes a revised definition of ξ that makes it directly measurable by the NO.

If ξ is externally measured, the only remaining private parameter is the benefit factor β. Misreporting only β prevents an SP from attaining the full optimal allocation, and the profit from misreporting is limited by the true value of β. Moreover, since β captures time-sensitivity of service delivery, it can serve as a priority indicator: declaring a higher β grants lower-latency responses at the expense of higher payments.

This introduces a trade-off: under-declaring β raises retained utility but reduces response priority,

whereas over-declaring β improves latency at the cost of lower net gains. When combined with post-allocation market mechanisms previously discussed, where SPs can buy or sell allocation shares after the initial assignment based on their declared utility, the benefits of truthful β reporting can outweigh the misreporting incentives, thus reinforcing honest declarations and balancing latency against utility retention.

3.16 Limits of the Independent Contribution Property

As seen in previous sections, the model simplifications we found were possible as a consequence of the independent contribution property. While this property greatly simplifies the mathematical and computational aspects of the game, it is important to acknowledge that it assumes that no externalities or synergies among SPs are introduced. The presence of such factors could significantly alter the dynamics and outcomes of the game. Below are some scenarios that would disrupt the independent contribution property and, in consequence, invalidate the equivalence between how the model was originally proposed and our simplified version:

- Global Coupling Constraint: Introducing a global constraint such as requiring the total allocation to fall within a minimum or maximum threshold for co-investment to take place imposes a dependency among SPs' value functions and breaks the independent contribution property. When the sum of individually optimal allocations falls outside this range, players will not be able to achieve their individual optimal allocations, resulting in strictly worse net utility. If the total allocation is below the minimum, co-investment may not occur or may yield reduced returns for all players. Conversely, if it exceeds the maximum, players may be forced to scale down their allocations, even when their individual optimum is higher.
- Variable Resource Unit Pricing: Moving away from a fixed price per resource unit, to a sublinear pricing model would give a greatter Shapley value to SPs with greatter needed allocations.
- Dynamic Allocation: Allowing allocations to vary by time-slot rather than being fixed would introduce variability in utility calculations and strategic considerations. This would also imply working with each time-slot load instead of the average load. In this case, SPs whose peak load is offset from that of the others would benefit.
- Energy Costs Variability: Incorporating the cost of electricity per allocated resource unit, if these costs vary or allow for the deactivation of resources in certain time-slots, would introduce a cost factor that fluctuates based on external conditions or operational decisions.
- Demand Interdependence (Complementarity or Substitutability): If the value generated by one SP depends on the allocations of others because their services are complementary or substitutable, then contributions can no longer be assessed independently.

3.16.1 Considering Positive Externalities and Synergies

Adding positive externalities or synergies among players would necessitate reevaluating the individual contribution approach. In such cases, the simplified model may no longer accurately capture the strategic and economic dynamics of the system. However, under certain conditions, this simplified version can still provide meaningful insights.

If only some of the previously discussed modifications are introduced, such as sublinear per-unit pricing, per time-slot allocation, or demand interdependence in the case of complementary services, the simplified model can still serve as a valid baseline. Although these changes disrupt the independent contribution assumption, the simplified model remains a useful benchmark. In particular, it can be interpreted as a "worst-case" scenario for individual Service Providers, as the lack of synergies or cooperative gains results in lower potential utility compared to a model that explicitly accounts for them.

In Chapter 5, we incorporate some of the previously discussed modifications that disrupt the independent contribution property. Specifically, we analyze the effects of introducing a sublinear per-unit pricing scheme, enabling dynamic allocation across time-slots, and enforcing minimum and maximum total allocation thresholds required for co-investment to occur. Then, in Chapter 6, we visualize the impact of these changes and compare the outcomes of the modified model with those of the baseline scenario developed under the assumption of independent contributions.

Chapter 4

Empirical Illustration and Results Analysis of the Originally Proposed Edge-Computing Model

4.1 Introduction

This chapter presents numerical experiments and visualizations designed to corroborate the theoretical insights established in the previous chapter. We begin by reproducing and critically reviewing the simulation results from the original article (Section V, "APPLICATION TO EDGE COMPUTING"). We then conduct a sensitivity analysis on all the variables, visualize the results through charts, and interpret them. Finally, we show how misestimation of ρ and ξ influences optimal allocation, net utility, and SP strategic gains through numerical examples.

4.2 Analyzing the Simulations Results of the Referenced Article

To conclude our analysis of the referenced article, and before conducting our own simulations and proposing modifications to the model, we first interpret the simulation results presented in Section "V. APPLICATION TO EDGE COMPUTING" of the original article. To support this interpretation, we reproduce charts and extracts directly from that article. The load and utility functions defined in Section "A. Parameters" have already been discussed in the previous analytical chapter. Therefore, we begin our analysis directly with Section "B. Scenario with 2 SPs of the same type".

4.2.1 B. Scenario with 2 SPs of the same type

Below we copy the extract where the scenario is set:

> In this case, there are two SPs of the same type: $\beta^{SP^1} = \beta^{SP^2} = \hat{p}$ where $\hat{p} \triangleq \frac{d}{D \cdot T}$ is the price, $d = 0.05$ dollars/millicores, amortized over each of T time-slots over the investment duration, D. SP^1 and SP^2 have the same temporal trends, but $l_t^1 = 4l_t^2 \quad \forall t$. For simplicity, we use the simplified equations for one time-slot, using the amortized CPU price.

In this section, the authors set the value of β equal to the amortized price d' (we call it d' to keep consistency with the previous definitions but is the same as $\hat{p}$). In practice, these variables would

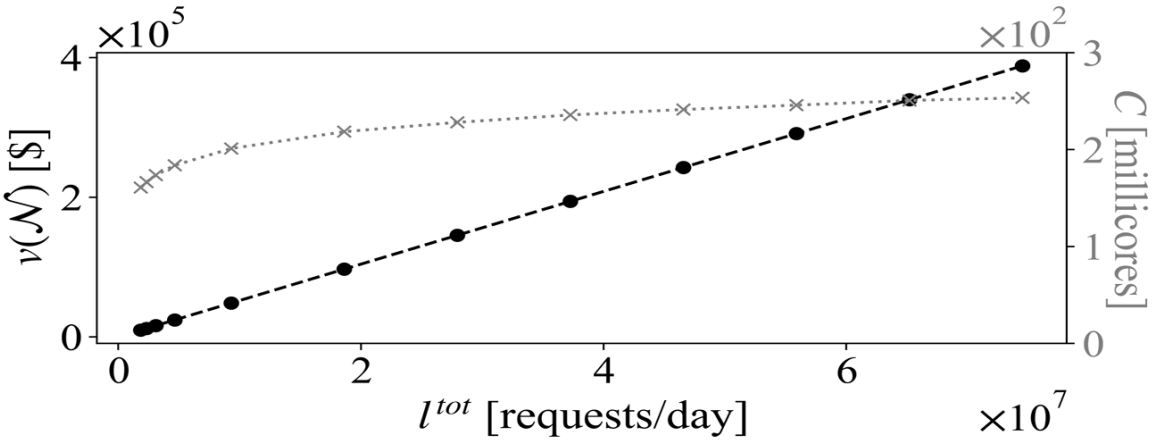


Figure 4.1: Figure 1. Scenario with 2 SPs of the same type

not typically be related, since β should be a free parameter set by the SP, representing how requests served at the edge translate into monetary units while d' is a time-slot dependent variable. There is no intrinsic reason for relating them. However, as we will observe, these two variables cancel each other out in the optimal allocation equation, simplifying it considerably. The value of ξ is not explicitly provided in this section, but since the scenario involves two SPs "of the same type", it must be the case that their parameters match, $\xi^{\mathrm{SP1}} = \xi^{\mathrm{SP2}}$.

Recall the original net utility equation:

$$U_{\mathrm{ts\_net}} = \rho\left(1 - e^{-\xi h}\right) - d'h.$$

The corresponding optimal allocation equation is:

$$h^* = \frac{1}{\xi} \ln\left(\frac{\rho\,\xi}{d'}\right).$$

With the chosen condition $\beta = d'$, these equations simplify to:

$$h^* = \frac{1}{\xi} \ln(\xi).$$

Substituting this simplified allocation back into the time-slot net utility equation gives:

$$U_{\mathrm{ts\_net}} = \frac{\rho(\xi - 1) - d' \ln(\xi)}{\xi}.$$

In the figure 4.1 we paste the first chart from this simulation along with its interpretation as presented in the original article:

> In Fig. 4.1, we show the capacity of purchased CPU and the value of the grand coalition, as a function of the daily total load, $l^{\mathrm{tot}} = \sum_{t=1}^{T} l_t^{\mathrm{tot}}$. We observe that, the more the load the more the capacity installed to serve it. However, recall that, the utility functions follow a diminishing return with respect to the resources, so, the trend of the capacity C is sublinear. We observe a linear trend for coalitional value because the value function is linearly dependent on the load (see Eqn. (17)).

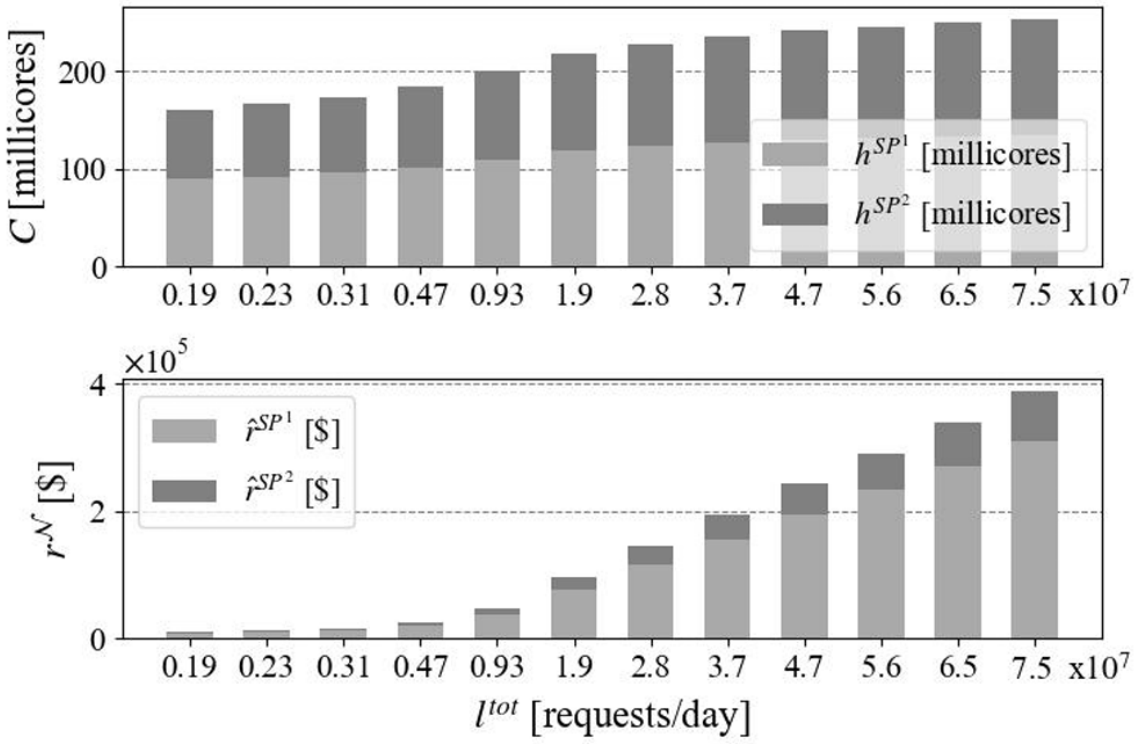


Figure 4.2: Figure 2. Scenario with 2 SPs of the same type

The authors correctly identify that the trend of C is sublinear; we further clarify that, specifically, this relationship is logarithmic. Additionally, they claim the coalitional value exhibits a linear trend. Since the coalitional value equals the sum of individual net utilities, we add a minor correction here. Because each individual net utility $U_{\text{ts\_net}}$ is strictly convex in its load argument (having a positive second derivative), their sum is also strictly convex with respect to the total load l^{tot}. Therefore, the grand coalition value function $V(l^{\text{tot}})$ exhibits increasing marginal returns at moderate loads (additional load generates increasingly larger marginal gains), eventually approaching a linear asymptote. Thus, rather than strictly linear, the grand-coalition value initially experiences accelerated growth and later transitions to linear scaling behavior.

In figure 4.2 we reproduce the second chart from this simulation, and below we present an extract of the interpretation provided in the original article:

> We observe in Fig. 4.2 capacity sharing between the SPs, SP^1 receives a larger capacity: it has to serve a larger part of the requests. Note that, even if the load of SP^1 is 4 times the load of SP^2, the difference between the resource allocated to them is not that big: a consequence of the diminishing return.

It is important to clarify here that the observed behavior is indeed a consequence of diminishing returns, but specifically the diminishing returns of the optimal allocation with respect to the variables l^{tot} and β, not directly the diminishing returns of utility with respect to allocation. Additionally, as a direct consequence of this phenomenon, the proportional difference in allocation shrinks as load grows larger.

To illustrate this clearly, consider a simple numerical example under conditions identical to the scenario presented:

- $\beta^{SP^1} = \beta^{SP^2} = d'$

- $l_t^1 = 4\, l_t^2$
- $d' = \frac{0.05}{D \cdot T}$

Case 1:

Set the number of daily time-slots to 100 and assume a daily load of $l_t^1 = 400,000$, evenly distributed. Since SPs are of the same type, we set $\xi = 0.1$ for both. Then, $\beta = d' = \frac{0.05}{D \cdot T} = 4.566 \times 10^{-7}$. The optimal allocation is computed as follows:

$$h_i^* = \frac{\log(l_t^i \cdot \xi)}{\xi}$$

Then:

$$h_{1_l=4000}^* = \frac{log(400)}{0.1} \approx 59.91$$

$$U_{\text{ts\_net\_l=4000}}^1 = 4000 \cdot 4.566 \times 10^{-7} \left(1 - e^{-5.991}\right) - 4.566 \times 10^{-7} \cdot 59.91 \approx 1.79 \times 10^{-3}$$

$$h_{2_l=1000}^* = \frac{log(100)}{0.1} \approx 46.05$$

$$U_{\text{ts\_net\_l=1000}}^2 = 1000 \cdot 4.566 \times 10^{-7} \left(1 - e^{-4.605}\right) - 4.566 \times 10^{-7} \cdot 46.05 \approx 4.31 \times 10^{-4}$$

The resulting relationships are:

$$\frac{h_1^*}{h_2^*} \approx 1.2, \quad \frac{U_{\text{ts\_net}}^1}{U_{\text{ts\_net}}^2} \approx 4.2.$$

Case 2:

For the same 100 daily time-slots but now with $l_t^1 = 40,000$:

$$h_{1_l=400}^* = \frac{log(40)}{0.1} \approx 36.88$$
$$U_{\text{ts\_net\_l=400}}^1 = 400 \cdot 4.566 \times 10^{-7} \left(1 - e^{-3.688}\right) - 4.566 \times 10^{-7} \cdot 36.88 \approx 1.61 \times 10^{-4}$$
$$h_{2_l=100}^* = \frac{log(10)}{0.1} \approx 23.02$$
$$U_{\text{ts\_net\_l=100}}^2 = 100 \cdot 4.566 \times 10^{-7} \left(1 - e^{-2.302}\right) - 4.566 \times 10^{-7} \cdot 23.02 \approx 3.06 \times 10^{-5}$$

Now, the relationships are:

$$\frac{h_1^*}{h_2^*} \approx 1.6, \quad \frac{U_{\text{ts\_net}}^1}{U_{\text{ts\_net}}^2} \approx 5.3.$$

Observe that when the load is reduced to 10% of its initial value, the optimal allocation reduces to approximately 61% for SP^1 and approximately 50% for SP^2. This confirms that as the total load grows, the relative allocation difference diminishes.

Regarding net utility, when the load drops to 10% of its initial value, the utility shrinks to approximately 9% for SP^1 and approximately 7% for SP^2. Furthermore, notice that the utility ratio approaches 4 as load increases while maintaining the ratio $l_t^1 = 4\, l_t^2$. Formally, for SPs with identical β:

$$\lim_{load \to \infty} \frac{U_{\text{net}}(n \cdot load)}{U_{\text{net}}(load)} = n.$$

This result holds true even if the SPs have different ξ values, because verifying this limit for players with the same β but different ξ values is equivalent to proving:

$$\lim_{\rho\to\infty} \frac{U_{\text{net}}(\rho,\xi)}{\rho} = 1.$$

We proceed with the demonstrations of this last equation. We begin with an equivalent form of the net utility for the optimal allocation expression 3.25

$$U_{\text{net}}(\rho,\xi) = \rho\Big(1 - e^{-\xi\, h^*}\Big) \;-\; d\, h^* = \rho \;-\; \frac{d}{\xi}\Big[\,1 + \ln\!\big(\tfrac{\rho\,\xi}{d}\big)\Big].$$

Divide through by ρ:

$$\frac{U_{\text{net}}(\rho,\xi)}{\rho} = 1 \;-\; \frac{d}{\xi\,\rho}\Big[\,1 + \ln\!\big(\tfrac{\rho\,\xi}{d}\big)\Big].$$

As $\rho \to \infty$, the term $\frac{d}{\xi\,\rho}\big[1 + \ln(\frac{\rho\,\xi}{d})\big]$ vanishes, since $\ln\rho$ grows much more slowly than ρ. Hence:

$$\lim_{\rho\to\infty} \frac{U_{\text{net}}(\rho,\xi)}{\rho} = 1,$$

establishing that $U_{\text{net}}(\rho,\xi)$ tends to ρ as their ratio converges to one.

In conclusion, two SPs cannot be considered "of the same type" even if they share identical values for β and ξ, since their relative allocation cost weighting strongly depends on their respective loads. Additionally, our analysis confirms that the model inherently favors SPs with larger loads. This phenomenon results from the exponential saturation function $(1 - e^{-h\,\xi})$, whose argument grows larger with increasing load, diminishing the relative significance of allocation costs.

To illustrate this behavior visually, we present the exponential saturation function chart 4.3:

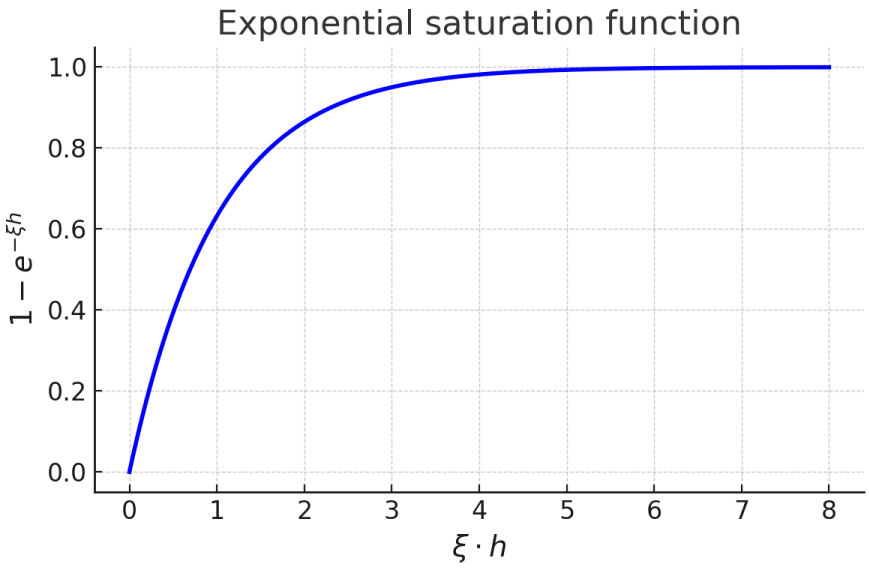


Figure 4.3: Exponential Saturation Function

Players of the Same Type

Two SPs can be considered "of the same type" when the product $\xi \cdot h$ is equal for both, implying that the relative weighting of allocation cost is identical, or in other words, the fraction of the total requests served at the Edge is identical. This condition is equivalent to stating that both players share the same value of the product $\rho \cdot \xi$. Below, we provide a formal demonstration of this equivalence.

From the first-order conditions of the utility function, we have:

$$h_i^* = \frac{1}{\xi_i} \ln\left(\frac{\rho_i \xi_i}{d}\right), \quad i = 1, 2.$$

Multiplying by ξ_i, we obtain:

$$\xi_i \, h_i^* = \ln\left(\frac{\rho_i \xi_i}{d}\right).$$

Imposing the equality condition:

$$\xi_1 \, h_1^* = \xi_2 \, h_2^*,$$

it follows that:

$$\ln\left(\frac{\rho_1 \xi_1}{d}\right) = \ln\left(\frac{\rho_2 \xi_2}{d}\right) \quad \Longrightarrow \quad \rho_1 \xi_1 = \rho_2 \xi_2.$$

Regarding the section of payments and revenues for this scenario, the original article states:

> The contribution of SP^i to the coalitional revenues is defined as $\hat{r}^i = D \cdot \sum_{t=1}^{T} u^i(l_t^i, h^i)$. We denote the grand coalitional revenues as $r^N = \sum_{i \in N} r^i$, where r^i is the result obtained in (12)–(14), i.e. the payoff of each player without considering the component of the payment. The term $\hat{r}^i$ is the amount of revenues produced by SP i, due to the served load during the overall duration of the coinvestment.
>
> Fig. 2 shows that most contribution comes from SP^1, since its load is four times higher than that of SP^2, and the utility of any SP (and thus its contribution to the grand coalitional revenues) is proportional to the served load; indeed, we observe that $\hat{r}^{\mathrm{SP}^1} = \frac{1}{4} \hat{r}^{\mathrm{SP}^2}$. Note that h^{NO} and r^{NO} are not in the figure, as the NO does not use resources, because its load is null; this implies that its utility is null so it does not produce revenues to the grand coalition by serving a load.

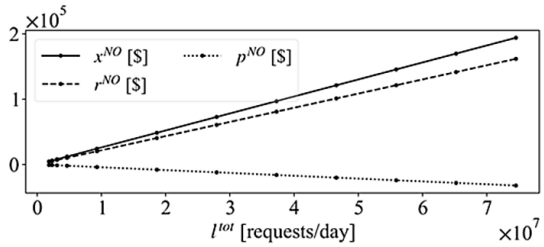
We first address what appears to be a typographical error in the quoted text. Rather than stating $\hat{r}^{\mathrm{SP}^1} = \frac{1}{4} \hat{r}^{\mathrm{SP}^2}$, the authors likely intended to express $\hat{r}^{\mathrm{SP}^1} = 4 \, \hat{r}^{\mathrm{SP}^2}$, given that SP^1's load is four times higher.

However, as we previously demonstrated, the relationship $\hat{r}^{\mathrm{SP}^1} = 4 \, \hat{r}^{\mathrm{SP}^2}$ does not hold unless the players satisfy the condition $\rho_1 \xi_1 = \rho_2 \xi_2$. To meet this condition given that $\rho_1 = 4 \rho_2$, the parameters ξ must be adjusted accordingly, specifically $\xi_1 \, 4 = \xi_2$.

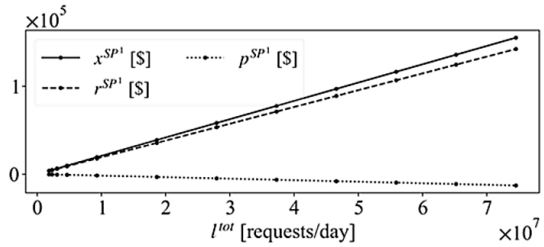
This result aligns with the behavior we previously discussed in Section 3.10.

To complete our analysis of the scenario involving two SPs "of the same type", we reproduce below the charts from the referenced article, provide the authors' original interpretations, and supplement them with our own observations.

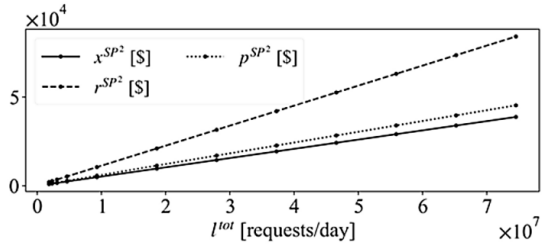
> Fig. 4.4 shows the outcome of the game, i.e. the payment p^i, the revenue r^i and the payoff x^i of each player $i \in \mathcal{N}$ given by the Shapley value. We first observe that the payoff of anyone increases with the total load, which is obvious as the revenues of the grand coalition is the sum of the utilities of each player, which in turn increase with the number of served requests. It is interesting to notice that only SP^2 actually pays to deploy the resources



(a) Payoff, revenues and payment by NO



(b) Payoff, revenues and payment by SP^1



(c) Payoff, revenues and payment by SP^2

Fig. 3: Shapley value: payoffs, revenues and payments.

Figure 4.4: Shapley value: payoffs, revenues and payments

> at the Edge, while the NO and SP^1 have negative payments, so they are not paying, but are being paid. In the case of SP^1, this means that it enjoys an additional gain from the coinvestment in the Edge resources, which sums to the revenue r^{SP^1} directly coming from its customers. The "privilege" of the NO and SP^1 can be explained by the fact that they are the most important for the coalition: NO is the veto player; SP^1 brings to the coalition most of the revenues collected from users (Fig. 2).

As described in Section 3.6.2, the referenced article proposes a framework (Equations 3.9 to3.11) to define payments and revenues. We identified several issues with this framework, as it results in a system of $n+1$ equations and $2n$ variables, which allows an infinite number of possible solutions. In this part of the referenced article, payments seem to have been calculated solely to cover allocation costs, satisfying the condition $\sum_{i \in S} p^i = d \cdot \mathcal{C}$, rather than to achieve fairness.

As previously mentioned in quotation 4.2.1, the NO should not generate any revenues, contrary to what is depicted in the first chart of Fig. 4.4. Additionally, as discussed in Section 3.4, SPs' revenues must be greater than twice their payoffs, and no SP should exhibit negative payments (which implies receiving money); this should only be the case for the NO.

Despite trying different interpretations, we could not reproduce these results. We believe this discrepancy arises from having $n-1$ free parameters in their framework. Therefore, we propose adopting the alternative framework described in Section 3.6.2, as it uniquely defines payments and clearly distinguishes payments intended to cover initial allocation costs from the ones aimed to ensure fairness through the Shapley value.

4.2.2 C. Scenario with 2 SPs of different type

In this scenario, the load relationship remains the same as before, with $l_t^1 = 4\, l_t^2 \quad \forall t$. Also, they maintain the condition $\beta^{\text{tot}} = \beta^{\text{SP1}} + \beta^{\text{SP2}} = 2\,\hat{d}$, where $\hat{d}$ denotes the amortized price. However, they introduce variability by changing the value of β according to the parameter ω:

$$\beta_{\text{SP1}} = (1-\omega)\,\beta_{\text{tot}}, \quad \beta_{\text{SP2}} = \omega\,\beta_{\text{tot}}, \quad \omega \in [0.5,\, 1].$$

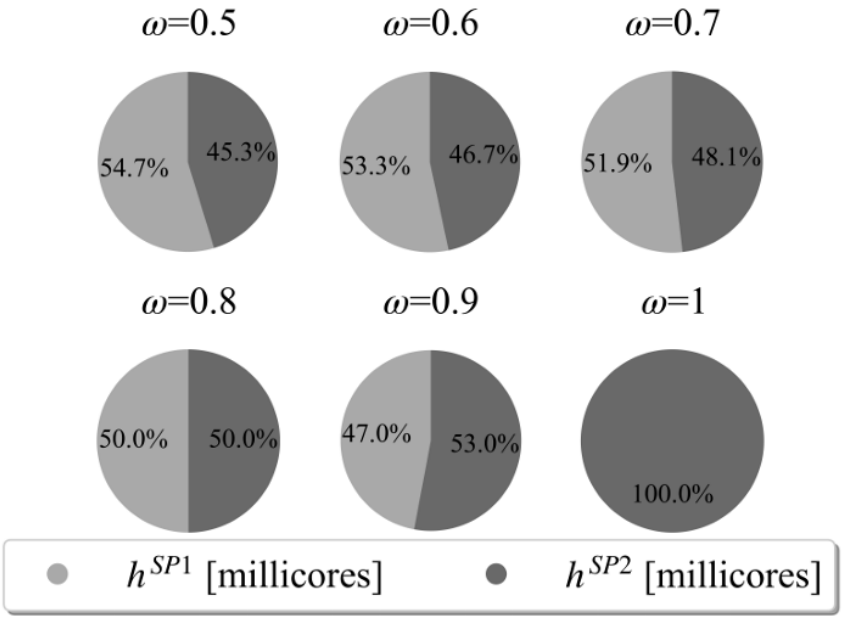
At $\omega = 0.5$, both SPs have the same characteristics as in the previous scenario. As ω increases, SP2 better monetizes its served load.

The authors then vary ω and highlight two main effects. First, Fig. 4.5 shows SP1's CPU allocation decreasing despite having four times the load of SP2, eventually reaching zero at $\omega = 1$, while SP2's allocation correspondingly increases. Second, Fig. 4.6 shows that SP2's Shapley value (the payoff) grows as ω increases, while SP1's payoff diminishes. The NO consistently receives half of the total coalition value.

We are not in disagreement with the results or their interpretation. Nevertheless, we quote below a fragment from the original article's interpretation of this scenario that is worth commenting on:

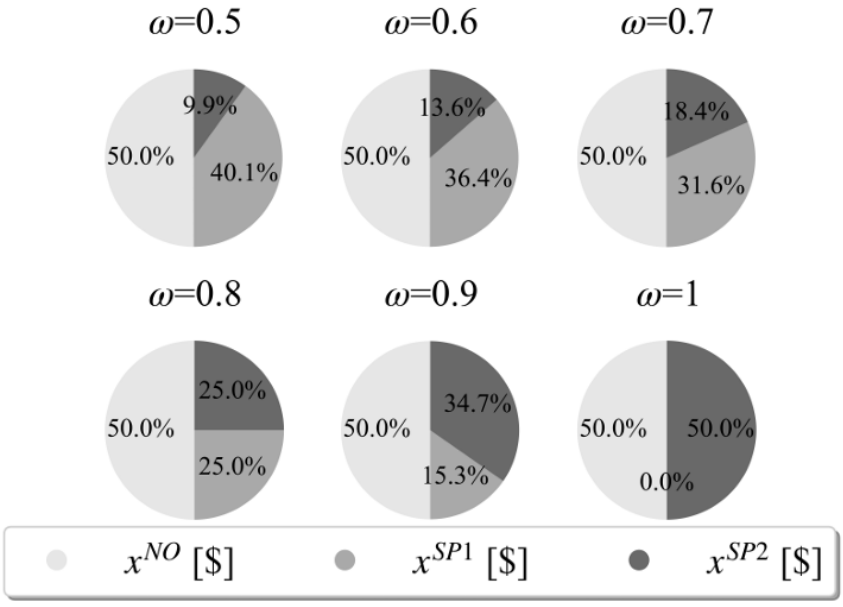
> Figure 4.5 shows how the CPU allocation evolves with ω. Increasing ω shifts capacity away from SP1, until at $\omega = 1$ SP1 receives no CPU and SP2 receives all of it. Although SP1 still attracts four times the load, its lower benefit factor at high ω makes it less advantageous to allocate resources there. This illustrates that edge allocation must consider both load and service time-sensitivity.
>
> Figure 4.6 shows the corresponding Shapley-value payoffs. As ω increases, SP2's payoff slice grows and SP1's shrinks, while the network owner maintains half of the total coalition value.

It is important to clarify that "time-sensitivity" here is captured by the parameter β. Thus, the authors indicate that both the load l and the benefit factor β should be considered when allocating resources



(a) Capacity subdivision among the players

Figure 4.5: Capacity subdivision among the players



(b) Payoff sharing

Figure 4.6: Payoff sharing

at the edge, as both directly influence the produced value of SP. We add here that changes in β and changes in l have exactly the same effect. Therefore, performing a sensitivity analysis by varying β is equivalent to performing the same analysis by varying l. For example, we could take the symmetric case:

$$l_{\text{SP1}} = (1-\omega)\, l_{\text{tot}}, \quad l_{\text{SP2}} = \omega\, l_{\text{tot}}, \quad \omega \in [0.5,\, 1],$$

with the constraint $\beta^1 = 4\,\beta^2$, and we would obtain identical results. This symmetry is the one that allowed us to introduce the substitution $\rho = l\,\beta$.

Furthermore, note that when $\omega = 0.8$, we reach the scenario described in Section 4.2.1, where players satisfy condition $\rho_1\,\xi_1 = \rho_2\,\xi_2$. Since in this case $l_{\text{SP1}} = 4 \cdot l_{\text{SP2}}$ is exactly compensated for by $\beta_{\text{SP1}} \cdot 4 = \beta_{\text{SP2}}$, we can say that at $\omega = 0.8$ players are not just of "the same type" but precisely identical. This explains why they receive exactly the same proportion of both the total allocation and the total coalitional value.

Since we do not know the value of ξ, neither the value of l they used, and we want to reproduce their results, we start by confirming that indeed $\xi_1 = \xi_2$ since at $\omega = 0.8$ we know $\rho_1 = \rho_2$. Then we can fix any number for l_1 and l_2 as long as $l_1 = l_4 \cdot 4$, and any value for β_1 and β_2 as long as: $\beta_1 + \beta_2 = \hat{d} \quad \textit{and} \quad \beta_1 = (1-w) \cdot \beta_1 + \beta_2 \quad \text{and} \quad \beta_2 = w\,\beta_1 + \beta_2$

We take these simple initial values:

- $l_1 = 4.0, l_2 = 1.0$
- $d = 1.0$
- $\beta_1 + \beta_2 = 2.0$

Because the published charts report only relative allocation and payoff shares, we can reverse the solution for the value of ξ that exactly reproduces their curves. In particular, at $\omega = 0.5$ the two SPs receive a similar amount of allocation shares, yet SP^1 contributes nearly four times the utility of SP^2. Such a large disparity implies that the marginal cost term is effectively negligible and therefore ξ must be very large. Carrying out a "reverse engineering" on ξ to reproduce their results, we find that:

$$\xi = 900.$$

Interestingly, the obtained value $\xi = 900$ coincides exactly with the duration of each of the 96 daily time-slots in seconds. This numerical match strongly suggests that ξ was intentionally calibrated using the relationship $\xi = \mu\,\Delta t$, with $\Delta t = 900$ seconds and $\mu = 1$, rather than emerging coincidentally. Note that here μ indicates the amount of requests served at the Edge by one millicore in one second. In other words, the parameter ξ appears designed to scale proportionally with the length of each time-slot. Thus, we interpret the numerical equality $\xi = 900$ for time-slots of 900 seconds as a deliberate modeling decision rather than mere coincidence.

Note that values of ξ of such a magnitude would produce unrealistic low allocations. If, for example, we have 48,530 requests in one time-slot, a number used as average load in the article (A. P. Vela A. Vía & Velasco, 2016) from where the traffic load is modeled, and a β value of 1.5×10^{-6} (both numbers used in some of their simulations), we would get an allocation of ≈ 0.018 millicores, meaning that with one core we could have $\approx$ 55,000 SPs with these characteristics.

In Section 5.2, we revisit the meaning of the parameter ξ, providing a theoretical discussion on its definition and computation. Subsequently, in Section 5.3, we introduce modifications to the utility function to address the unrealistically low allocations that such large values of ξ produce.

Figures 4.7 and 4.8 respectively plot the CPU allocation share and the total coalition payoff share as functions of the parameter ω.

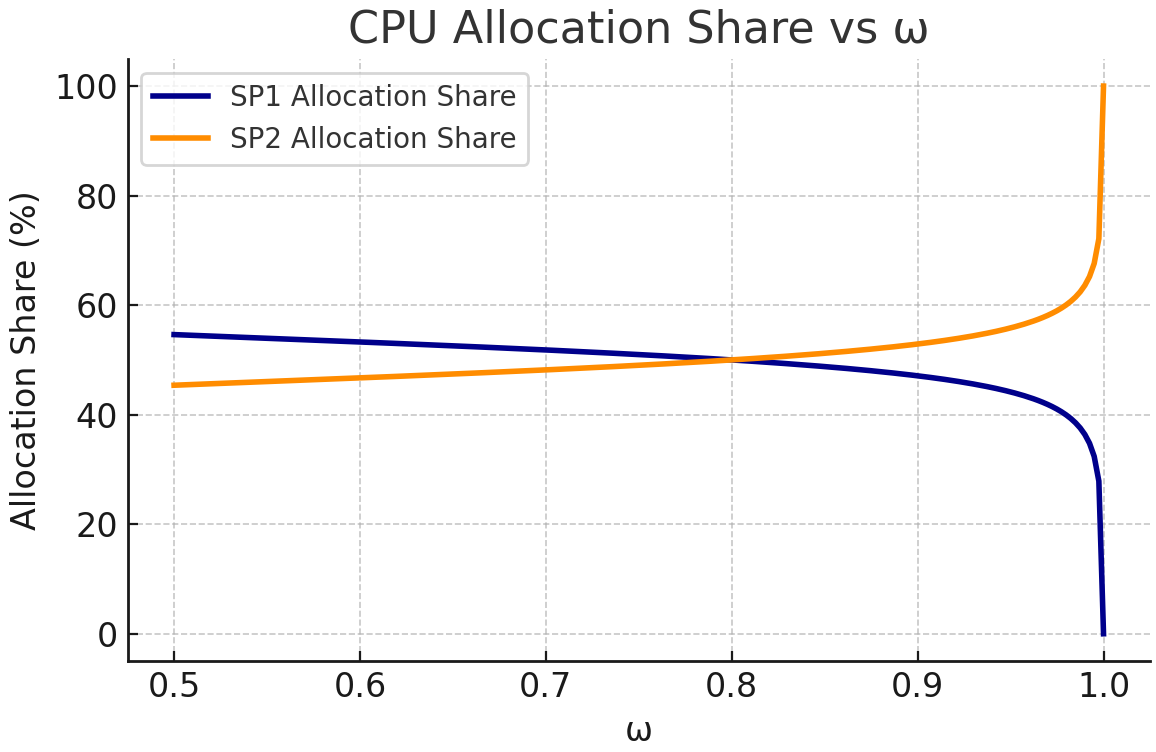


Figure 4.7: CPU Allocation Share vs ω

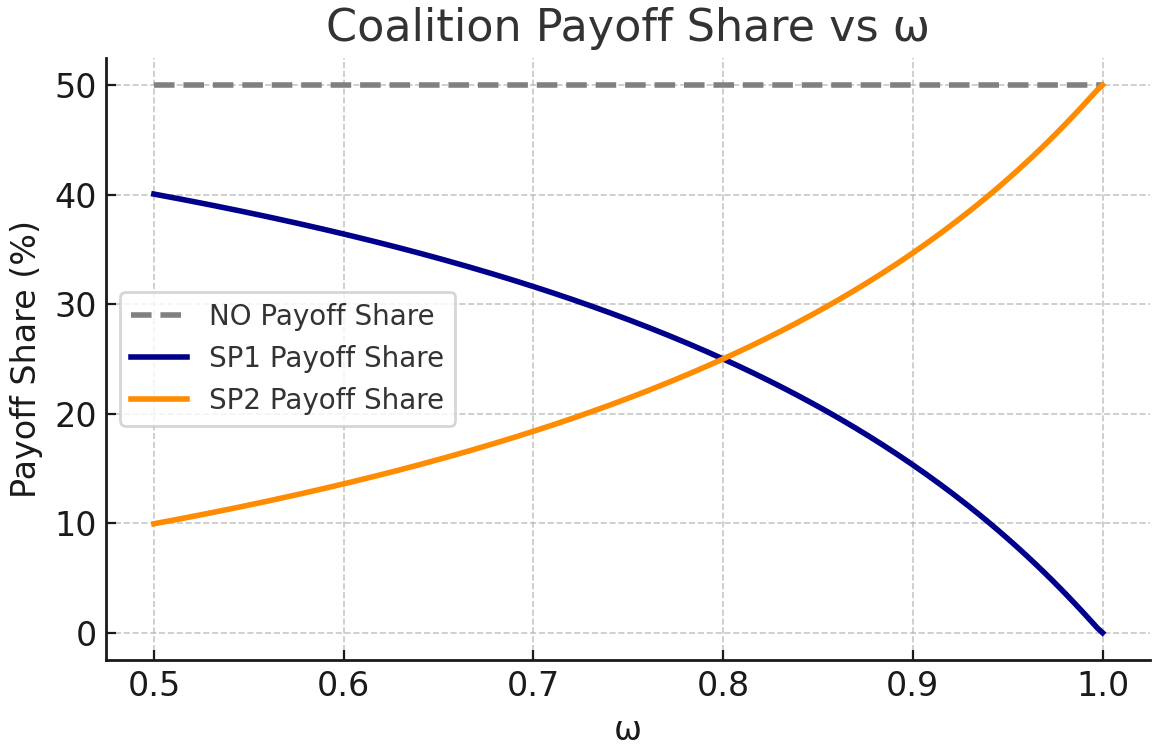


Figure 4.8: Coalitional Payoff Share vs ω

4.2.3 D. Price sensitivity analysis

In this scenario, a sensitivity analysis with respect to price d is conducted, in figure 4.9 we show the corresponding charts from the referenced article and below we quote their interpretation of them.

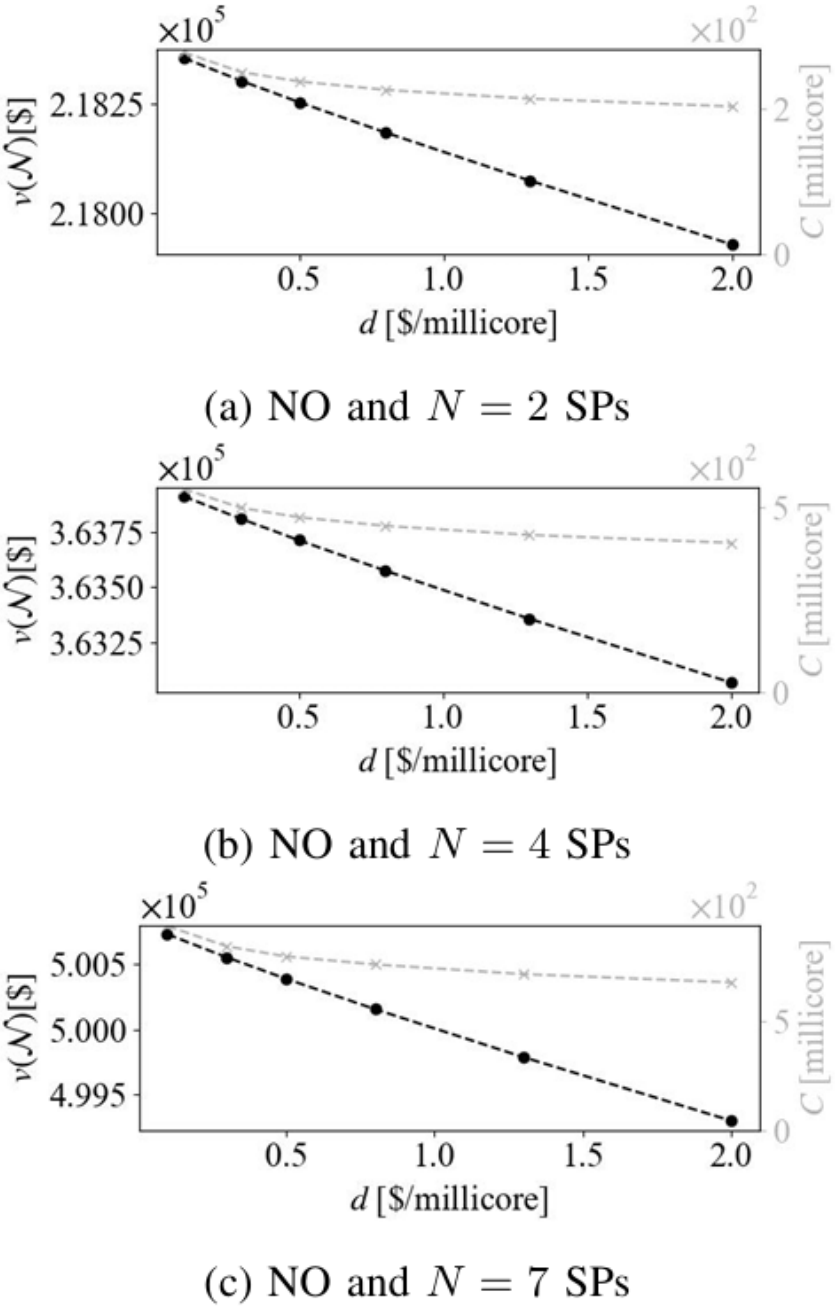


Figure 4.9: Price Sensitivity Analisis

Here, we assess our model to show its behavior varying the number of SPs. The consequence of increasing the price is that (Fig. 5(i)) the purchased capacity is reduced but in a sublinear way, because of (17), and the coalitional value decreases linearly. These trends remain consistent when changing the number N of SPs. Fig. 5 also confirms that adding a player to the game brings a higher benefit in terms of the coalition value v. This is in line with the supermodularity of v (Th. 1) and hence the convexity and stability of the game.

As previously discussed in Scenario B (Section 4.2), we confirm here that the purchased capacity indeed decreases logarithmically. However, contrary to the authors' interpretation, the coalitional value does not decrease linearly. Below we provide a formal demonstration.

Increasing the unit price d' affects the optimal per time-slot net utility as follows:

$$U^*(d') = \rho - \frac{d'}{\xi}\left[1 + \ln\left(\frac{\rho\,\xi}{d'}\right)\right].$$

Taking the first derivative with respect to d':

$$\frac{dU^*}{dd'} = \frac{1}{\xi}\ln\left(\frac{d'}{\rho\,\xi}\right) < 0,$$

and the second derivative is:

$$\frac{d^2U^*}{dd'^2} = \frac{1}{\xi\,d'} > 0.$$

Thus, $U^*(d')$ is strictly convex and decreasing with respect to d'. Consequently, the coalitional value defined as

$$V(d') = \sum_{t=1}^{T} U^*(d')$$

also decreases strictly convexly, flattening as d' increases. Hence, the coalitional value does not decline linearly but exhibits a convex pattern with diminishing marginal losses as the price grows.

4.3 Sensitivity Analysis

In this section, we perform a sensitivity analysis for each variable, presenting the corresponding charts. Our aim is to validate the observations discussed in both the theoretical and previous sections, as well as to highlight additional insights derived from this analysis.

4.3.1 Price Sensitivity Analysis

In figure 4.10 we can observe that as price d' rises, the optimal allocation h^* declines sharply at first and then more gradually, reflecting the inverse logarithmic relationship. Small increases in price when costs are low therefore lead to substantial cuts in allocated resources, whereas once prices are high, further increases have only a modest additional impact. The curve reaches zero at the boundary of the positive allocation condition ($d' = \rho\,\xi$). This behavior captures the trade-off between marginal benefit and marginal cost under diminishing returns. It also mirrors the price-sensitivity effects discussed in Section 4.2.3.

We recall the net utility equation subject to the optimal allocation

$$U(h^*) \;=\; \rho \;-\; \frac{d'}{\xi}\Big[1 + \ln\Big(\tfrac{\rho\,\xi}{d'}\Big)\Big].$$

In figure 4.11 we can observe that as d' rises, net utility $U(h^*)$ falls more sharply at first and then more gradually, reflecting its strictly convex dependence on price. Small increases in d' when costs are low produce large utility losses, whereas at higher d' further price hikes have diminishing impact. Finally, $U(h^*)$ reaches zero at the threshold $d' = \rho\,\xi$, where marginal cost equals marginal benefit. This curve therefore captures the balance between revenue generation and allocation cost under diminishing returns. It also mirrors the price-sensitivity effects discussed in Section 4.2.3.

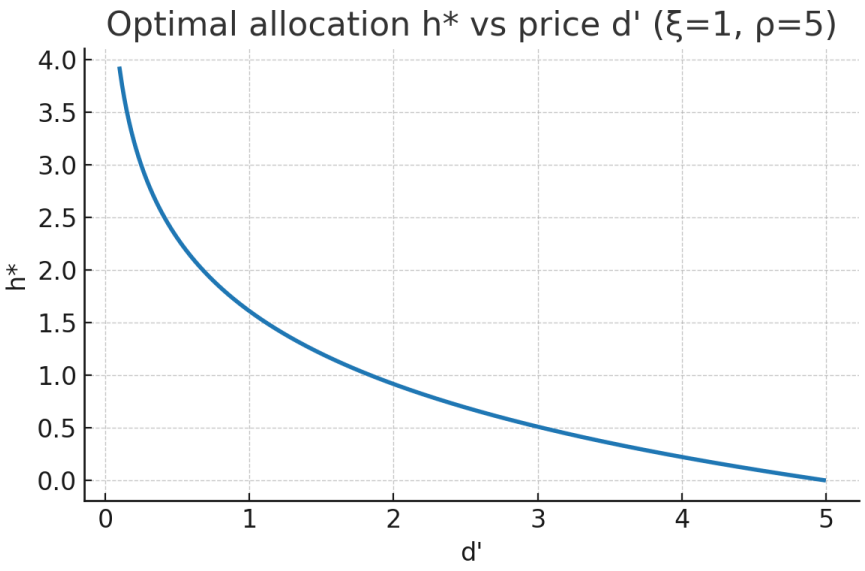


Figure 4.10: Optimal Allocation vs Price

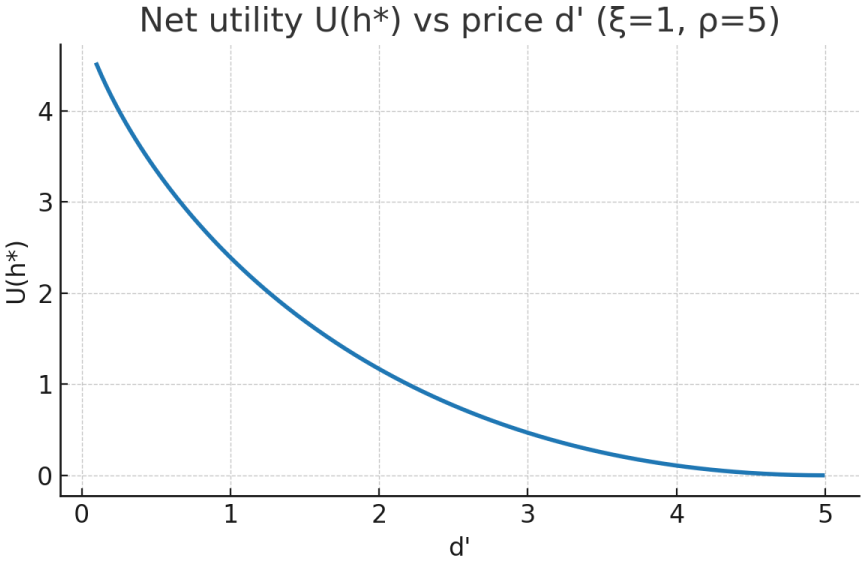


Figure 4.11: Net Utility at h U(h) vs Price

4.3.2 Potential Monetization Sensitivity Analysis

The figure 4.12 shows that the curve of h^* versus ρ is logarithmic: it rises very steeply when ρ is small and then gradually flattens as ρ grows. Economically, this means that increases in the SP's combined parameter ρ (whether from higher average load or greater benefit per request) prompt significant additional resource allocation at low levels but yield ever-smaller marginal increases once ρ is already large. This concave shape embodies the principle of diminishing returns: each additional unit of ρ produces a smaller incremental allocation of computing resources.

This diminishing returns in allocation may not accurately reflect how computing resources should escalate with respect to ρ, this is something we should address when proposing changes to the net utility function.

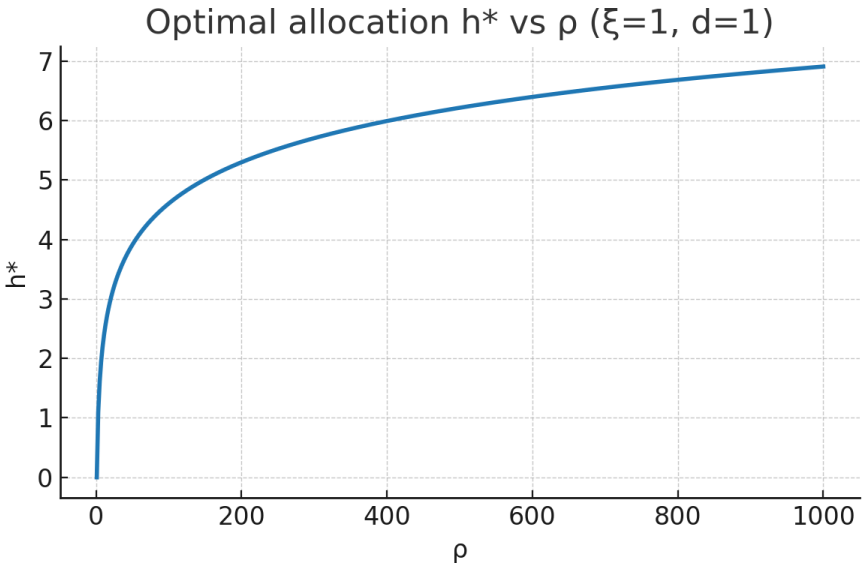


Figure 4.12: Optimal Allocation (h*) vs ρ

The figure 4.13 shows that the net utility under optimal allocation grows sublinearly with the combined parameter ρ, reflecting the trade-off between revenue and allocation cost. For $\xi = 1$ and $d' = 1$, we have

$$h^* = \ln(\rho), \qquad U(h^*) = \rho - \left[1 + \ln(\rho)\right]$$

At low ρ, the term $1 + \ln(\rho)$ nearly cancels ρ, so $U(h^*)$ remains close to zero until ρ exceeds about e. As ρ increases, $U(h^*)$ rises steadily but always stays below the line $U(h^*) = \rho$, since each additional unit of ρ must cover both the direct revenue and the logarithmically growing cost of extra resources. In the limit $\rho \to \infty$, $\ln(\rho)/\rho \to 0$, so the gap between $U(h^*)$ and ρ vanishes and $U(h^*) \sim \rho$, illustrating that costs become negligible at very large scales.

This confirms what we established in Section 4.2.2 indicating that for a fixed value of ξ the fraction of requests served at the Edge changes when the value of ρ changes.

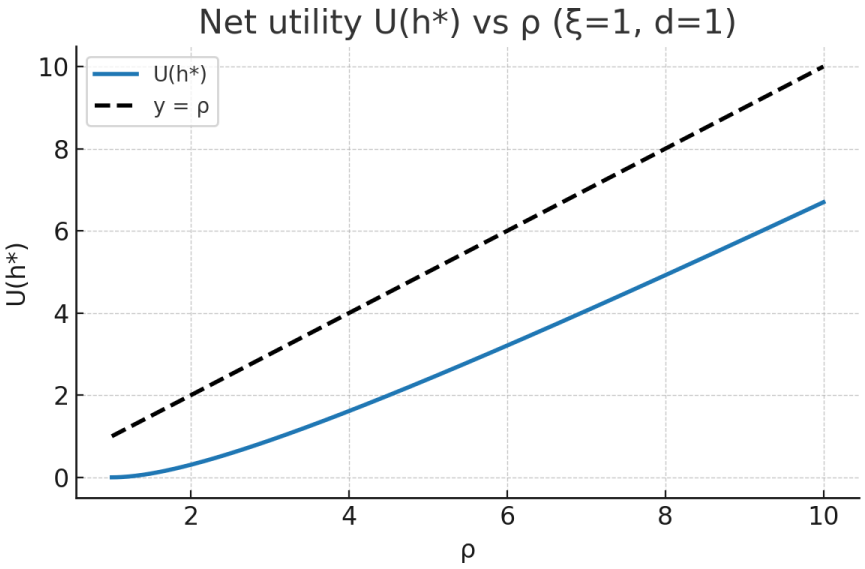


Figure 4.13: Net Utility at h U(h*) vs ρ

4.3.3 diminishing returnEffect Sensitivity Analysis

To complete the sensitivity analysis, we proceed with the diminishing returns parameter ξ. Given the more complex impact of this parameter on the h^* and $U(h^*)$ charts, we include heatmap visualizations for these functions to clearly illustrate the interaction between the parameters ρ and ξ.

The figure 4.14 visualizes the optimal allocation as a function of the sensitivity parameter ξ for various benefit-to-price ratios ρ/d. No resources are allocated until the positive allocation condition $\rho\,\xi > d$ is satisfied. As ξ increases slightly above d/ρ, the marginal benefit term $\ln(\rho\,\xi/d)$ grows faster than the cost of increasing ξ, causing h^* to rise. The allocation attains its maximum when

$$\frac{d}{d\xi}\big[\ln(\rho\,\xi/d) - \ln\xi\big] = 0 \quad \Longrightarrow \quad \xi_{\text{peak}} = \frac{e\,d}{\rho}, \quad h^*_{\max} = \frac{\rho}{e\,d}.$$

Beyond ξ_{peak}, the $1/\xi$ factor dominates and h^* decays toward zero, reflecting the stronger effect of diminishing returns. Higher values of ρ/d not only raise the peak allocation but also shift ξ_{peak} to lower values, meaning that more profitable configurations can sustain larger allocations even under greater sensitivity to resource increases.

The figure 4.15 shows a heatmap for the optimal allocation equation:

$$h^*(\xi,\rho) = \frac{1}{\xi}\ln\!\Big(\frac{\rho\,\xi}{d}\Big)$$

across the (ξ,ρ) plane. The white dashed line marks the locus of points satisfying $h^* = 1$. Below and to the left of this contour, the allocation would be less than one unit (and effectively zero below the feasibility boundary $\xi = d/\rho$). Above and to the right, the allocation exceeds one. The shape of this contour illustrates that maintaining $h^* = 1$ at higher values of ρ requires the sensitivity parameter ξ to increase sublinearly, reflecting the logarithmic relationship between h^* and ρ. The pronounced curvature observed near low values of ξ and ρ emphasizes the model's heightened sensitivity when operating near the viability threshold. This confirms our earlier observation that, in order to maintain a constant fraction of the load served at the Edge, it is necessary to carefully adjust ξ whenever ρ varies, typically as a consequence of changes in load.

In the figure corresponding to the optimal net utility 4.16, the curves remain zero until the feasibility condition $\rho\,\xi > d$ is met, since no allocation can occur below that threshold. As ξ increases beyond this point, $U(h^*)$ initially rises with a rapid growth of h^*, then its speed diminishes once h^* peaks.

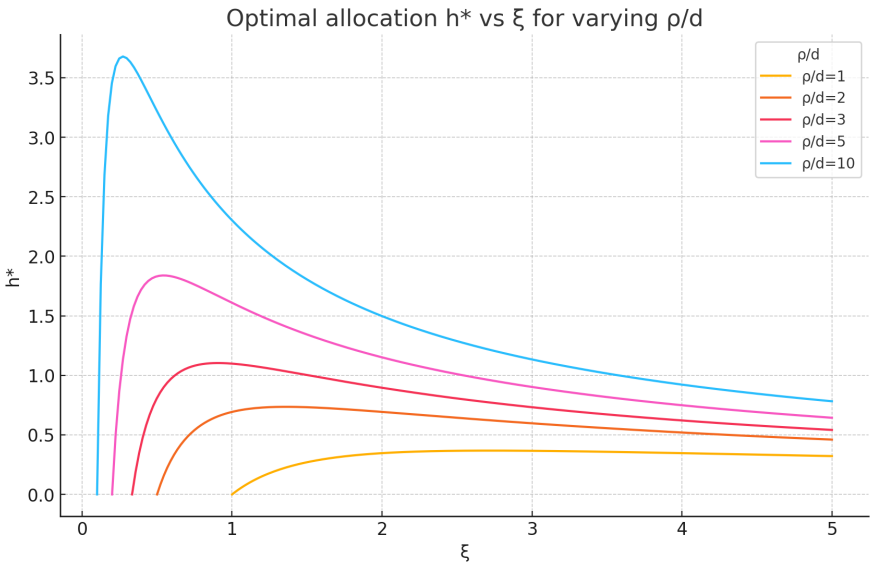


Figure 4.14: Optimal allocation $h^*(\xi)$ versus ξ for varying ρ/d.

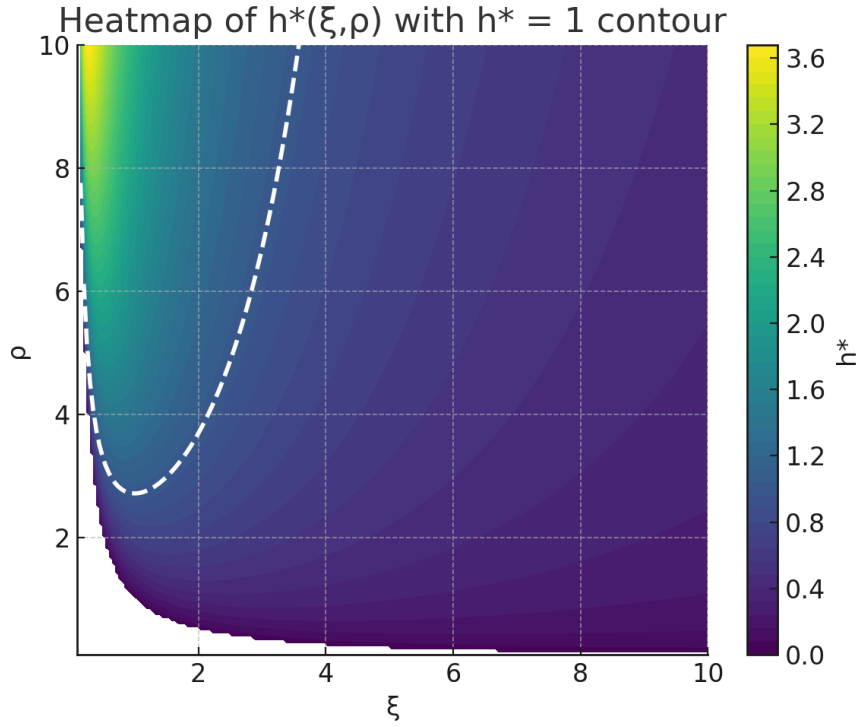


Figure 4.15: Heatmap of $h^*(\xi, \rho)$ with the contour $h^* = 1$.

Recalling the equation:

$$U\big(h^*(\xi)\big) = \rho - \frac{d'}{\xi}\Big[1 + \ln\Big(\frac{\rho\,\xi}{d'}\Big)\Big],$$

so as $\xi \to \infty$ the logarithmic term cancels out with the linear one and $U(h^*) \to \rho$. Higher ratios of $\frac{\rho}{d}$ shift all curves upward and move their turnover points to smaller ξ, illustrating that more lucrative settings sustain larger payoffs even under strong diminishing returnsensitivity.

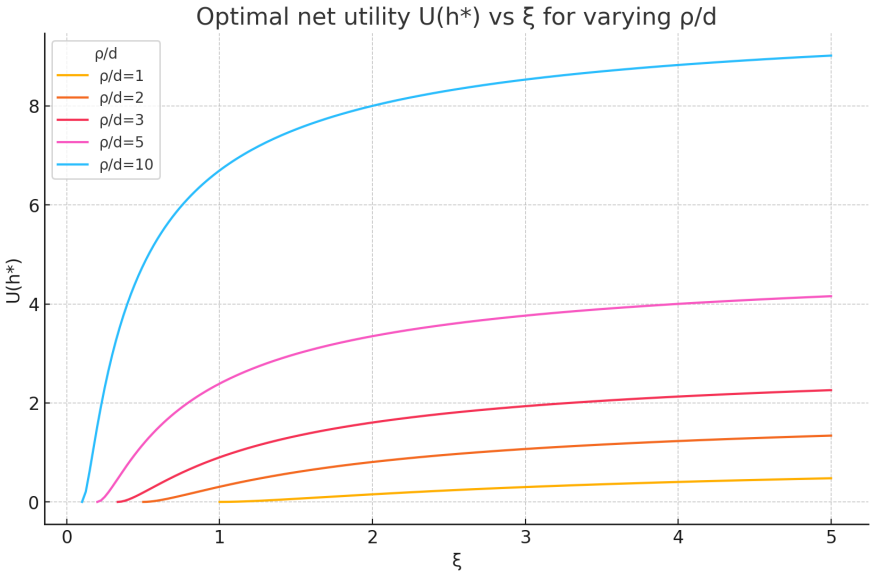


Figure 4.16: Optimal net utility U(h*) vs ξ for varying $\frac{\rho}{d}$

The figure 4.17 shows the heatmap of $U(h^*)(\xi, \rho)$ with the contour $U = \rho/2$. This heatmap displays the net utility over the same parameter plane. The white dashed curve is the points where $U(h^*) = \frac{1}{2}\,\rho$, marking half-saturation of the benefit factor. Points above this contour achieve more than half of the maximum possible utility, while points below achieve less. The shape of the contour shows that larger ρ or moderate ξ are required to reach 50% of ρ, while very high sensitivity actually suppresses utility. This boundary identifies the region of parameter space in which the system captures a majority of its potential payoff.

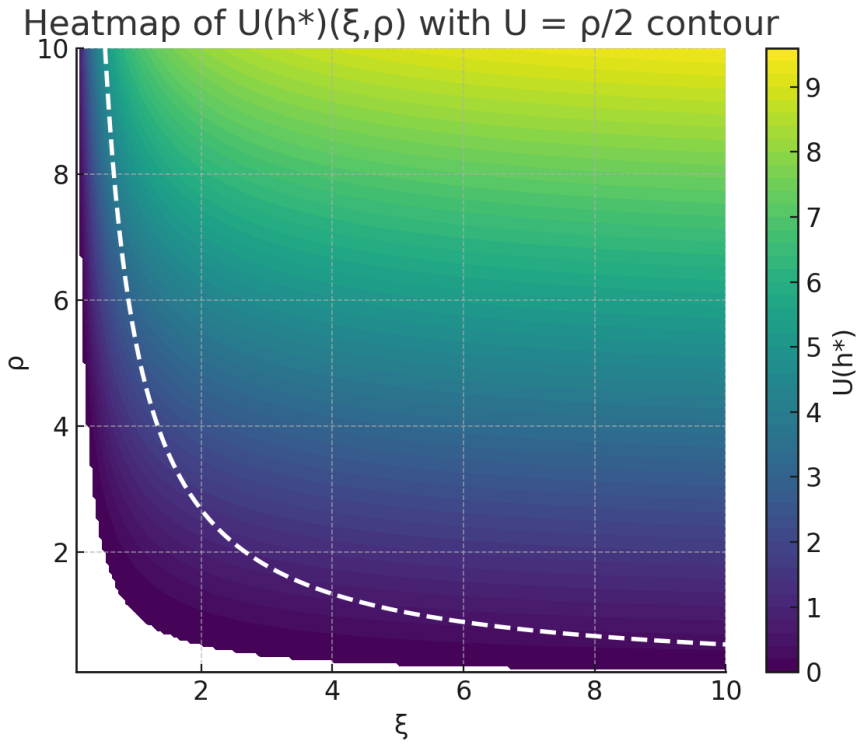


Figure 4.17: Heatmap of $U(h^*)$

4.4 Practical Illustration of the Strategic Implications

To conclude this practical chapter, we numerically analyze the strategic implications of economic risks associated with misestimating the parameter ρ.

Then we present three numerical examples illustrating how a Service Provider might strategically misrepresent its net utility. In the first case, misestimation occurs in both parameters (ξ and ρ), where the optimal allocation can still be maintained. In the second case, misestimation occurs only in parameter ξ, and in the third case, the misestimation occurs only in the parameter ρ.

4.4.1 Practical Illustration of Parameter Misestimation

In this practical section, we numerically illustrate the theoretical insights developed in Section 3.12. Through specific numerical examples and visualizations, we show the effects of parameter misestimation.

Recall that ρ_{exp} represents the expected value of ρ, determining an optimal allocation h_{exp} and yielding an expected net utility U_{exp}, while ρ_{real} refers to the actual realized value of ρ, producing the real net utility U_{real}.

Consider two different scenarios characterized by different real parameter values of ξ:

- Case A: $\xi = 0.02718$, $\rho = 1$, $d' = 0.01$ (Note that this is ξ_{peak})
- Case B: $\xi = 0.2$, $\rho = 1$, $d' = 0.01$

Optimal Allocation and Expected Utility

The optimal allocation h_{exp} and expected net utility U_{exp} are

$$h_{\text{exp},A} = \frac{1}{0.02718}\ln\!\left(\tfrac{1\cdot 0.02718}{0.01}\right) \approx 36.79, \quad U_{\text{exp},A} = 1\left(1-e^{-1}\right) - 0.01\times 36.79 \approx 0.2642,$$

$$h_{\text{exp},B} = \frac{1}{0.2}\ln\!\left(\tfrac{1\cdot 0.2}{0.01}\right) \approx 14.98, \quad U_{\text{exp},B} = 1\left(1-e^{-2.996}\right) - 0.01\times 14.98 \approx 0.8000.$$

Effects of Misestimation After Initial Investment

Recall that when h is fixed, the opportunity cost here is not applicable.

Overestimation ($\rho_{\text{real}} = 0.5$):

$$\text{Loss}_A = 0.5\left(1-e^{-1}\right) \approx 0.3160, \quad \text{Loss}_B = 0.5\left(1-e^{-2.996}\right) \approx 0.4750.$$

Case A becomes unprofitable under a 50% drop, while Case B remains profitable.

$$U_{\text{exp},A} - U_{\text{exp},A} = -0.0518$$
$$U_{\text{exp},B} - U_{\text{exp},B} = 0.325$$

Break-Even Point

Solve $U = 0$ for ρ:

$$\rho_{100,A} = \frac{0.01\times 36.79}{1-e^{-1}} \approx 0.5819 \quad\Longrightarrow\quad p_{100,A} = 1 - 0.5819 \approx 0.4181\ (41.81\%),$$

$$\rho_{100,B} = \frac{0.01\times 14.98}{1-e^{-2.996}} \approx 0.1577 \quad\Longrightarrow\quad p_{100,B} = 1 - 0.1577 \approx 0.8423\ (84.23\%).$$

This is consistent with the previous results since dropping ρ by a 50% becomes case A unprofitable, while case B should drop more than ≈ 84.23 to become unprofitable.

Effects of Misestimation Before Initial Investment

Recalculate h and U for actual ρ:

Underestimation ($\rho_{\text{real}} = 1.5$):

$$h_{\text{real},A} = \frac{1}{0.02718}\ln\!\left(\tfrac{1.5\cdot 0.02718}{0.01}\right) \approx 51.71, \quad U_{\text{real},A} = 1.5\left(1-e^{-1.406}\right) - 0.01\times 51.71 \approx 0.6150,$$

$$h_{\text{real},B} = \frac{1}{0.2}\ln\!\left(\tfrac{1.5\cdot 0.2}{0.01}\right) \approx 17.01, \quad U_{\text{real},B} = 1.5\left(1-e^{-3.402}\right) - 0.01\times 17.01 \approx 1.2799.$$

Overestimation ($\rho_{\text{real}} = 0.5$):

$$h_{\text{real},A} = \frac{1}{0.02718}\ln\!\left(\tfrac{0.5\cdot 0.02718}{0.01}\right) \approx 11.29, \quad U_{\text{real},A} = 0.5\left(1-e^{-0.308}\right) - 0.01\times 11.29 \approx 0.0192,$$

$$h_{\text{real},B} = \frac{1}{0.2}\ln\!\left(\tfrac{0.5\cdot 0.2}{0.01}\right) \approx 11.51, \quad U_{\text{real},B} = 0.5\left(1-e^{-2.3026}\right) - 0.01\times 11.51 \approx 0.3349.$$

Generalizing Misestimations

We begin with the equilibrium formula 3.37 valid for any misestimation fraction ϵ:

$$\rho_{eq} = \frac{d'\, h\, \epsilon}{\sinh\big(\xi\, h\, \epsilon\big)\, e^{-\xi\, h}}$$

Using this expression, we compute the thresholds for three values of ϵ:

Small misestimation ($\epsilon = 0.2$)

$$\rho_{eq,A} = \frac{0.01 \times 36.79 \times 0.2}{\sinh\big(0.02718 \times 36.79 \times 0.2\big)\, e^{-1}} \approx 0.9934, \quad \rho_{eq,B} = \frac{0.01 \times 14.98 \times 0.2}{\sinh\big(0.2 \times 14.98 \times 0.2\big)\, e^{-2.996}} \approx 0.9400.$$

Moderate misestimation ($\epsilon = 0.5$)

$$\rho_{eq,A} = \frac{0.01 \times 36.79 \times 0.5}{\sinh\big(0.02718 \times 36.79 \times 0.5\big)\, e^{-1}} \approx 0.9596, \quad \rho_{eq,B} = \frac{0.01 \times 14.98 \times 0.5}{\sinh\big(0.2 \times 14.98 \times 0.5\big)\, e^{-2.996}} \approx 0.7053.$$

Extreme misestimation ($\epsilon = 1$)

$$\rho_{eq,A} = \frac{0.01 \times 36.79 \times 1}{\sinh\big(0.02718 \times 36.79 \times 1\big)\, e^{-1}} \approx 0.8510, \quad \rho_{eq,B} = \frac{0.01 \times 14.98 \times 1}{\sinh\big(0.2 \times 14.98 \times 1\big)\, e^{-2.996}} \approx 0.3020.$$

SPs compare their ρ_{exp} against these thresholds:

- If $\rho_{\text{exp}} > \rho_{eq}$: overestimating allocation is riskier.
- If $\rho_{\text{exp}} < \rho_{eq}$: underestimating allocation is riskier.

Interpretation of Equilibrium Thresholds

Why is $\rho_{eq,A} > \rho_{eq,B}$ for the same values of ϵ.

- Case A has a much smaller ξ than Case B, so its optimal allocation h_A is larger.
- The numerator $d'\, h\, \epsilon$ therefore grows more in A than in B.
- Meanwhile the denominator $\sinh(\xi\, h\, \epsilon)\, e^{-\xi\, h}$ grows more rapidly in B (since $\xi_B\, h_B$ is larger), reducing its ratio.
- Consequently the threshold ρ_{eq} remains higher in Case A than in Case B for any fixed ϵ.

Why ρ_{eq} decreases as ϵ increases.

- The numerator grows linearly in ϵ.
- The denominator contains $\sinh(\xi\, h\, \epsilon)$, which increases super-linearly (exponentially) in ϵ.
- As ϵ grows, the denominator outpaces the numerator, driving ρ_{eq} down.

Why ρ_{eq} never exceeds 1 (the original ρ_{exp}).

- In the limit $\epsilon \to 0$, $\sinh(\xi\, h\, \epsilon) \approx \xi\, h\, \epsilon$, so

$$\rho_{eq} \longrightarrow \frac{d'\, h}{\xi\, h\, e^{-\xi\, h}} = \rho_{\text{exp}} = 1.$$

- For any $\epsilon > 0$, the super-linear growth of sinh makes the denominator strictly larger than its $\epsilon \to 0$ approximation, so $\rho_{eq} < \rho_{\text{exp}} = 1$.
- Hence the equilibrium threshold always lies below the original expected value.

In conclusion, and from a more intuitive point of view, we can assess that overestimation always poses a greater risk than underestimation. These results are consistent with the theoretical section 3.12.4 where we exposed the reasons behind this asymmetry.

4.4.2 Numerical Illustration of Optimal Allocation with the Two Misrepresented Parameters

We demonstrate practically how an SP can achieve the optimal allocation while declaring a lower net utility using a detailed numerical scenario. Consider the following real parameters:

$$\xi = 0.02, \quad \rho = 1, \quad d' = 0.01$$

Step 1: Calculate the optimal allocation h^*:

$$h^* = -\frac{1}{\xi} \ln\left(\frac{d'}{\rho\, \xi}\right) = -\frac{1}{0.02} \ln\left(\frac{0.01}{1 \times 0.02}\right) \approx 34.66$$

Step 2: Compute the corresponding time-slot real net utility U_{real}:

$$U_{\text{real}} = \rho \left(1 - e^{-\xi h}\right) - d'h = 1 \times (1 - e^{-0.02 \times 34.66}) - 0.01 \times 34.66 \approx 0.1534$$

Suppose the SP wants to declare exactly half of this real net utility:

$$U_{\text{declared}} = \frac{U_{\text{real}}}{2} \approx 0.0767$$

Step 3: Using the constructive method (see Section 3.13), the declared parameters ξ_{dec} and ρ_{dec} can be found.

First, express the declared time-slot net utility solely in terms of the SP's declared parameter ξ_{dec} and the fixed parameters d' and h^* :

$$U_{\text{net}}^{\text{declared}}(\xi_{dec}) = \frac{d' e^{\xi_{dec} h^*}}{\xi_{dec}} \left(1 - e^{-\xi_{dec} h^*}\right) - d'h^*$$

Step 4: Numerically solve for ξ_{dec} to achieve the declared utility of 0.0767:

$$0.0767 = \frac{0.01\, e^{\xi_{dec} \times 34.66}}{\xi_{dec}} \left(1 - e^{-\xi_{dec} \times 34.66}\right) - 0.01 \times 34.66$$

Solving numerically yields:

$$\xi_{dec} \approx 0.01118$$

Step 5: Using the obtained ξ_{dec}, calculate the corresponding ρ_{dec}:

$$\rho_{dec} = \frac{d' e^{\xi_{dec} h}}{\xi_{dec}} = \frac{0.01\, e^{0.01118 \times 34.66}}{0.01118} \approx 1.3177$$

Thus, the SP maintains the optimal allocation $h = 34.66$, declaring parameters:

$$\xi_{dec} \approx 0.01118, \quad \rho_{dec} \approx 1.3177,$$

achieving to declare exactly half the real net utility. This numerical illustration clearly shows the vulnerability of the model to strategic misrepresentation by SPs. Note that this percentage

4.5 Numerical Illustration of Misreporting ρ

Take

$$\rho_{\text{real}} = 10, \quad \xi = 1, \quad d' = 1.$$

Then $h_{\text{dec}} = \ln \rho_{\text{dec}}$, and we evaluate at:

ρ_{dec}	h_{dec}	U_{dec}	U_{real}	U_{SP}	Benefit
10	$\ln 10 \approx 2.30$	6.70	6.70	3.35	0%
8	$\ln 8 \approx 2.08$	4.92	6.67	4.21	25.7%
5	$\ln 5 \approx 1.61$	2.39	6.39	5.20	55.1%
2	$\ln 2 \approx 0.69$	0.31	4.31	4.15	24.0%

Table 4.1: Comparison of declared vs. real utilities and retained SP utility

As ρ_{dec} decreases, U_{SP} peaks at $\rho_{\text{dec}} = 5$ with a 55.1% gain over truthful reporting. This is the maximum benefit attainable by misreporting ρ in this scenario. Note that this percentage may vary depending on the real value of ρ.

4.6 Numerical Illustration of Misreporting ξ

As previously established, the value of ξ can be measured directly, a fact we formalize in the next chapter. To illustrate how the declaration of this parameter influences the model's strategy-proofness, we present an example demonstrating how an SP can benefit from misreporting its value. As shown earlier in the theoretical section, if the actual value satisfies $\xi_{\text{real}} > \xi_{\text{peak}}$, an SP can choose a declared value $\xi_{\text{dec}} < \xi_{\text{peak}}$ that yields the same allocation. Moreover, declaring an even lower value, while resulting in a smaller allocation, can further increase the SP's payoff from misreporting. Below, we consider an illustrative example where $\xi_{\text{real}} = \xi_{\text{peak}}$, a particularly insightful scenario.

Take

$$\rho_{\text{real}} = 1, \quad \xi_{\text{real}} = \xi_{\text{peak}} = 0.1\, e \approx 0.2718, \quad d' = 0.1.$$

Then

$$h_{\text{dec}} \;=\; \frac{1}{\xi_{\text{dec}}} \ln\!\Big(\frac{\xi_{\text{dec}}}{0.1}\Big),$$

and we evaluate at:

ξ_{dec}	h_{dec}	U_{dec}	U_{real}	U_{SP}	Benefit
0.2718	≈ 3.6788	0.2642	0.2642	0.1321	0%
0.4000	≈ 3.4657	0.4034	0.2636	0.0619	-53.1%
0.3000	≈ 3.6620	0.3005	0.2642	0.1140	-13.7%
0.2000	≈ 3.4657	0.1534	0.2636	0.1869	41.5%
0.1500	≈ 2.7031	0.0630	0.2501	0.2186	65.4%
0.1250	≈ 1.7851	0.0215	0.2059	0.1952	47.7%
0.1000	0.0000	0.0000	0.0000	0.0000	-100.0%

Table 4.2: Comparison of declared vs. real utilities and retained SP utility for various ξ_{dec}.

Chapter 5

Analytical Investigation of the Proposed Modifications to the Edge-Computing Model

5.1 Introduction

In this chapter, we present changes to the baseline model, addressing its primary limitations and enabling it to more accurately represent real-world co-investment dynamics. Our main goals for this chapter are:

- Reinterpretation and redefining the meaning of ξ: In the referenced article, the varible ξ is described as "a parameter that governs the diminishing returns", without much clarification beyond its mathematical meaning. In the practical Section 4.2.1, we grasped some insights of its real word meaning that we intend to formalize.

- Linearization of the allocation/utility scaling: In the original model, optimal allocation grows logarithmically with ρ, while net utility approaches linearity as $\rho \to \infty$. This behavior does not realistically reflect proportional scaling.

- Differentiating the load demand l and the benefit factor β: In the original model, l and β appear multiplicatively as $\beta\, l$, which justifies replacing them with ρ, although they represent distinct concepts from the real world. To better capture their separate influences, we extend the model to allow these variables to play different roles. Specifically, we want to be able to differenciate them in the optimal allocation equation, where the load demand should induce a more linear-like scaling behavior, while the benefit factor may retain its current logarithmic relationship.

- Improving Strategy-Proofness: We propose modifications to the utility function aimed at limiting the lack of strategy-proofness, or at least preventing SPs from arbitrarily declaring their utility while still receiving the optimal allocation.

- Enabling interdependent contributions: By incorporating positive externalities or synergies among SP, the independent contribution assumption no longer holds. This allows us to capture richer cooperative behaviors, where SPs jointly amplify each other's utility, while the baseline serves as a "worst-case" scenario. One of these synergies is the ime-slot-specific allocations. Instead of allocating resources based solely on the average load, each player now receives potentially different allocations per time-slot. This approach captures the heterogeneity in daily demand patterns that was previously overlooked and reveals additional synergies among SPs with complementary load curves.

5.2 Reinterpreting and Redefining the Meaning of the Diminishing Returns Parameter

In the original article, the parameter ξ is introduced as an exogenous shape parameter that controls the rate of diminishing returns. Specifically, it determines how rapidly the exponential saturation function

$$1 - e^{-\xi\, h}$$

approaches its upper bound of 1 as the allocated resource h increases. However, the original article does not provide a real-world interpretation, practical measurement methods, or guidelines on empirically estimating ξ. Additionally, the original formulation implicitly treats ξ as uniform across Service Providers (SPs). In practice, different SPs typically require distinct amounts of resources to handle the same number of requests.

In Section 4.2.2 of the empirical chapter, we identified a practical interpretation for the parameter ξ. Specifically, the parameter ξ^i represents the number of requests that SP^i can serve with one millicore during a single time-slot. Formally, if SP^i can serve on average μ^i requests per second per millicore, and the day is divided into T slots, each of length Δt seconds (e.g., $T = 96$, $\Delta t = 900$), then the corresponding parameter for each time-slot is defined as:

$$\xi^i_{\Delta t} \;=\; \underbrace{\mu^i}_{\text{req/sec/millicore}} \;\times\; \underbrace{\Delta t}_{\text{s per time-slot}} \;.$$

Although this interpretation clarifies the meaning of ξ and makes it measurable in practical terms, it introduces an important model limitation: the resulting optimal allocation explicitly depends on the chosen time-slot length. To illustrate this, consider the optimal allocation expression:

$$h^* = \frac{1}{\xi_{\Delta t}} \ln\!\Big(\frac{l_{\Delta t}\,\beta\,\xi_{\Delta t}}{d'_{\Delta t}}\Big).$$

Each parameter in this equation scales linearly with the duration of the time-slot Δt, causing the resulting allocation h^* to vary with the time-slot length. A more detailed demonstration of this dependency is provided in Section B.8.5 of the demonstrations chapter.

To resolve this limitation and maintain consistency with the original model, we redefine the exponential saturation function using the per-second parameter μ^i, thus eliminating explicit dependency on Δt:

$$1 - e^{-\mu^i\, h}.$$

Under this redefinition, the saturation function remains consistent regardless of the chosen time-slot length. Therefore, we conclude that defining $\xi^i = \mu^i$ best aligns with our original model's goals, ensuring clarity, empirical relevance, and time-slot length independence for the optimal allocation.

Typical practical values for the parameter μ (requests per second per millicore) across common edge service profiles are listed below:

- Static content server: $\mu \approx 5$ to 15
- Dynamic web application: $\mu \approx 1$ to 5
- Complex database queries: $\mu \approx 0.2$ to 1
- Real-time and IoT services (audio, video, gaming, AR): $\mu \approx 0.1$
- CPU-intensive tasks (scientific computing, ML inference): $\mu \approx 0.01$ to 0.1

This refined interpretation of the parameter ξ provides enhanced model clarity and empirical grounding. Furthermore, using $\xi = \mu$ gives a more realistic range of allocation values in our original model.

5.3 Motivating Changes in the Net Utility Function

Given our redefinition of $\xi^i_{\Delta t}$ as the number of requests that SP^i can serve at the edge with one millicore within a time-slot of duration Δt, under the assumption of a perfectly uniform request distribution, the quantity

$$\frac{l_{i,\Delta t}}{\xi_{i,\Delta t}}$$

represents the minimum number of millicores necessary for SP^i to serve all $l_{i,ts}$ requests at the edge within that time-slot.

In real-world scenarios, requests are not perfectly evenly distributed, and even under ideal conditions, resource utilization at or near this theoretical limit would degrade service quality and latency. Therefore, it is economically reasonable for SPs to allocate resources above this minimum threshold whenever the marginal benefit justifies the associated marginal cost.

By redefining ξ in measurable terms, we provide the NO, which controls the infrastructure, with the practical capability to measure both request volume and resource usage. Although specific virtualization implementations fall outside the scope of this work, Section C.3 in the Appendix describes possible approaches for effective measurement and monitoring of these metrics.

With a measurable and practically interpretable definition of ξ, we now turn to addressing two key objectives introduced at the beginning of this chapter:

- Linearization of the relationship between resource allocation and net utility with respect to l.
- Clear differentiation of the roles of the load demand l and the benefit factor β.

To achieve these goals, we propose modifying the original net utility function accordingly. After implementing these modifications, and taking into account the measurability of ξ, we must reassess the strategy-proofness of the resulting model, specifically examining how these changes affect the incentives and potential strategic misrepresentation by SPs.

5.4 Introducing Modified Net Utility Functions

In the next sections, we propose different modifications to the original utility function that allow us to address the previously discussed limitations. Our goal is to preserve both the linear price term, which maintains economic realism, and the original article's intent to model diminishing returns through an exponential saturation function multiplying the potential monetization. To achieve this, we introduce a new auxiliary function:

$$f(\xi,\, l,\, \beta,\, d')$$

This function multiplies the allocation h in the exponential saturation term and may depend on any combination of the parameters ξ, l, β, and d'. It also may include additional constant factors. Note that by setting $f(\xi,\, l,\, \beta,\, d') = \xi$ we obtain the original model function.

Our modified new utility equations will be expressed as:

$$U = b \cdot l \cdot \left(1 - e^{-h \cdot f(\xi, l, \beta, d')}\right) - h \cdot d'$$

The corresponding first-order condition, giving the optimal allocation, is:

$$h^* = \frac{1}{f(\xi, l, \beta, d')} \ln\Big(\frac{b \cdot l \cdot f(\xi, l, \beta, d')}{d'}\Big)$$

Subject to the positive allocation constraint:

$$b \cdot l \cdot f(\xi, l, \beta, d') > d'$$

And defining the utility at this optimal allocation as:

$$U(h^*) = b\, l - \frac{d'}{f} - \frac{d'}{f} \ln\left(\frac{b\, l\, f}{d'}\right)$$

5.5 Modified Net Utility Function Case 1

To address the previously mentioned models' limitations, we define:

$$f(\xi, l) = \frac{\xi}{l}$$

The resulting net utility function is:

$$U = \beta \cdot l \cdot \left(1 - e^{-h \cdot \frac{\xi}{l}}\right) - h \cdot d' \tag{5.1}$$

The corresponding first-order condition for the optimal allocation is:

$$\frac{\partial U}{\partial h} = \beta\, \xi\, e^{-h \cdot \xi / l} - d' = 0 \quad \Longrightarrow \quad e^{-h^* \cdot \xi / l} = \frac{d'}{\beta \cdot \xi}$$

$$h^* = \frac{l}{\xi} \ln\left(\frac{\beta \cdot \xi}{d'}\right)$$

The positive allocation constraint is:

$$\frac{\beta \cdot \xi}{d'} > 1$$

We show that this optimal allocation equation does not depend on the time-slot length:

$$h^* = \frac{l_{\Delta_t}}{\xi_{\Delta_t}} \ln\Big(\frac{\beta\, \xi\, \Delta_t}{d_{\Delta_t}}\Big) = \frac{l}{\xi} \ln\Big(\frac{\beta \ \cdot \xi}{d}\Big)$$

Since we could eliminate the length of the time slot Δ_t the allocation remains constant for any time-slot duration.

5.5.1 Parameter Interpretation and Economic Intuition for the Optimal Allocation Function

The optimal allocation h^* is given by:

$$h^* = \frac{l}{\xi} \ln\left(\frac{\beta\xi}{d'}\right) \quad \text{with} \quad \frac{\beta\xi}{d'} > 1. \tag{5.2}$$

Interpretation of Allocation Terms

- First term: $\frac{l}{\xi}$

  This is the amount of request in the time-slot, divided by the amount of requests one millicore can serve in that time-slot, giving exactly the allocation needed to serve the requests if these were evenly distributed and using 100% of the allocated resources, this is a pure technical value.

- Second term: $\ln \frac{\beta \cdot \xi}{d'}$ This term relates the technical parameter with the economical ones. When the value of this term decreases from 1, the optimal allocation is less than the amount of "technically needed" resources, indicating that the relative weight of the allocation cost prevents the SP from serving all the potential requests, contrary, when the value of this term raises from 1, it indicates that the relative weight of the allocation cost allows the SP to have "extra resources" to ensure that at any time most of the requests are served at the Edge.

As in the original equation we have a value of ξ at which the allocation has maximum, the value of this ξ_{peak} is:

$$\xi_{\text{peak}} = \frac{e \ \cdot d'}{\beta}$$

The value of the allocation at this point is:

$$h^*(\xi_{\text{peak}}) = \frac{l \cdot \beta}{e \cdot d'}$$

And the value of the net utility subject to the optimal allocation at this point is:

$$U(\xi_{\text{peak}}) = l \cdot \beta \left(1 - \frac{2}{e}\right) \approx l * \beta \cdot 0.264$$

Note that the only difference from the ξ_{peak} of the original model is that l is not present in the definition of ξ_{peak} while $h^*(\xi_{\text{peak}})$ and $U(\xi_{\text{peak}})$ remain equal.

In the Section B.8.4 of the demonstrations Appendix, we derive this function for each variable. On table 5.1 , we present a summary of the derivative signs.

Parameter	**Sign of Derivative**	**Interpretation**	**Mathematical relationship**
β	Positive	Higher monetization justifies higher allocation.	Logarithmic
l	Positive	Larger load requires more allocation.	Linear
ξ	Positive until ξ_{peak}, negative after ξ_{peak}	Benefit-driven rise, then less need of resources-driven fall.	Both linear and logarithmic
d'	Negative	Increased marginal cost reduces allocation.	Logarithmic

Table 5.1: Summary of parameter influences on optimal allocation based on derivatives.

Note that we keep the sign of the derivative of each variable with respect to the original optimal allocation function, but could differentiate the role of l and β

5.5.2 Parameter Interpretation and Economic Intuition for the Net Utility Function at the Optimal Allocation

We consider the utility at the optimal allocation:

$$U(h^*) = \beta\, l \left(1 - \frac{d'}{\beta\xi}\right) - d' \frac{l}{\xi} \ln\left(\frac{\beta\xi}{d'}\right), \quad \text{with} \quad \frac{\beta\xi}{d'} > 1. \tag{5.3}$$

Interpretation of Utility Terms

- First term: $\beta l(1 - \frac{d'}{\beta\xi})$

  Represents the total revenue generated by serving a fraction $1 - \frac{d'}{\beta\xi}$ of the load l at per-unit benefit β.

- Second term: $-d' \frac{l}{\xi} \ln\left(\frac{\beta\xi}{d'}\right)$ This term equals the total cost of allocating the resources h^* at amortized marginal cost d', since $h^* = \frac{l}{\xi} \ln\left(\frac{\beta\xi}{d'}\right)$.

In the Section B.8.3 of the demonstrations Appendix, we derive this function for each variable. On table 5.2, we present a summary of the derivative signs.

Parameter	Sign of Derivative	Interpretation
β	Positive	Increasing monetization increases utility
l	Positive	Higher load directly increases utility
ξ	Positive	Increased capacity to serve requests increases utility
d'	Negative	Higher marginal costs reduce utility

Table 5.2: Summary of parameter influences on net utility based on derivatives.

Note that we keep the sign of the derivative of each variable with respect to the original utility function, but could differentiate the role of l and β.

Considering that now the NO can measure l and ξ, and d' is fixed externally, the only parameter vulnerable to misreporting is β. Intuitively, we can observe that β influences the optimal allocation logarithmically, but in the net utility it appears in a dominant linear revenue term and a smaller logarithmic cost term, so its overall impact on utility is predominantly linear.

For that reason, an SP may understate β so as to reduce its declared net utility by a larger proportion than its optimal allocation and its actual net utility. This asymmetry in the influence of optimal allocation motivates the following alternative net utility modification, that although it doesn't enforce strategy-proofness, we think is worth mentioning.

5.6 Modified Net Utility Function Case 2

The proposed f function that we are defining below is not a definitive one, but it is expressed to show a line of reasoning. In this alternative net utility formulation, we aim for the allocation to depend linearly on β, l, and d'. To achieve this, we introduce a constant λ, tied to the percentage of requests served at the Edge, a quantity measurable by the NO. By fixing λ, we also fix the exponential-saturation level:

$$1 - e^{-h \cdot f(\xi, l, \beta, d')} = 1 - \frac{1}{\lambda}$$

which univocally defines each SP's type. In order to achieve that, we define the auxiliary function as

$$f(\rho,\, d',\, \lambda) \;=\; \frac{d'}{\rho}\,\lambda,$$

leading to the net utility function:

$$U \;=\; \rho\Big(1 - e^{-\,h\,\frac{d'}{\rho}\,\lambda}\Big) \;-\; d'\,h. \tag{5.4}$$

The first-order condition for the optimal allocation is:

$$h^* \;=\; \frac{\rho}{d'\,\lambda}\,\ln(\lambda),$$

valid whenever $\lambda > 1$.

Thus, substituting h^* into the exponential saturation term yields $1 - 1/\lambda$, which appears in

$$U(h^*) = \rho\Big(1 - \tfrac{1}{\lambda} - \tfrac{1}{\lambda}\ln(\lambda)\Big) \tag{5.5}$$

If λ is defined via the observed fraction p of requests handled at the edge,

$$p \;=\; 1 - e^{-\lambda} \quad\Longrightarrow\quad \lambda \;=\; -\,\ln\big(1-p\big) \tag{5.6}$$

then the optimal allocation

$$h^* = \frac{\rho}{d'\,\lambda}\,\ln(\lambda)$$

requires $\lambda > 1$ for $h^* > 0$. Equivalently,

$$\ln(\lambda) > 0 \iff \lambda > 1 \iff -\ln\big(1-p\big) > 1 \iff p > 1 - e^{-1} \approx 0.632$$

This formulation therefore represents only those SPs whose fraction of requests served at the edge exceeds approximately 63.2%. That limitation motivates a new definition of $f(\xi, l, \beta, d')$, where λ can vary over $(0, 1)$ by appearing in the denominator of the exponent. For that, we define the auxiliary function as:

$$f(\rho,\, d',\, \lambda) = \frac{d'}{\rho\,\lambda}$$

yielding the net utility function:

$$U(h) = \rho\Big(1 - e^{-\,h\,\frac{d'}{\rho}\,\frac{1}{\lambda}}\Big) - d'\,h$$

The optimal allocation from the first-order condition is:

$$h^* = \frac{\rho\,\lambda}{d'}\,\big(-\ln\lambda\big)$$

with the positive-allocation constraint $0 < \lambda < 1$.

In an analogous way to what we did before, we can show that the exponential saturation equation is indeed:

$$1 - e^{-h \cdot f(\xi, l, \beta, d')} = 1 \; - \; \lambda$$

5.6.1 Parameter Interpretation and Economic Intuition for the Optimal Allocation Function

The optimal allocation h^* is given by:

$$h^* = \frac{\rho\,\lambda}{d'}\left(-\ln\lambda\right) \quad \text{with} \quad 0 < \lambda < 1 \tag{5.7}$$

Interpretation of Allocation Terms

The expression splits into two factors:

- $\frac{\rho\,\lambda}{d'}$: shows ρ and λ being directly proportional to h^*, and d' inversely proportional.
- $-\ln\lambda$: is the logarithmic amplification, which grows as λ decreases.

As in the original formulation and the previously proposed modified utility functions, there is a value of λ at which the optimal allocation reaches its maximum. For this inverse λ formulation, the peak occurs at:

$$\lambda_{\text{peak}} = \frac{1}{e}$$

The optimal allocation at this point remains equal to the previous cases:

$$h^*(\lambda_{\text{peak}}) = \frac{\rho}{e \cdot d'}$$

Similarly, the net utility at the optimal allocation remains unchanged compared to previous cases:

$$U(\lambda_{\text{peak}}) = l \cdot \beta \left(1 - \frac{2}{e}\right) \approx l \cdot \beta \cdot 0.264$$

Thus, the only difference from the previous cases is the value of λ_{peak} itself, while the optimal allocation and its corresponding utility at this point remain consistent. On table 5.3, we present a summary of the derivative signs.

Parameter	**Sign of $\frac{\partial h^*}{\partial \cdot}$**	**Interpretation**	**Relationship**
ρ	Positive	More monetization yields more allocation.	Linear
λ	Positive until λ_{peak}, then negative	Benefit-driven rise, then less need of resources-driven fall.	Both
d'	Negative	Higher marginal cost reduces allocation.	Both

Table 5.3: Effects of parameters on the optimal allocation h^*.

5.6.2 Parameter Interpretation and Economic Intuition for the Net Utility Function at the Optimal Allocation

We consider the utility at the optimal allocation:

$$U(h^*) = \rho\Big(1 - \lambda + \lambda \ln \lambda\Big), \quad \text{with} \quad 0 < \lambda < 1 \tag{5.8}$$

Interpretation of Utility Terms

- $\rho\big(1-\lambda\big)$: revenue from serving a fraction $1-\lambda$ of total requests.
- $\rho\,(\lambda \ln \lambda)$: cost term from the allocated resources (note $\ln \lambda < 0$ for $\lambda < 1$).

On table 5.4, we present a summary of the derivative signs.

Influence of Parameters on $U(h^*)$

Parameter	**Sign of $\partial U(h^*)/\partial\cdot$**	**Interpretation**
ρ	Positive	Greater potential monetization raises net utility
λ	Negative	Higher λ reduces net utility (smaller edge share)
d'	Zero	Cost changes shift allocation but net utility remains unchanged

Table 5.4: Effects of parameters on net utility at the optimal allocation.

Note that in this case we inverted the influence of λ compared to previous equations where the greater the value of μ or ξ the greater the percentage of requests served at the Edge.

Similarly to what we did on equation 5.6 the NO can infer λ from the observed fraction p of requests served at the edge:

$$p \;=\; 1 \;-\; e^{-\,1/\lambda} \quad\Longleftrightarrow\quad e^{-\,1/\lambda} = 1 - p \quad\Longleftrightarrow\quad -\frac{1}{\lambda} = \ln\big(1-p\big) \quad\Longleftrightarrow\quad \lambda = -\,\frac{1}{\ln\big(1-p\big)}$$

Because the Network Owner measures p directly (by counting edge-served requests versus total), the value

$$\lambda = -\,\frac{1}{\ln\big(1-p\big)} \tag{5.9}$$

is empirically observable. This procedure yields a valid $\lambda \in (0,1)$ whenever $0 < p < 1$, and it does not impose a lower bound of 63% or any lower bound at all.

In this formulation, the parameter λ uniquely determines the edge-served fraction:

$$p = 1 - e^{-\,1/\lambda}$$

Consequently, once λ (and thus p) is observed, the marginal cost d' only affects the allocation

$$h^* = \frac{\rho\,\lambda}{d'}\big(-\ln \lambda\big)$$

but drops out of the net utility at optimum:

$$U(h^*) = \rho\big(1 - \lambda + \lambda \ln \lambda\big).$$

Because $U(h^*)$ no longer depends on d', a change in cost only changes h^*, not the achieved net utility. This is mathematically consistent but economically restrictive, since it implies that higher costs are fully absorbed by adjusting allocation without reducing net benefit. For applications requiring that a higher cost actually lowers utility, one should use a model in which d' appears explicitly in the final utility expression rather than only in the allocation.

5.7 Modified Net Utility Function Case 3

In this formulation, we remove the marginal cost d' from the exponential saturation term. Specifically, we define

$$f(\rho, \xi, d') \;=\; \frac{\xi}{\rho}$$

then the net utility function becomes:

$$U(h) \;=\; \rho\Big(1 - e^{-\,h\,\frac{\xi}{\rho}}\Big) \;-\; d'\,h \tag{5.10}$$

The corresponding first-order condition for the optimal allocation is:

$$\frac{\partial U}{\partial h} \;=\; \xi\, e^{-\,h\,\xi/\rho} \;-\; d' \;=\; 0 \quad\Longrightarrow\quad e^{-\,h^*\,\xi/\rho} \;=\; \frac{d'}{\xi}$$

hence, the optimal allocation equation is:

$$h^* \;=\; \frac{\rho}{\xi}\,\ln\!\Big(\frac{\xi}{d'}\Big), \tag{5.11}$$

with the positive-allocation constraint:

$$\frac{\xi}{d'} > 1 \quad\Longleftrightarrow\quad \xi > d'$$

Note that in this case we are using the same definition of ξ as in case one; this is the amount of requests that can be served at the Edge by one millicore in one time-slot of duration Δ_t. Observe that the time-dependent variables cancel each other, making h^* time-slot length invariant.

$$h^* \;=\; \frac{l\Delta_t\,\beta}{\xi_{\Delta_t}}\,\ln\!\Big(\frac{\xi_{\Delta_t}}{d'_{\Delta_t}}\Big),$$

5.7.1 Influence of Parameters on Optimal Allocation

In table 5.5 we present the influence of the parameter on the optimal allocation h^*.

5.7.2 Net Utility at Optimal Allocation and Its Interpretation

Substituting h^* into $U(h)$ yields

Parameter	$\partial h^*/\partial \cdot$	Interpretation	Relationship
ρ	Positive	Greater potential monetization increases the scale of allocation h^*.	Linear
ξ	Positive until ξ_{peak}, negative after ξ_{peak}	Greater serving capacity initially drives up allocation, then yields diminishing returns.	Both linear and logarithmic
d'	Negative	Higher marginal cost reduces optimal allocation.	Logarithmic

Table 5.5: Effects of parameters on the optimal allocation h^*.

$$U(h^*) = \rho\Big(1 - e^{-\,h^*\,\xi/\rho}\Big) \;-\; d'\,h^* = \rho\Big(1 - \tfrac{d'}{\xi}\Big) - d'\,\frac{\rho}{\xi}\,\ln\Big(\frac{\xi}{d'}\Big),$$

which can also be written as

$$U(h^*) = \rho\Big(1 - \tfrac{d'}{\xi} - \tfrac{d'}{\xi}\,\ln\big(\tfrac{\xi}{d'}\big)\Big). \tag{5.12}$$

Interpretation of Utility Terms

- $\rho\Big(1 - \frac{d'}{\xi}\Big)$: revenue from serving a fraction $\big(1 - \frac{d'}{\xi}\big)$ of the potential demand, since ξ/d' measures technical capacity relative to cost. Note that the positive allocation constraint allows us to represent any fraction of the total requests to be served at the Edge.
- $-\,\dfrac{\rho\,d'}{\xi}\,\ln\Big(\frac{\xi}{d'}\Big)$: total cost of allocated resources at price d', adjusted by diminishing returns.

In table 5.6 we present a summary of parameters' interpretation on the net utility at the optimal allocation.

Parameter	$\partial U(h^*)/\partial \cdot$	Interpretation
ρ	Positive	Greater monetization increases net utility
ξ	Positive (for $\xi > d'$)	Higher service rate raises utility
d'	Negative	Higher marginal cost reduces net utility

Table 5.6: Effects of parameters on net utility at the optimal allocation.

5.8 Formal Demonstration of Non-Strategy-Proofness

We provide a formal proof demonstrating why all previously introduced utility functions fail to meet the conditions necessary for strategy-proofness, along with an explicit equation to quantify the profit an SP can gain from misreporting the benefit factor β.

General Formulation

Consider the general net utility function:

$$U(\beta, l, h) = \beta\,l\Big(1 - e^{-\,h\,f(\xi,\beta,l,d')}\Big) - h\,d'$$

where:

- β is declared by the SP.
- l and ξ are measurable and d' is exogenous.
- The allocation is chosen to maximize the declared utility, so we solve

$$\frac{\partial U}{\partial h} = 0 \quad \Longrightarrow \quad h^*(\beta_{\text{dec}})$$

SP's Retained Utility

After the allocation h^* is determined using β_{dec}, the SP's actual realized utility is computed using β_{real}, and the declared utility is computed using β_{dec}. Since the SP transfers half of its declared net utility to the NO, the SP keeps:

$$U_{\text{SP}}(\beta_{\text{dec}}) = U_{\text{real}}(\beta_{\text{real}}, h^*) - \tfrac{1}{2}\, U_{\text{dec}}(\beta_{\text{dec}}, h^*).$$

Where:

$$U_{\text{dec}} = \beta_{\text{dec}}\, l\Big(1 - e^{-\,h^*\, f(\xi, \beta_{\text{dec}}, l, d')}\Big) - h^*\, d',$$

is the utility computed with the declared benefit factor, and

$$U_{\text{real}} = \beta_{\text{real}}\, l\Big(1 - e^{-\,h^*\, f(\xi, \beta_{\text{dec}}, l, d')}\Big) - h^*\, d',$$

is what the SP actually earns when using the same allocation h^*. Thus U_{SP} captures the fact that the SP's payoff depends both on the allocation (which itself depends on β_{dec}) and on the true benefit β_{real}.

Condition for Strategy-Proofness

To show whether truth telling is optimal for the SP, we must check that no deviation from reporting β_{real} can increase the SP's retained utility U_{SP}. Concretely, the standard strategy-proofness conditions are that, at $\beta_{\text{dec}} = \beta_{\text{real}}$

$$\frac{\partial U_{\text{SP}}}{\partial \beta_{\text{dec}}}\Big|_{\beta_{\text{dec}}=\beta_{\text{real}}} = 0, \quad \frac{\partial^2 U_{\text{SP}}}{\partial \beta_{\text{dec}}^2}\Big|_{\beta_{\text{dec}}=\beta_{\text{real}}} < 0$$

These two conditions ensure that the truthful report is a stationary point of U_{SP}, and that it is a local maximum.

Because U_{SP} depends on β_{dec} both through the allocation $h^*(\beta_{\text{dec}})$ and the declared utility term, we use the chain rule and write

$$\frac{\partial U_{\text{SP}}}{\partial \beta_{\text{dec}}} = \frac{\partial U_{\text{real}}}{\partial h^*}\frac{\partial h^*}{\partial \beta_{\text{dec}}} - \tfrac{1}{2}\frac{\partial U_{\text{dec}}}{\partial \beta_{\text{dec}}}$$

- The term $\frac{\partial U_{\text{real}}}{\partial h^*}$ measures how the SP's real utility would change if the allocation h^* shifted slightly, holding β_{dec} fixed in the function.
- The factor $\frac{\partial h^*}{\partial \beta_{\text{dec}}}$ captures how the allocation itself changes when β_{dec} changes.
- The term $\frac{1}{2}\frac{\partial U_{\text{dec}}}{\partial \beta_{\text{dec}}}$ captures the direct marginal effect on the declared utility from changing β_{dec} (and in consequence the fraction the SP must transfer).

At the truthful report $\beta_{\text{dec}} = \beta_{\text{real}}$, the allocation h^* was chosen to maximize the SP's declared utility. Because the first-order condition for h^* is

$$\frac{\partial U}{\partial h}\Big(\beta_{\text{dec}}, h^*\Big) = 0,$$

it follows that $\partial U_{\text{real}}/\partial h^* = 0$ when $\beta_{\text{dec}} = \beta_{\text{real}}$. Therefore the first term in the chain-rule expression vanishes at truth. The only way to have

$$\frac{\partial U_{\text{SP}}}{\partial \beta_{\text{dec}}}\Big|_{\beta_{\text{dec}}=\beta_{\text{real}}} = 0$$

is then to force

$$\frac{\partial U_{\text{dec}}}{\partial \beta_{\text{dec}}}\Big|_{\beta_{\text{dec}}=\beta_{\text{real}}} = 0$$

In other words, truthful reporting can be a local maximizer of retained utility only if changing β_{dec} has zero first-order effect on the declared utility term at the truthful point. Since our exponential saturation form never satisfies that condition $\partial U_{\text{dec}}/\partial \beta_{\text{dec}}$ is nonzero at $\beta_{\text{dec}} = \beta_{\text{real}}$.

We have proven that no utility function with an exponential saturation equation to model the diminishing returns can be strategy-proof.

Explicit Equation for Profit from Misreporting

Even though truthful declarations fail to be a local maximum, we can still quantify how much an SP gains by choosing the best misreported value; for that, we define:

$$\text{Profit } (\%) = \frac{U_{\text{SP}}(\beta^*_{\text{dec}}) \; - \; U_{\text{SP}}(\beta_{\text{real}})}{U_{\text{SP}}(\beta_{\text{real}})} \times 100\%,$$

where

$$\beta^*_{\text{dec}} = \arg\max_{\beta_{\text{dec}}} U_{\text{SP}}(\beta_{\text{dec}}).$$

5.8.1 Misreporting Benefit Factor for a Generic Auxiliary Function f

We assume the NO measures ξ but the SP may declare $\beta_{\text{dec}} \neq \beta_{\text{real}}$. We then write:

- Declared allocation:
$$h_{\text{dec}} = \frac{1}{f(\xi, \beta_{\text{dec}}, l, d')} \ln\Big(\frac{\beta_{\text{dec}}\, l\, f(\xi, \beta_{\text{dec}}, l, d')}{d'}\Big)$$
- Declared utility (computed using β_{dec}):
$$U_{\text{dec}} = \beta_{\text{dec}}\, l\Big(1 - e^{-\,h_{\text{dec}}\, f(\xi, \beta_{\text{dec}}, l, d')}\Big) - d'\, h_{\text{dec}}$$
- Actual utility (using the true β_{real} but the same allocation):
$$U_{\text{real}} = \beta_{\text{real}}\, l\Big(1 - e^{-\,h_{\text{dec}}\, f(\xi, \beta_{\text{dec}}, l, d')}\Big) - d'\, h_{\text{dec}}$$
- Utility retained by the SP:
$$U_{\text{SP}}(\beta_{\text{dec}}) = U_{\text{real}} - \tfrac{1}{2}\, U_{\text{dec}}$$

Substitute the formula for h_{dec} into U_{dec} and U_{real}. We find:

$$U_{\text{SP}}(\beta_{\text{dec}}) = \beta_{\text{real}}\, l - \frac{\beta_{\text{real}}\, l\, d'}{\beta_{\text{dec}}\, l\, f(\xi, \beta_{\text{dec}}, l, d')} - \frac{d'\, h_{\text{dec}}}{2} - \tfrac{1}{2}\, \beta_{\text{dec}}\, l$$

Finally, solving by solving:

$$\frac{d\, U_{\text{SP}}}{d\, \beta_{\text{dec}}} = 0$$

we get the best misreport β^*_{dec}, and the percentage of gain is:

$$\text{Gain } (\%) = \frac{U_{\text{SP}}(\beta^*_{\text{dec}}) - U_{\text{SP}}(\beta_{\text{real}})}{U_{\text{SP}}(\beta_{\text{real}})} \times 100\%.$$

5.8.2 Finding Optimal Declared Benefit Factor for Each of the Proposed Utility Function Modifications

Similarly to what we did in section 3.13.2 where we provided an equation for the optimal misreporting of ρ in the original formulation, we aim to find the optimal misreporting of β for the alternative net utility functions. We will find a closed form for β^*_{dec} for Case 1, and show that there is no closed form for β^*_{dec} in Cases 2 and 3, since the optimal declaration for beta is $\beta^*_{\text{dec}} \to 0^+$. We will subsequently analyze why this occurs and provide an alternative interpretation of strategy-proofness, under which this result can indeed be beneficial.

Case 1: $f(\xi, \beta, l, d') = \frac{\xi}{l}$

We don't reproduce all calculations here since they are very similar to what we did in section 3.13.2. We just recall the h^* equation for this case:

$$h_{\text{dec}} = \frac{l}{\xi} \ln\!\Big(\frac{\beta_{\text{dec}}\, \xi}{d'}\Big)$$

then solving $dU_{\text{SP}}/d\beta_{\text{dec}} = 0$ yields

$$\xi\, \beta^2_{\text{dec}} + d'\, \beta_{\text{dec}} - 2\, d'\, \beta_{\text{real}} = 0$$

$$\beta^*_{\text{dec}} = \frac{-\, d' + \sqrt{d'^2 + 8\, d'\, \beta_{\text{real}}\, \xi}}{2\, \xi}, \qquad \text{Gain } (\%) \to 100\% \text{ as } \beta_{\text{real}} \to \infty \tag{5.13}$$

If we assume that the declared value of β is always β^*_{dec} then we can calculate the real value of β as:

$$\beta_{\text{real}} = \frac{\beta^*_{\text{dec}}\big(\xi\, \beta^*_{\text{dec}} + d'\big)}{2\, d'} \tag{5.14}$$

Case 2: $f(\rho, d', \lambda) = \dfrac{d'}{\rho\, \lambda}$

In this formulation, $0 < \lambda < 1$ so that $-\ln \lambda > 0$. Hence, the declared allocation is:

$$h_{\text{dec}} = \frac{1}{\frac{d'}{\beta_{\text{dec}}\, l}\ \frac{1}{\lambda}} \ln\!\Big(\frac{\beta_{\text{dec}}\, l\ \left(\frac{d'}{\beta_{\text{dec}}\, l}\ \frac{1}{\lambda}\right)}{d'}\Big) = \frac{\beta_{\text{dec}}\, l\, \lambda}{d'}\,(-\ln\lambda)$$

Because h_{dec} depends linearly on β_{dec}, we have

$$h_{\text{dec}} = \frac{\beta_{\text{dec}}\, l\, \lambda}{d'}\,(-\ln\lambda)$$

Next, note that

$$e^{-\,h_{\text{dec}}\, f} = e^{-\left(\frac{\beta_{\text{dec}}\, l\, \lambda}{d'}\,(-\ln\lambda)\right)\times\left(\frac{d'}{\beta_{\text{dec}}\, l}\ \frac{1}{\lambda}\right)} = e^{-\ln\lambda} = \lambda$$

Therefore

$$U_{\text{dec}} = \beta_{\text{dec}}\, l\,\big(1-\lambda\big)\ -\ d'\, h_{\text{dec}} = \beta_{\text{dec}}\, l\,(1-\lambda)\ -\ \beta_{\text{dec}}\, l\,\lambda\,(-\ln\lambda) = \beta_{\text{dec}}\, l\Big[\,(1-\lambda)\ -\ \lambda\,(-\ln\lambda)\Big]$$

and

$$U_{\text{real}} = \beta_{\text{real}}\, l\,\big(1-\lambda\big)\ -\ d'\, h_{\text{dec}} = \beta_{\text{real}}\, l\,(1-\lambda)\ -\ \beta_{\text{dec}}\, l\,\lambda\,(-\ln\lambda)$$

The utility retained by the SP is

$$U_{\text{SP}}(\beta_{\text{dec}}) = U_{\text{real}}\ -\ \tfrac{1}{2}\,U_{\text{dec}} = \beta_{\text{real}}\, l\,(1-\lambda)\ -\ \beta_{\text{dec}}\, l\,\lambda\,(-\ln\lambda)\ -\ \tfrac{1}{2}\Big[\beta_{\text{dec}}\, l\,(1-\lambda)\ -\ \beta_{\text{dec}}\, l\,\lambda\,(-\ln\lambda)\Big]$$

Collecting like terms gives

$$U_{\text{SP}}(\beta_{\text{dec}}) = \beta_{\text{real}}\, l\,(1-\lambda)\ -\ \frac{\beta_{\text{dec}}\, l}{2}\Big[\,(1-\lambda)\ +\ \lambda\,(-\ln\lambda)\Big]$$

Since $(1-\lambda) > 0$ and $(-\ln\lambda) > 0$ for $0 < \lambda < 1$, one sees

$$\frac{d\,U_{\text{SP}}}{d\,\beta_{\text{dec}}} = -\,\frac{l}{2}\Big[\,(1-\lambda)\ +\ \lambda\,(-\ln\lambda)\Big] < 0$$

Consequently $U_{\text{SP}}(\beta_{\text{dec}})$ is strictly decreasing in β_{dec}, and the SP's optimal misreport is

$$\beta^{*}_{\text{dec}}\ \to\ 0^{+}$$

Case 3: $f(\rho,\xi,d') = \dfrac{\xi}{\rho}$

Here $\rho = \beta\, l$ and the positive allocation constraint is $\xi > d'$. We compute:

$$h_{\text{dec}} = \frac{1}{\frac{\xi}{\beta_{\text{dec}}\, l}} \ln\!\Big(\frac{\beta_{\text{dec}}\, l\ \frac{\xi}{\beta_{\text{dec}}\, l}}{d'}\Big) = \frac{\beta_{\text{dec}}\, l}{\xi}\ \ln\!\Big(\frac{\xi}{d'}\Big)$$

Since $e^{-\,h_{\text{dec}}\, f} = e^{-\ln(\frac{\xi}{d'})} = \frac{d'}{\xi}$, one finds

$$U_{\text{dec}} = \beta_{\text{dec}}\, l\left(1 - \tfrac{d'}{\xi}\right) \;-\; d'\, h_{\text{dec}} = \beta_{\text{dec}}\, l\left(\tfrac{\xi - d'}{\xi}\right) \;-\; \beta_{\text{dec}}\, l\, \tfrac{d'}{\xi}\, \ln\!\left(\tfrac{\xi}{d'}\right)$$

$$U_{\text{real}} = \beta_{\text{real}}\, l\left(1 - \tfrac{d'}{\xi}\right) \;-\; d'\, h_{\text{dec}} = \beta_{\text{real}}\, l\left(\tfrac{\xi - d'}{\xi}\right) \;-\; \beta_{\text{dec}}\, l\, \tfrac{d'}{\xi}\, \ln\!\left(\tfrac{\xi}{d'}\right)$$

$$U_{\text{SP}}(\beta_{\text{dec}}) = U_{\text{real}} - \tfrac{1}{2}\, U_{\text{dec}} = \beta_{\text{real}}\, l\left(\tfrac{\xi - d'}{\xi}\right) \;-\; \frac{\beta_{\text{dec}}\, l}{2\,\xi}\left[(\xi - d') + d' \ln \tfrac{\xi}{d'}\right]$$

Because $(\xi - d') > 0$ and $\ln(\xi/d') > 0$ when $\xi > d'$, we have

$$\frac{d\, U_{\text{SP}}}{d\, \beta_{\text{dec}}} = -\,\frac{l}{2\,\xi}\left[(\xi - d') + d' \ln \tfrac{\xi}{d'}\right] < 0$$

Thus $U_{\text{SP}}(\beta_{\text{dec}})$ decreases in β_{dec}, and the SP maximizes its retained utility by choosing $\beta^*_{\text{dec}} \to 0^+$.

5.8.3 Summary and Interpretation

As with the original formulation, we derived a closed-form expression for β^*_{dec} in Case 1. In Cases 2 and 3, however, the optimal declared value is $\beta_{\text{dec}} \to 0^+$. At first glance, letting $\beta \to 0$ would force the optimal allocation $h^* \to 0$, which initially appears paradoxical. This could indicate a problematic net utility function design; nevertheless, these cases might still be practically viable. Notably, in both Cases 2 and 3, the net utility at the optimal allocation depends linearly on β. Specifically:

- In Case 2:
$$U(h^*) \;=\; \rho\Big(1 - \tfrac{1}{\lambda} - \tfrac{1}{\lambda}\ln(\lambda)\Big), \quad \rho = \beta\, l.$$
- And in Case 3:
$$U(h^*) = \rho\Big(1 - \tfrac{d'}{\xi} - \tfrac{d'}{\xi}\, \ln\!\big(\tfrac{\xi}{d'}\big)\Big), \quad \rho = \beta\, l.$$

The factor in parentheses does not depend on β. As long as the SP picks β_{dec} so that the first-order condition is satisfied, that same constant multiplies ρ and fully determines the utility. Hence β appears only inside the allocation formula but only linearly in the final utility expression.

Because the SP's retained utility is
$$U_{\text{SP}} = U_{\text{real}} - \tfrac{1}{2}\, U_{\text{dec}},$$
making β_{dec} smaller drives both the declared allocation h^* and the declared utility U_{dec} toward zero, but it does not reduce the real utility U_{real} by a proportional amount. In fact, the ratio $\frac{U_{\text{SP}}}{U_{\text{real}}}$ approaches one as $\beta_{\text{dec}} \to 0^+$. That is why no finite interior solution exists: the best 'solution' is always to push β_{dec} as low as the rules allow.

A Different Notion of Strategy-Proofness

Under the classical definition of strategy-proofness, namely, that no SP can misdeclare β in a way that raises its retained utility, Cases 2 and 3 clearly fail; moreover, we could say that these functions fail to represent the net utility of an SP. However, suppose we shift our focus away from maximizing retained net utility and instead ask:

> Is there any way for an SP to declare a lower net utility (smaller β) while simultaneously securing a proportionally less lower allocation?

Under this alternative definition:

- In Case 2 the optimal allocation is:

$$h^* = \frac{\rho\,\lambda}{d'}\big(-\ln\lambda\big) = \frac{\beta\,l\,\lambda}{d'}\big(-\ln\lambda\big).$$

  Here h^* is proportional to β. If an SP lowers β_{dec}, then $\rho_{\text{dec}} = \beta_{\text{dec}}\,l$ falls and h^* falls in exactly the same proportion.

- In Case 3 the optimal allocation is:

$$h^* = \frac{\rho}{\xi}\,\ln\!\big(\tfrac{\xi}{d'}\big) = \frac{\beta\,l}{\xi}\,\ln\!\big(\tfrac{\xi}{d'}\big).$$

  Again h^* is proportional to β. Any under-declaration of β strictly reduces h^* in the same proportion.

Hence, if we adopt the criterion "no SP can declare a lower β and thereby get a proportionally larger portion of h^*" then Cases 2 and 3 do satisfy this.

Note that in Case 1 as well as in the original formulation, this is not the case, since an SP may underdeclare β and get a proportionally larger allocation. The key difference in these formulations is that in Case 2 and 3, h^* has a linear relationship with β while in the first two this relationship is logarithmic.

Viewed in this light, the formulas for Cases 2 and 3 admit an alternative interpretation; suppose the SPs are partially or fully not identified by the net utility equations, but they are taken as price functions. Since other parameters are even measurable or exogenous, each SP then "buys" allocation h^* at $\frac{U(h^*)}{2} - d'\,h$ which combines the allocation cost with that SP's share of the Network Owner's Shapley value.

5.8.4 Summary of Strategy-Proofness Results and Real-World Implications

To complete our analysis on strategy-proofness, we summarize our findings and propose new mechanisms to counter the lack of strategy-proofness based on the quality of service the NO provides.

Case 0 and Case 1: Bounded Misreport Gains

In the original model (Case 0) and the first modified one (Case 1), we showed:

- If only β remains private, the SP still benefits by understating it, the SP's optimal misreport of β can be computed in closed form. Thus, although the model fails strategy-proofness, there is a finite upper bound on how much net utility can be retained by the SP by giving an untruthful report of β.

Case 2 and Case 3: Allocation-Price Forms

- The optimal allocation h^* depends linearly on β. As a result, any under-declaration of β reduces the SP's actual allocation by exactly the same proportion.

- Under the usual definition of strategy-proofness (no SP can lie and end up with strictly more retained utility), Cases 2 and 3 completely fail since $\beta_{\text{dec}} \to 0$ lets the SP keep nearly 100% of its true utility.

- However, under the criterion "no SP can get a proportionally larger allocation by misdeclaring β", then Cases 2 and 3 become strategy-proof: From a purely allocation-oriented perspective, no SP can buy more CPU for less money.

Interpreting β as Time-Sensitivity

In the underlying economic model, β represents a per-unit benefit: every request served at the Edge delivers β monetary units to the SP. As the original article notes:

> Time-sensitivity here is captured by the parameter β. Resource allocation at the Edge must be based not only on load, but also on the nature of the services, and in particular on time-sensitivity, which is reflected in a different benefit per unit of load satisfied at the Edge.

Thus:

- A higher β means each request served at the Edge is worth more, typical of real-time or ultra-low-latency services (e.g. industrial IoT, real-time analytics).
- A lower β corresponds to less time sensitive or best effort workloads (e.g. background data processing, non-urgent updates).

Real-World QoS Mechanisms to Encourage Honest β Reporting

Quality of Service (QoS) refers to technologies and mechanisms network providers use to manage bandwidth, latency, and reliability, ensuring differentiated performance levels for various types of network traffic. Since β encodes time sensitivity, the NO, typically an Internet Service Provider (ISP), can tie the declared value of β to measurable QoS guarantees and priorities. Specifically:

Prioritized Internet Slicing Based on β:

- The NO assigns distinct network slices with differentiated latency and throughput characteristics for each declared value of β. Higher declared values of receive slices with better latency and throughput.
- Each SP thus automatically obtains service quality strictly ordered by their declared β. Under declaring β leads directly to assignment on a lower quality slice, providing an inherent penalty for dishonesty.

Post Initial Allocation Market Mechanisms:

Under this new scenario, a secondary market as the one described in Section 3.15.2 may not even be necessary since the incentives for misreporting can be entirely countered by the quality slice SPs sit in. Nevertheless, if we were to consider such mechanisms, we could include several variants:

- Enforce or not SPs to sell their allocation.
- Buying or selling, depending on their updated expected load and market price, could be useful to mitigate misestimations or even to profit from pure speculation.
- If SP1 buys allocation from SP2 such that $\beta_1 > \beta_2$ then it has to pay SP2 for what SP2 would get until the end of the co-investment, and at this point, it has two options:
  - Option 1: Keep the value of β_2 as it was and have some allocated resources in a lower quality slice.
  - Option 2: Make the value of $\beta_2 = \beta_1$ increase the amount of allocated resources at the same quality slice it originally has, but having to pay proportionally more tho the NO.
- Allow or not to buy allocations from an SP with a greater β value.

These options, along with others, create new scenarios that, although they fall outside of the scope of this work, we think are worth exploring.

5.9 Positive Price Externality and Dynamic Allocation Synergies

In this section, we extend the baseline allocation model by incorporating two distinct mechanisms. The first is a positive externality of volume discount pricing, where the per-millicore price diminishes as the coalition's total allocation grows. The second is a player synergy enabled by dynamic allocation, which allows SPs to vary their resource shares across time-slots to exploit time-varying allocation benefits. We consider three model variants: (i) externality only; (ii) synergy only; and (iii) both effects together. This enables us to isolate each mechanism's impact and to assess their joint effect on aggregate utility.

5.9.1 Linear Volume-Discount (Sublinear) Per-millicore Pricing

Recall that C denotes the total CPU allocation in millicores, with $C_{\min} \leq C \leq C_{\max}$. We impose a linear volume-discount schedule (also known as declining-block pricing) by defining a per-millicore price

$$p(C) = p_{\max} - \frac{C - C_{\min}}{C_{\max} - C_{\min}} (p_{\max} - p_{\min}),$$

so that $p(C_{\min}) = p_{\max}$ and $p(C_{\max}) = p_{\min}$. The total cost is then

$$\text{Cost}(C) = C\, p(C) = \left[p_{\max} - \frac{C - C_{\min}}{C_{\max} - C_{\min}} (p_{\max} - p_{\min}) \right] C.$$

Because $p(C)$ decreases linearly in C, $\text{Cost}(C)$ is strictly concave (sublinear) in C. Economically, this models a positive scale externality: as the coalition's total allocation grows, each additional millicore becomes cheaper.

5.9.2 Introducing Dynamic Allocation

We propose a dynamic allocation mechanism in which each SP's allocation can vary across time-slots while ensuring that the entire deployed capacity C is always used. Because the cost of hardware deployment is paid upfront and there is no variable charge for millicore consumption, any idle capacity in a slot sacrifices strictly positive benefit without lowering cost. Under these conditions, the marginal benefit of allocating each additional millicore remains positive, and the total benefit is maximized by allocating at the capacity boundary in every time-slot. We do not consider the possibility of saving money through reduced energy cost, as energy expenditures are not modeled in our framework.

Under dynamic allocation, the exponential term in the saturation component of the original net utility equation (Case 0) can vary across time-slots. This variability occurs because the diminishing returns parameter remains constant despite changes in load, preventing service providers from maintaining a consistent fraction of requests served at the edge throughout the day. In contrast, the modified net utility equations (Cases 1, 2, and 3) explicitly incorporate load into this diminishing returns parameter, thereby ensuring that each SP consistently serves the same percentage of requests at the edge in every time-slot.

It is important to note that neither of these modeling approaches is inherently incorrect. They simply represent distinct design decisions regarding how the percentage of requests served at the edge should respond to changes in load. In the original model (Case 0), the fraction of requests varies with load, reflecting resource allocation that does not directly adapt to changing demand. In contrast, the modified models (Cases 1, 2, and 3) enforce a consistent fraction of requests served, thereby explicitly adapting resource allocation to load variations. Each approach is valid depending on the specific objectives and assumptions of the analysis.

5.9.3 Consequences of Interdependent Service Providers' Contributions

Introducing interdependent contributions significantly alters several key insights derived from the original model. As previously discussed, the original independent contribution scenario can now be viewed as a worst-case benchmark. Below we list specific claims from the analytical investigation of the originally proposed model (Section 3) that no longer hold under interdependent contributions:

- Optimal allocation cannot be calculated independently: Optimal resource allocations must now be determined using optimization algorithms designed to maximize the total grand coalition value, rather than computed individually per SP.
- Shapley value computation becomes significantly more complex: Instead of a straightforward individual calculation, evaluating the Shapley value now requires determining the value of each possible fecoalition. This approach exponentially increases computational complexity as the number of SPs grows. To mitigate this complexity, we adopt approximation techniques employing Monte Carlo methods, sampling random coalition formations, and caching results to avoid redundant computations.
- The core of the game is no longer a single point: It is now represented by a set of feasible payoffs. The values that SPs would receive under independent contributions establish the minimum payoff boundary. By definition, the Shapley value remains within the core.
- Computational complexity escalates: The original formulation, when evaluating all potential coalitions for the Shapley value, has a computational complexity of $\mathcal{O}(2^n)$, where n represents the number of SPs.

5.10 Generalization of the Shapley Value for any Amount of Network Owners

In this final section we explore what would happen if more than one NO is interested in taking part in the co-investment; specifically, we analyze how the Shapley value is calculated in this new scenario. As we saw in equation 3.8 of section 3.4, the Shapley value and consequently the payoff of the NO is half of the sum of each SP's net utility. This arises because the NO is a veto player, and the set of all SPs, when considered as a single super-player, is also a veto player. As stated in the referenced article:

> The intuition behind this equal sharing of the Shapley value between the NO and the SPs is based on each game is decomposed into a weighted sum of unanimity games in which the Shapley value assigns an equal share of a unit to each veto player." In our case if the set of SPs is considered as one super-player, it is actually a veto player as well, because the value function is zero if no SP is in the coalition, since it would not be possible to collect revenues from users utilization.

These results raise important questions regarding fairness. Although the formal definition of fairness in cooperative game theory is achieved through the Shapley Value, it may seem inequitable that a player who makes no economic investment, assumes no risk, and collects no revenue receives half of the grand coalition's total value. Moreover, the advantageous position of the NO is so attractive that additional NOs may be interested in participating in the coinvestment by providing the infrastructure location, which could substantially alter the distribution of the coalition's value. In this section, we calculate the Shapley value under two scenarios:

- Scenario 1: Multiple NOs can participate in the coinvestment simultaneously.
- Scenario 2: Many potential NOs exist, but only one NO is allowed per feasible coalition.

We show that for the SPs, from the point of view of the payoff, these scenarios are equal. In contrast, for the NOs, the allocation differs: in Scenario 1, all NOs should agree to impose their veto power, whereas in Scenario 2, the entire veto power is allocated to the single effective NO.

Let us consider n Network Owners named as $\mathrm{NO}_1, \mathrm{NO}_2, \ldots, \mathrm{NO}_n$ and m Service Providers named as $\mathrm{SP}_1, \mathrm{SP}_2, \ldots, \mathrm{SP}_m$. We assume independent contributions for SPs (each SP provides a fixed contribution c_i,) this assumption does not affect the NO Shapley values but simplifies the calculation for SPs.

In both scenarios, let

$$V = \{\mathrm{NO}_1, \ldots, \mathrm{NO}_n\} \quad \text{and} \quad I = \{\mathrm{SP}_1, \ldots, \mathrm{SP}_m\}$$

with the total set $N = V \cup I$. The characteristic function is defined as:

For Scenario 1 (feasible if at least one NO is present):

$$v(S) = \begin{cases} 0, & \text{if } S \cap V = \varnothing, \\ \displaystyle\sum_{\mathrm{SP}_i \in S} c_i, & \text{if } S \cap V \neq \varnothing \end{cases}$$

For Scenario 2 (feasible only if exactly one NO is present):

$$v(S) = \begin{cases} 0, & \text{if } |S \cap V| \neq 1, \\ \displaystyle\sum_{\mathrm{SP}_i \in S} c_i, & \text{if } |S \cap V| = 1 \end{cases}$$

The grand coalition's value is:

$$v(N) = \sum_{i=1}^{m} c_i$$

Calculation of the Shapley Value:

The Shapley value for a player i is given by

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big[v(S \cup \{i\}) - v(S) \Big]$$

For an independent Service Provider SP_i, its contribution is activated by the presence of at least one NO. Considering the ordering of SP_i with n NOs, the probability that SP_i appears after at least one NO is

$$1 - \frac{1}{n+1} = \frac{n}{n+1}$$

Hence,

$$\phi(\mathrm{SP}_i) = \frac{n}{n+1}\, c_i$$

For a Network Owner NO_j, the marginal contribution is nonzero only when it is the first NO to appear in the permutation. When a NO appears as the first veto, it activates the contributions of all SPs preceding it; for each SP, the probability of appearing before any NO is $\frac{1}{n+1}$. Therefore, the expected contribution activated is

$$\frac{1}{n+1} \sum_{i=1}^{m} c_i$$

Since the NOs are symmetric, the probability that a given NO is the first is $\frac{1}{n}$. Thus, in Scenario 1, the Shapley value for an NO is

$$\phi(\mathrm{NO}_j) = \frac{1}{n} \cdot \frac{1}{n+1} \sum_{i=1}^{m} c_i = \frac{1}{n(n+1)} \sum_{i=1}^{m} c_i, \quad j = 1, \ldots, n$$

In Scenario 2, only coalitions with exactly one NO are feasible (all other coalitions yield zero value). Consequently, in any permutation, the only effective NO receives the entire NO share, which is

$$\frac{1}{n+1} \sum_{i=1}^{m} c_i$$

Thus, if a coalition features a unique NO, that NO's Shapley value is

$$\phi(\mathrm{NO}) = \frac{1}{n+1} \sum_{i=1}^{m} c_i$$

Verification of Efficiency:

The efficiency property requires that

$$\sum_{j=1}^{n} \phi(\mathrm{NO}_j) + \sum_{i=1}^{m} \phi(\mathrm{SP}_i) = \sum_{i=1}^{m} c_i$$

Indeed, in Scenario 1,

$$\sum_{j=1}^{n} \phi(\mathrm{NO}_j) = n \cdot \frac{1}{n(n+1)} \sum_{i=1}^{m} c_i = \frac{1}{n+1} \sum_{i=1}^{m} c_i$$

and

$$\sum_{i=1}^{m} \phi(\mathrm{SP}_i) = \frac{n}{n+1} \sum_{i=1}^{m} c_i$$

Thus,

$$\frac{1}{n+1} \sum_{i=1}^{m} c_i + \frac{n}{n+1} \sum_{i=1}^{m} c_i = \sum_{i=1}^{m} c_i$$

In Scenario 2, since only one NO is effective in each coalition, the entire NO share, $\frac{1}{n+1} \sum_{i=1}^{m} c_i$, is allocated to that single NO, and the SPs' Shapley values remain unchanged.

Summary of Results:

$$\phi(\mathrm{SP}_i) = \frac{n}{n+1} c_i, \quad i = 1, \ldots, m$$

$$\phi(\mathrm{NO}_j) = \frac{1}{n(n+1)} \sum_{i=1}^{m} c_i, \quad j = 1, \ldots, n \quad \text{(Scenario 1)}$$

$$\phi(\mathrm{NO}) = \frac{1}{n+1} \sum_{i=1}^{m} c_i \quad \text{(Scenario 2)}$$

Thus, while the SPs' payoffs are identical in both scenarios, they substantially benefit from increasing the amount of NOs willing to take part in the coinvestment. Meanwhile, the interpretation for NOs differs. In Scenario 1, the veto power is shared among the n NOs, whereas in Scenario 2, the single active NO in a coalition obtains the entire NO share, making the coinvestment opportunity even more attractive for NOs.

In our model, the Shapley value for a Network Owner (NO) is given by $\phi(\mathrm{NO}_j) = \frac{1}{n(n+1)} \sum_{i=1}^{m} c_i$, where n is the number of NOs and m the number of Service Providers (SPs). This result remains unchanged even if the independent contribution property for SPs is not assumed because the NO's veto power, ensuring that any coalition without an NO has zero value, dominates its marginal contribution. Thus, while the SPs' payoffs may vary with different contribution structures, the NO's share is invariant and solely reflects its essential role as a veto player.

Chapter 6

Empirical Illustration and Results Analysis of the Proposed Modifications to the Edge-Computing Model

6.1 Summarizing and Comparing the Different Utility Functions

Before conducting the sensitivity analysis to compare our modified versions of the utility functions, we first summarize their main characteristics in Tables 6.1 and 6.2.

Table 6.1 provides an organized overview of each utility function, highlighting the optimal allocation (h^*), the conditions ensuring positive allocations, and the peak values of the diminishing-return parameters. This arrangement simplifies referencing each case and clearly illustrates which variables influence both the optimal allocation and the diminishing returns parameter at its peak. Note that for Cases 2 and 3, the parameter β does not appear in the expression for the diminishing-return parameter at peak. This absence occurs because, in these cases, β impacts the optimal allocation linearly, rather than through a nonlinear or logarithmic relationship.

Table 6.2 decomposes the utility at the optimal allocation ($U(h^*)$) into three components: (1) potential monetization, identical across all cases, (2) fraction of requests served at the edge, and (3) cost of allocation. For Cases 2 and 3, β similarly does not influence the fraction of requests served at the edge nor the allocation cost, consistent with its linear relationship with the optimal allocation. Consequently, service providers in these scenarios could strategically misreport the value of β without affecting their allocation, thus enabling them to maximize profits without immediate negative repercussions.

Case	$U(h)$	h^*	Positive Allocation Constraint	Diminishing Returns Parameter Peak
0	$\rho\big(1 - e^{-\mu\, h}\big) \;-\; d'\, h$	$\dfrac{1}{\mu}\, \ln\!\big(\frac{\rho\,\mu}{d'}\big)$	$\dfrac{\rho\,\mu}{d'} > 1$	$\mu_{\text{peak}} \;=\; e\,\dfrac{d'}{\rho}$
1	$\rho\big(1 - e^{-\frac{\xi}{l}h}\big) \;-\; d'\, h$	$\dfrac{l}{\xi}\, \ln\!\big(\frac{\beta\,\xi}{d'}\big)$	$\dfrac{\beta\,\xi}{d'} > 1$	$\xi_{\text{peak}} \;=\; e\,\dfrac{d'}{\beta}$
2	$\rho\Big(1 - e^{-\frac{d'}{\rho\,\lambda}\, h}\Big) \;-\; d'\, h$	$\dfrac{\rho\,\lambda}{d'}\,\big(-\ln\lambda\big)$	$0 < \lambda < 1$	$\lambda_{\text{peak}} \;=\; e^{-1}$
3	$\rho\Big(1 - e^{-\,h\,\frac{\xi}{\rho}}\Big) \;-\; d'\, h$	$\dfrac{\rho}{\xi}\, \ln\!\Big(\frac{\xi}{d'}\Big)$	$\xi > d'$	$\xi_{\text{peak}} \;=\; e\, d'$

Table 6.1: Summary of the modified utility formulations, their optimal allocations, feasibility constraints, and the peak value of each diminishing-return parameter.

Case	$U(h^*)$ breakdown		
	Potential Monetization	Fraction of Requests Served at the Edge	Cost of Allocation
0	ρ	$1 - \dfrac{d'}{\rho\,\mu}$	$\dfrac{d'}{\mu}\, \ln\!\Big(\dfrac{\rho}{d'}\Big)$
1	ρ	$1 - \dfrac{d'}{\beta\,\xi}$	$\dfrac{d'\,l}{\xi}\, \ln\!\Big(\dfrac{\beta\,\xi}{d'}\Big)$
2	ρ	$1 - \lambda$	$\rho\,\big(\lambda\, \ln\lambda\big)$
3	ρ	$1 - \dfrac{d'}{\xi}$	$\rho\,\Big(\frac{d'}{\xi}\, \ln\!\big(\frac{\xi}{d'}\big)\Big)$

Table 6.2: $U(h^*)$ breakdown: potencial monetización, fracción de requests served at the Edge and allocation cost.

6.2 Unification of All Cases by Calibration to Case 0

In this section, we demonstrate that, by selecting appropriate values for μ, ξ, and λ, we can calibrate these parameters so that any modified utility formulation collapses into another. Although we present the collapse into Case 0 for concreteness, the same reasoning applies to transforming any case into any other. This universality arises because each formulation retains the original structure; potential monetization multiplied by an exponential saturation term, minus the resource allocation cost. Consequently, adjusting μ, ξ, or λ simply rescales the exponential coefficient without altering the fundamental utility form, allowing complete equivalence among all cases under proper calibration.

Note that although we can collapse one into each other, once we fix the value of the diminishing returns parameter, they have different behaviors with respect to changes in their variables.

Collapsing Case 1 into Case 0

In Case 0, the exponent coefficient on h is μ. In Case 1, it is $\frac{\xi}{l}$. To match them, set

$$\frac{\xi}{l} \;=\; \mu \;\;\Longleftrightarrow\;\; \xi = \mu\, l$$

Since $\rho = \beta\, l$ may remain arbitrary, the utility functions become identical:

$$U_1(h)\big|_{\xi=\mu\, l} = \rho\Big(1 - e^{-\frac{\mu\, l}{l}\, h}\Big) - d'\, h = \rho\Big(1 - e^{-\mu\, h}\Big) - d'\, h = U_0(h)$$

Likewise, the optimal allocation collapses:

$$h_1^* = \frac{l}{\xi}\, \ln\Big(\frac{\beta\, \xi}{d'}\Big)\Big|_{\xi=\mu\, l} = \frac{l}{\mu\, l}\, \ln\Big(\frac{\beta\, (\mu\, l)}{d'}\Big) = \frac{1}{\mu}\, \ln\Big(\frac{\rho\, \mu}{d'}\Big) = h_0^*$$

Thus, with

$$xi = \mu\, l$$

Case 1 is equivalent to Case 0.

Collapsing Case 2 into Case 0

In Case 2, the exponent coefficient on h is $\frac{d'}{\rho\, \lambda}$. Equate this to μ:

$$\frac{d'}{\rho\, \lambda} \;=\; \mu \quad\Longleftrightarrow\quad \lambda = \frac{d'}{\rho\, \mu}$$

Substituting into $U_2(h)$ yields

$$U_2(h)\big|_{\lambda=\frac{d'}{\rho\, \mu}} = \rho\Big(1 - e^{-\frac{d'}{\rho\, \left(\frac{d'}{\rho\, \mu}\right)}\, h}\Big) - d'\, h = \rho\Big(1 - e^{-\mu\, h}\Big) - d'\, h = U_0(h).$$

The optimal allocation likewise matches:

$$h_2^* = \frac{\rho\, \lambda}{d'}\, \big(-\ln(\lambda)\big)\Big|_{\lambda=\frac{d'}{\rho\, \mu}} = \frac{\rho\, \left(\frac{d'}{\rho\, \mu}\right)}{d'}\, \big(-\ln(\tfrac{d'}{\rho\, \mu})\big) = \frac{1}{\mu}\, \ln\Big(\frac{\rho\, \mu}{d'}\Big) = h_0^*$$

Hence, with

$$\lambda = \frac{d'}{\rho\,\mu}$$

Case 2 reduces to Case 0.

Collapsing Case 3 into Case 0

In Case 3, the exponent coefficient on h is $\frac{\xi}{\rho}$. Set this equal to μ:

$$\frac{\xi}{\rho} \;=\; \mu \quad\Longleftrightarrow\quad \xi = \rho\,\mu$$

Then

$$U_3(h)\big|_{\xi=\rho\,\mu} = \rho\Big(1 - e^{-\,h\,\frac{\rho\,\mu}{\rho}}\Big) - d'\,h = \rho\Big(1 - e^{-\mu\,h}\Big) - d'\,h = U_0(h)$$

The optimal allocation also matches:

$$h_3^* = \frac{\rho}{\xi}\,\ln\!\Big(\frac{\xi}{d'}\Big)\Big|_{\xi=\rho\,\mu} = \frac{\rho}{\rho\,\mu}\,\ln\!\Big(\frac{\rho\,\mu}{d'}\Big) = \frac{1}{\mu}\,\ln\!\Big(\frac{\rho\,\mu}{d'}\Big) = h_0^*$$

Therefore, choosing

$$\xi = \rho\,\mu$$

collapses Case 3 into Case 0.

Summary of Calibrations to Case 0

- Case 1 $\rightarrow$ Case 0: $\xi = \mu\,l$.
- Case 2 $\rightarrow$ Case 0: $\lambda = \frac{d'}{\rho\,\mu}$.
- Case 3 $\rightarrow$ Case 0: $\xi = \rho\,\mu$.

In each instance, the choice of parameter values makes $U_i(h) = U_0(h)$ and $h_i^* = h_0^*$. Thus, all four formulations collapse into Case 0 under these calibrations.

6.3 Sensitivity Analysis and Cases Comparative

In this section, we perform a sensitivity analysis with respect to each parameter for the original utility formulation (Case 0) and its three proposed modifications (Cases 1, 2, and 3). For each utility function, we individually vary each parameter to analyze how these changes influence the optimal allocation (h^*) and the associated optimal utility ($U(h^*)$). By comparing results across cases, we identify common behaviors and differences, providing a characterization of the net utility function.

6.3.1 Sensitivity Analysis for the diminishing returnsParameter

The figure 6.1 illustrates the optimal allocation (h^*) as a function of the normalized diminishing returns parameter for each of the four utility function cases. To enable a consistent comparison, we normalize the diminishing returns parameter as a percentage relative to its respective peak value. For Cases 0, 1, and 3, the parameter ranges from the positive allocation constraint to the peak value, and then continues up to four times this peak. In contrast, Case 2 provides simpler handling, as its diminishing returns parameter has clearly defined upper and lower bounds.

Upon normalization, Cases 0, 1, and 3 exhibit identical behaviors, indicating a shared structural relationship between the diminishing returns parameter and optimal allocation. Conversely, Case 2 demonstrates a fundamentally different functional dependency, distinctly differentiating its allocation pattern from that of the other cases. More formally:

- Characterization of the diminishing returns parameter for Case 0, 1 and 3, (we represent them with ξ but it is equivalent for μ):
  - increasing from the positive allocation constraint to the peak
  - maximum at ξ_{peak}
  - decreasing for $\xi > \xi_{peak}$
  - $\lim_{\xi \to \infty} h^*(\xi) = 0$
- Characterization of the diminishing returns parameter for Case 2 λ:
  - $h^*(\lambda) = -\frac{\rho\,\lambda}{d'} \ln(\lambda)$
  - $h^*(\lambda) = 0$ at $\lambda = 0$ and $\lambda = 1$
  - increasing for $\lambda \in \left(0,\, \frac{1}{e}\right)$
  - maximum at $\lambda_{peak} = \frac{1}{e}$
  - decreasing for $\lambda \in \left(\frac{1}{e},\, 1\right)$

The figure 6.2 shows the utility at optimal allocation ($U(h^*)$) as a function of the normalized diminishing returns parameter for each case. After normalization, Cases 0, 1, and 3 show identical behaviors, indicating a consistent positive relationship between the diminishing returns parameter and the utility achieved. Conversely, Case 2 exhibits an inverse relationship, where the highest utility is obtained at the minimal value of the diminishing returns parameter. More formally:

- Characterization of the diminishing returns parameter for Case 0, 1, and 3:

$$U(h^*) \;=\; \rho \;-\; \frac{d'}{\alpha}\Big[1 + \ln\!\Big(\tfrac{\rho\,\alpha}{d'}\Big)\Big] \quad \text{with} \quad \alpha = \begin{cases} \mu & \text{(case 0)} \\ \frac{\beta\,\xi}{l} & \text{(case 1)} \\ \frac{\xi}{\rho} & \text{(case 3)} \end{cases}$$

  - Threshold of positivity: $U(h^*) = 0$ for $\alpha \le \frac{d'}{\rho}$
  - Supralinear growth just above $\alpha_0 = \frac{d'}{\rho}$ (convex region)
  - Inflection point at $\alpha = e\,\frac{d'}{\rho}$ where growth changes from convex to concave
  - Monotonic increase in α for $\alpha > \frac{d'}{\rho}$
  - Asymptotic bound: $\lim\limits_{\alpha \to \infty} U(h^*) = \rho$
- Characterization of the diminishing returns parameter for Case 2 λ:
  - Domain: interior solution for $0 < \lambda < 1$ (else $h^* = 0,\ U = 0$)

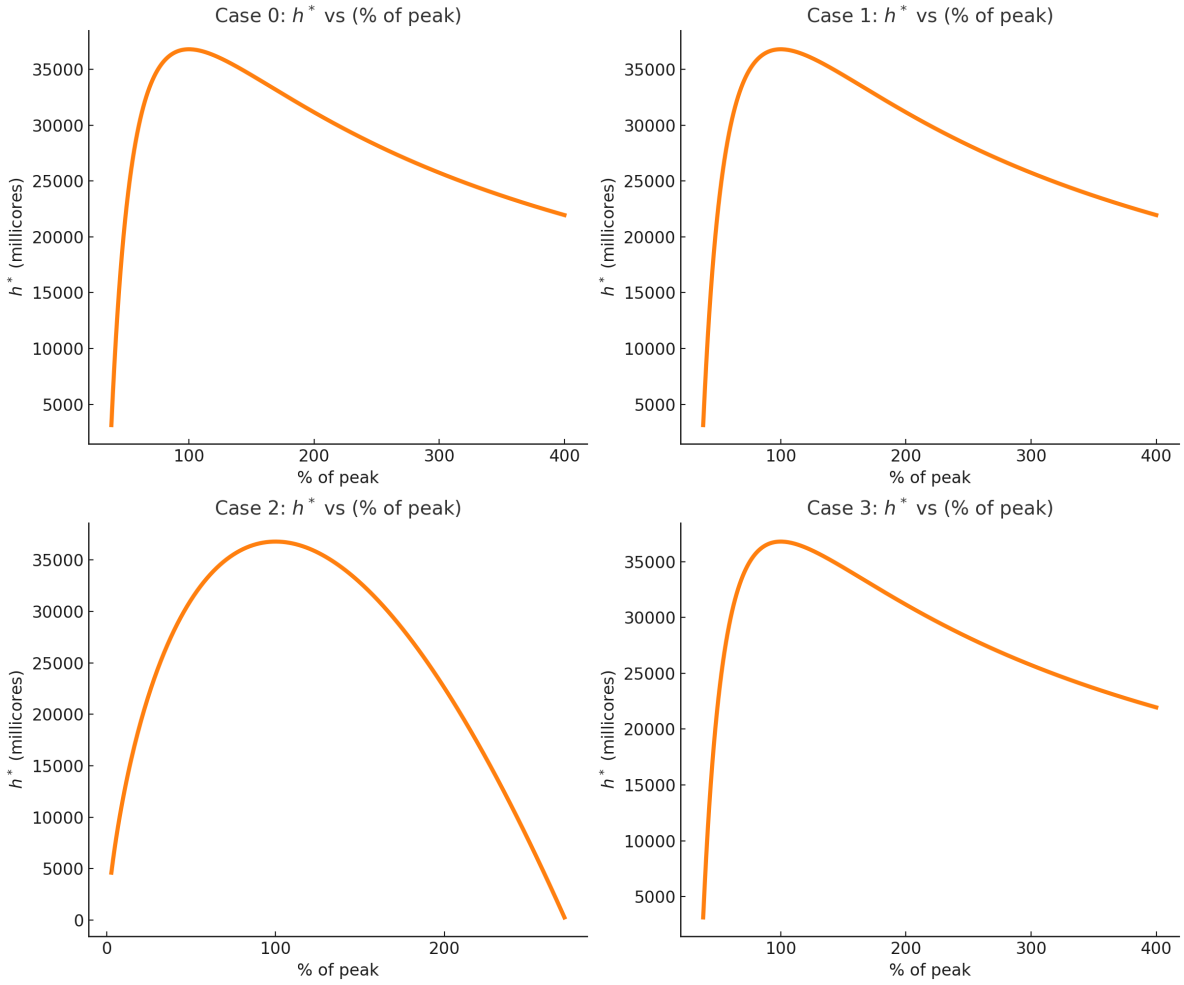


Figure 6.1: Comparative of diminishing returns parameters in the optimal allocation

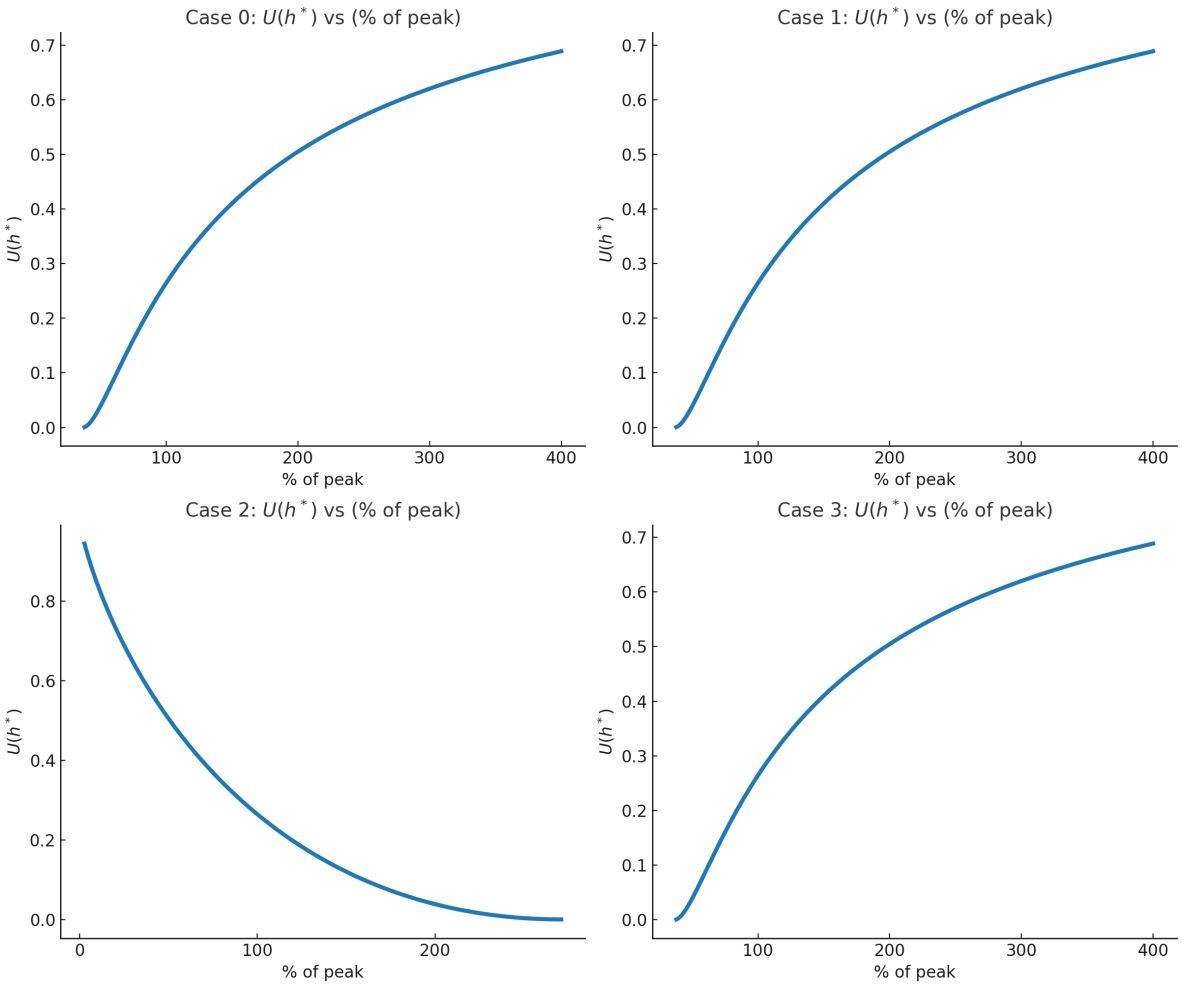


Figure 6.2: Comparative of diminishing returns parameters in the utility at optimal allocation

- Monotonicity: $U(h^*)$ decreases in λ on $(0, 1)$ since $\dfrac{dU}{d\lambda} = \rho \ln(\lambda) < 0$
- Convexity: $\dfrac{d^2U}{d\lambda^2} = \rho/\lambda > 0$
- Asymptotic behavior:
  - $U \to \rho$ as $\lambda \to 0^+$
  - $U = 0$ at $\lambda = 1$ (horizontal tangent)

The charts in figure 6.3 emphasize again the similarity among Cases 0, 1, and 3, highlighting that these cases collapse into identical behaviors once normalized. Conversely, Case 2 demonstrates the opposite relationship, attaining its maximum fraction of requests served at minimal values of the diminishing-return parameter.

More significantly, these charts illustrate a fundamental limitation present in all cases: the existence of a single parameter modeling both the fraction of total requests served at the edge and the linear cost component, specifically through the number of requests a single millicore can serve within a given time-slot duration. Due to this shared parameter, increasing the fraction of requests served at the edge inherently reduces the linear cost, thus restricting the range of achievable SP profiles. For instance, scenarios requiring both high fractions of requests served at the edge and high linear costs are unattainable within this framework.

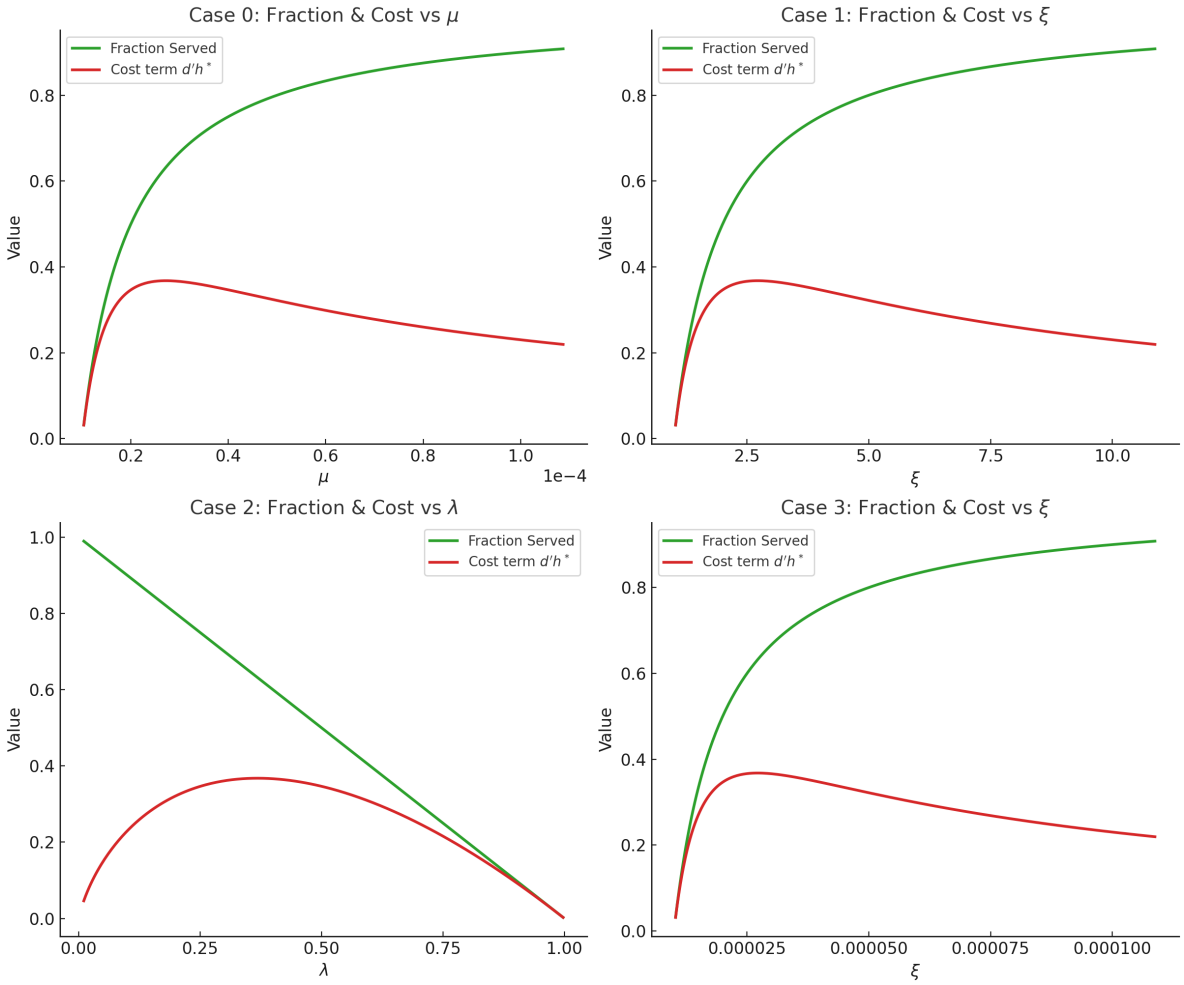


Figure 6.3: Influence of diminishing returns parameters in the fraction sedved at the Edge and the allocation cost

To overcome this limitation and allow greater flexibility in modeling diverse SPs profiles, it is needed to introduce separate parameters: one parameter to independently control the fraction of requests served at the edge, and another distinct parameter dedicated to defining the hardware requirement per request.

6.3.2 Sensitivity Analysis for the Load Parameter

The charts presented in figure 6.4 illustrate that Cases 1, 2, and 3 successfully achieve optimal allocations (h^*) that scale linearly with respect to the total number of requests. Conversely, Case 0 reveals a problematic sublinear scaling, characterized by a slowly increasing logarithmic function. This behavior poses limitations in adequately representing scenarios requiring proportional resource allocation growth.

The linear scaling observed in the modified versions (Cases 1, 2, and 3) is more desirable, as it enables straightforward and predictable resource allocation in direct proportion to the load. While one could potentially use other supralinear or sublinear functional forms, a linear or nearly-linear scaling is preferable, particularly avoiding logarithmic relationships.

The charts presented in figure 6.5 show the utility at optimal allocation ($U(h^*)$) relative to the number of requests. Once again, Cases 1, 2, and 3 exhibit identical, strictly linear behaviors, while Case 0

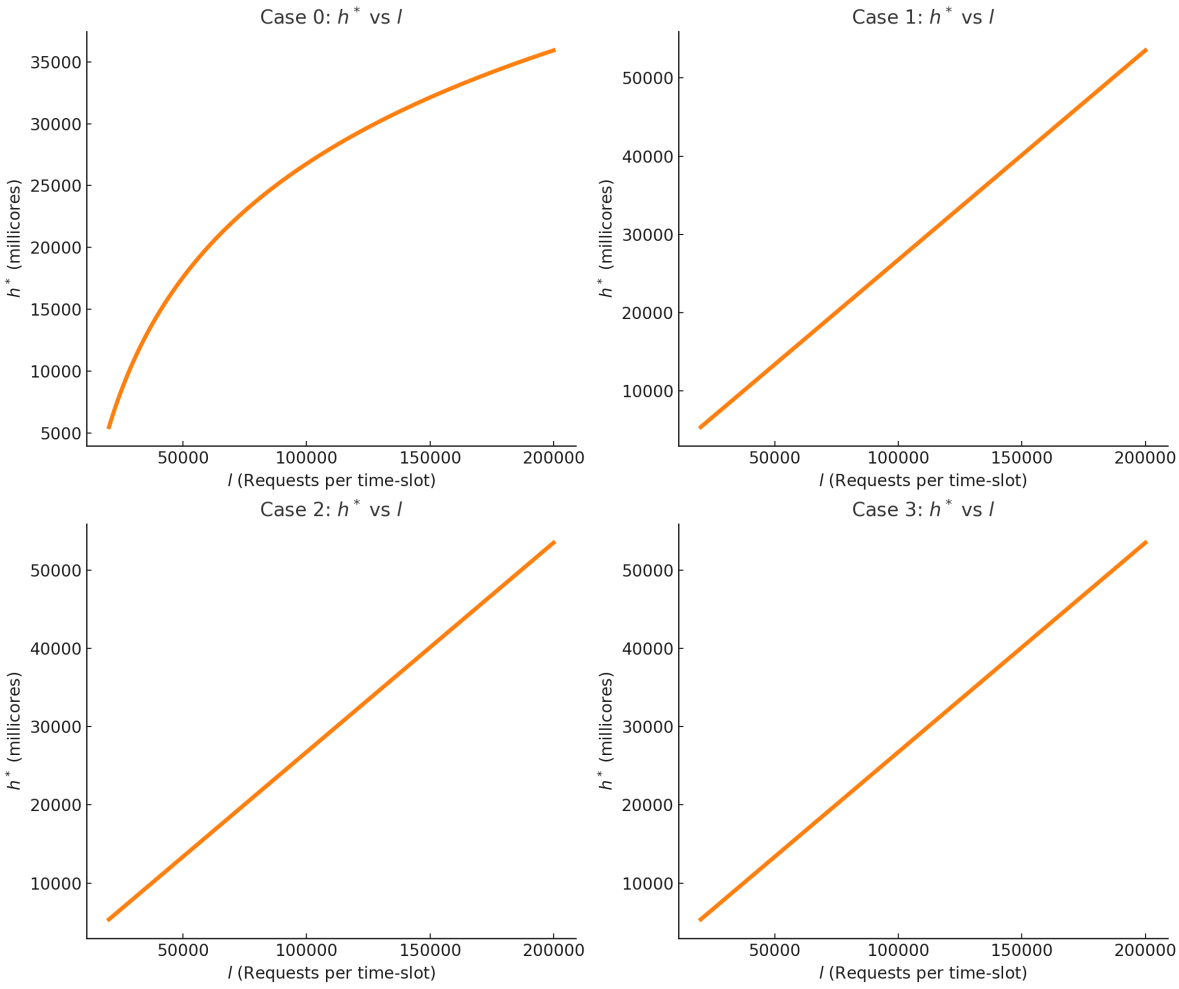


Figure 6.4: Comparative analysis of load effects on optimal allocation

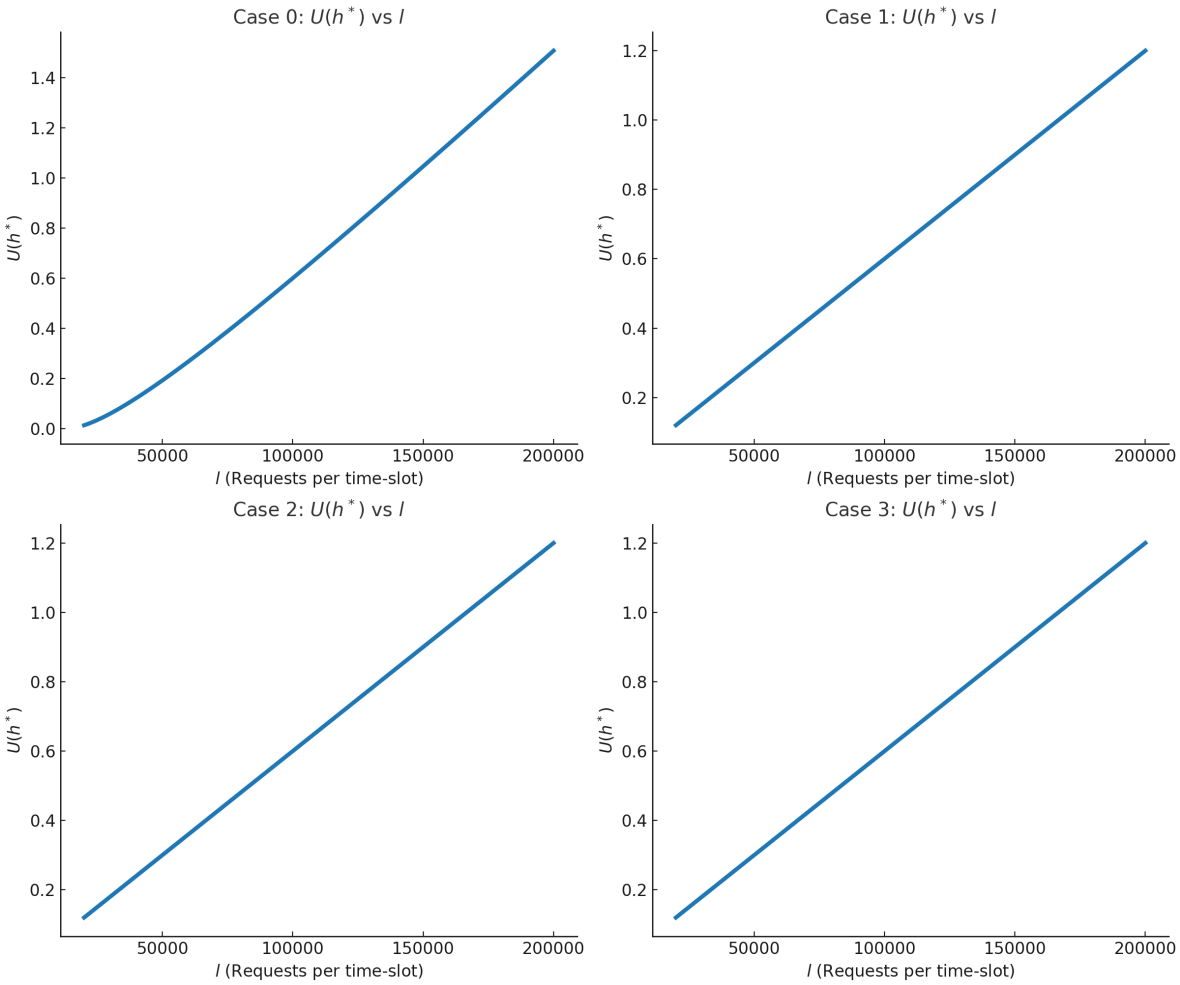


Figure 6.5: Comparative analysis of load effects on utility at optimal allocation

differs distinctly. Specifically, Case 0 demonstrates an asymptotically linear (sub-linear) growth of utility, consistent with its previously observed logarithmic scaling in allocation. Conversely, Cases 1, 2, and 3 maintain a precisely linear relationship.

Note that the achieved utility at optimal allocation $U(h*)$ reaches a higher value for Case 0; this is due to the sublinear influence of the load in allocation and, as a consequence, on the cost.

6.3.3 Sensitivity Analysis for the Benefit Factor Parameter

The charts presented in figure 6.6 illustrate distinct behaviors of optimal allocation (h^*) with respect to the benefit factor (β). Cases 0 and 1 display sublinear scaling, consistent with the logarithmic relationship previously observed between load and allocation in Case 0. In contrast, Cases 2 and 3 exhibit strictly linear behavior, indicating direct proportionality between the benefit factor and the optimal allocation. Although this linear relationship mirrors the one observed for load in Case 1, 2 and 3 there is no inherent requirement for the economic benefit from requests served at the edge to linearly determine optimal allocation; rather, it is sufficient, and economically intuitive, that this relationship is positive.

The charts in figure 6.7 for utility at the optimal allocation ($U(h^*)$) versus the benefit factor (β) demonstrate consistency with the previous observations regarding load. Specifically, Cases 0 and 1 display an asymptotic, sub-linear behavior reflecting the logarithmic relationship of the benefit factor

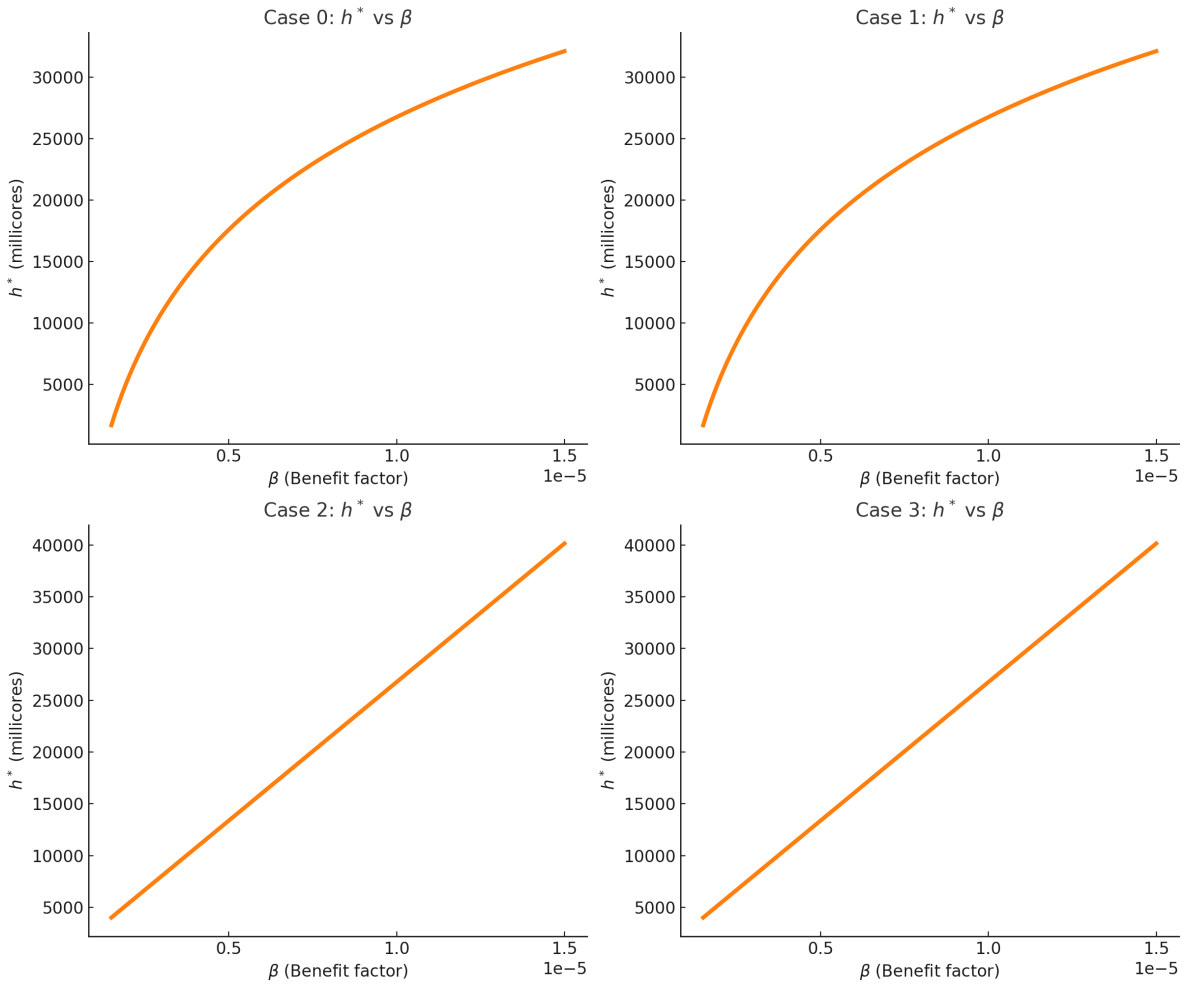


Figure 6.6: Comparative analysis of beta factor effects on optimal allocation

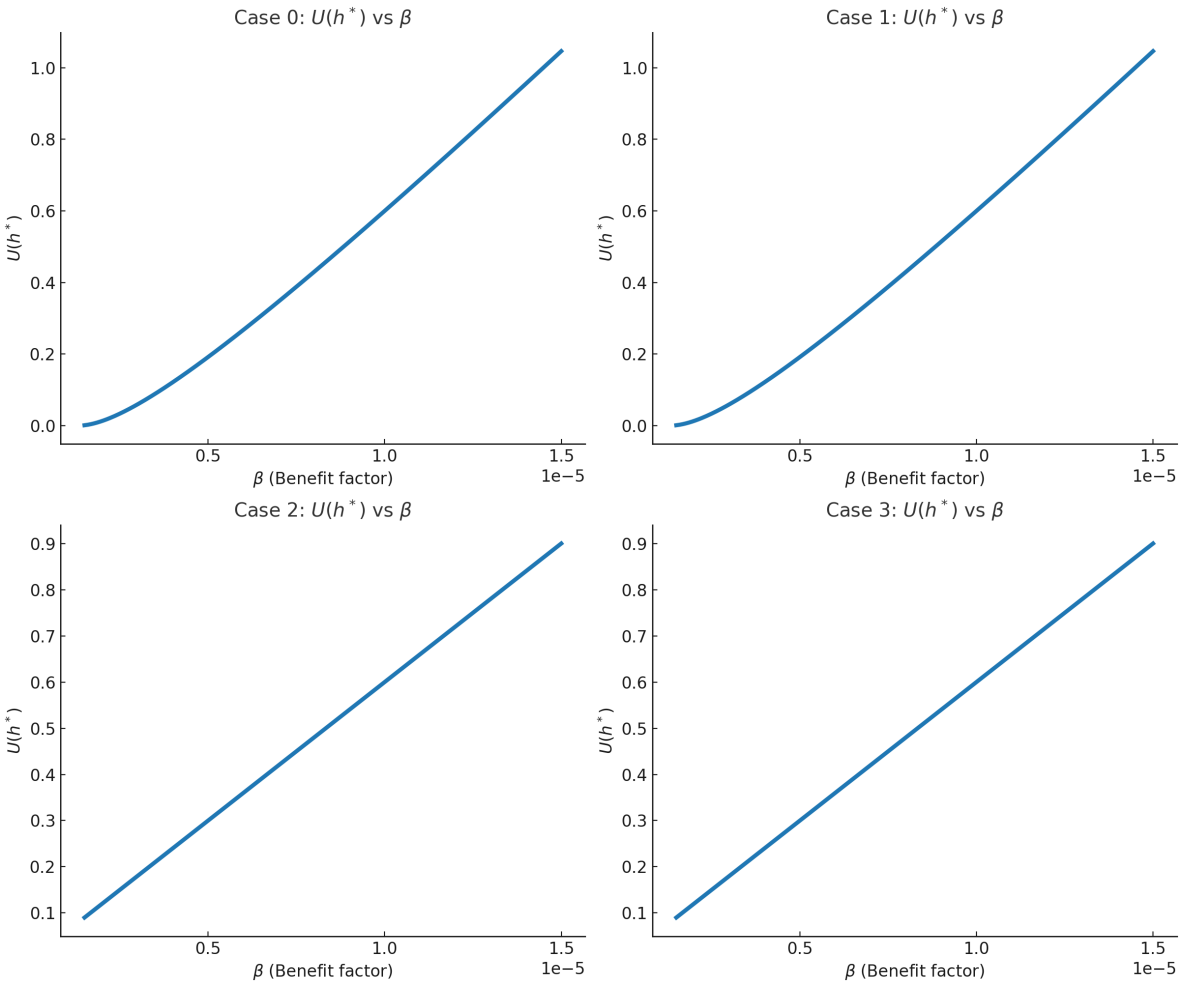


Figure 6.7: Comparative analysis of beta factor effects on utility at optimal allocation

with allocation, similar to what was previously observed in utility vs. load charts.

Conversely, Cases 2 and 3 exhibit completely linear relationships, due to the explicit linear influence of β in their allocation functions, translating directly into linear behavior in the optimal utility. This distinction arises precisely because, in Cases 2 and 3, the benefit factor β explicitly appears within the exponent of the exponential saturation term, thus yielding linear scaling in optimal allocation and, consequently, linear growth in the utility at the optimal allocation. Cases 0 and 1, lacking β in the exponent, inherently display sub-linear, asymptotic utility behavior.

Note that analogously to what we observed for the load in Case 0, here it is present in case 1 and 2. This is; the achieved utility at optimal allocation $U(h*)$ reaches a higher value, this is due to the sublinear influence of the β in allocation and as a consequence in the cost term.

6.3.4 Sensitivity Analysis for the Amortized Price Parameter

The charts presented in figure 6.8 show optimal allocation (h^*) versus the amortized per-millicore price (d') clearly illustrate identical behaviors across Cases 0, 1, and 3, all of which exhibit an inversely logarithmic relationship. Case 2, however, differs significantly, displaying an inversely linear relationship. This distinct behavior in Case 2 arises explicitly due to the presence of the amortized unit price d' within the exponent of the exponential saturation function, whereas in the other cases, d' is not included in this exponent, resulting in their characteristic logarithmic scaling.

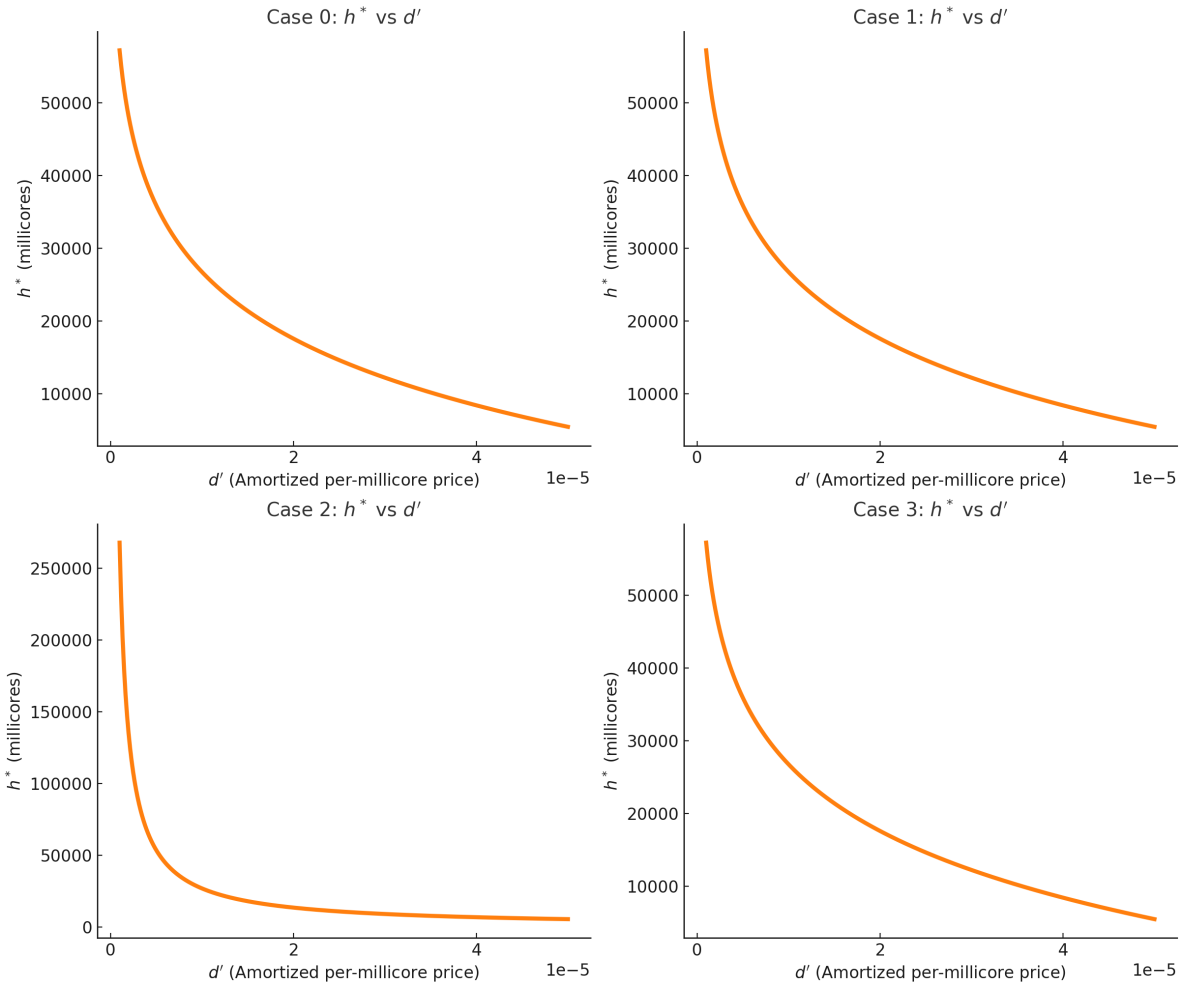


Figure 6.8: Comparative analysis of amortized price effects on optimal allocation

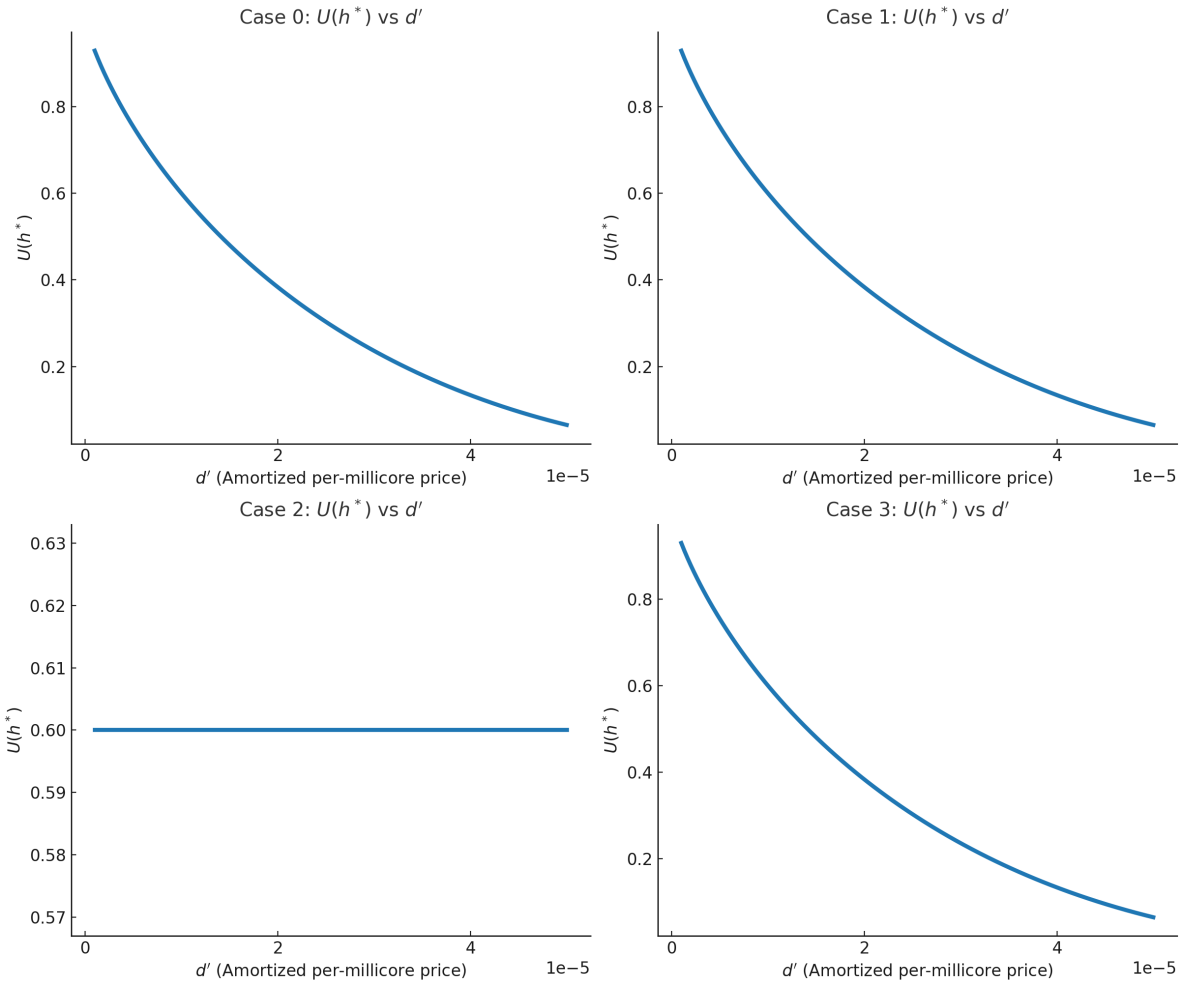


Figure 6.9: Comparative analysis of amortized price effects on utility at optimal allocation

The charts presented in figure 6.9 show utility at optimal allocation ($U(h^*)$) versus the amortized per-millicore price (d'). They reveal that Cases 1 and 3 retain the behavior observed in the original formulation (Case 0), showing a diminishing net utility as the price increases. In contrast, Case 2 distinctly demonstrates that the price parameter (d') does not influence the net utility at optimal allocation. Economically, this implies that in Case 2, the price solely determines the amount of resources purchased without affecting the overall net utility outcome. Such a scenario is plausible if the marginal benefit of resource allocation precisely offsets the incremental cost at every price level, resulting in a constant optimal net utility regardless of unit pricing variations.

To complete this comparative between the different net utility functions, we present table 6.3 where the influence of each term in the optimal allocation function is compared.

	Load (ℓ)	**Benefit (b)**	**Diminishing-return**	**Price (d')**
Case 0	Logarithmic	Logarithmic	Mixture (log and linear)	Inverse-logarithmic
Case 1	Linear	Logarithmic	Mixture (log and linear)	Inverse logarithmic
Case 2	Linear	Linear	Mixture (log and linear)	Linear
Case 3	Linear	Linear	Mixture (log and linear)	Inverse logarithmic

Table 6.3: Influence of each variable on the optimal allocation h^*.

To analyze how parameters influence the optimal net utility $U(h^*)$, we separate the utility equation into two different components. First, in table 6.4, we present the fraction of requests served at the edge, multiplied by the potential monetization factor ρ, corresponding to the gross utility. Then in table 6.5 we show the influence of these variables in the cost term, and finally in table 6.6 we show the combined effect of these variables.

Case	Benefit Term	Load	Benefit Factor	Diminishing Return	Price
0	$\rho\left(1-\frac{d'}{\rho\,\mu}\right)$	Linear	Linear	Inverse Linear	Inverse Linear
1	$\beta\,l\,(1-\frac{d'}{\beta\,\xi})$	Linear	Linear	Inverse Linear	Linear
2	$\rho\,(1-\lambda)$	Linear	Linear	Inverse Linear	–
3	$\rho\left(1-\frac{d'}{\xi}\right)$	Linear	Linear	Inverse Linear	Inverse Linear

Table 6.4: Influence of each parameter on the gross utility.

Case	Cost Term	Load	Benefit Factor	Diminishing Return	Price
Case 0	$\frac{d'}{\mu}\ln\left(\frac{\rho\,\mu}{d'}\right)$	Logarithmic	Logarithmic	Mixed (Inverse Linear + Logarithmic)	Mixed (Linear + Inverse Logarithmic)
Case 1	$\frac{d'\,l}{\xi}\ln\left(\frac{\beta\,\xi}{d'}\right)$	Linear	Logarithmic	Mixed (Inverse Linear + Logarithmic)	Mixed (Linear + Inverse Logarithmic)
Case 2	$\rho\,(\lambda\ln\lambda)$	Linear	Linear	Mixed (Linear + Logarithmic)	–
Case 3	$\rho\,\frac{d'}{\xi}\ln\left(\frac{\xi}{d'}\right)$	Linear	Linear	Mixed (Inverse Linear + Logarithmic)	Mixed (Linear + Inverse Logarithmic)

Table 6.5: Influence of each parameter on the cost.

Case	Load (l)	Beta (β)	Diminishing Return	Price (d')
Case 0	Asymptotic linear (sublinear→linear)	Asymptotic linear (sublinear→linear)	Direct (monotonic increase)	Mixed (steep drop→flat)
Case 1	Linear	Asymptotic linear (sublinear→linear)	Direct (monotonic increase)	Mixed (steep drop→flat)
Case 2	Linear	Linear	Inverse (monotonic decrease)	None (no effect)
Case 3	Linear	Linear	Direct (monotonic increase)	Mixed (steep drop→flat)

Table 6.6: Influence of each parameter on the optimal net utility.

6.4 Numerical Illustration of Misreporting the Benefit Factor

Since we have already established that, for Cases 2 and 3, an SP can retain all the utility, because β does not affect the fraction of requests served at the edge, and we have already made a numerical illustration of this misreporting for Case 0 in section 4.5 we limit our analysis to Case 1.

We examine two distinct service provider profiles. The first profile (SP1) features a high value of β, allowing it to retain a significant portion of the potential monetization. The second profile (SP2), by contrast, retains approximately 26% of the potential monetization, corresponding to the scenario where β results in $\xi = \xi_{\text{peak}}$. Typically, SP1 corresponds to real-time applications with latency constraints, while SP2 represents services less sensitive to response time. We expect that SP1 can profit more from misreporting than SP2. We define the same values for l, ξ and d' for both:

$$l = 10000, \qquad \xi = 60, \quad d' = 5 \times 10^{-7}$$

SP 1 :$\beta_{\text{real}} = 1 \times 10^{-6}$

The interior optimum solves

$$\frac{dU_{\text{SP}}}{d\beta_{\text{dec}}} = 0 \quad \Longrightarrow \quad \beta^*_{\text{dec}} = \frac{-\,d' + \sqrt{d'^2 + 8\,d'\,\beta_{\text{real}}\,\xi}}{2\,\xi} \;\approx\; 4.04 \times 10^{-7}$$

At truthful declaration, the retained utility is

$$U_{\text{SP}}(\beta_{\text{real}}) = \tfrac{1}{2}\Big[\,\beta_{\text{real}}\,l - \frac{l\,d'}{\xi} - \frac{l\,d'}{\xi}\,\ln\!\big(\tfrac{\beta_{\text{real}}\,\xi}{d'}\big)\Big] \approx 4.76 \times 10^{-3}$$

We evaluate U_{SP} at ten values of β_{dec}:

β_{dec}	h_{dec}	U_{dec}	U_{real}	U_{SP}	Benefit
1.0×10^{-6}	797.92	0.00952	0.00952	0.00476	0.0%
5.0×10^{-7}	682.39	0.00458	0.00949	0.00720	51.4%
2.0×10^{-7}	529.68	0.00165	0.00932	0.00849	78.5%
1.0×10^{-7}	414.15	0.00071	0.00896	0.00860	80.8%
6.0×10^{-8}	329.01	0.00035	0.00845	0.00827	73.8%
5.0×10^{-8}	298.63	0.00027	0.00818	0.00805	69.2%
4.0×10^{-8}	261.44	0.00019	0.00779	0.00769	61.7%
3.0×10^{-8}	213.49	0.00011	0.00712	0.00706	48.4%
2.0×10^{-8}	145.91	0.00004	0.00576	0.00574	20.6%
1.0×10^{-8}	30.39	0.00000	0.00165	0.00165	-65.3%

Table 6.7: Declared vs. real utilities and retained SP utility for Case 1a

The SP's retained utility U_{SP} rises to a maximum at $\beta_{\text{dec}} \approx 4.04 \times 10^{-7}$ and then declines as β_{dec} is reduced further.

SP 2 :$\beta_{\text{real}} = 2.26 \times 10^{-8}$

The interior optimum solves

$$\frac{dU_{\text{SP}}}{d\beta_{\text{dec}}} = 0 \quad \Longrightarrow \quad \beta^*_{\text{dec}} = \frac{-\,d' + \sqrt{d'^2 + 8\,d'\,\beta_{\text{real}}\,\xi}}{2\,\xi} \;\approx\; 1.57 \times 10^{-8}$$

At truthful declaration, the retained utility is

$$U_{\text{SP}}(\beta_{\text{real}}) = \tfrac{1}{2}\Big[\beta_{\text{real}}\,l\big(1 - e^{-\,\xi\,h^*/l}\big) - d'\,h^*\Big] \approx 2.97 \times 10^{-5}$$

We evaluate U_{SP} at several β_{dec}:

The SP's retained utility U_{SP} peaks at $\beta_{\text{dec}} \approx 1.57 \times 10^{-8}$ and then falls, becoming negative for sufficiently small β_{dec}.

β_{dec}	h_{dec}	U_{dec}	U_{real}	U_{SP}	Gain
2.26×10^{-8}	166.7	5.95×10^{-5}	5.95×10^{-5}	2.97×10^{-5}	0.0%
2.00×10^{-8}	145.8	4.39×10^{-5}	5.89×10^{-5}	3.70×10^{-5}	24.5%
1.60×10^{-8}	108.7	2.90×10^{-5}	5.72×10^{-5}	4.27×10^{-5}	43.9%
1.40×10^{-8}	86.5	3.10×10^{-5}	5.71×10^{-5}	4.16×10^{-5}	40.0%
1.00×10^{-8}	30.4	1.51×10^{-6}	2.26×10^{-5}	2.18×10^{-5}	-26.5%

Table 6.8: Comparison of declared vs. real utilities and retained SP utility for Case 1b

6.5 Studying the Effects of Interdependent Contribution

The effects of implementing sublinear per-millicore pricing (Section 5.9.1) are straightforward and independent of the characteristics of the service providers (SPs). In contrast, dynamic allocation introduces more intricate behaviors, which we examine in this section.

We begin our analysis by considering an artificial but illustrative, scenario. We assume two SPs with identical values for the diminishing return parameter, benefit factor, and average load, differing only by reversing the signs of the hyperparameters $\{a_k\}$ for one SP. This construction results in two perfectly complementary SPs. Next, we compute the diminishing return parameter for each case. This can be done either by applying the equations from Section 6.2, which unify the cases, or more simply by directly defining the diminishing return parameter as a fraction of the total load served at the edge. By adopting this approach, we compare Case 0 against Cases 1, 2, and 3.

We expect identical behavior for Cases 1, 2, and 3 under varying loads because, in these scenarios, the exponential term of the saturation function scales linearly with load. Conversely, Case 0 behaves differently, as the load does not explicitly scale the exponential term. Consequently, SPs do not maintain consistent behavior across all time slots in Case 0. Here, the diminishing return parameter defined as a fraction of load is valid only at the average load. This behaviour of Case 0 is consistent with the claim we did section 4.2 stating that two SPs are "of the same type" defining the same type as they serve the percentage of requests in the edge, or in other words the relative weight of their cost is the same. only when their diminishing return parameters have equal value and at the same time, for SP1 and SP2 $l^1\,\beta^1 = l^2\,\beta^2$.

This distinct characteristic in Case 0 produces two notable effects:

- When normalizing the allocation and load curves, they coincide only at the average load. For load levels above the average, allocations fall slightly below the load, while for loads below the average, allocations are slightly higher than the load. This effect is illustrated in Figure 6.10, where 66% of requests are served at the edge.
- When the target fraction of requests served at the edge is reduced, allocations may drop to zero for SPs during periods of low load. This occurs when the positive allocation constraint is violated, resulting in all resources being allocated to the complementary SP. This phenomenon appears mildly in Figure 6.11, with 40% of requests served at the edge, and more prominently in Figure 6.12, where the served requests drop to 10%. Note that contrary to the previous case here, when load drops from the average, allocation drops even more, and when load is higher than the average, allocation is even higher.

Formalizing these observations, Tables 6.1 and 6.2 clearly demonstrate that only in Case 0 do both the positive allocation constraint and the peak value μ_{peak} explicitly depend on the load. Consequently, the fraction of requests served at the edge also varies with load exclusively in this scenario. Thus, when the positive allocation constraint is violated in Case 0, allocation for the affected SP drops to zero, reallocating resources entirely to the other SP to maximize net utility.

Note that for Case 1, 2 and 3 as a consequence of the load not determining the fraction of requests served at the edge, the curves of load and allocation perfectly overlap each other regardless of the fraction of the total load served at the edge. This can be observed in figure 6.13 where 99% of the requests are served at the edge, and in figure 6.14 where only 10% of requests are served at the edge.

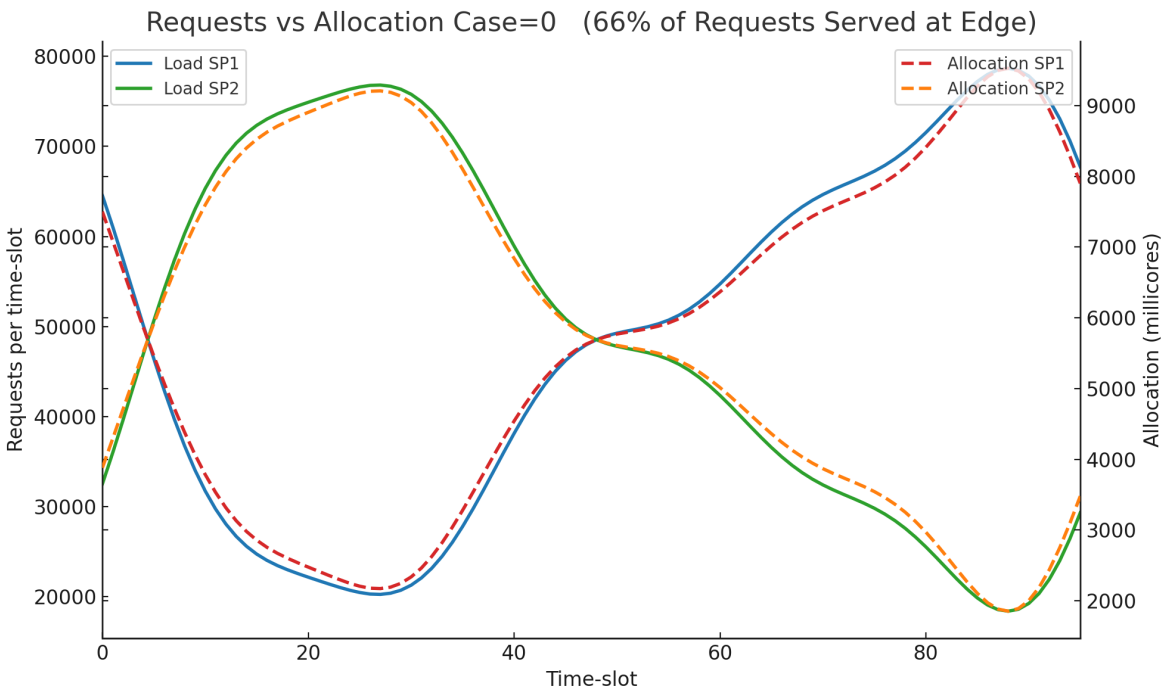


Figure 6.10: Load vs Allocation for Case 0, 66% of Requests Served at the Edge

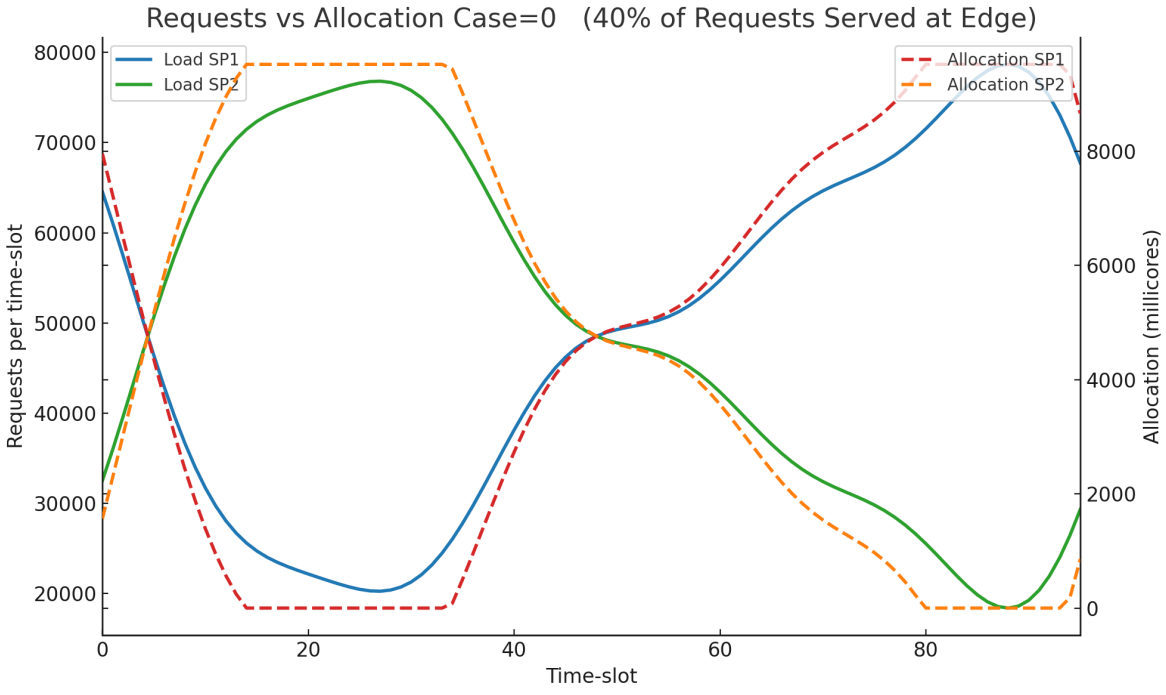


Figure 6.11: Load vs Allocation for Case 0, 40% of Requests Served at the Edge

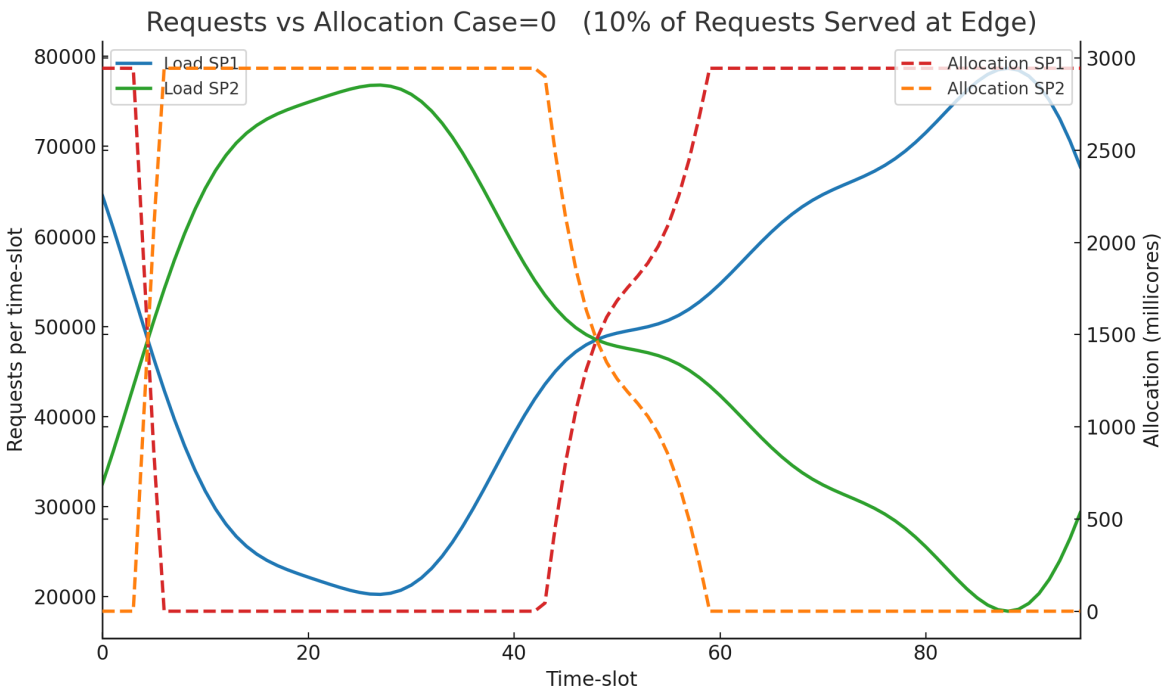


Figure 6.12: Load vs Allocation for Case 0, 10% of Requests Served at the Edge

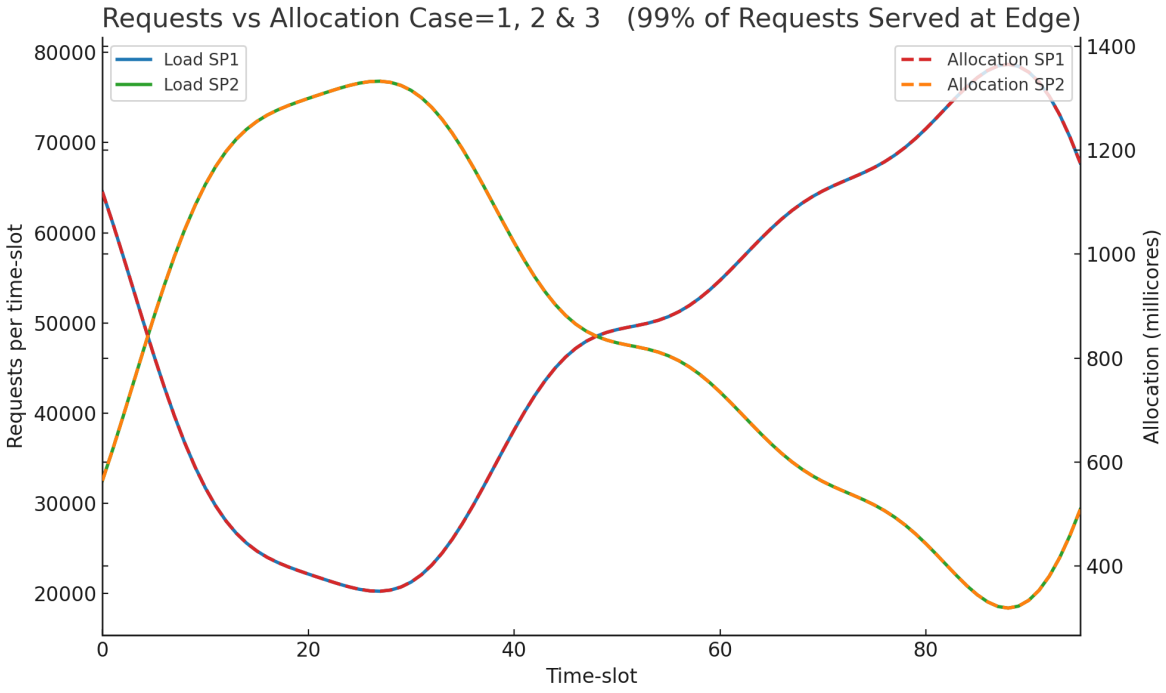


Figure 6.13: Load vs Allocation for Case 1, 2 and 3 with 99% of Requests Served at the Edge

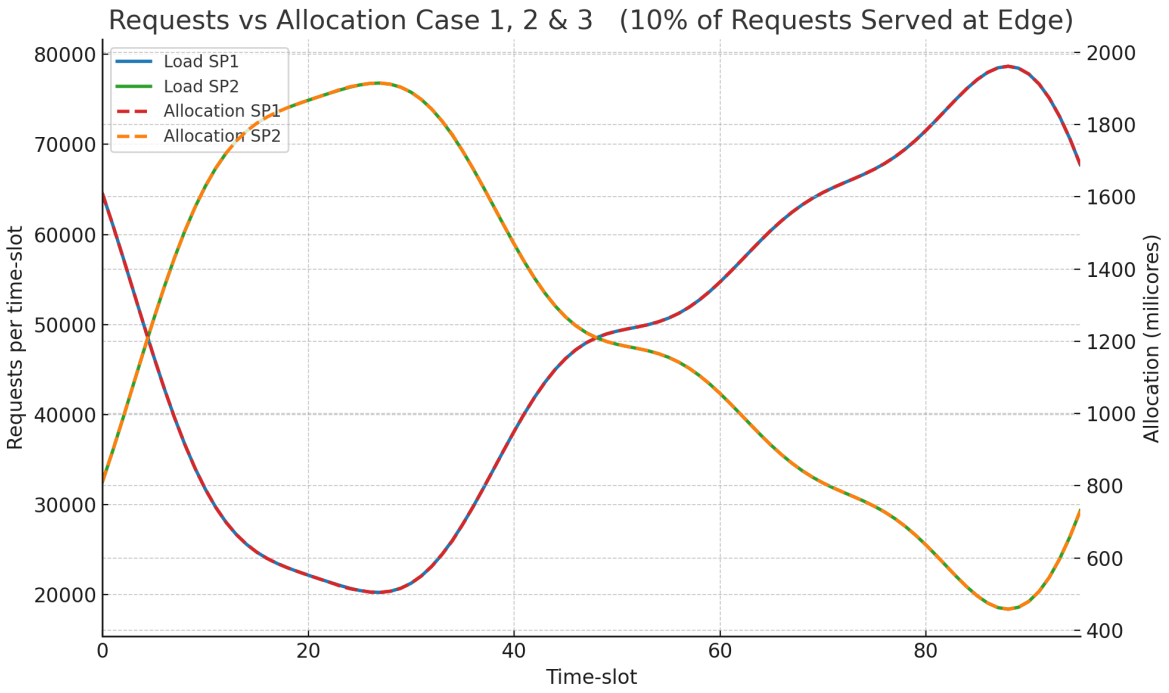


Figure 6.14: Load vs Allocation for Case 1, 2 and 3 with 10% of Requests Served at the Edge

To evaluate how synergic interdependency improves the SPs net utility, we compare their outcomes under independent contribution versus interdependent contribution. Our analysis focuses on Case 1, but results for Cases 2 and 3 are identical, and for Case 0 they are similar, except for previously mentioned differences. Specifically, load and allocation exactly overlap only at the average load, and when the positive allocation constraint is unmet, the entire allocation is reassigned to the other SP.

We present results for one SP, but given the symmetry of the scenario, outcomes for both SPs are identical. Table 6.9 presents gross utility, net utility, and costs for both scenarios. Additionally, we compute the absolute difference in net utility and normalize this difference by dividing it by the independent net utility and multiplying by 100, expressing the result as a percentage gain from interdependent cooperation. Based on these results, we provide the following observations:

- Gross utility depends exclusively on the percentage of requests, thus it remains identical in both interdependent and independent scenarios. Differences arise only due to cost sharing.
- Minor numerical differences in gross utility result from the independent scenario being computed through a deterministic algorithm, whereas the interdependent scenario involves maximization of net utility.
- At lower percentages of requests (e.g., 1% or 10%), the relative gain from perfect complementarity is substantial. As the percentage increases, the benefit diminishes significantly. At high utilization (95% or 99%), the additional net utility is reduced to around 6% and 3%, respectively, above the independent scenario.
- The above point illustrates the dual influence of the diminishing returns parameter. Increasing this parameter results in a higher percentage of requests being served at the edge but simultaneously reduces the relative weight of allocation cost, by reducing the amount of needed resources per request.
- We have presented a case of two perfectly complementary SPs and compared it with the independent contribution case. In practice, interdependent real-world scenarios will lie between these

two extremes. Perfect complementarity represents the best-case gain from cooperation, while independent contribution is the worst-case baseline. Other intermediate load curves will produce gains that fall somewhere in the middle.

Load (%)	Interdependent			Independent			Normalized Diff (%)
	Gross	Net	Cost	Gross	Net	Cost	Net
1%	62.81	24.18	38.63	62.94	0.32	62.63	7456.10
10%	637.02	264.79	372.23	636.28	32.80	603.48	707.29
20%	1292.40	582.00	710.40	1289.12	137.38	1151.74	323.64
30%	1961.43	952.28	1009.15	1961.10	325.00	1636.09	193.01
40%	2654.79	1393.53	1261.26	2655.42	610.58	2044.83	128.23
50%	3379.64	1947.16	1432.48	3376.19	1014.70	2361.49	91.93
60%	4128.97	2603.14	1525.83	4128.72	1567.26	2561.46	66.06
70%	4921.37	3362.92	1558.45	4919.97	2315.10	2604.87	45.27
80%	5759.40	4292.76	1466.64	5759.28	3340.61	2418.68	28.50
90%	6659.10	5481.77	1177.33	6659.81	4823.48	1836.33	13.64
95%	7139.71	6259.86	879.85	7139.68	5887.96	1251.73	6.32
99%	7544.70	7330.48	214.22	7544.65	7130.00	414.66	2.82

Table 6.9: Comparison of net utilities and normalized gain

Chapter 7

Developed Software Description

This section provides a description of the simulation software developed to investigate cooperative game-theoretical models for edge computing. The software is built using Python, employs a MySQL database for data management, uses .YAML configuration files to define simulation setups and facilitate sensitivity analysis. Additionally, Metabase is used for data visualization.

In this context, specific terminology is adopted to distinguish between individual simulations and groups of simulations clearly. The term "game" refers explicitly to the execution of a single instance of the simulation process, characterized by a specific parameter configuration. Conversely, the term "simulation" denotes a collection of multiple games that share the same set of service providers but differ in specific parameter values. Each simulation is uniquely identified by a simulation name, facilitating grouped management and organized analysis of related games.

Software Architecture Overview

The software structure is organized into six key components:

- Configuration Management
- Simulation Setup and Initialization
- Game Controllers
- Optimization Algorithms
- Load Function Generation
- Database Interaction

These components interact to execute simulations, manage data, and perform analyses.

Configuration Management

The Python configuration class ("config.py") defines technical parameters and specifies the type of simulation to execute. In this file, we do not set the numerical values of game parameters, but rather we configure the simulation framework itself. Specifically, this includes:

- Logging levels: DEBUG, INFO, ERROR.
- Database connection settings.

- Database maintenance operations: (Drop, Truncate, Create)
- Optional flags to save detailed simulation data (load and utility values). This is useful since the load function may include randomnes, making results non-deterministic.
- Simulation type (additive deterministic, non-additive deterministic, non-additive estimation). These simulation types are described in detail later.
- Allocation settings (per-time-slot or dynamic allocation), and CPU pricing strategies (variable or fixed).

Simulation Setup and Initialization

The YAML configuration files enable users to configure simulation runs. The possible parameters to configure are the following:

- Simulation identification: unique names to group related executions of simulations named as "games".
- Resource constraints: maximum possible amount of cores hosted at the edge.
- Investment duration: timeframes (years) to evaluate investment impacts.
- Daily granularity: number of time slots per day.
- Utility function selection: choice among predefined cases (0–3), new cases can be defined here.
- Pricing strategy: fixed or variable CPU prices.
- Service Provider specifications:
  - Name: It identifies an SP inside its corresponding simulation name, by defining multiple name for the same SP definition, it wil create multiple "different" SP with the exact same parameters values.
  - Load function:
    - Average load.
    - Hyperparameters for sinusoidal load function.
    - Variability (sigma).
  - Benefit factor.
  - diminishing returns parameter:
    - Directely configured through the variable xi (that also represents λ or μ).
    - Fractional service level (fraction of requests served at Edge).
  - Optimization parameters (latter described in more depth) :
  - `GTOL`: the gradient tolerance.
  - `FTOL`: the function tolerance.
  - `XTOL`: the step tolerance.
  - `JTOL`: the Jacobian tolerance.
  - `EPS`: the finite-difference step size.
  - `MAXITER`: the maximum number of iterations.

The YAML files allow defining parameters in three different ways: using a single value (e.g., `price_cpu: 0.5`), an explicit list of values (e.g., `price_cpu: [0.5, 0.6]`), or a range defined by an initial value, a final value, and the number of intermediate points (e.g., `price_cpu: ['0.5:1:5']`). The software automatically runs one game for each possible combination of these parameters. Users should carefully manage the configuration, as this approach can rapidly increase the total number of simulations.

Regarding the Utility function selection, the file allows us to modify or even create new utility functions by defining the exponents (i, j, k, m) of the parameters in the auxiliary function $f(l, \beta, \xi, d') = l^i \, \beta^j \, \xi^k \, d'^m$ such that the utility function for a time-slot is:

$$l\,\beta(1 - e^{-h\,f(l,\beta,\xi,d')}) - d'\,h$$

As stated in Section 5.9.2, the fraction of requests served at the edge remains invariant with respect to the load for Cases 1, 2, and 3. In contrast, for Case 0, when the diminishing returns parameter is defined through a fraction of requests, this fraction is valid only at the average load, from which the constant value of ξ is derived.

YAML Configuration example

```
// General properties of the game
# Only one value accepted
# Identifies the set of games defined in this file
simulation_name : Sim_127

# All the following properties in this file have 3 different possible assignment configurations:
# Immediate number, ex: price_cpu: 0.5
# Explicit list of values, ex: price_cpu: [0.5, 0.6]
# Implicit list of values with the format [Initial_val:Final_val:amount_of_values], e.g. cpu_price: ['0.5:1:5']

max_cores_hosted: [44000] # Max amount of cores hosted at the edge

years: [3,4]            # Investment period (years), can take decimals
daily_timeslots: [24,96,192]   # Time slots per day

# Utility function redefinition:
# The four numbers correspond to the exponent of the parameters in the auxiliary function f(l,\beta,\xi,d'), new cases can be defined here

cases:
  0: [0, 0, 1, 0]
  1: [-1, 0, 1, 0]
  2: [-1, -1, -1, 1]
  3: [-1, -1, 1, 0]

selected_case: 0 #This corresponds to the original utility function

# In the config.py it is defined if it take the fixed or variable cpu price
prices:
  when_fixed:
    cpu_price: [0.05:0.5:10] # Used if 'variable_cpu_price': False, if not it is ignored
  when_variable:
    # Following 3 work together and need equal amount of elements
    min_cores_hosted: [0] # Min amount of cores hosted at the Edge, this may make small coalition to have 0 value
    min_cpu_price:   [0.1]  # Price per unit when allocating max_cores_hosted resources
    max_cpu_price:   [0.5]  # Price per unit when allocating min_cores_hosted

service_providers:
  - service_provider_name: ['SP1']
    sigma: [0] # standard deviation for calculating the load, 0 will make the load function deterministic
```

```
    avg_load: [48530] # average load
     # hyperparameters: used to define the load function, a_k determining the
    amplitude (requests) and t_k the offset (time in seconds)
    a_k:      - [25313, -8832, 1757, -2873]
    t_k:      - [47340, 49080, 44520, 44880]
    benefit_factor: [1.5e-06,3e-06,6e-06]
    frac_of_requests: [0.99]  #alternatively we can define the value of the
    diminishing returns parameter e.g. xi: [0.08]

- service_provider_name: ['SP2','SP3','SP4']
  sigma: [0]
  avg_load: [48530]
  a_k:      - [-25313, 8832, -1757, 2873]
  t_k:      - [47340, 49080, 44520, 44880]
  benefit_factor: [3.5-06]
  frac_of_requests: [0.5,0.6,0.7,0.8]
```

In the example above, each simulation named as "game" will have four SPs, one named as SP1, and three named as SP2, SP3, and SP4, with the same characteristics.

Given the following parameters for which we declared multiple values:

daily_timeslots: [24, 96, 192] (defining at least 3 simulations) years: [3, 4] (defining at least 2 simulations) cpu_price: [0.05:0.5:10] (defining at least 10 simulations) (suppose that the price is set as fixed) for SP1: benefit_factor: [1.5e-06, 3e-06, 6e-06] (defining at least 3 simulations) for SP2, SP3 and SP4: frac_of_requests: [0.5, 0.6, 0.7, 0.8] (defining at least 4 simulations)

This will result in $3 \cdot 2 \cdot 10 \cdot 3 \cdot 4 = 720$, simulations, with 4 SPs each one. We can observe that this mechanism facilitates the sensitivity analysis, but if we are not careful, the total amount of simulations can grow extremely fast. Also note that while under the independent contribution, each simulation is executed in less than a second; when the contribution is interdependent, particularly in the case of dynamic allocation, running 720 simulations with 4 SPs each one can take a few minutes even if we relax the epsilon and tolerance values of the Sequential Least Squares Programming maximization method.

GameBuilderController Class

The GameBuilderController interprets YAML files, initializes simulations and individual games. Each game generated is uniquely identifiable by the Simulation it belongs to and all the parameter values. A group of games sharing the same simulation name can be updated: when executing a simulation with an existing name, the software adds new cases if parameters differ, and overrides existing results if parameters match exactly. The number and names of service providers must match exactly; otherwise, the software raises an error indicating an incompatible simulation name.

Game Controllers

Deterministic Additive Value Game Controller

This controller addresses scenarios with independent contributions from SPs, significantly simplifying calculations. It implements the pseudocode given in Section 3.10.2, eliminating the need for numerical optimization and greatly enhancing computational speed.

Estimation Non-Additive Solver Game Controller

This controller manages scenarios in which service provider contributions are interdependent, especially when the number of service providers is large or dynamic allocation is enabled. Instead of calculating values for all possible coalitions explicitly, it calculates only the grand coalition value to determine resource allocations. The Shapley values are then approximated by sampling random coalitions, making

computations feasible in a reasonable amount of time. It also incorporates caching mechanisms to prevent redundant calculations, balancing computational efficiency with analytical accuracy.

Deterministic Non-Additive Solver Game Controller

This controller manages scenarios where service provider contributions are interdependent, but the number of service providers is relatively small or dynamic allocation is not enabled, keeping computational complexity manageable. Under these conditions, the controller explicitly calculates the value for each possible coalition to precisely determine the Shapley value.

Optimization Algorithms

The `OptimizationController` class is crucial for handling complex optimization scenarios. This class numerically maximizes coalition utilities, particularly under dynamic, per-time-slot resource allocations and variable CPU pricing strategies. Key functionalities include:

- Computing utility functions based on the selected net utility function case and input parameters.
- Employing the `trust-constr` optimization method from SciPy, which is particularly well-suited for constrained, smooth, nonlinear optimization problems.
- Explicitly computing and supplying the Jacobian (first-order derivative) and Hessian (second-order derivative) matrices to improve numerical precision and convergence efficiency.
- Enforcing feasibility constraints to ensure all resource allocations stay within the bounds of available resources.
- Reading optimization parameters from the configuration file, including:
  - `GTOL`: the gradient tolerance, which defines the stopping criterion based on the projected gradient norm.
  - `XTOL`: the step tolerance, which controls convergence by checking the magnitude of relative changes in the decision variables.
  - `BARRIER_TOL`: the barrier tolerance, controlling the accuracy in constraint satisfaction and convergence of barrier subproblems.
  - `JTOL`: the Jacobian tolerance, applied to constraints, ensuring that constraint violations remain within acceptable bounds.
  - `MAXITER`: the maximum number of iterations allowed during the optimization process to prevent infinite loops or excessive computation time.

The following table summarizes how these parameters influence the speed and precision of the optimization process:

Load Function Generation

The `LoadFunctionController` generates the load profiles for each SP, utilizing sinusoidal expansions that mimic typical daily fluctuations in demand. It can introduce controlled randomness through a configurable sigma parameter, enabling the exploration of load uncertainty impacts. This functionality provides support for sensitivity analyses under varying load conditions, to better model real-world operational scenarios.

Parameter	Description	Effect on Speed	Effect on Precision
`GTOL`	Gradient-norm threshold for convergence	Lower values slow convergence	Improves accuracy near optimum
`XTOL`	Tolerance on variable changes	Reduces early stopping	Helps locate stable solutions
`BARRIER_TOL`	Tolerance for constraint barrier subproblems	Stricter feasibility slows convergence	Enhances constraint satisfaction
`MAXITER`	Cap on number of iterations	Slows if set very high	Avoids premature termination

Table 7.1: Optimization parameters and their influence on performance.

Simulation Execution

The `SimulationController` oversees the complete simulation execution, orchestrating three primary processes:

- Maximization of the coalition's total payoff using optimization algorithms.
- Calculation of individual contributions (Shapley values) for each SP and the NO; this implies calculating the payoff maximization for each feasible coalition.
- Determination of financial outcomes, explicitly calculating payments for each participant based on their Shapley value.

Additionally, it logs execution details, including computational resource usage and timing information, aiding performance analysis and scalability evaluations.

Database Interaction

The `DAOController` class manages interactions with a MySQL database, storing and updating simulation results. Its functionalities include:

- Storing simulation inputs (parameters, configurations).
- Managing updates and insertions of game data, merging results for the same simulation name and same SPs.
- Saving generated load functions, utility values, when this option is selected in the configuration file.

Main Execution

The main execution script (`main.py`) serves as the software's central operational entry point. It systematically performs the following actions:

- Initializes the logging environment and database connections as specified in the Python configuration class.
- Sequentially reads YAML configuration files, generating and initializing simulation scenarios using the GameBuilderController.

- Executes each simulation, managing individual game processes through selected Game Controllers and Optimization modules.
- Logs resource utilization and execution timing information, enhancing transparency and aiding resource management.

Interactive Load Function Editor

The script `create_load_function.py` provides a browser-based tool to define any daily load pattern visually and extract the corresponding sinusoidal hyperparameters and average load. Its main features are:

- A parametric model of the load curve as the sum of K sinusoids.
- An interactive plot (using "Bokeh" library) spanning 0 to 24 hours, where the user can drag sample points up or down.
- On each `Submit & Fit Curve` action, a nonlinear least squares fit (SciPy `curve_fit`) recomputes. $\{a_k\}$, $\{t_k\}$ and avg_load
- A "copy to clipboard" button that outputs the result in a YAML format to directly paste it into the setup file.

This tool makes it simple to generate different daily demand patterns without manual computation of phase shifts or amplitudes. To execute it just run the file named `create_load_function.py` inside the utils folder.

Conclusion

The presented simulation software provides a framework to analyze cooperative resource allocation models in edge computing environments. It integrates optimization algorithms, configurable load modeling, and systematic data management, supporting economic evaluations and strategic analyses of cooperative scenarios.

Chapter 8

Conclusions and Future Work

In this final chapter, we synthesize the results obtained throughout the thesis and reflect on the extent to which our original goals were met. We also outline possible directions for future work based on the insights gained.

This document focused primarily on developing a deep understanding of the original model proposed in the referenced academic article. Rather than immediately pursuing enhancements or extensions, we first chose to validate and dissect the model's structure, simplify its formulation where possible, and evaluate its behavior under different conditions. This approach allowed us to identify key limitations, interpret the economic meaning of its parameters, and explore both the computational and strategic implications of its assumptions.

While there are numerous opportunities to expand the model or incorporate additional complexities, we deliberately limited the scope of this thesis to avoid superficial treatments. The contributions made here aim to establish a solid analytical foundation that clarifies the model's internal mechanics. With this understanding now in place, we are better positioned to pursue meaningful extensions in future research, including the formal integration of alternative utility functions, dynamic resource allocation mechanisms, and mechanisms to improve incentive compatibility.

The sections that follow present a structured overview of the main contributions and propose a concrete path for future development.

8.1 Conclusions

Analytical Simplifications and Efficiency Gains

We identified several simplifications of the original model, significantly reducing analytical complexity and computational requirements. These simplifications enabled simulations to run in linear rather than exponential time, greatly enhancing practical feasibility. They also allowed for a more comprehensive and straightforward analysis of the model's core equations.

Identification of Constraints and Limitations

Constraints such as the requirement for positive allocation, along with limitations of the original model, were identified, especially conditions under which its assumptions did not hold or resulted in unrealistic outcomes. To overcome these limitations, we proposed alternative utility function formulations and systematically compared their implications for resource allocation and incentive compatibility.

Insights into Model Variables and Parameters

We gained insights into the internal dynamics of the model, particularly regarding the role and interpretation of each variable. The diminishing return parameter was notably clarified, and we showed that it can be related to measurable real-world parameters, helping to explain its influence on optimal resource allocation and utility.

Risk Analysis and Strategic Implications

We conducted a risk analysis focusing on the consequences of misestimating key parameters, providing equations that allow service providers to evaluate potential losses and support more informed strategic decisions. We found that overstating the load or benefit factor β consistently results in greater risk than understating them. This asymmetry aligns with the lack of incentive compatibility identified in the model, where underdeclaring β is a dominant strategy for SPs seeking to maximize their retained utility.

Alternative Utility Function Formulations

We proposed and analytically investigated three alternative utility function formulations, each addressing specific limitations of the original model. These new formulations allowed us to explore different economic interpretations of resource allocation and saturation effects, and provided more flexible representations of SP behavior. We compared their mathematical properties, strategic implications, and practical consequences, highlighting the trade-offs involved in selecting each version.

Strategy-Proofness Across Different Formulations

We demonstrated that the lack of strategy-proofness in the model is limited exclusively to the benefit factor β, since all other variables are either directly measurable or exogenous to the service provider. This finding holds true across all the utility function formulations analyzed: the original model and the three proposed alternatives. Furthermore, we showed analytically that the absence of strategy-proofness is not specific to the exponential saturation equation used initially but rather applies to any possible saturation function employed to model diminishing returns. In particular, we found that for the original and the first proposed utility functions, the gains from strategic under-declaration can be effectively bounded. In contrast, under the second and third utility function formulations, the retained utility obtained by misreporting β cannot be similarly constrained. However, if we adopt an alternative interpretation of strategy-proofness, specifically, preventing SPs from proportionally increasing their allocation while simultaneously declaring a lower net utility, Case 2 and Case 3 utility function formulations satisfy this second criterion of strategy-proofness.

Proposed Mechanisms to Enhance Strategy-Proofness

To address this lack of strategy-proofness, we proposed mechanisms such as a secondary market where service providers can trade their allocation. In this setup, service providers are required to sell allocation based on their declared net utility, while buying remains optional. This arrangement encourages truthful declarations, as underdeclaring utility could result in being forced to sell allocation at a declared value lower than its actual worth. We also suggested interpreting the benefit factor β as a priority indicator linked to response time. In this context, the network owner can employ internet slicing to offer differentiated service levels, associating higher declared values of β with faster and more reliable responses. These adjustments introduce a strategic trade-off that discourages misreporting by making honest declarations more attractive.

Enhanced Realism Through Dynamic Allocation

By introducing additional complexity into the model, particularly allowing service providers to have varying allocations across different time-slots, we achieved a more realistic representation of edge computing scenarios. Although these enhancements invalidated the initial simplifications, the simplified model remains valuable as a clearly defined worst-case scenario for comparative analysis. In this context, it represents a conservative baseline for SPs, meaning their net utility under the more complex models will always be equal to or greater than what they would obtain under the original version.

Simulation Framework and Computational Scalability

We developed a software tool to support extensive simulation analyses, including scenario configuration, execution, data storage, and visualization. This tool facilitated the comparison of various model formulations and provided deeper insight into their behaviors. The framework supports both independent and interdependent contribution models, enabling simulations under dynamic allocation and variable pricing strategies. When interdependencies between players are present, particularly for the calculation of the Shapley value, simulation runtimes can significantly increase. To address this computational challenge, we employed techniques such as Monte Carlo sampling and caching, which substantially reduce the required computational time, allowing simulations to scale effectively to larger numbers of players. Additionally, by adjusting the optimization parameters directly in the configuration files, users can explicitly manage the trade-off between computational precision and execution speed, choosing parameter values that best match their analytical priorities and computational resources.

Integration of Ideas from Related Literature

We analyzed the set of academic articles provided with the thesis and summarized their main findings to identify which ideas could be integrated into our model. Among these, we highlight the concept of a secondary market where service providers can optionally buy or sell allocation. Unlike the previously described scenario, in which service providers are forced to sell based on their declared utility, here trading is voluntary. Service providers can freely sell portions of their allocation for any subset of time slots and durations, thereby providing flexibility to adapt to changing needs or to address parameter misestimations effectively. Additionally, the PyMarket library, presented in one of the reviewed articles, provides a simple yet powerful Python-based toolkit designed specifically for the prototyping, implementation, and evaluation of auction-based market mechanisms. PyMarket supports various market designs, including multi-unit double auctions (MUDA) and peer-to-peer trading, both particularly suitable for facilitating such secondary market exchanges. Its compatibility with Python ensures seamless integration with our existing simulation framework, significantly simplifying the development and testing of these auction mechanisms. Furthermore, we adopted the concept of ramp constraints from these articles, which must be considered when dynamic allocation is introduced. Reassigning resources at every time slot could otherwise introduce delays in response time, and ramp constraints offer a way to realistically account for these transition costs. The reviewed literature also highlighted the benefits of combinatorial auctions, enabling participants to bid simultaneously across multiple time slots, effectively capturing flexibility in resource utilization. This combinatorial approach is especially relevant when considering dynamic allocations, where efficient redistribution of resources must be managed across consecutive periods to enhance overall performance and fairness.

Decoupling Edge-Served Request Fraction from Hardware Requirements for Enhanced Model Flexibility

We found that the original diminishing return parameter implicitly defined two distinct and independently measurable real-world quantities: the fraction of requests served at the edge, and the amount of hardware (in millicores) required to process a given number of requests in a given time. Since these two aspects do not necessarily need to be coupled, we proposed separating them into two different

parameters. One parameter explicitly controls the fraction of requests served at the edge, reflecting service responsiveness and saturation behavior. The other parameter defines the hardware requirements per request, which depend on the computational intensity of the service. By using a single variable to simultaneously define both dimensions, the model only allows a narrow set of combinations between responsiveness and hardware usage. This restriction limits the diversity of service provider profiles that the model can represent. For instance, a high value of the original diminishing return parameter implies that a very large fraction of requests is served at the edge with minimal hardware allocation, which may not reflect realistic service behaviors. By introducing two separate parameters, we allow for a much broader range of configurations, enabling the model to represent service providers with high responsiveness but also high hardware demands, or any other meaningful combination. This separation increases the expressiveness of the utility function and allows the model to more accurately reflect the physical and performance constraints of edge computing environments.

Potential Directions for Future Work

We outlined potential directions for future work. As discussed in the previous point, one possibility is to redefine the utility function using two separate parameters to decouple the fraction of requests served at the edge from the hardware required to process them. Additionally, some of the ideas and mechanisms presented in the reviewed academic articles, such as secondary markets, ramp constraints, or dynamic coordination mechanisms, can also be adapted and incorporated into future versions of the model. Whether to implement the proposed separation of parameters or adopt one of the other modified utility functions presented in this work is a design decision that should be made in coordination with the rest of the research team. Once this decision is made, the next step would be to proceed with the formal definition and analysis of a selected new utility function.

8.2 Future Work

8.2.1 Introducing Separate Variables for Edge Request Fraction and Hardware Requirements

As discussed in the conclusions, a promising future direction involves redefining the utility function by introducing two separate variables, each explicitly representing a distinct, measurable real-world metric. One variable would define the fraction of requests served at the edge, while the other independently specifies hardware requirements per request. Although a detailed analysis of these new utility functions exceeds the scope of this thesis, we provide their initial formulations, demonstrating that each has a well-defined first-order optimality condition and possesses a unique global maximum. This separation has the potential to substantially increase model flexibility and realism in future formulations of the Edge computing model.

We define λ as the variable that controls the exponential saturation of the fraction of requests served at the edge, and ξ as the number of requests a single millicore can serve in one time-slot. While we keep d' as the amortized price per millicore and l, β and h as previously defined.

Utility Formulation A

$$U_A(h) = l\,\beta\big(1 - e^{-\lambda h}\big) - d'\,\frac{l}{\xi}\,h$$

Benefit saturates with λ, while cost is linear in h. The linear cost term is scaled by $\frac{l}{\xi}$ that converts load into the number of millicores required to serve it during the time-slot.

First-order condition:

$$\frac{\partial U_A}{\partial h} = l\,\beta\,\lambda\,e^{-\lambda h} - \frac{d'}{\xi} = 0 \quad \Longrightarrow \quad h_A^* = \frac{1}{\lambda}\,\ln\!\left(\frac{\beta\,\xi\,\lambda}{d'}\right)$$

Where $\beta\,\xi\,\lambda \le d'$ is the positive allocation constraint.

Uniqueness:

$$\frac{\partial^2 U_A}{\partial h^2} = -l\,\beta\,\lambda^2 e^{-\lambda h} < 0 \quad \forall h > 0$$

Since $U_A(h)$ is strictly concave, the stationary point h_A^* is the unique global maximizer. Because a closed-form expression for h_A^* was obtained, we can directly substitute this value into the net utility function $U_A(h)$ to obtain a closed-form expression for the optimal utility $U_A(h_A^*)$:

$$U_A(h_A^*) = l\,\beta\left(1 - \frac{d'}{\beta\,\xi\,\lambda}\right) - \frac{d'\,l}{\xi\,\lambda}\,\ln\!\left(\frac{\beta\,\xi\,\lambda}{d'}\right).$$

Utility Formulation B

$$U_B(h) = l\,\beta\big(1 - e^{-\lambda h}\big) - d'\,\frac{l}{\xi}\,h\left(1 - e^{-\frac{\lambda h}{l}}\right)$$

Both benefit and cost employ the same saturation rate λ,capturing diminishing returns in revenue and in marginal cost as allocation grows relative to load.

First-order condition:

$$l\,\beta\,\lambda\,e^{-\lambda h} - \frac{d'}{\xi}\left(1 - e^{-\frac{\lambda h}{l}} + \tfrac{\lambda h}{l} e^{-\frac{\lambda h}{l}}\right) = 0$$

A closed form for h_B^* is not available because λh appears both inside and outside exponentials.

Uniqueness:

$$\frac{\partial^2 U_B}{\partial h^2} = -\,l\,\beta\,\lambda^2 e^{-\lambda h} - \frac{d'}{\xi}\left(\frac{\lambda}{l}\right)^2 h\,e^{-\frac{\lambda h}{l}} < 0 \quad \forall h > 0$$

so $U_B(h)$ is strictly concave and the root of the first-order condition is unique, hence there is a unique optimal allocation. Because of strict concavity, any monotone root finding method, such as the Newton Raphson iteration or a bracketed bisection search, converges to h_B^*.

Utility Formulation C

$$U_C(h) = l\,\beta\big(1 - e^{-\lambda h}\big) - d'\,\frac{l}{\xi}\,h\big(1 - e^{-\xi h}\big)$$

Benefit saturation depends on λ (edge-served fraction), while cost saturation depends on ξ (hardware efficiency), allowing independent tuning of these two effects.

First-order condition

$$l\,\beta\,\lambda\,e^{-\lambda h} - \frac{d'}{\xi}\big(1 - e^{-\xi h} + \xi\,h\,e^{-\xi h}\big) = 0$$

Again no closed form exists, but

$$\frac{\partial^2 U_C}{\partial h^2} = -\,l\,\beta\,\lambda^2 e^{-\lambda h} - d'\,l\,e^{-\xi h}\left(\frac{\xi}{\xi}\right)^2 < 0 \quad \forall h > 0$$

so $U_C(h)$ is strictly concave and the stationary point h_C^* is unique. Strict concavity guarantees convergence of Newton–Raphson or bisection methods to h_C^*.

Table 8.1 lists the three formulations and their optimal allocation conditions.

Form	**Utility** $U(h)$	**First-order condition**	**Optimal allocation** h^*
A	$l\,b\,(1-e^{-\lambda h}) - \frac{d\,l}{\xi}\,h$	$b\lambda e^{-\lambda h} = \frac{d}{\xi}$	$\frac{1}{\lambda}\ln\left(\frac{b\,\xi\,\lambda}{d}\right)$
B	$l\,b\,(1-e^{-\lambda h}) - \frac{d\,l}{\xi}\,h\,(1-e^{-\frac{\lambda h}{l}})$	$b\lambda e^{-\lambda h} = \frac{d}{\xi}\left(1-e^{-\frac{\lambda h}{l}}+\frac{\lambda h}{l}e^{-\frac{\lambda h}{l}}\right)$	Solve numerically
C	$l\,b\,(1-e^{-\lambda h}) - \frac{d\,l}{\xi}\,h\,(1-e^{-\xi h})$	$b\lambda e^{-\lambda h} = \frac{d}{\xi}\left(1-e^{-\xi h}+\xi h e^{-\xi h}\right)$	Solve numerically

Table 8.1: Comparison of three utility functions and their optimal allocations

8.2.2 Improving Computational Efficiency for Shapley Value Estimation

The most computationally intensive task in the current model is calculating each service provider's Shapley value. For the network owner, this is not problematic; as detailed in Section 5.10, the network owner consistently retains half of the grand coalition's value due to its role as the veto player. However, computing Shapley values for individual SP requires evaluating all possible feasible coalitions, a process that grows exponentially with the number of SPs involved.

In our primary approach, we addressed this challenge by applying Monte Carlo sampling methods. While effective as an approximation, Monte Carlo methods may perform inconsistently when there is significant heterogeneity among service providers. For example, including or excluding a provider with a highly complementary load profile can substantially alter the estimated Shapley value purely due to random sampling variability.

To overcome this limitation, we propose developing estimators based on two key factors: the distinctiveness of a provider's load profile relative to others, and the proportion of total requests served at the edge, which determines the relative weight of costs in the utility function. By computing a coefficient capturing these characteristics, we could directly scale an SP's marginal contribution within the grand coalition. This approach would allow for a more computationally efficient and reliable estimation of Shapley values, particularly when scaling the model to include a large number of SPs. Specifically, this coefficient would be greater for providers with more distinct load profiles and for those serving a lower fraction of requests at the edge.

8.2.3 Extending the Model for Multi-resource Optimization and Quality-of-Service Differentiation

Throughout this thesis, we have focused on optimizing a single computational resource, specifically CPU allocation measured in millicores. While this simplification allowed us to rigorously analyze the core model and clearly interpret its economic implications, real-world edge computing scenarios typically require simultaneous consideration of multiple resource types. Resources such as memory, storage, bandwidth, and computing capability each have distinct usage patterns and constraints.

Moreover, we could interpret that the existing model associates CPU allocation with corresponding quantities of memory and storage. This interpretation, while simplifying, is overly restrictive, as different edge computing services exhibit heterogeneous resource demands. By explicitly modeling

multiple resource dimensions, including CPU capacity, memory allocation, storage capability, and bandwidth, we can more accurately capture the trade-offs faced by service providers with varying service profiles.

Furthermore, Quality-of-Service (QoS) metrics such as latency, reliability, and response time have only been briefly considered through preliminary ideas about internet slicing by the network operator. Initially proposed as a mechanism to counteract the lack of strategy-proofness in the benefit factor, internet slicing could also enhance the realism of the model. Explicitly incorporating QoS metrics as constraints or additional utility terms would allow SPs to associate higher benefit factors directly with improved latency, reliability, and bandwidth guarantees.

Implementing such multi-resource and QoS-aware modeling will undoubtedly increase the complexity of the optimization problem, as it now involves multi-dimensional allocation strategies. Nevertheless, this extended model would enable a more realistic and nuanced representation of practical edge computing environments.

8.2.4 Incorporating Uncertainty, Risk Preferences, and Extended Dynamic Allocation

Throughout this thesis, variability was introduced only into the load function by adding randomness to simulate stochastic demand scenarios. However, a detailed analysis of the implications of such uncertainty on optimal allocations and coalition stability was beyond the scope of this work. Future extensions of the model could systematically incorporate stochastic processes not only in demand but also in other critical utility parameters, such as resource prices, benefits, and diminishing returns factors.

Additionally, the current formulation assumes the daily load pattern remains constant throughout the entire investment period, allowing only daily dynamic allocation. A further enhancement could involve modeling weekly load patterns or incorporating longer-term variations across the full investment horizon. Such an extended dynamic allocation would imply a unique allocation for each individual time slot of each day throughout the investment period, significantly increasing the complexity and computational demands of the model.

8.2.5 Integration of Secondary Markets and Auction-based Mechanisms

As discussed in the conclusions and throughout this thesis, introducing secondary markets or auction-based mechanisms represents a promising avenue for future extensions. Such mechanisms can serve two primary purposes: first, to address the identified lack of strategy-proofness related to the benefit factor; and second, to enhance service providers' overall utility. By allowing service providers to trade their allocated resources at any point during the investment period, secondary markets could provide a practical means to mitigate risks associated with parameter misestimations and dynamically improve individual utilities through flexible resource reallocation.

8.2.6 Spatial Considerations and Network Topology

Another valuable direction for extending the model involves incorporating spatial dimensions and network topology considerations. The current framework assumes a single edge site, disregarding location-specific characteristics such as resource costs and latency variations. A natural generalization would involve multiple edge sites interconnected via a backbone network, each characterized by unique pricing structures, availability of resources, and distinct latency constraints. In such an extended model, service providers would face more complex allocation and resource transfer decisions, influenced not only by local resource prices but also by the geographical distribution of users, infrastructure capabilities, and network latency constraints. Additionally, coalition formation and stability analyses would need to explicitly account for spatial interdependencies, resource transfer costs, and potential strategic behaviors driven by site-specific advantages.

Bibliography

A. P. Vela A. Vía, M. R., F. Morales, & Velasco, L. (2016). Traffic generation for telecom cloud-based simulation. *ICTON*, 1–4.

Das, S., & Mukherjee, A. (2021). An empirical study in edge computing using aws. In *Proceedings of the 11th ieee annual computing and communication workshop and conference (ccwc)* (pp. 0734–0739). doi: 10.1109/CCWC51732.2021.9375892

Diego Kiedanski, D. K., Ana Bušić, & Orda, A. (2020). Efficient distributed solutions for sharing energy resources at local level: a cooperative game approach. *59th IEEE Conference on Decision and Control (CDC)*, 1–9.

Diego Kiedanski, D. K., Ariel Orda. (2019). The effect of ramp constraints on coalitional storage games. *Proceedings of the Tenth ACM International Conference on Future Energy Systems (e-Energy '19)*, 1–13.

Diego Kiedanski, D. K., Ariel Orda. (2020a). Combflex: a linear combinatorial auction for local energy markets. *IEEE SmartGridComm*, 1–8.

Diego Kiedanski, D. K., Ariel Orda. (2020b). Discrete and stochastic coalitional storage games. *Eleventh ACM International Conference on Future Energy Systems (e-Energy)*, 1–12.

Diego Kiedanski, D. K., Lauren Kuntz. (2020). Benchmarks for grid flexibility prediction: Enabling progress and machine learning applications. *ICLR Workshop on Tackling Climate Change with Machine Learning*, 1–6.

Diego Kiedanski, J. H., Daniel Kofman. (2020). Pymarket - a simple library for simulating markets in python. *Journal of Open Source Software*, 1–3.

Diego Kiedanski, P. M., Daniel Kofman, & Horta, J. (2020). Misalignments of objectives in demand response programs: a look at local energy markets. *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 1–7.

European Telecommunications Standards Institute (ETSI). (2019). *Edge Computing: From standard to actual infrastructure deployment and software development.* https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp36_EdgeComputing.pdf. (Accessed: 2025-05-03)

Inc., M. (2024). *Metabase documentation.* https://www.metabase.com/docs/latest/. (Accessed: 2025-06-12)

Juan Vanerio, S. S., Lily Hugerich. (2024). Tero: Offloading cdn traffic to massively distributed devices. In (p. 1-13).

Khatri, H., Lambert, A., Cenzano, J., & Broilo, R. (2020). How we scale live streaming for millions of viewers simultaneously. *Facebook Engineering Blog*. (Available at: https://engineering.fb.com/2020/10/12/video-engineering/live-streaming/)

Kiedanski, D., Kofman, D., Horta, J., & Menga, D. (2019). Strategy-proof local energy market with sequential stochastic decision process for battery control. *IEEE*, 1–9.

Li, Y., & Gao, W. (2018). Muvr: Supporting multi-user mobile vr with resource-constrained edge cloud. In (pp. 1–16).

Moura, J., & Hutchison, D. (2019). Game theory for multi-access edge computing: Survey, use cases, and future trends. *IEEE Communications Surveys & Tutorials*, *21*(1), 260–288. doi: 10.1109/COMST.2018.2877280

Nguyen, V.-D., & Huh, E.-N. (2021). Efficient solution for large-scale iot with proactive edge–cloud brokers clustering. *Sensors*, 1–21.

Rosario Patanè, A. A., Chahed, T., Kiedanski, D., & Kofman, D. (2023). Coalitional game-theoretical approach to coinvestment with application to edge computing. *20th Consumer Communications*

& Networking Conference (CCNC), 1–6.

Tabarez, S. (2025). *Edge computing simulations.* Retrieved from https://github.com/Santiago-Tabarez/edge-computing-simulations (Accessed: 2025-06-12)

Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y., & Vinel, A. (2019). Coalitional games for computation offloading in noma-enabled multi-access edge computing. *IEEE Transactions on Vehicular Technology*, *68*(12), 12131–12143. doi: 10.1109/TVT.2019.2948679

Appendix A

Summary of Related Articles

In addition to an in-depth analysis of the primary Edge Computing article, this thesis examines nine other articles provided as complementary thesis material. The core concepts of these articles relate, to varying degrees, to the Edge Computing problem explored in this work. These articles serve as reference points and sources of inspiration, offering valuable insights and suggesting possible enhancements to our current model. In the following sections, we summarize each article, highlight their main contributions, and discuss how their findings may either complement our existing approach or guide potential improvements through the integration of new concepts.

A.1 Strategy-Proof Local Energy Market With Sequential Stochastic Decision Process For Battery Control

The article (Kiedanski, Kofman, Horta, & Menga, 2019) proposes a new framework for coordinating household energy storage systems (batteries) and local renewable energy generation through a local energy market. The key elements are described below.

Strategy-Proof Local Energy Market

- Uses a multi-unit double auction (MUDA) that is strategy-proof in both price and quantity.
- Prosumers (households producing and consuming electricity) are incentivized to bid their true preferences.
- This design encourages fairness and transparency, enabling participants to buy or sell energy at prices determined by competition.

Battery Control Via Stochastic Decision Process

- Each household manages its battery through a sequential stochastic decision process.
- Uncertainty in load and generation forecasts is handled using approximate dynamic programming or reinforcement learning methods.
- The control policy optimizes when to charge, discharge, buy, or sell energy, considering expected rewards and costs.

Combined Benefits

- Integrating the local market with adaptive battery control reduces reliance on the main grid, lowering both congestion and costs.
- Prosumers benefit economically by locally matching supply and demand, rather than depending solely on utility rates.
- Distribution system operators experience a more stable aggregate load profile within the neighborhood.

Relation To The Edge Computing Problem

Although this article centers on household energy markets rather than hardware capacity sharing, it focuses on mechanisms through which participants benefit from truthful bidding. In edge computing coinvestments, a similar requirement emerges: handling uncertainty in resource allocation demands and incentivizing honest declarations of resource needs or valuations. The core ideas of strategy-proof market design and real-time decision-making provide valuable insights for the fair and efficient allocation of edge computing resources among multiple stakeholders.

In contrast, our Edge Computing model directly assigns resources to maximize the grand coalition value, thus inherently providing optimal allocation. However, a significant issue identified in our approach is the lack of strategy-proofness. Introducing auction mechanisms, such as MUDA, for resource assignment would fundamentally alter the existing model's intent. Nonetheless, incorporating a market mechanism after resource allocation, particularly when discrepancies arise between declared and actual utility parameters, could allow SPs to trade their allocated resources. This approach might be especially advantageous in scenarios involving dynamic allocation, discussed in section 5.9.2. Here, SPs can be analogous to prosumers, where the "produced" resources correspond to their allocated capacities, and the "consumed" resources reflect their operational demands. The potential implementation of such post-allocation market mechanisms is explored further in section 3.15.2 and section 5.8.4

A.2 The Effect Of Ramp Constraints On Coalitional Storage Games

The article (D. K. Diego Kiedanski Ariel Orda, 2019) explores how ramp constraints limits on battery charge and discharge rates affect the formation of coalitions for jointly purchasing and operating energy storage systems. Using cooperative game theory, it identifies scenarios where consumer groups benefit from shared battery investments, demonstrating how ramp constraints significantly influence coalition profitability and stability.

Shared Battery Investment and Cooperative Games

- Consumers jointly invest in batteries, dividing costs and benefits based on cooperative game rules.
- Without ramp constraints, sharing provides minimal or no advantage over individual investments.
- Introducing ramp constraints makes certain joint investments profitable but can also destabilize coalitions (potentially resulting in an empty core).

Key Theoretical Findings

- Constant ramp constraints, independent of battery size, can lead to an empty core, indicating unstable profit divisions.

- When ramp constraints scale linearly with capacity, stable and fair cost allocation is achievable, and the Shapley value resides within the core.
- Certain scenarios relate closely to classic "glove-market" games, allowing explicit identification of stable solutions.

Practical Relevance

- Real battery products have significant ramp constraints, influencing coalition profitability for smaller groups.
- Empirical data confirms ramp constraints critically impact the viability of coalition formation.
- Analysis offers guidelines for coalition formation and stable benefit-sharing strategies.

Relation To The Edge Computing Problem

Originally developed for battery storage, ramp constraints parallel practical limitations in edge computing resources—such as maximum data throughput, concurrency limits, and reallocation delays. These constraints influence coalition incentives, impacting stability and fairness in resource-sharing. Recognizing such analogies helps build realistic and fair cost-sharing models for edge computing environments, especially when immediate resource reallocation is impractical.

Currently, ramp constraints are not integrated into our model but may be relevant in future iterations, particularly for dynamic allocations over multiple time slots. Increasing the frequency of allocation adjustments under such constraints may not always yield improved performance or efficiency. The significance of ramp constraints also varies depending on virtualization practices: duplicating a full virtual environment or database could impose substantial constraints, whereas expanding resources within an existing environment is generally quicker.

A.3 PyMarket: A Simple Library For Simulating Markets in Python

The article (J. H. Diego Kiedanski Daniel Kofman, 2020) introduces PyMarket, a library written in Python designed to facilitate the development and evaluation of diverse market mechanisms (https://pypi.org/project/pymarket/). Its primary use is in contexts such as local energy trading or bandwidth allocation, where auctions and market algorithms require systematic prototyping and evaluation. PyMarket offers an interface for bid submission, market clearing processes, and visualization of results, including supply and demand curves.

Key Features and Market Mechanisms

- Implements standard market designs, including multi-unit double auctions and peer-to-peer trading schemes.
- Provides Python APIs for defining bids and clearing mechanisms, with built-in statistical analysis and plotting capabilities.
- Facilitates replication and adaptation of existing market designs for academic research and new applications.

Practical Utility

- Enables quick prototyping and validation of new auction and market mechanisms.
- Offers a framework for reproducible experiments with consistent data, metrics, and visualizations.
- Proposes market clearing routines, simplifying the integration of customized auctions or bidding rules.

Relation to the Edge Computing Problem

Although PyMarket was primarily developed for energy-related auctions, its toolkit can be applied effectively to resource allocation scenarios in edge computing. Coinvestments SPs or operators could create real-time markets for resources such as CPU capacity, storage, or network throughput. Additionally, PyMarket can support the implementation of post-allocation market mechanisms as discussed in Section 3.15.2. Its flexible design offers:

- Simple bidding processes for shared edge resources.
- Visual tools to assess real time capacity allocation and pricing.
- An extendable base for incorporating specialized constraints, such as latency demands or fairness conditions, crucial for cooperative edge computing scenarios.

Moreover, the simulation toolkit developed in this thesis uses Python, making PyMarket highly compatible with our existing codebase. However, a complete integration of PyMarket into our framework is beyond this thesis's scope, as it would significantly expand its complexity and extent.

A.4 Efficient Distributed Solutions for Sharing Energy Resources at the Local Level: A Cooperative Game Approach

This article (D. K. Diego Kiedanski Ana Bušić & Orda, 2020) investigates how multiple households (players in our context), each owning energy generation or storage equipment, can jointly reduce electricity costs by cooperatively sharing resources. By formulating this as a cooperative game, the authors demonstrate that at least one stable payoff allocation exists, ensuring that all participants are satisfied (non-empty core). Specifically, the article models the demand and battery constraints of each household.

Main Contributions

- A linear programming formulation demonstrating that a stable payoff allocation in the core can always be found. This addresses concerns regarding the computational complexity of finding stable cost distributions.
- An efficient centralized algorithm that computes the payoff allocation. Unlike naive methods requiring exhaustive evaluation of all coalitions, the proposed solution scales linearly with the number of players.
- A distributed algorithm that allows households to iteratively converge to a stable payoff without fully disclosing their private load or resource data. The algorithm achieves convergence through exchanging partial results among neighboring nodes in a communication network.
- A graph analysis highlighting that certain network topologies, such as expander graphs or selected overlay networks, accelerate iterative convergence, whereas complete or sparse ring topologies slow it down.

Relation to the Edge Computing Problem

The article's exploration of cooperative investment and resource-sharing among multiple participants under specific constraints aligns directly with our edge computing co-investment problem. Concepts such as distributed information exchange without revealing full data and scalable distributed algorithms can inform our approach. Moreover, both scenarios face the computational challenge of evaluating all possible coalitions for calculating the Shapley value, especially when considering interdependent contributions. Nevertheless, the extensive use of linear programming techniques in the article presents a significant obstacle for direct integration into our nonlinear model. Adapting our nonlinear utility functions, characterized by diminishing returns, into linear equivalents would not only be challenging but would also fundamentally alter the nature and practical realism of our edge computing model.

A.5 Design of a Combinatorial Double Auction for Local Energy Markets

This article (D. K. Diego Kiedanski Ariel Orda, 2020a) proposes an approach to local energy trading, shifting from multiple individual auctions (one per time-slot) toward a single combinatorial double auction covering multiple time-slots simultaneously. Participants (households) specify their desired aggregate daily energy usage, leveraging battery flexibility or shifting loads. The auction maximizes the total market value while ensuring each participant is at least as well off as they would be without participating.

Main Contributions

- A combinatorial bidding format enabling participants to bid on multi-slot energy usage profiles rather than separate bids for each time-slot.
- An allocation rule that selects net consumption or production profiles to maximize local energy trade value, benchmarked against tariffs paid to the main grid.
- A settlement (payment) mechanism that ensures buyers and sellers within the coalition achieve mutual economic improvement compared to traditional utility arrangements.
- Illustrative examples demonstrating the advantages of the combinatorial auction over sequential single-slot auctions by exploiting synergy across time-slots.

Relation to the Edge Computing Problem

This article addresses multi-interval auctions, a scenario closely related to the allocation of edge computing resources across multiple time-slots. The principles of combinatorial allocation, ensuring individual rationality, and bundling resource requests to simplify complexity are particularly relevant. Such concepts are especially useful when extending our model to dynamic allocation contexts, where resources must be efficiently redistributed based on changing demands over consecutive periods. Rather than comparing benefits against an external baseline, as done in the energy context, we can compare improvements against an initial fixed allocation scenario. This approach ensures that each stakeholder clearly recognizes the advantages provided by cooperative resource reallocation.

A.6 Benchmarks for Grid Flexibility Prediction: Enabling Progress and Machine Learning Applications

This article (D. K. Diego Kiedanski Lauren Kuntz, 2020) highlights the absence of standardized benchmarks for evaluating grid flexibility and demand response (DR) programs. As renewable energy de-

ployment increases, effective DR schemes become crucial for aligning intermittent generation with consumption patterns. The authors propose a benchmark framework for generation profiles, detailed grid topology, consumer appliance models, and standardized performance metrics. This structured approach allows consistent evaluation and comparison of DR initiatives across realistic scenarios.

Main Contributions

- Benchmark Architecture: Provides guidelines for specifying data on local renewable generation, grid structure, and consumer appliances (flexible and inflexible). Recommendations include optimal time granularity, baseline operating modes without DR, and performance metrics to assess DR effectiveness.
- Grid Flexibility Metrics: Focuses on aligning renewable energy production with actual consumption to minimize unmet demand and energy curtailment, rather than exclusively targeting peak load reduction.
- Machine Learning Opportunities: Illustrates how standardized benchmarks facilitate machine learning applications, including deep learning methods for predicting maximum achievable flexibility and reinforcement learning techniques for real-time management of flexible assets.

Relation to the Edge Computing Problem

Although centered on power systems and renewable energy integration, the proposed benchmarking framework is applicable to evaluating multi-tenant edge computing environments. The distinction between flexible and inflexible electrical loads corresponds directly to edge computing workloads, which may either be deferred or must be executed immediately. The concept of grid flexibility aligns with metrics used to quantify resource mismatches in edge infrastructures, such as underutilized hardware or unmet computational demand. Additionally, standardized benchmarks could support machine learning approaches, including reinforcement learning, for managing dynamic and adaptive resource allocation in edge computing scenarios. While these ideas have not been incorporated into the current model, they offer promising avenues for future development.

A.7 Discrete and Stochastic Coalitional Storage Games

This article (D. K. Diego Kiedanski Ariel Orda, 2020b) addresses the idea of sharing energy storage capacity among multiple residential consumers (players) cooperatively. Unlike traditional models that assume a continuous (infinitely divisible) battery size and deterministic demand profiles, this article considers two important practical extensions:

- Stochastic Demand Profiles: Each player's daily consumption is uncertain and represented by a probability distribution or scenario set (rather than a fixed, deterministic profile).
- Discrete Battery Sizes: Instead of allowing for any fraction of a storage device to be purchased, real-world constraints are introduced by having battery sizes come only in integral multiples of a base module (e.g., 13.5 kWh units).

The article demonstrates how these two extensions affect the classical model of "coalitional storage games" and how a stable cost-allocation method (i.e., in the core) can or cannot be obtained.

A.7.1 Cooperative Investment in Energy Storage

- Households form a coalition to jointly invest in battery storage, sharing the total costs and subsequent benefits.

- Each household's random energy usage is handled using a two-stage stochastic optimization:
  - First stage: the coalition decides how many battery modules to buy (discrete).
  - Second stage: the realized consumption and storage usage is optimized for each scenario, minimizing day-to-day operational costs.
- By pooling resources and risks, households can acquire a larger effective storage capacity than would be viable individually.

A.7.2 Core Existence and Approximate Stability

- In the continuous (infinitely divisible) battery model, the article shows the core is guaranteed to be nonempty (so there is a stable cost split).
- Under discrete battery-size constraints, the game may fail to admit a cost allocation in the core. Hence, an approximate or relaxed notion of stability (the ε-core) is introduced.
- An approximate solution is proposed: first find a stable allocation in the continuous case, then adjust or "round" the result for discrete battery sizes. The article derives theoretical bounds on the "rounding error."

A.7.3 Relation To The Edge Computing Problem

In edge computing scenarios, as in shared battery investment models, resource sharing among multiple entities is a strategy to reduce capital expenditure and operational costs. The article's approach to modeling stochastic demand closely parallels the time-varying workloads encountered in edge environments. Discrete battery sizes are conceptually similar to the modular capacity increments found in edge infrastructure. The coalitional analysis presented in the article, which ensures that no group of participants can benefit by deviating from the grand coalition, is also relevant to multi-tenant edge systems where isolated subgroups may fail to capture the full benefits of cooperation. In the extended version of our model, we address fluctuations in service providers' demand by incorporating the standard deviation of their load; however, our model currently assumes identical load values across different days for a given time-slot. Allowing for varying loads across days is a possible direction for future iterations.

A.8 CombFlex: A Linear Combinatorial Auction for Local Energy Markets

This article (D. K. Diego Kiedanski Ariel Orda, 2020a) proposes a market mechanism named CombFlex to facilitate local energy trading among prosumers within low-voltage distribution grids. Its main contributions include the following:

Motivation and Bid Format

- Traditional local energy markets typically use sequential auctions or peer-to-peer trading schemes, each conducted per time-slot.
- Such methods fail to leverage the flexibility in prosumers' loads, which arises from energy storage capabilities.
- CombFlex introduces a combinatorial bidding format, allowing participants to submit bids specifying prices and quantities across multiple time-slots simultaneously. This bid format explicitly captures the flexibility of prosumers' battery usage schedules.

Winner Determination Problem

- CombFlex aggregates bids for all time-slots into a single linear programming (LP) formulation.
- The LP objective is to maximize the total value of local energy trades, aligning aggregate buying and selling volumes while adhering to battery constraints.
- The resulting optimization problem has linear complexity relative to the number of participants and time-slots, making it computationally manageable.
- By exploiting flexibility across multiple time-slots, CombFlex enables more profitable local trading, reducing dependence on external grids.

Price and Payment Rule

- After determining an optimal allocation, clearing prices for each time-slot are set within the range bounded by the highest accepted selling price and the lowest accepted buying price.
- The payment rule can optionally split participants into two groups, ensuring that each group's clearing prices are influenced only by the opposite group's supply and demand. This prevents participants from manipulating their own trading prices.

Relation to the Edge Computing Problem

The CombFlex mechanism primarily addresses flexibility in local energy exchanges among prosumers. However, its core concepts have potential relevance to edge computing resource allocation. Specifically, the combinatorial auction approach could be used in scenarios where resources must be allocated dynamically across multiple time-slots, particularly when adapting allocations due to changes in expected utility parameters. Additionally, as discussed in section 3.15.2, employing such combinatorial auction mechanisms post-allocation could limit strategic misreporting by service providers, thus enhancing the overall fairness and effectiveness of the cooperative edge computing framework.

A.9 Misalignments of Objectives in Demand Response Programs: A Look at Local Energy Markets

The article (P. M. Diego Kiedanski Daniel Kofman & Horta, 2020) examines Local Energy Markets (LEMs) within demand response programs. It focuses on scenarios where households, possibly equipped with energy storage, trade their energy surplus or deficit in a local market, while also having the option to transact directly with a Traditional Energy Company (TEC). The primary motivation is to investigate how agents schedule their consumption and production when interacting with potentially more attractive local markets, and how these decisions might inadvertently create negative impacts on the overall electricity demand profile. A key finding is that certain market designs can misalign participant incentives, causing unexpected peaks or imbalances in demand, ultimately undermining the intended benefits.

Main Idea and Contributions

- Introduces a multi-stage stochastic game to model prosumers' decision-making processes, considering both local markets and traditional energy suppliers.
- Proposes a simplified model where agents form and continually update beliefs about future market prices, adjusting their battery schedules accordingly.

- Demonstrates through numerical experiments that purely rational agent strategies can unexpectedly synchronize, leading to undesirable consumption peaks.
- Identifies how specific tariff structures (such as certain time-of-use pricing ratios) and agent expectations contribute to these harmful consumption peaks and explores ways to mitigate them.

Relation to the Edge Computing Problem

While local energy markets and edge computing both involve resource management at a local scale, a direct parallel is not immediately apparent. However, a relevant analogy can be drawn concerning dynamic resource allocations. In edge computing scenarios where resources are frequently reallocated, simultaneous adjustments across all service providers within identical time-slot intervals could create throughput bottlenecks. Such synchronized reallocations could similarly produce undesirable peaks, negatively impacting response times during resource redistribution. This aspect highlights a potential parallelism with the challenges identified in the article and suggests the need for careful consideration of synchronized reallocations in edge computing environments.

A.10 Summary of Ideas Related to The Edge Computing Problem

Although significant differences exist between energy markets, including ramp constraints for battery storage and resource reallocation processes in edge data centers, numerous parallels can still be drawn. The main body of this thesis synthesizes the key insights obtained from reviewing these related articles, highlighting how their findings can inform enhancements and refinements to our current edge computing coinvestment model.

Appendix B

Demonstrations

The detailed mathematical demonstrations are included in this Appendix to improve the readability and overall flow of the thesis. While these proofs provide necessary rigor and validate our analytical framework, they are not essential for understanding the main contributions and conclusions of the work. Readers who are interested in the full details of the mathematical proofs can refer to this appendix at their convenience.

B.1 Positive Net Utility Constraint

The total net utility for player i over the investment period T is given by:

$$U^i_{tot\_net} = D \cdot T \cdot \left(\beta^i \cdot l^i_{avg} \cdot \left(1 - e^{-\xi^i \cdot h^i}\right)\right) - d \cdot h^i.$$

To ensure that the total net utility is positive, we require:

$$U^i_{tot\_net} > 0.$$

Substituting the expression for $U^i_{tot\_net}$:

$$D \cdot T \cdot \left(\beta^i \cdot l^i_{avg} \cdot \left(1 - e^{-\xi^i \cdot h^i}\right)\right) - d \cdot h^i > 0.$$

Rearranging the inequality:

$$D \cdot T \cdot \beta^i \cdot l^i_{avg} \cdot \left(1 - e^{-\xi^i \cdot h^i}\right) > d \cdot h^i.$$

Considering that $\left(1 - e^{-\xi^i \cdot h^i}\right) < 1$, when ξ is negative and h is positive, we can substitute this expression with 1. Thus, the condition required for the total net utility to be positive is as follows:

$$d \cdot h^i < D \cdot T \cdot \beta^i \cdot l^i_{avg}.$$

Note that this expression is valid for any positive allocation h, not only the optimal one.

B.2 Optimal Allocation and Positive Allocation Constraint

Starting from the equation of total net utility 3.3, we calculate the first derivative with respect to h^i and set it equal to zero:

$$D \cdot T \cdot \left(\beta^i \cdot l^i_{\text{avg}} \cdot \xi \cdot e^{-\xi \cdot h^i}\right) - d = 0 \tag{B.1}$$

To solve for h^i, we isolate the exponential term:

$$e^{-\xi \cdot h^i} = \frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}$$

Taking the natural logarithm of both sides:

$$-\xi \cdot h^i = \ln\left(\frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}\right)$$

Thus, h^i is given by:

$$h^i = -\frac{1}{\xi} \cdot \ln\left(\frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}\right) \tag{B.2}$$

To more intuitively interpret how the optimal allocation relates to the utility function parameters, we can rewrite the previous equation as:

$$h^i = \frac{1}{\xi} \cdot \ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right) \tag{B.3}$$

The second derivative, indicating the nature of the critical point, is calculated as follows: :

$$\frac{d^2 U^i_{\text{tot\_net}}}{(dh^i)^2} = -D \cdot T \cdot \left(\beta^i \cdot l^i_{\text{avg}} \cdot \xi^2 \cdot e^{-\xi \cdot h^i}\right) \tag{B.4}$$

This function will always have negative values because it has a negative sign at the beginning and is the product of positive terms. This indicates that the critical point is a maximum aligned with the diminishing-return effect expressed in the net utility equation.

Having established that the optimal allocation h^i maximizes the net utility and is unique, we must next ensure that h^i is positive. This is certain when.

$$\frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi^i} < 1$$

Therefore, the condition for h^i to be positive is:

$$d < D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi^i \tag{B.5}$$

B.3 Positive Net Utility Constraint

We want to investigate whether the net utility subject to the optimal allocation condition can be negative. We prove that it is always non-negative.

We start with the net utility equation

$$U^i_{\text{tot\_net}} = D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \left(1 - e^{-\xi \cdot h^i}\right) - d \cdot h^i, \tag{B.6}$$

subject to the optimal allocation

$$h^i = \frac{1}{\xi} \cdot \ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right). \tag{B.7}$$

First, we substitute the optimal allocation expression into the exponential term $e^{-\xi \cdot h^i}$ of the net utility expression.

$$e^{-\xi \cdot h^i} = e^{-\xi \cdot \frac{1}{\xi} \ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right)}.$$

Notice that we can simplify the exponent since $-\xi \cdot \frac{1}{\xi} = -1$. Therefore, the expression simplifies to

$$e^{-\xi \cdot h^i} = e^{-\ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right)}.$$

Now we apply the exponential logarithm identity. For any positive number x, $e^{-\ln(x)} = \frac{1}{x}$. From the positive allocation constraint we know that x is positive.

$$e^{-\ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right)} = \frac{1}{\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}}.$$

Taking the reciprocal of the fraction gives

$$e^{-\xi \cdot h^i} = \frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}.$$

Now, we compute

$$1 - e^{-\xi \cdot h^i} = 1 - \frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}.$$

Substituting into the original equation yields

$$U^i_{\text{tot\_net}} = D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \left(1 - \frac{d}{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}\right) - d \cdot h^i.$$

The first term simplifies to

$$D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} - \frac{d}{\xi}.$$

Substituting the expression for h^i in the second term, we have

$$d \cdot h^i = \frac{d}{\xi} \ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right).$$

Thus, the net utility becomes

$$U^i_{\text{tot\_net}} = D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} - \frac{d}{\xi} - \frac{d}{\xi} \ln\left(\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}\right).$$

To simplify the equation, we define a new variable z

$$z = \frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d}.$$

Then, the expression for the net utility becomes

$$U^i_{\text{tot\_net}} = \frac{d}{\xi} \left[z - 1 - \ln(z) \right].$$

Determine the condition for positivity: Since $\frac{d}{\xi} > 0$, the sign of $U^i_{\text{tot\_net}}$ depends on

$$g(z) = z - 1 - \ln(z).$$

We can see that:

- The derivative $g'(z) = 1 - \frac{1}{z}$ is zero at $z = 1$.
- The second derivative $g''(z) = \frac{1}{z^2} > 0$ for $z > 0$, so $z = 1$ is a minimum.
- At $z = 1$, we have $g(1) = 1 - 1 - \ln(1) = 0$.

Thus, $g(z) \geq 0$ for all $z > 0$ and with $g(z) = 0$ only when $z = 1$. Therefore, $U^i_{\text{tot\_net}}$ is strictly positive if and only if

$$g(z) > 0 \quad \Longleftrightarrow \quad z \neq 1.$$

Recalling that

$$z = \frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d},$$

the condition $z \neq 1$ is equivalent to

$$\frac{D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi}{d} \neq 1,$$

or, rearranging,

$$d \neq D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi.$$

Thus, starting from the initial equations and substituting the given expression for h^i, we arrive at the conclusion that $U^i_{\text{tot\_net}}$ is nonnegative and equal to zero only if:

$$d = D \cdot T \cdot \beta^i \cdot l^i_{\text{avg}} \cdot \xi.$$

B.4 Derivation of Different Expressions for the Utility Function

In the following sections, we present step-by-step demonstrations showing how to derive alternative expressions for the net utility function. Each version is obtained by isolating one of the variables from the optimal allocation condition and substituting it into the net utility equation.

Getting $U_{net}(\xi, \rho, h)$

Starting with the original net utility function:

$$U_{\text{net}}(\xi, \rho, h, d') = \rho \left(1 - e^{-\xi h}\right) - d'h$$

Substitute d' from Equation (3.23):

$$d' = \frac{\xi \cdot \rho}{e^{h \cdot \xi}}$$

Perform the following substitution:

$$U_{\text{net}}(\xi, h, d') = \rho \left(1 - e^{-\xi h}\right) - \left(\frac{\xi \cdot \rho}{e^{h \cdot \xi}}\right) h$$

Simplifying the expression:

$$U_{\text{net}}(\xi, h, d') = \rho \left(1 - e^{-\xi h} - \frac{\xi h}{e^{\xi h}}\right)$$

Getting $U_{net}(\xi, \rho, d')$

We start with the utility function, where ρ, ξ, and h are variables, and d' is a constant. As we want to eliminate h, we proceed to substitute it from the 3.19 equation into the utility equation 3.18. This results in the following equation:

$$U_{\text{ts\_net}}(\xi, \rho, d') = \rho \cdot \left(1 - e^{-\xi \cdot \left(-\frac{1}{\xi} \ln\left(\frac{d'}{\rho \cdot \xi}\right)\right)}\right) - d' \cdot \left(-\frac{1}{\xi} \ln\left(\frac{d'}{\rho \cdot \xi}\right)\right)$$

The exponent simplifies as follows:

$$-\xi \cdot \left(-\frac{1}{\xi} \ln\left(\frac{d'}{\rho \cdot \xi}\right)\right) = \ln\left(\frac{d'}{\rho \cdot \xi}\right)$$

Thus, the expression for U_{net} becomes:

$$U_{\text{ts\_net}}(\xi, \rho, d') = \rho \cdot \left(1 - e^{\ln\left(\frac{d'}{\rho \cdot \xi}\right)}\right) + d' \cdot \frac{1}{\xi} \ln\left(\frac{d'}{\rho \cdot \xi}\right)$$

Since $e^{\ln(a)} = a$, the equation further simplifies to:

$$U_{\text{ts\_net}}(\xi, \rho, d') = \rho \cdot \left(1 - \frac{d'}{\rho \cdot \xi}\right) + d' \cdot \frac{1}{\xi} \ln\left(\frac{d'}{\rho \cdot \xi}\right)$$

Getting $U_{net}(\xi, h, d')$

To obtain the net utility only in terms of ξ, d' and h^* , we substitute 3.22 in 3.25.

$$U_{\text{ts\_net}}(\xi, h, d') = \frac{d' \cdot e^{\xi \cdot h}}{\xi} \left(1 - \frac{d'}{\frac{d' \cdot e^{\xi \cdot h}}{\xi} \xi}\right) + \frac{d'}{\xi} \ln\left(\frac{d'}{\frac{d' \cdot e^{\xi \cdot h}}{\xi} \xi}\right)$$

$$U_{\text{ts\_net}}(\xi, h, d') = \frac{d' \cdot e^{\xi \cdot h}}{\xi} \left(1 - \frac{1}{e^{\xi \cdot h}}\right) + \frac{d'}{\xi} \ln\left(\frac{1}{e^{\xi \cdot h}}\right)$$

Using logarithmic identity $\ln(\frac{1}{x}) = -\ln(x)$:

$$U_{\text{ts\_net}}(\xi, h, d') = \frac{d' \cdot e^{\xi \cdot h}}{\xi}(1 - e^{-\xi \cdot h}) - \frac{d'}{\xi}(\xi \cdot h)$$

$$U_{\text{ts\_net}}(\xi, h, d') = \frac{d' \cdot e^{\xi \cdot h}}{\xi} e^{-\xi \cdot h}(e^{\xi \cdot h} - 1) - d' \cdot h$$

Factoring out $\frac{d'}{\xi}$:

$$U_{\text{ts\_net}}(\xi, h, d') = \frac{d'}{\xi}(e^{\xi \cdot h} - 1) - d' \cdot h \tag{B.8}$$

Utility function expressed in terms of ξ, ρ, and h

$$U_{\text{ts\_net}}(\xi, \rho, h) = \rho \left(1 - e^{-\xi h} - \frac{\xi h}{e^{\xi h}}\right) \tag{B.9}$$

This expression simplifies the analysis by allowing us to study how the net utility evolves with h without directly involving the cost parameter d'. The term $1 - e^{-\xi h}$ captures the gross benefit, highlighting how revenue grows with increasing allocation h, subject to diminishing returns governed by ξ. The subtracted term $\frac{\xi h}{e^{\xi h}}$ emerges from the cost component after substituting the optimal allocation condition. Together, the combined expression $1 - e^{-\xi h} - \frac{\xi h}{e^{\xi h}}$ concisely represents the balance between growing revenues and increasing costs. This formulation improves the interpretability of the net utility function and supports a more efficient and stable numerical evaluation.

Utility function expressed in terms of ξ, d', and h

$$U_{\text{ts\_net}}(\xi, h, d') = \frac{d'}{\xi}\left(e^{\xi h} - 1\right) - d'\, h \tag{B.10}$$

This expression provides the net utility per time-slot in terms of the diminishing returns parameter ξ, the allocation h, and the amortized cost d'. The term $\frac{d'}{\xi}\left(e^{\xi h} - 1\right)$ represents the nonlinearly increasing benefit as h grows, while subtracting $d'\, h$ accounts for the linear cost. By eliminating the monetized load ρ, this expression focuses directly on ξ and d', simplifying both sensitivity analyses and numerical optimizations. Overall, this compact representation highlights the core trade-off between rising benefits and increasing costs, serving as a practical tool for theoretical studies and simulation-based evaluations.

Among the three previous expressions, this is generally preferable in terms of numerical stability and conditioning. This is because it directly captures the trade-off between the non-linear benefit component and the linear cost component. The term $\frac{d'}{\xi}\left(e^{\xi h} - 1\right)$ reflects the benefit that grows nonlinearly with h, while the subtraction of $d'\, h$ covers the cost. As long as $U_{\text{ts\_net}}$ is not very close to zero, this formulation avoids cancellation errors that may occur in the first expression and circumvents the sensitivity issues associated with logarithms in the second one. Moreover, it is easier to evaluate numerically over a broad range of $\xi \cdot h$, making it a robust and well-conditioned representation for further analysis.

B.5 Finding ξ_{peak} Max Allocation of the Function $h(\xi)$

Starting with equation 3.19 we can define $h(\xi)$ as the function that, for a fixed value of ρ and d', returns the value of h that maximizes the net utility.

$$h(\xi) = \frac{1}{\xi} \ln\left(\frac{\rho \cdot \xi}{d'}\right)$$

To find the critical points, we first take the derivative of $h(\xi)$ with respect to ξ and set it to zero:

$$h'(\xi) = \frac{d}{d\xi}\left(\frac{1}{\xi} \ln\left(\frac{\rho \cdot \xi}{d'}\right)\right)$$

Using the product rule $(uv)' = u'v + uv'$:

$$u = \frac{1}{\xi}, \quad v = \ln\left(\frac{\rho \cdot \xi}{d'}\right)$$

First, find u' and v':

$$u' = \frac{d}{d\xi}\left(\frac{1}{\xi}\right) = -\frac{1}{\xi^2}$$

$$v' = \frac{d}{d\xi}\left(\ln\left(\frac{\rho \cdot \xi}{d'}\right)\right) = \frac{1}{\xi}$$

Now, apply the product rule:

$$h'(\xi) = u'v + uv' = -\frac{1}{\xi^2} \ln\left(\frac{\rho \cdot \xi}{d'}\right) + \frac{1}{\xi} \cdot \frac{1}{\xi}$$

$$h'(\xi) = -\frac{1}{\xi^2} \ln\left(\frac{\rho \cdot \xi}{d'}\right) + \frac{1}{\xi^2}$$

$$h'(\xi) = \frac{-\ln\left(\frac{\rho \cdot \xi}{d'}\right) + 1}{\xi^2}$$

To find the critical points, we set the derivative equal to zero, which is equivalent to making the numerator equal to zero:

$$-\ln\left(\frac{\rho \cdot \xi}{d'}\right) + 1 = 0$$

Solve for ξ:

$$\ln\left(\frac{\rho \cdot \xi}{d'}\right) = 1$$

$$\frac{\rho \cdot \xi}{d'} = e$$

$$\xi = \frac{d' \cdot e}{\rho}$$

Now, we substitute this critical point back into the original allocation function to find h at $\xi = \frac{d'e}{\rho}$:

$$h = \frac{1}{\frac{d' \cdot e}{\rho}} \ln\left(\frac{\rho \cdot \frac{d' \cdot e}{\rho}}{d'}\right)$$

Simplify the logarithm:

$$h = \frac{\rho}{d' \cdot e} \ln(e)$$

$$h = \frac{\rho}{d' \cdot e}$$

Thus, the maximum allocation at the critical point $\xi = \frac{d'e}{\rho}$ is:

$$h = \frac{\rho}{d' \cdot e}$$

Moreover, at this peak sensitivity one finds

$$\xi_{\text{peak}}\, h(\xi_{\text{peak}}) \;=\; \frac{d'e}{\rho} \times \frac{\rho}{d'e} = 1,$$

so the product $\xi\, h$ equals one exactly at ξ_{peak}.

To find the equation for U_{net} with ξ_{peak}, we substitute $h(\xi_{\text{peak}}) = \frac{\rho}{d' \cdot e}$ into the net utility function:

$$U_{\text{ts\_net}} = \rho \cdot \left(1 - e^{-\xi\, h}\right) - d' \cdot h.$$

First, we substitute $\xi = \frac{d' \cdot e}{\rho}$:

$$e^{-\xi\, h} = e^{-1}.$$

Substitute into the net utility equation:

$$U_{\text{ts\_net}} = \rho \left(1 - \frac{1}{e}\right) - d' \cdot \frac{\rho}{d'\, e} = \rho \frac{e-1}{e} - \frac{\rho}{e} = \frac{\rho(e-2)}{e}.$$

Thus, the net utility at the peak sensitivity is

$$U_{\text{ts\_net\_xi\_peak}} = \frac{\rho(e-2)}{e}.$$

B.6 Analysis of Utility Function Different Expressions

In the following sections, we examine how the net utility for a single time-slot responds to variations in the key parameters used in the different expressions.

Analyzing $U_{\text{net}}(\xi, \rho, h, d')$

$$U_{\text{net}}(\xi, \rho, h, d') = \rho \left(1 - e^{-\xi h}\right) - d'h$$

To understand how U_{net} changes with each variable, we compute the partial derivatives with respect to ξ, ρ, d', and h.

The partial derivative with respect to ξ:

$$\frac{\partial U_{\text{net}}}{\partial \xi} = \rho h e^{-\xi h}.$$

All terms are positive, which ensures that $\frac{\partial U_{\text{net}}}{\partial \xi} > 0$. Thus, U_{net} increases with an increase in ξ.

Partial derivative with respect to ρ:

$$\frac{\partial U_{\text{net}}}{\partial \rho} = 1 - e^{-\xi h}.$$

Since $e^{-\xi h} < 1$ for all positive values of ξ and h, it follows that $\frac{\partial U_{\text{net}}}{\partial \rho} > 0$. Therefore, the net utility increases as ρ increases.

The partial derivative with respect to h is

$$\frac{\partial U_{\text{net}}}{\partial h} = \rho \xi e^{-\xi h} - d'.$$

The sign of this derivative depends on the relationship between $\rho \xi e^{-\xi h}$ and d'. Specifically, $\frac{\partial U_{\text{net}}}{\partial h} > 0$ if $\rho \xi e^{-\xi h} > d'$, and $\frac{\partial U_{\text{net}}}{\partial h} < 0$ if $\rho \xi e^{-\xi h} < d'$. This indicates that U_{net} increases with h when $\rho \xi e^{-\xi h}$ exceeds d' and decreases when it is below.

Partial derivative with respect to d':

$$\frac{\partial U_{\text{net}}}{\partial d'} = -h.$$

Given that $h > 0$, this implies $\frac{\partial U_{\text{net}}}{\partial d'} < 0$. Consequently, U_{net} decreases as d' increases.

In summary, the net utility $U_{\text{net}}(\xi, \rho, h, d')$ behaves as follows: it increases with ρ and ξ, decreases with d', and increases with h up to the optimal value $h = \frac{1}{\xi} \ln\left(\frac{\rho \xi}{d'}\right)$ before decreasing beyond this point.

Analyzing $U_{\text{net}}(\xi, \rho, h)$

$$U_{\text{net}}(\xi, \rho, h) = \rho \left(1 - e^{-\xi h} - \frac{\xi h}{e^{\xi h}}\right)$$

Partial derivative with respect to ρ:

$$\frac{\partial U_{\text{net}}}{\partial \rho} = 1 - e^{-\xi h} - \frac{\xi h}{e^{\xi h}}$$

If we define a new variable $x = \xi h$, where $x > 0$. Substituting x into the expression, we have:

$$1 - e^{-x} - \frac{x}{e^x} = 1 - e^{-x}(1 + x)$$

Consequently, the term $1 - e^{-\xi h} - \frac{\xi h}{e^{\xi h}}$ is positive, but less than 1. This indicates that the net utility U_{net} increases with an increase in ρ. However, the rate of this increase is moderated by the exponential terms, which means that while U_{net} grows as ρ increases, the growth rate decreases due to the diminishing contributions of the exponential components.

Partial derivative with respect to h:

$$\frac{\partial U_{\text{net}}}{\partial h} = \rho \xi^2 h e^{-\xi h} > 0$$

All terms are positive, so U_{net} increases with h.

Partial derivative with respect to ξ:

$$\frac{\partial U_{\text{net}}}{\partial \xi} = \rho \xi h^2 e^{-\xi h} > 0$$

All terms are positive, so U_{net} increases with ξ.

In summary, this expression, $U_{\text{net}}(\xi, \rho, h)$ increases with ρ, h, and ξ.

[

Analyzing $U_{\text{net}}(\xi, \rho, d')$]Analyzing $U_{\text{net}}(\xi, \rho, d')$

$$U_{\text{net}}(\xi, \rho, d') = \rho \left(1 - \frac{d'}{\rho \xi}\right) + d' \cdot \frac{1}{\xi} \ln \left(\frac{d'}{\rho \xi}\right)$$

To examine the behavior of the utility function with respect to ρ, we compute the partial derivative:

$$\frac{\partial U_{\text{net}}}{\partial \rho} = 1 - \frac{d'}{\rho \xi}$$

The condition for the derivative to be positive is:

$$1 - \frac{d'}{\rho \xi} > 0 \implies \rho \xi > d'$$

This inequality shows that the net utility function is marginally supra-linear with respect to ρ when the positive allocation constraint 3.20 is met . In other words, increases in ρ lead to a proportionally greater increase in net utility, indicating that the system benefits more than linearly from an increase in ρ.

Partial derivative with respect to d':

$$\frac{\partial U_{\text{net}}}{\partial d'} = -\frac{\rho}{\rho \xi} + \frac{1}{\xi} \ln \left(\frac{d'}{\rho \xi}\right) + \frac{1}{\xi} = -\frac{1}{\xi} + \frac{1}{\xi} \ln \left(\frac{d'}{\rho \xi}\right) + \frac{1}{\xi} = \frac{1}{\xi} \ln \left(\frac{d'}{\rho \xi}\right) < 0$$

Since $\frac{d'}{\rho \xi} < \frac{1}{e}$, the logarithm is negative. Thus, U_{net} decreases with d'.

The partial derivative with respect to ξ:

$$\frac{\partial U_{\text{net}}}{\partial \xi} = \frac{d'}{\xi^2} - \frac{d'}{\xi^2} \ln \left(\frac{d'}{\rho \xi}\right) - \frac{d'}{\xi^2} = -\frac{d'}{\xi^2} \ln \left(\frac{d'}{\rho \xi}\right) > 0$$

The negative logarithm multiplied by the negative sign yields a positive result. Therefore, U_{net} increases with ξ.

In this expression, $U_{\text{net}}(\xi, \rho, d')$ increases with ρ and ξ, but decreases with d'.

Analyzing $U_{\text{net}}(\xi, d', h)$

$$U_{\text{net}}(\xi, d', h) = \frac{d'}{\xi}(e^{\xi h} - 1) - d' h$$

Partial derivative with respect to d':

$$\frac{\partial U_{\text{net}}}{\partial d'} = \frac{1}{\xi}(e^{\xi h} - 1) - h = \frac{e^{\xi h} - 1 - \xi h}{\xi} > 0$$

Since $e^{\xi h} - 1 - \xi h > 0$ (due to the exponential function growing faster than any polynomial), U_{net} increases with d'.

Partial derivative with respect to h:

$$\frac{\partial U_{\text{net}}}{\partial h} = d' e^{\xi h} - d' = d'(e^{\xi h} - 1) > 0$$

All terms are positive, so U_{net} increases with h.

Partial derivative with respect to ξ:

$$\frac{\partial U_{\text{net}}}{\partial \xi} = d' e^{\xi h} \left(h - \frac{1}{\xi} \right)$$

Given the constraint $\xi h > 1$ (since $\xi h = \ln\left(\frac{\rho \xi}{d'}\right) > 1$), we have $h > \frac{1}{\xi}$. Therefore, $\frac{\partial U_{\text{net}}}{\partial \xi} > 0$, and U_{net} increase with ξ.

In summary, this expression, $U_{\text{net}}(\xi, d', h)$ increases with d', h, and ξ.

B.7 Equating Overestimation and Underestimation

To find the point where the impact of overestimating and underestimating h by ϵ is equal, we set the two previous equations equal:

$$\rho \left(1 - e^{-\xi h(1+\epsilon)}\right) - d' h(1+\epsilon) \;=\; \rho \left(1 - e^{-\xi h(1-\epsilon)}\right) - d' h(1-\epsilon).$$

Expanding both sides:

$$\rho \left(1 - e^{-\xi h} e^{-\xi h \epsilon}\right) - d' h(1+\epsilon) \;=\; \rho \left(1 - e^{-\xi h} e^{\xi h \epsilon}\right) - d' h(1-\epsilon).$$

Rearranging terms leads to:

$$\rho\, e^{-\xi h} \left(e^{\xi h \epsilon} - e^{-\xi h \epsilon}\right) \;=\; 2\, d'\, h\, \epsilon.$$

Using the definition $\sinh(x) = \frac{e^x - e^{-x}}{2}$, we obtain:

$$\rho\, e^{-\xi h} \cdot 2\, \sinh(\xi h \epsilon) \;=\; 2\, d'\, h\, \epsilon \quad \Longrightarrow \quad \rho\, e^{-\xi h}\, \sinh(\xi h \epsilon) \;=\; d'\, h\, \epsilon.$$

Dividing both sides by $e^{-\xi h}\, \sinh(\xi h \epsilon)$ yields the generalized equilibrium expression:

$$\rho_{eq} \;=\; \frac{d'\, h\, \epsilon}{\sinh(\xi\, h\, \epsilon)\, e^{-\xi\, h}}.$$

Notice that if we formally set $\epsilon = 0$, the left-hand side of the starting equation becomes an identity ($\rho(1 - e^{-\xi h}) - d' h = \rho(1 - e^{-\xi h}) - d' h$), so no new information is gained directly from equating. Instead, to recover the same equilibrium value of ρ when $\epsilon = 0$, we turn to the optimal allocation condition:

$$h^* \;=\; \frac{1}{\xi}\, \ln\!\left(\frac{\rho\, \xi}{d'}\right).$$

Solving this for ρ gives

$$\rho \;=\; \frac{d'}{\xi}\, e^{\,\xi\, h^*}.$$

Since $e^{\xi\, h^*} = e^{\xi\, h}$ when $h = h^*$, this matches exactly the limit of (3.37) as $\epsilon \to 0$. In other words, setting $\epsilon = 0$ in the generalized condition is equivalent to using the optimal allocation equation to solve for ρ. Thus the two approaches coincide, confirming consistency without invoking a Taylor expansion.

B.8 Strategy-Proof

In the next subsections, we use both a constructive and an analytical approach to show that the model is not strategy-proof.

B.8.1 Constructive Method Demonstration

We give a complete demonstration and find the expression for ρ_{dec} in terms of ξ_{dec} and h. We start with the optimal allocation, formula 3.19:

$$h = -\frac{1}{\xi} \ln\left(\frac{d'}{\rho\xi}\right)$$

Now we use the optimal allocation solved for ρ_{dec} 3.22

$$\rho_{dec} = \frac{d' \cdot e^{\xi_{dec} \cdot h}}{\xi_{dec}}$$

Substitute it into the declared net utility:

$$U_{\text{net}}^{\text{declared}}(\xi) = \frac{d' \cdot e^{\xi_{dec} \cdot h}}{\xi_{dec}} \cdot \left(1 - \frac{1}{e^{\xi_{dec} h}}\right) - d'h$$

This equation represents the net utility declared in terms of the declared parameter ξ_{dec} as a variable, and the allocation h is a fixed constant with the value we previously calculated.

Next, we use the equation 3.38 and solve it for ρ_{dec}:

$$\rho_{dec} = \frac{U_{\text{net}}^{\text{declared}}(\xi) + d'h}{1 - e^{-\xi_{dec} h}}$$

Now, we substitute the expression for $U_{\text{net}}^{\text{declared}}(\xi)$ in the previous equation. Then we only need to operate to simplify the equation.

$$\rho_{dec} = \frac{\left(\frac{d' \cdot e^{\xi_{dec} \cdot h}}{\xi_{dec}} \cdot \left(1 - \frac{1}{e^{\xi_{dec} h}}\right) - d'h\right) + d'h}{1 - e^{-\xi_{dec} h}}$$

Simplifying the expression, the $-d'h + d'h$ terms cancel out, leaving us with:

$$\rho_{dec} = \frac{\frac{d' \cdot e^{\xi_{dec} \cdot h}}{\xi_{dec}} \cdot \left(1 - \frac{1}{e^{\xi_{dec} h}}\right)}{1 - e^{-\xi_{dec} h}}$$

Next, we factor the $e^{\xi_{dec} h}$ terms in parentheses:

$$\rho_{dec} = \frac{\frac{d' \cdot e^{\xi_{dec} \cdot h}}{\xi_{dec}} \cdot \frac{e^{\xi_{dec} h} - 1}{e^{\xi_{dec} h}}}{1 - e^{-\xi_{dec} h}}$$

Now, cancel out the $e^{\xi_{dec} h}$ terms in the numerator:

$$\rho_{dec} = \frac{\frac{d' \cdot (e^{\xi_{dec} h} - 1)}{\xi_{dec}}}{1 - e^{-\xi_{dec} h}}$$

Finally, notice that $e^{-\xi_{dec} h} = \frac{1}{e^{\xi_{dec} h}}$, so we can express the denominator $1 - e^{-\xi_{dec} h}$ in terms of $e^{\xi_{dec} h}$:

$$\rho_{dec} = \frac{d' \cdot (e^{\xi_{dec} h} - 1)}{\xi_{dec} \cdot (1 - e^{-\xi_{dec} h})}$$

The denominator becomes:

$$\xi_{dec} \cdot (1 - e^{-\xi_{dec} h}) = \xi_{dec} \cdot \frac{e^{\xi_{dec} h} - 1}{e^{\xi_{dec} h}} = \frac{\xi_{dec}(e^{\xi_{dec} h} - 1)}{e^{\xi_{dec} h}}$$

Now, substitute the simplified denominator back into the original expression:

$$\rho_{dec} = \frac{d' \cdot (e^{\xi_{dec} h} - 1)}{\frac{\xi_{dec}(e^{\xi_{dec} h} - 1)}{e^{\xi_{dec} h}}}$$

The terms $e^{\xi_{dec} h} - 1$ in the numerator and denominator cancel out (assuming $e^{\xi_{dec} h} \neq 1$):

$$\rho_{dec} = \frac{d'}{\frac{\xi_{dec}}{e^{\xi_{dec} h}}}$$

Dividing by a fraction is the same as multiplying by its reciprocal:

$$\rho_{dec} = d' \frac{e^{\xi_{dec} h}}{\xi_{dec}}$$

So the simplified expression for ρ_{dec} is:

$$\rho_{dec} = \frac{d' e^{\xi_{dec} h}}{\xi_{dec}}$$

B.8.2 Analytical Method Demonstration

To demonstrate that the net utility can decrease while maintaining the optimal allocation, we will use the utility equation $U_{\text{net}}(\xi, h, d')$ B.9. Our next step is to show that the derivative of this equation is never zero, which we will now calculate:

$$\frac{dU_{\text{net}}}{d\xi} = \frac{(d' \cdot h \cdot e^{\xi \cdot h}) \cdot \xi - d' \cdot e^{\xi \cdot h} + d'}{\xi^2}$$

Simplifying further:

$$\frac{dU_{\text{net}}}{d\xi} = \frac{d' \cdot (h \cdot e^{\xi \cdot h} \cdot \xi - e^{\xi \cdot h} + 1)}{\xi^2} \tag{B.11}$$

This expression provides the rate of change of U_{net} with respect to ξ. If there is no value of ξ that would make this expression equal to zero, this implies that there is a direction in which, by modifying ξ, the net utility, U_{net} decreases, indicating that the game is not strategy-proof.

To analyze whether the previous expression can be equal to zero, we take the expression in parentheses to define the auxiliary function $f(\xi)$ and make it equal to zero:

$$f(\xi) = h \cdot e^{\xi \cdot h} \cdot \xi - e^{\xi \cdot h} + 1 = 0$$

Factoring out $e^{\xi \cdot h}$, we obtain:

$$e^{\xi \cdot h}(h \cdot \xi - 1) + 1 = 0$$

This rearranges to:

$$e^{\xi \cdot h} = \frac{-1}{h \cdot \xi - 1}$$

The denominator must be negative, implying we should add the following condition to our model

$$h \cdot \xi - 1 < 0 \implies \xi < \frac{1}{h}$$

We can see that this condition doesn't hold in our model and contradicts what we have previously seen. To complete the demonstration, we calculate the derivative of the function $f(\xi)$:

$$\begin{aligned} f'(\xi) &= h^2 \cdot e^{\xi \cdot h} \cdot \xi + h \cdot e^{\xi \cdot h} - h \cdot e^{\xi \cdot h} \\ f'(\xi) &= h^2 \cdot e^{\xi \cdot h} \cdot \xi \end{aligned} \tag{B.12}$$

Given that $\frac{dU_{\text{net}}}{d\xi} > 0$ for all $\xi > 0$, the function $f(\xi)$ and in consequence the net utility U_{net} increases with ξ when h is kept constant. This means that by decreasing ξ, a Service Provider can reduce U_{net} while maintaining the same allocation h. Even with the restriction $\xi > \xi_{\text{peak}} = \frac{d'e}{\rho}$, there is room to adjust ξ downward (but still above ξ_{peak}) to achieve a lower net utility.

B.8.3 Analysis of Modified Utility Function Using Derivatives

- With respect to β:

$$\frac{\partial U}{\partial \beta} = l\left(1 - \frac{d'}{\beta\xi}\right)$$

- With respect to l:

$$\frac{\partial U}{\partial l} = \beta\left(1 - \frac{d'}{\beta\xi}\right) - \frac{d'}{\xi}\ln\left(\frac{\beta\xi}{d'}\right)$$

- With respect to ξ:

$$\frac{\partial U}{\partial \xi} = -\frac{d'l}{\xi^2}\ln\left(\frac{\beta\xi}{d'}\right)$$

- With respect to d':

$$\frac{\partial U}{\partial d'} = -\frac{l}{\xi}\ln\left(\frac{\beta\xi}{d'}\right)$$

B.8.4 Analysis of Modified Optimal Allocation Using Derivatives

- With respect to β:

$$\frac{\partial h^*}{\partial \beta} = \frac{l}{\beta\xi} > 0$$

- With respect to l:
$$\frac{\partial h^*}{\partial l} = \frac{1}{\xi} \ln\left(\frac{\beta\xi}{d'}\right) > 0$$
- With respect to ξ:
$$\frac{\partial h^*}{\partial \xi} = -\frac{l}{\xi^2} \ln\left(\frac{\beta\xi}{d'}\right) + \frac{l}{\xi^2}$$
(This derivative is positive until ξ_{peak} and negative after it.)
- With respect to d':
$$\frac{\partial h^*}{\partial d'} = -\frac{l}{\xi d'} < 0$$

ξ_{peak} Parameter Modified Derivation

Consider the optimal allocation:
$$h^*(\xi) = \frac{l}{\xi} \ln\left(\frac{\beta\,\xi}{d'}\right)$$

To find the optimal sensitivity parameter ξ_{peak}, we set the first-order derivative equal to zero:
$$\frac{\partial h^*}{\partial \xi} = 0$$

First-order condition

We compute the derivative:
$$\frac{\partial h^*}{\partial \xi} = -\frac{l}{\xi^2} \ln\left(\frac{\beta\,\xi}{d'}\right) + \frac{l}{\xi^2} = 0$$

Multiplying by $\frac{\xi^2}{l}$:
$$-\ln\left(\frac{\beta\,\xi}{d'}\right) + 1 = 0$$

Solve for ξ:
$$\ln\left(\frac{\beta\,\xi}{d'}\right) = 1$$

Exponentiating both sides gives:
$$\frac{\beta\,\xi}{d'} = e$$

Thus, we obtain the optimal ξ, denoted by ξ_{peak}:
$$\xi_{\text{peak}} = \frac{e\,d'}{\beta}$$

Allocation at ξ_{peak}

Evaluating h^* at ξ_{peak}:
$$h^*(\xi_{\text{peak}}) = \frac{l}{\frac{e\,d'}{\beta}} \ln\left(\frac{\beta \cdot \frac{e\,d'}{\beta}}{d'}\right) = \frac{l\,\beta}{e\,d'} \ln(e)$$

Since $\ln(e) = 1$, we have:
$$h^*(\xi_{\text{peak}}) = \frac{l\,\beta}{e\,d'}$$

Utility at ξ_{peak}

The original utility function is given by:

$$U = l\,\beta\left(1 - e^{-\frac{\xi\,h}{l}}\right) - d'h$$

Evaluating at ξ_{peak} and $h^*(\xi_{\text{peak}})$:

First, simplify the exponent:

$$\frac{\xi_{\text{peak}}\,h^*(\xi_{\text{peak}})}{l} = \frac{\frac{e\,d'}{\beta}\cdot\frac{l\,\beta}{e\,d'}}{l} = 1$$

Thus, the utility simplifies to:

$$U(\xi_{\text{peak}}) = l\,\beta\left(1 - e^{-1}\right) - d'\cdot\frac{l\,\beta}{e\,d'}$$

Simplifying further:

$$U(\xi_{\text{peak}}) = l\,\beta\left(1 - \frac{1}{e}\right) - \frac{l\,\beta}{e} = l\,\beta\left(1 - \frac{2}{e}\right)$$

Therefore, the utility at the optimal point is:

$$U(\xi_{\text{peak}}) = l\,\beta\left(1 - \frac{2}{e}\right)$$

B.8.5 Allocation is Time-dependent When ξ is Time-dependent

If we add the $_{\Delta t}$ suffix for the variables dependent on time-slot length and consider that all variables scale proportionally to it. By recalling the h^* equation

$$h^* = \frac{1}{\xi_{\Delta t}}\ln\Big(\frac{l_{\Delta t}\,\beta\,\xi_{\Delta t}}{d'_{\Delta t}}\Big).$$

We suppose each "per–slot" parameter scales linearly with Δt:

$$l_{\Delta t} \propto \Delta t, \quad \xi_{\Delta t} \propto \Delta t, \quad d'_{\Delta t} \propto \Delta t.$$

Then inside the logarithm

$$\frac{l_{\Delta t}\,\beta\,\xi_{\Delta t}}{d'_{\Delta t}} \;=\; \frac{(\Delta t)(\Delta t)}{\Delta t}\,C \;=\; \Delta t\,C,$$

where C is independent of Δt. Hence

$$h^* = \frac{1}{\xi_{\Delta t}}\ln(\Delta t\,C) = \frac{1}{\xi_{\Delta t}}\big[\ln C + \ln(\Delta t)\big] = \frac{\ln C}{\xi_{\Delta t}} + \frac{\ln(\Delta t)}{\xi_{\Delta t}}.$$

The term $\dfrac{\ln(\Delta t)}{\xi_{\Delta t}}$ cannot cancel out unless $\ln(\Delta t) = 0$ (i.e. $\Delta t = 1$). Thus Δt cannot be eliminated from the per–slot expression for h^*.

Demonstration of the Exponential Term at h^*

Starting from

$$U \;=\; \rho\Big(1 - e^{-\,h\,\frac{d'}{\rho}\,\lambda}\Big) \;-\; d'\,h,$$

and using

$$h^* \;=\; \frac{\rho}{d'\,\lambda}\,\ln(\lambda),$$

we compute the exponent:

$$-\,h^*\,\frac{d'}{\rho}\,\lambda = \;-\Big(\frac{\rho}{d'\,\lambda}\,\ln(\lambda)\Big)\,\frac{d'}{\rho}\,\lambda = \;-\,\ln(\lambda)\,\big(\tfrac{\rho}{d'\,\lambda}\cdot\tfrac{d'}{\rho}\cdot\lambda\big) = \;-\,\ln(\lambda).$$

$$e^{-\,h^*\,\frac{d'}{\rho}\,\lambda} \;=\; e^{-\,\ln(\lambda)} = \;\frac{1}{\lambda},$$

$$1 - e^{-\,h^*\,\frac{d'}{\rho}\,\lambda} \;=\; 1 - \frac{1}{\lambda}.$$

Appendix C

Further Considerations and Real-World Implications

This annex presents additional analyses that complement our theoretical and numerical results. The discussions included here are placed outside the main body of the thesis because they address secondary aspects, involving potential implementations and real-world considerations that exceed the primary scope of this research. Specifically, we explore general properties of the independent contribution scenario, consider the introduction of the $\xi > \xi_{\text{peak}}$ constraint for improved economic efficiency and stability, examine practical edge computing scenarios that exhibit sublinear resource scaling, and outline monitoring techniques for accurately tracking per-tenant request rates. These discussions bridge the gap between theoretical findings and practical applications, highlighting how different assumptions and constraints impact outcomes in realistic environments.

C.1 Some More General Observations Regarding the Independent Contribution

To complete the first part of the model's analytical investigation, we review several general theoretical consequences of the independent contribution property and discuss their impact on the Edge Computing model.

- Strong Stability: This property guarantees that the payoff assigned to any coalition exactly equals the coalition's total value, thereby preventing any subgroup from having an incentive to deviate. Although this creates a stable structure by discouraging defection, it also limits the motivation of existing SPs to recruit new members, since their individual payoffs remain unchanged with the addition of new participants. This motivation only holds for the NO.
- Core Non-Emptiness: A common challenge in cooperative game theory is to ensure that the core is not empty. When the independent contribution property is satisfied, it often guarantees a non-empty core. This occurs because matching a coalition's value with the sum of its members' payoffs naturally satisfies the condition of coalitional rationality for all coalitions, including the grand coalition. In the Edge Computing model, while the existence of the core has been established in the referenced article, we have shown that, as a result of the independent contribution property, the core is reduced to a unique point.
- Fairness and equity: This property means that each player's contribution is clearly recognized and fairly rewarded. In general, this creates a sense of fairness among participants. However, in some cases, the use of the Shapley value and the dependence on other players' utility functions could seem unclear or even unfair. In our model, this concern is reduced because the independent contribution property ensures that each Service Provider's payoff depends only on its own

parameters. This avoids incentives for strategic manipulation or bargaining between SPs, since no SP can influence the payoff of another. The Network Owner, however, can still be affected by inaccurate reports from SPs. We will explore this in more detail in section 3.13.

- Simplified Negotiation: When the independent contribution property holds, surplus or deficit divisions are transparently linked to each player's independent contribution, thereby simplifying negotiations. In our case, there is no surplus or deficit to negotiate, effectively eliminating any scope for disputes over payoff divisions.
- Multiple Participation for Diversified Services: A Service Provider offering different services can participate in the coinvestment as if it were multiple distinct SPs. This approach allows for a clearer identification of the revenues and payoffs associated with each service, enhancing transparency and facilitating the assessment of each service's financial performance.

Considering Adding the $\xi > \xi_{peak}$ Constraint

Although adding this constraint is not mandatory in our model, it could bring several advantages:

- Economic Efficiency: When $\xi \leq \xi_{\text{peak}}$, small changes in ξ can cause large increases in allocation, leading to inefficient overconsumption of resources. The enforcement of $\xi > \xi_{\text{peak}}$ ensures a more stable and controlled allocation behavior.
- Predictability: Staying above ξ_{peak} leads to smoother and more manageable allocation changes, reducing the risks associated with estimation errors in the parameters of the utility function.
- Strategic Robustness: In the region $\xi \leq \xi_{\text{peak}}$, the high marginal utility of additional resources may incentivize SPs to underreport parameters in order to secure larger allocations. Keeping ξ above the peak mitigates this risk.
- Feasibility and Long-term Sustainability: When $\xi > \xi_{\text{peak}}$, the system operates in a more stable regime where the marginal gain from additional allocation naturally tapers off, aligning costs with true economic value.
- Numerical and Computational Stability: Operating in the region $\xi > \xi_{\text{peak}}$ leads to smoother gradients and more stable behavior in the allocation function. This improves simulation robustness.

Rather than setting this threshold as a strict rule, a more robust approach may be to define a small buffer:

$$\xi \geq \xi_{\text{peak}} + \epsilon = \frac{d'e}{\rho} + \epsilon$$

where $\epsilon > 0$ is a small constant. This ensures that allocation remains stable even when parameters drift slightly and avoids configurations where it sits too close to an unstable maximum.

Although the allocation decreases beyond ξ_{peak}, the corresponding net utility function continues to increase slowly and becomes more stable. This is because:

- As $\xi \to \infty$, $h(\xi) \to 0$
- The term $e^{-\xi h} \to 0$, so the gross utility approaches ρ
- The cost term $d'h \to 0$ even faster.

This implies that for $\xi > \xi_{\text{peak}}$ net utility enters a regime of diminishing sensitivity and greater predictability.

C.2 Edge Computing Services with Sublinear Resource Scaling

Although using a logarithmic relationship between monetized load (ρ) and resource allocation offers analytical simplicity, it may underestimate hardware requirements in real-world edge computing. As request volumes grow, demands from load balancing, redundancy, high availability, and latency guarantees typically cause resource usage to increase at least linearly.

The monetized load is defined as $\rho = \beta \cdot l_{\text{avg}}$, where β represents the effectiveness of each request in generating revenue, and l_{avg} is the average number of requests per time-slot. While in the allocation function modeling β logarithmically is reasonable due to diminishing returns, applying a logarithmic model to l_{avg} does not align well with practical observations. Typically, higher request loads necessitate proportional or greater hardware expansion.

Nonetheless, there are scenarios in edge computing where resource usage scales sublinearly with increasing load. Such cases occur when computational tasks or data-serving efforts can be amortized across multiple requests via caching, sharing, or coordinated mechanisms. Below, we highlight practical service examples where sublinear scaling occurs:

- Content Delivery Networks (CDNs):CDNs commonly exhibit sublinear scaling due to effective caching at edge nodes. Frequent requests are served locally, significantly reducing upstream traffic. This efficiency is facilitated by Zipf-like content popularity distributions, where the frequency of requests for a given item is inversely proportional to its popularity rank, meaning a small set of items accounts for most requests. Once popular content is cached, additional requests yield minimal upstream load increases. For instance, Vanerio (Juan Vanerio, 2024) demonstrates that edge caching can reduce upstream traffic by 44% relative to a baseline, with routing overhead growing sublinearly with request rates.

- Real-time video streaming: Edge computing supports sublinear resource scaling through request coalescing, combining simultaneous requests for identical content segments. Platforms like Facebook utilize cache sharding and multicast delivery to serve millions of live viewers without proportional growth in network and server resources. Coalescing techniques significantly reduce redundant upstream fetches, maintaining nearly constant upstream load despite increasing viewer numbers (Khatri, Lambert, Cenzano, & Broilo, 2020).

- Augmented and Virtual Reality (AR/VR): Real-time AR/VR applications benefit from edge computing by reusing computational tasks among users sharing the same environment. Systems like MUVR cache rendered frames and intermediate results, enabling the reuse of common components such as background scenes. Experiments report over 90% savings in computation and a 95% bandwidth reduction compared to individual rendering per user (Li & Gao, 2018), demonstrating significant sublinear scaling in computational and network resources as user numbers increase.

- Real-time Internet of Things (IoT): Edge nodes aggregate, filter, or summarize sensor data before upstream transmission, resulting in sublinear growth in network load relative to sensor count. Nguyen et al. (Nguyen & Huh, 2021) propose a two-tier edge-cloud architecture where clustered edge brokers manage local traffic and directly exchange summarized data, reducing inter-broker relay traffic to approximately 7.77% compared to non-clustered systems.

These examples illustrate how edge computing achieves sublinear resource scaling by leveraging mechanisms such as caching, task reuse, and data aggregation. By avoiding redundant computations and transmissions—such as repeated rendering, cached video segments, or aggregated sensor data—resource demands can scale sublinearly or even remain nearly constant as load increases. While locality primarily helps reduce latency and backbone congestion, combining it with reuse strategies is critical for achieving efficient scaling. Empirical evidence from the discussed examples underscores the feasibility and practical significance of sublinear resource growth in edge computing.

C.3 Monitoring Per-Tenant Request Rates

Monitoring the request rates of tenants (individual users or organizations using shared infrastructure) is critical for efficient resource allocation. Several techniques are available, ranging from low-level system metrics to high-level application instrumentation.

- Hypervisor Metrics: Hypervisors such as KVM or VMware ESXi track basic statistics like network packets and bytes transmitted by each virtual machine. Management tools like libvirt (a library for virtual machine management) or VMware's vSphere API allow periodic retrieval of these counters. By measuring how these counters change over time, the request rate for each tenant can be estimated.
- Virtual Switch Monitoring: Virtual switches, such as Open vSwitch, connect virtual machines to networks. These can use protocols like sFlow or IPFIX (standard methods for sampling network flows) to monitor network traffic. Each captured flow (a record of packets traveling through the network) indicates traffic associated with specific virtual machines. Aggregating these records provides an estimate of per-tenant request rates.
- Network Fabric Monitoring: Physical network devices often support standard flow monitoring protocols such as NetFlow or sFlow. When tenants' network traffic is isolated using mechanisms like VLAN (Virtual LAN), VXLAN (Virtual Extensible LAN), or VRF (Virtual Routing and Forwarding), traffic flows can be separated and analyzed. Aggregating these flows yields the request rates per tenant.
- In-Guest Instrumentation: Another method involves installing lightweight software agents directly inside the virtual machine or container. These agents use technologies such as eBPF (extended Berkeley Packet Filter, which efficiently observes kernel-level events) or OpenTelemetry (a toolkit for application-level telemetry). These tools precisely track requests at the application or network socket level, providing accurate measurements of request counts and rates.
- Cloud and Container Orchestration APIs: Public cloud services and container orchestration platforms often provide built-in monitoring. Services like AWS CloudWatch, Azure Monitor, or Kubernetes metrics-server expose standardized metrics related to network and application activity. These metrics can be easily associated with individual tenants or application instances, facilitating straightforward measurement of per-tenant request rates.

Appendix D

Edge Computing Simulations Manual

D.1 Introduction

The software (Tabarez, 2025) was developed primarily to simulate the model proposed in the referenced academic article, along with the three alternative formulations introduced in Section 5.4. Additionally, it extends and illustrates the underlying mathematical theory presented in that work. To capture more realistic scenarios and improve functionality, the software also incorporates additional considerations discussed in the previous sections of this thesis.

This software defines a "game" as a single simulation run and a "simulation" as a set of games. A simulation defines a collection of games, and this set of games can be updated or extended by reusing the same simulation name in subsequent executions. The software verifies that the simulation name and the amount and names of SPs match those already present in the database to correctly merge the results. Additionally, the software supports simulations with either independent or interdependent SP contributions. In the case of interdependent contributions, the software allows estimations to be performed using deterministic or non-deterministic approaches.

The simulator imports data from a YAML file containing general game parameters, such as the investment duration (in years), the number of time slots, the price per CPU unit, and the maximum allowed CPUs. Additionally, the file includes data for each SP, specifying their load and utility functions.

The load function describes the SP's demand profile, modeled by a sinusoidal function as detailed in article (A. P. Vela A. Vía & Velasco, 2016). This function is parameterized by the average load and a pair of hyperparameters a_k and t_k, representing the amplitude (in requests) and the time offset (in seconds), respectively. Additionally, the YAML input file can optionally specify a standard deviation for the load, enabling the simulation of stochastic dynamics within the game.

Alternatively, the software provides a visual tool for manually defining different daily load profiles. This functionality is implemented in the script `create_load_function.py`, located in the `utils` folder, and uses the interactive visualization capabilities of the "Bokeh" library. Upon executing this script, a browser window opens, allowing the user to define any desired load profile by interactively dragging points corresponding to specific loads at different time slots. Each time the user submits the modified points, the tool automatically computes the corresponding sinusoidal hyperparameters and the average load. The resulting parameters can then be copied directly in YAML format into the simulator's setup file. This interactive approach simplifies the definition of load profiles, avoiding manual computation or explicit specification of sinusoidal hyperparameters.

The utility function defined for each SP takes as inputs the load function, the benefit factor (β), and the diminishing return parameter (ξ). To facilitate switching between different utility function cases without having to recalibrate the diminishing return parameter, as we previously did in section 6.2,

the software alternatively allows defining this parameter through the percentage of load served at the edge.

The input variables, including both generic game parameters and SP's specific data, have been designed to facilitate straightforward execution of multiple sensitivity analyses and customized scenarios. Multiple YAML files can be processed simultaneously by placing them within the folder `Simulations to process`, enabling the evaluation of diverse scenarios through a single execution.

The data generated from the simulations is stored in an SQL database, integrated with Metabase (Inc., 2024), a visualization platform that enables the creation of various charts and supports direct execution of SQL queries. Metabase also includes a querying interface that suggests relevant columns for joins based on foreign keys, significantly simplifying and streamlining the querying process. This integration offers users an effective tool for analyzing and interpreting the outcomes of simulations.

D.2 Installation

In the following subsections, we provide step-by-step instructions for installation and explain how to use the Vagrant file included in the project. Both approaches were tested using Ubuntu 22.04, but the simulator and its dependencies should be straightforward to install on any operating system.

D.2.1 Installation Steps

Assuming a clean virtual or local machine, first update the available system packages by running:

```
sudo apt-get update
```

Then we install all the needed operating system dependencies:

```
sudo apt-get install -y \
  software-properties-common \
  build-essential \
  zlib1g-dev \
  libssl-dev \
  libffi-dev \
  curl \
  libbz2-dev \
  libsqlite3-dev \
  liblzma-dev \
  mysql-server \
  libmysqlclient-dev \
  pkg-config \
  openjdk-11-jdk
```

Although any Python 3 version should be compatible, here we show how to install the exact version used during development directly from the source:

```
if ! command -v python3.12; then
  curl -O https://www.python.org/ftp/python/3.12.1/Python-3.12.1.tgz
  tar -xf Python-3.12.1.tgz
  cd Python-3.12.1
  ./configure --enable-optimizations
  make -j $(nproc)
  sudo make altinstall
```

```
  cd ..
  rm -rf Python-3.12.1
  rm Python-3.12.1.tgz
fi
```

Next, we install the Python package manager (`pip`):

```
curl -sS https://bootstrap.pypa.io/get-pip.py | sudo python3.12
```

Start and enable the MySQL service:

```
sudo systemctl start mysql
sudo systemctl enable mysql
```

Create Database and MySQL user:

```
sudo mysql -e "CREATE DATABASE edge_computing;"
sudo mysql -e "CREATE USER 'admin'@'%' IDENTIFIED BY 'admin';"
sudo mysql -e "GRANT ALL PRIVILEGES ON edge_computing.* TO 'admin'@'%';"
sudo mysql -e "FLUSH PRIVILEGES;"
```

Now we proceed to install git if necessary, clone the project repository, and navigate to its main file:

```
sudo apt install git
git clone https://github.com/Santiago-Tabarez/edge-computing-simulations
cd edge-computing-simulations
```

Create a virtual environment for the project, activate it, install the required dependencies, and update the Python path (replace `path_to_project` with your project's actual path):

```
python3.12 -m venv \~/env
source ~/env/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
export PYTHONPATH=/path_to_project/edge-computing-simulations
```

At this point, the simulator has been installed. To verify the installation, execute:

```
python3 main/main.py
```

To also install Metabase, first download it using the following command:

```
curl -Lo \~/metabase.jar https://downloads.metabase.com/v0.44.6/metabase.jar
```

Next, create the configuration file for the Metabase system service:

```
echo "[Unit]
Description=Metabase
After=syslog.target
After=network.target

[Service]
```

```
User=$USER
ExecStart=/usr/bin/java -jar /home/$USER/metabase.jar
Restart=always
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=metabase

[Install]
WantedBy=multi-user.target" | sudo tee /etc/systemd/system/metabase.service
```

Reload `systemd`, then enable and start the Metabase service:

```
sudo systemctl daemon-reload
sudo systemctl enable metabase
sudo systemctl start metabase
```

To verify if Metabase is correctly installed, open a browser and go to localhost:3000.

D.2.2 Vagrant Configuration

If the user or developer prefers using Vagrant for deployment, a Vagrant file has been provided in the main folder of the project repository.

After installing Vagrant and VirtualBox, follow these steps to set up and run the simulator inside the virtual machine:

First, open a terminal window (on Windows or Unix), navigate to the project's main folder containing the Vagrant file, and create the virtual machine by running:

```
vagrant up
```

Once the setup process is complete, connect to the virtual machine with:

```
vagrant ssh
```

Inside the virtual machine's console, activate the Python virtual environment:

```
source /home/vagrant/env/bin/activate
```

Navigate to the project's main directory:

```
cd /home/vagrant/edge-computing-simulation/main
```

Finally, execute the simulator:

```
python3 main.py
```

The Metabase service can be accessed through the web browser by navigating to `localhost:3000`.

D.2.3 Dependencies

All dependencies are listed in the `requirements.txt` file and include the following:

- `psutil`: Used to monitor system resources such as CPU and memory usage during simulations, specifically to identify whether a given simulation is CPU-intensive, memory-intensive, or both.
- `mysql`: Manages the MySQL database that stores simulation results and configurations.
- `mysql-connector-python`: Provides the necessary tools for connecting to the MySQL database, executing queries, and managing database transactions.
- `scipy`: Offers advanced mathematical functions and algorithms essential to the simulations, particularly for optimizing coalition values.
- `numpy`: Facilitates numerical operations, efficient array management, and mathematical computations.
- `PyYAML`: Handles parsing and generation of YAML configuration files used as inputs for the simulations.

While the previous dependencies are necessary to run simulations, the following packages are optional. They have been included to facilitate the creation of static and interactive charts for reproducing the results presented in this thesis, as well as for manually defining daily load patterns.

- `plotly`: Used for creating interactive charts.
- `matplotlib`: Used for generating static charts.
- `bokeh`: Provides interactive visualizations, particularly used for the manual definition and fitting of daily load profiles.

D.3 Configuration

D.3.1 Logging Configuration

The simulator supports three logging levels to accommodate varying detail requirements during simulation monitoring:

- `ERROR`: Records only critical errors that affect the validity of simulation results and halt the execution.
- `INFO`: Captures intermediate and final simulation results, such as resource allocations, payoffs, Shapley values, as well as payments and revenues.
- `DEBUG`: Provides comprehensive and detailed information, including the type of value function used, duration of each game and the overall simulation, and system resource usage. Additionally, in dynamic allocation scenarios (allocation per time slot), it logs the distribution of allocations across the different time slots.

D.3.2 Database Configuration

In addition to the standard database connection settings such as host, port, username, password, and database name, this software provides a special database management configuration under the `DATABASE_MANAGEMENT_CONFIG` section. This configuration manages the execution of three SQL scripts located in the `sql scripts` folder. Each script can be individually activated by setting its corresponding variable to `True`:

- `truncate`: Erases all existing data while preserving the database structure when set to `True`.
- `drop`: Completely removes the database when set to `True`. This is useful when the database and its data are no longer needed, or before running the `create` script to accommodate structural changes.
- `create`: Creates the database structure from scratch when set to `True`. It assumes no existing database with the same name and will produce an error if such a database already exists.

Simulations will not execute until all three configuration variables are set to `False`. Only one of these scripts should be activated at any given time; activating more than one simultaneously will result in an error displayed in the console. Developers should update these scripts whenever modifications to the database structure are made.

This software has been developed and tested with MySQL 8.4, but it should be compatible with other SQL databases with minimal or no adjustments required.

D.3.3 Optimization Parameters for TRUST-CONSTR Method

The simulator uses the `trust-constr` optimization method because the utility function employed is nonlinear, smooth, and has analytically computable derivatives, Jacobian, and Hessian. This method leverages these analytical properties, offering efficient convergence and robust constraint handling for nonlinear optimization problems.

The following optimization parameters can be configured specifically for the `trust-constr` method:

- `GTOL`: The gradient tolerance, defining the stopping criterion based on the projected gradient norm.
- `XTOL`: The step tolerance, controlling convergence by monitoring the relative changes in decision variables.
- `BARRIER_TOL`: The barrier tolerance, governing the accuracy in constraint satisfaction and convergence of barrier subproblems.
- `JTOL`: The Jacobian tolerance applied to constraints, ensuring constraint violations remain within acceptable bounds.
- `MAXITER`: The maximum number of iterations permitted during optimization to prevent infinite loops or excessive computation times.

By carefully tuning these parameters, users can adjust the trade-off between computational precision and execution speed, enhancing one at the expense of the other according to their specific needs.

D.3.4 Save Function

The `SAVE_FUNCTION` configuration section allows users to specify whether the load function, the utility function, or both should be stored in the database, recording their values at each time-slot. This functionality is particularly beneficial when randomness is introduced into the load, as simulation outcomes will vary between executions and therefore cannot be exactly reproduced.

D.3.5 Simulation Mode and Extra Considerations

The following subsections are closely interrelated; they define the computational methods for executing simulations and additional considerations influencing their execution.

Value Function Mode

The `VALUE_FUNCTION_MODE` in the configuration section determines how the coalition values are computed, directly influencing how the games are simulated. There are three available options; exactly one must be set to `True` at any given time.

It is important to note that the terms additive and non-additive used here correspond respectively to the current notation of independent and interdependent contributions of SPs.

- `additive`: This mode defines the value function as additive, implying that contributions from players to a coalition are independent. Consequently, only the grand coalition's value is computed, and the load curves of SPs are not considered since the same results are obtainable using the average load. Simulations in this mode run in linear time, making it efficient and scalable for scenarios involving hundreds or thousands of players. However, this mode does not capture externalities or synergies within coalitions. Although running additive simulations with synergy or externality options is permissible (and will trigger a warning), this mode primarily serves as a rapid and straightforward estimation tool rather than providing precise results. Its outcomes represent a worst-case scenario for SP utility.
- `non_additive_deterministic`: This mode treats the value function as non-additive (interdependent), meaning individual contributions cannot be isolated due to the presence of synergies or externalities. Since the values of all possible coalitions must be computed in order to calculate each player's Shapley value, simulations are computationally intensive and do not scale well beyond a small number of players.
- `non_additive_estimation`: This mode also treats the value function as non-additive (interdependent) but utilizes Monte Carlo methods to estimate coalition values more efficiently. This efficiency, however, sacrifices determinism, resulting in probabilistic rather than fixed outcomes. This method is suitable for scenarios prioritizing computational speed and scalability over exact precision.

Both non-additive modes were specifically developed to incorporate externalities or synergies, as detailed in the subsequent section. In both non-additive scenarios, the grand coalition value and the resulting optimal allocations remain the same. However, calculating the Shapley value, which requires evaluating all feasible sub-coalitions, is computationally intensive, as the number of sub-coalitions grows exponentially with the number of SPs. To address this computational challenge, the `non_additive_estimation` mode estimates the Shapley value using Monte Carlo methods combined with a caching strategy to avoid recalculating coalition values that have already been computed. Running these modes without activating externality or synergy options is permissible (though it triggers a warning message), primarily serving to verify correctness (in the deterministic case) or accuracy (in the estimation case) relative to expected results.

Extra Considerations

The "EXTRA_CONSIDERATIONS" determines if the value function can be considered as additive; if at least one is set to True, then the non_additive modes should be used for accurate results. They are called extra considerations since they were not present in the original model and were introduced to break the independent contribution property and, in consequence, introduce complexity into the model.

- per_time_slot_allocation: This consideration allows players to have different allocations per time-slot. Considering that we are running an optimization problem (optimizing SPs' allocations for each time-slot) inside another optimization problem (optimizing SPs' allocations for all time-slots), the simulation will be computing intensive. Since this dynamic allocation acts as a synergy between players, the outcomes should always be at least as beneficial for players as they are for a fixed allocation.

- `variable_cpu_price`: This consideration makes the per-unit CPU price sublinear. The price is defined using a maximum per-unit price for the minimum allowed total allocation and a minimum per-unit price for the maximum allowed total allocation. This setup is typically considered a positive externality, as all players benefit when new players join the coalition. However, to ensure that this mechanism acts as a positive externality, the minimum allocation should be set either to zero or at most equal to the smallest individual allocation among service providers, while the maximum allocation should be at least the sum of all individual allocations. If intermediate values are selected instead, the pricing can inadvertently create negative externalities. Under these conditions, if the resulting per-unit price from the additive value function mode is lower than the fixed price, results incorporating this consideration will always be at least as favorable as those obtained using the fixed CPU price.

Monte Carlo Variable

When using the non_additive_estimation mode, in this section, the amount of samples to calculate the Shapley value for each Service Provider should be set in this field. This number defines the amount of random coalitions without the current SP, that will be used to approximate the Shapley value. To equally account for coalitions with and without the NO (that would produce zero value). The implementation only considers the subgroups with the NO in it, then divides the result by two. It also uses a cache to avoid calculating the same coalition's value twice.

It is up to the user to define a reasonable number of samples, depending on the characteristics of the game.

D.4 Input Files

The `simulations to process` folder must contain one or more YAML files, each specifying the details of a simulation. Each of these files defines a collection of one or more games, grouped under the same simulation identifier.

Originally, the design involved reading the files from this folder, executing the corresponding simulations, and subsequently moving them to a different folder to systematically manage multiple simulation executions. However, to simplify the workflow and avoid repeatedly copying files during testing phases, when users frequently execute brief simulations rather than extensive ones, the files are now kept in the same folder. This approach prevents users from needing to move or copy files back for each execution.

In the following subsections, we describe each input parameter contained in these YAML files, specifying their format and behavior when defining multiple values for a single parameter.

D.4.1 Simulation name

This value is defined through the input data field, called "simulation_name". For this field, only one value is accepted in each simulation file, and it identifies a set of games. If this name is used multiple times, there are three distinct cases to consider:

- If the names of the SPs differ from a previous simulation with the same name, an error is generated in the console, and the simulation is interrupted.
- If the SPs' names are compatible with a previous simulation, for all the games where input values are identical, the software will overwrite the existing values after issuing a warning to the console.
- If the SP are compatible, but the input values differ, the software will merge the data from these simulations, allowing the combined data to be visualized together.

D.4.2 Input Values Format

With a few exceptions noted later, most input fields can accept values in any of the following three forms:

- A single numeric value, directly specified (e.g., `price_cpu: 0.05`).
- An explicit list of numeric values enclosed in square brackets (e.g., `price_cpu: [0.05, 0.06]`).
- A range of numeric values defined by three elements in the format `['Initial_val:Final_val:amount_of_values']`, where the first element specifies the initial value, the second the final value, and the third indicates how many values to generate (e.g., `price_cpu: ['0.05:0.1:5']`).

With these formats, each possible combination of values across all input fields generates a distinct game, facilitating the definition of many games within a single file. For instance, if there are 3 CPU price values, 3 co-investment duration values (in years), 3 average load values for Service Provider SP1, 3 benefit factor values for SP1, and 3 sigma values for Service Provider SP2, then the total number of games executed will be:

3 (cpu_price) $\times$ 3 (years) $\times$ 3 (SP1 avg_load) $\times$ 3 (SP1 benefit_factor) $\times$ 3 (SP2 sigma) $= 3^5 = 243$ distinct games

Having already described the input field `simulation_name`, we now proceed to describe the remaining input parameters, grouped into general game values and Service Provider-specific values.

D.4.3 Global Games Inputs

- `max_cores_hosted`: Defines the maximum number of millicores that can be hosted.
- `price`: This field includes two subfields. Which subfield is read depends on whether the variable `variable_cpu_price` in the configuration file is set to `True` or `False`:
  - If `variable_cpu_price` is set to `False`, the value under the subfield `when_fixed: cpu_price` is used. This results in a fixed per-unit CPU price for each game.
  - If `variable_cpu_price` is set to `True`, the values under the subfield `when_variable` are used. This subfield includes three values: `min_cores_hosted`, `min_cpu_price`, and `max_cpu_price`. These three input fields must each have the same number of values. They collectively determine the number of games processed, meaning that if, e.g., each has three values, the total number of games will be multiplied by three.
- `years`: Specifies the total duration of the co-investment period, measured in years. Fractional values are accepted, enabling detailed sensitivity analyses.
- `daily_timeslots`: Indicates the number of daily time-slots used in simulations. Increasing this value provides more load samples per day, resulting in a more precise representation of the load function. If `per_time_slot_allocation` is set to `False`, changes to this variable won't affect the final results, as any setting is equivalent to using a single time-slot with average load values. Conversely, if `per_time_slot_allocation` is set to `True`, simulations more accurately reflect differences between service providers' load patterns, generally leading to better outcomes for all participants due to improved allocation granularity. However, this increased precision also makes simulations computationally more intensive.
- `selected_case`: This input selects the specific utility function case to be used in the simulation. Case 0 corresponds to the originally proposed utility function, while cases 1 to 3 represent modified versions. These modifications alter the exponents of the terms in the exponential saturation equation, as described in section 5.4. Unlike other input parameters, `selected_case` accepts exactly one value per simulation.

D.4.4 Service Providers Inputs

Under the label `service_providers`, one or more Service Providers can be defined using the following input fields:

Service Providers Names

- `service_provider_name`: Identifies each Service Provider within a simulation. It accepts either a single name or a list of names. If a list is provided, it creates multiple Service Providers with different names but identical characteristics defined by the remaining input parameters. This facilitates scaling the number of players within simulations.

Service Providers Load Function

Each Service Provider defines its load function through three input fields:

- `sigma`: Specifies the standard deviation of the load for each time slot.
- `average_load`: Defines the average daily load.
- `hyperparameters`: Defined through two sets of four values, describing the sinusoidal function modeling the load shape. The first set of four values represents the amplitude (in requests per time slot), and the second set of four values represents the time offsets (in seconds). Together, these two sets characterize the load function for a single game.

Service Providers Utility Function

The load of each Service Provider is translated into monetary units via its utility function, defined by:

- `benefit_factor`: Represents the monetary benefit obtained by the Service Provider from serving one unit of load at the edge.
- `xi` and `frac_of_requests`: Both fields define the diminishing returns effect on the Service Provider's utility, but only one of them should be specified per Service Provider. The field `xi` directly sets the diminishing returns parameter, maintained primarily for consistency with previous work. Alternatively, the `frac_of_requests` field allows defining this effect indirectly via the fraction of load served at the edge, facilitating switching between different utility function cases (original and modified versions) without manual calibration. When `frac_of_requests` is used, the software automatically computes the corresponding diminishing returns parameter based on the selected utility function case as specified in section 6.2.

D.4.5 Additional Project Folders and Files

In addition to the previously described folders and files, the repository includes the following resources:

- `charts_reproduction`: A dedicated folder containing scripts and data to easily reproduce the charts presented in this thesis.
- `utils/create_load_function.py`: A script providing an interactive, browser-based tool for manually defining daily load profiles, simplifying the process of computing sinusoidal hyperparameters as described in the introduction.

Appendix E

Variable definitions

- l_t^i: Expected load for player i at time t, representing the average number of requests coming from users of SP i during timeslot t.
- β_i: Benefit factor of SP i
- ξ_i: Diminishing returns parameter of SP i, μ_i and λ_i are also used to represent the same parameter but measured in different ways.
- x_i: Payoff of any player i.
- r_i: Revenues of player i.
- p_i: Payment or capital cost of player i.
- C: Total computational capacity, measured in millicores.
- h_i: Resources allocated to player i.
- d: Price expressed in dollars per resource unit for the whole investment period
- d': Amortized price that corresponds to one timeslot, can be calculated as $d' = \frac{d}{D \cdot T}$
- u_{it}: Instantaneous utility for player i at time t, representing the revenues coming from the end users of the services.
- $U_{tot\_net}^i$: Total net utility for player i
- D: Duration of the investment in days.
- T: Number of timeslots in one day.
- $v(S)$: Value of coalition S.
- $\delta_i(S)$: Marginal contribution of player i to coalition S.
- ϕ_i: Shapley value of player i