Imperial College London
Department of Electrical and Electronic Engineering

Optimal Resource Management in Communication Networks: Theory and Algorithm Designs

Faheem

2020

Supervised by Dr. Kin K. Leung

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London
and the Diploma of Imperial College London

Statement of Originality

I declare that this thesis is the result of my own work. Information and ideas derived from the work of others has been acknowledged in the text and a list of references is given in the bibliography. The material of this thesis has not been and will not be submitted for another degree at any other university or institution.

Copyright Declaration

This thesis is dedicated to Baba jan.

Publications

Ph.D. Related Publications

Peer-reviewed journals

- **Faheem Zafari**, Prithwish Basu, Kin K. Leung, Jian Li, Ananthram Swami, & Don Towsley, "Resource Sharing in the Edge: A Distributed Bargaining-Theoretic Approach," *Submitted for possible publications in IEEE Transactions on Mobile Computing.*

- **Faheem Zafari**, Kin K. Leung, Don Towsley, Prithwish Basu, Ananthram Swami, & Jian Li, "Let's Share: A Game-Theoretic Framework for Resource Sharing in Mobile Edge Clouds," *Submitted for possible publication in IEEE Transactions on Network and Service Management (major revision).*

- **Faheem Zafari**, Jian Li, Kin K. Leung, Don Towsley & Ananthram Swami, "Optimal Energy Consumption for Communication, Computation, Caching, and Quality Guarantee, " IEEE Transactions on Control of Network Systems, vol. 7, no. 1, pp. 151-162, March 2020.

Peer-reviewed conferences

- **Faheem Zafari**, Kin K. Leung, Don Towsley, Prithwish Basu, Ananthram Swami & Jian Li, "Cooperative-Game Framework for Resource Sharing in SDC," *DAIS-ITA Annual Fall Meeting (AFM)*, 2019.

- **Faheem Zafari**, Kin K. Leung, Don Towsley, Prithwish Basu, & Ananthram Swami, "A Game-Theoretic Framework for Resource Sharing in Clouds," *IEEE/IFIP Wireless and Mobile Networking Conference (WMNC)*, 2019.

- **Faheem Zafari**, Jian Li, Kin K. Leung, Don Towsley & Ananthram Swami, "Optimal Energy trade-off among Communication, Computation and Caching with QoI-guarantee," *IEEE Global Communications Conference (GLOBECOM),* 2018.

- **Faheem Zafari**, Jian Li, Kin K. Leung, Don Towsley, & Ananthram Swami "A Game-Theoretic Approach to Multi-Objective Resource Sharing and Allocation in Mobile Edge," *Proceedings of ACM Mobicom workshop on Technologies for the Wireless Edge Workshop,* 2018.

- **Faheem Zafari**, Jian Li, Kin K. Leung, Don Towsley & Ananthram Swami, "Application of Optimization and Game Theory to Resource Allocation in SDC Slice," *DAIS-ITA Annual Fall Meeting (AFM),* 2018.
- **Faheem Zafari**, Jian Li, Kin K. Leung, Don Towsley & Ananthram Swami, "Energy Tradeoff among Communication, Computation, and Caching with QoI-guarantee in Wireless Sensor Networks," *DAIS-ITA Annual Fall Meeting (AFM),* 2017.

Other Publications

Peer-reviewed journals

- Nitish K. Panigrahy, Jian Li, **Faheem Zafari**, Don Towsley, & Paul Yu, "A TTL-based Approach for Content Placement in Edge Networks," *Submitted for possible publication in IEEE/ACM Transactions on Networking.*
- **Faheem Zafari**, Athanasios Gkelias, & Kin K. Leung, "A Survey of Indoor Localization Systems and Technologies," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2568-2599, third quarter 2019.

Peer-reviewed conferences

- Nitish K. Panigrahy, Jian Li, **Faheem Zafari**, Don Towsley, & Paul Yu, "Jointly Compressing and Caching Data in Wireless Sensor Networks," *IEEE International Conference on Smart Computing (SMARTCOMP),* 2019.
- Jian Li, **Faheem Zafari**, Don Towsley, Kin K. Leung, & Ananthram Swami, "Joint Data Compression and Caching: Approaching Optimality with Guarantees," *ACM/SPEC International Conference on Performance Engineering (ICPE),* 2018.
- Jian Li, **Faheem Zafari**, Don Towsley, Kin K. Leung & Ananthram Swami, "Joint Data Compression and Caching: Approaching Optimality with Guarantees," *DAIS-ITA Annual Fall Meeting (AFM),* 2018.

Abstract

Communication networks have enabled human beings and machines to communicate with each other using either wired or wireless technologies. With the ever-growing reliance on the networks, there is a need for efficiently managing and using the available infrastructures as the demand for the resources is more than their supply. Due to the advent of novel paradigms, such as the Internet of Things (IoT) and edge computing, and the growing demand for resources for data analytics and machine learning (among many other applications), existing resource management algorithms and frameworks may no longer be viable. Therefore, there is a need for novel resource management frameworks that will enable the *optimal* utilization and timely provision of available resources to different applications to maximize a single or multiple objectives. In particular, there is a need for multi-objective resource sharing frameworks that will enable different service providers (SPs), despite having different objectives or utilities, to share their resources in order to improve their utility and achieve higher application (user) satisfaction.

To achieve the above, we first study the single-objective optimization problem of allocating storage, communication and computation resources to reduce energy consumption in a communication network. This issue is important because energy efficiency is a fundamental requirement of communication systems, as reflected in much recent work on performance analysis of system energy consumption. However, most work has only focused on communication and computation energy consumption without considering data caching costs. Given the increasing interest in cache networks, this is a serious deficiency. We consider the problem of energy consumption in data communication, compression and caching (C3) with a quality-of-information (QoI) guarantee in a communication network. Our goal is to identify the optimal data compression rates and cache placement over the network that minimizes the overall energy consumption in the network. We formulate the problem as a *Mixed Integer Non-Linear Programming* (MINLP) problem with non-convex functions, which is NP-hard in general. We propose a variant of the spatial Branch-and-Bound algorithm (V-SBB) that can provide an ϵ-global optimal solution to the problem. By extensive numerical experiments, we show that the C3 optimization framework improves the energy efficiency by up to 88% compared to any optimization that only considers either communication and caching or communication and computation. Furthermore, the V-SBB technique provides comparatively better solutions than some other MINLP solvers at the cost of added computation time.

We then study the problem of multi-objective resource sharing in a communication net-

work with multiple SPs, such as in an edge computing setting, where resources belong to different SPs that have their own objectives or *utilities* to optimize. On the one hand, certain SPs may not have sufficient resources to satisfy their applications according to the associated service-level agreements (SLAs). On the other hand, some SPs may have additional unused resources. For the case where SPs treat their native and non-native (belonging to other SPs) applications uniformly, we propose a bargaining-theory based resource-sharing framework that enables different SPs to optimally manage their resources and improve the satisfaction level of applications subject to constraints, such as communication costs for sharing resources across SPs. For a specific class of concave utility functions, we present an N-person *Nash Bargaining Solution* (NBS) for resource management and sharing among SPs with the Pareto optimality guarantee. Furthermore, we propose a *distributed* algorithm to obtain the NBS by proving that the strong-duality property holds for the resultant resource-sharing optimization problem. Using synthetic and real-world data traces, we show numerically that the proposed NBS framework not only enhances the ability to satisfy applications' resource demands, but also improves the utilities of different SPs.

Finally, we propose a cooperative game-theoretic framework for resource sharing among SPs with multiple objectives that only requires the SPs to have monotonic, non-decreasing and non-negative utility functions. In contrast with our proposed NBS framework, the cooperative game-theoretic framework is applicable to more general settings at the cost of added computational complexity. Furthermore, the cooperative game-theoretic framework can support two different strategies employed by SPs. In the *uniform priority* strategy, SPs do not differentiate between their native and non-native applications for resource management. In the *non-uniform priority* strategy, SPs first allocate resources to their native applications and then share remaining resources with other SPs, if needed. For the uniform priority strategy, we prove that the proposed resource-sharing game is *canonical* and *cardinally* convex. Hence, the *core* is not empty and the grand coalition of SPs is stable. We propose *Game-theoretic Pareto optimal allocation for the Uniform Priority strategy* (GPUS), a centralized algorithm to obtain a Pareto optimal allocation from the core for the uniform priority strategy. We then modify our game-theoretic framework to enable the SPs to employ non-uniform priority as a strategy. We prove that despite the modification, our proposed resource-sharing game is canonical and cardinally convex. We propose two algorithms, referred to as the *Game-theoretic Pareto optimal allocation* (GPOA) and *Polyandrous-Polygamous Matching based Pareto Optimal Allocation* (PPMPOA), also to provide allocations from the core. Hence the obtained allocations are *Pareto* optimal and the grand coalition of all SPs is stable. Experimental results confirm that our proposed resource-sharing framework improves utilities of SPs and the degree of application request satisfaction.

Acknowledgments

I would first like to thank my advisor Professor Kin Leung who made it possible for me to join Imperial College London and provided me with his valuable feedback that helped me with my research. If it had not been for his response to my e-mail back in 2015, I would have not joined Imperial College. By making me part of the DAIS-ITA project, Professor Leung enabled me to interact with other great researchers. I am also thankful to my collaborators Don Towsley, Ananthram Swami and Prithwish Basu for the their help and collaboration throughout the course of my Ph.D.

I would like to thank both Sebastian Stein and Eric Kerrigan for agreeing to be my examiners and providing me with valuable feedback. Eric, throughout my time at Imperial, has helped me with his insights on my work and has taught me a lot. Sebastian, through our interactions in DAIS-ITA AFMs and workshops, has always been kind to share his thoughts and insights with me. Through their thought provoking questions, they made the examination a great learning experience for me.

I am greatly indebted to Thanos for being my best friend and mentor at Imperial College. The countless hours spent discussing research, politics, and movies were all worth it. Having you criticize my work, diet and everything that I did helped me and made my time at Imperial enjoyable. I am thankful to Enrico Piovano for his constant nagging and making me go crazy during the first three years of my Ph.D. I am also thankful to Spike for all his help and refreshing conversations throughout my time at Imperial. I would like to extend my gratitude to Amani, Eric, Krysia, Melanie, Charlotte, Nikki and Anderson for their help throughout my time at Imperial College. I am also thankful to HiPEDS CDT, Department of Electrical Engineering and DIAS-ITA for funding this research. I also would like to thank Dinesh Verma, Seraphin Calo, Liang Ma and Frank Le at IBM T.J. Watson Center for hosting me as a research scholar.

I would also like to thank my former advisor Ioannis Papapanagiotou for his help throughout the years. His energy was contagious that made me work harder than ever before in my life. I also am thankful to Gul Muhammad Khan and Ali Mahmud for their guidance and support back when I did not even know what research meant. Thank you for introducing the art of conducting research to me. I am forever grateful to Zafar Iqbal and Hanif Rasool for their support during my college years. And no acknowledgment can be complete for me without mentioning my school teachers Sir Akhtar, Sir Munir, Miss Myra and Miss Shabana who had to put up with so many of my quirks and eccentricities.

I appreciate the help, and hospitality of Fazal Mama, and his entire family who always welcomed me into their house with opens arms and made sure I have a place that I can call home in the UK. I also am thankful to my friend Fahd Humayun and his entire family for hosting me at their home whenever I traveled to the US during my four years at Imperial. I can't thank Aarif Ali enough for his support and making it possible for me to do my research without worrying much about things outside my research. I am also greatly indebted to my friend and brother Irfan Mohammad who has stood by my side, no matter what. Throughout the highs and lows in my life, he has been a constant support. I have not been anywhere near what you deserve as a friend, but you have still been putting up with me.

I am thankful to Papa and Mummy for their support and love over the years. I hope that I can make up to you guys for all the sacrifices that you made for me. I also am greatly indebted to my siblings Najim (Dada), Yasamin (Khurai), Samim and Kalsoom who have helped me throughout the course of my life. Dada made at possible for me to move to Turkey to pursue graduate studies. Khurai, my very first teacher, has been no less than a second mother to me. No words can do justice to the sacrifices you have made for me and I will never forget the taste of your "*roghan-e-joshi*" baked in blistering summers of Peshawar. Samim, through his prowess and excellence, always challenged me to improve my skill set in academics as well as games. Kalsoom, being the youngest, made sure that she drove me crazy.

I would also like to thank Baba Jan and Bibi jan for their love and support. Baba Jan! You have been in my thoughts, and prayers every single day and not a single day passes by without missing your presences, and your voice. I am greatly indebted to Shereen mama who has been a fatherly figure to me. I appreciate every single thing that you have done for all of us and I hope I become half as kind as you are. I also am grateful for the love, support and help of Khaista Mama. Your sense of humor is unparalleled and your jokes still make me giggle. I also owe a lot to my uncle Watan who has always been there for me and my family. He has been a great uncle, friend, confidante, mentor and most importantly my chess guru. I would also like to thank my aunts Anisa, Shahlu, Faiza, Laila and Wara Yasamin for their love and support over the years. I am also thankful to my maternal cousins for their love and support. Unfortunately, word and page limits do not allow me to list each and everyone of you.

I extend my deepest gratitude to Uncle Jürgen and Aunt Gerda for all their help and support. This thesis and my studies would have never been possible without their support. Last but not the least, I would like to thank my best friend and fiancé Hosai for all her unconditional support, love and for changing my life. Thank you for being there for me and making everything possible. I am looking forward to all the years ahead with you!

Faheem
London, June 2020.

Contents

List of Tables

List of Figures

1. Introduction

Communication networks have become ubiquitous due to the wide range of services that they provide. The growing demand for communication networks requires not only to increase the resources to meet the demand, but also to manage the existing infrastructure optimally from a number of perspectives, such as reducing the energy consumed by these networks, which in turn will reduce their carbon footprint.

Optimization theory and game theory have been used in the literature for *single objective* and *multi-objective* optimal resource management and have shown promising results. However, the advent of paradigms, such as edge computing, machine learning, and the Internet of Things (IoT) necessitates novel frameworks and algorithms that can efficiently manage and allocate resources. Furthermore, resource management problems may also be multi-objective that render existing single objective optimization frameworks inapplicable.

Therefore, there is a need to develop novel frameworks and algorithms for communication networks that guarantee the optimal management of available resources with respect to a particular objective (such as minimizing energy consumption). Furthermore, there is also a need for efficient multi-objective resource-sharing frameworks among different service providers (SP) that enable SPs to satisfy requests of as many applications as possible.

1.1. Background and Motivation

Communication networks have enabled human beings and machines[1] to communicate and provide a wide range of services, such as video calls, high speed data transfer, and live streaming, etc. All these services require different resources, such as communication, computation (processing) and storage resources. Communication resources enable data transfer from the transmitter to the receiver. Computation resources process data, such as compressing the transmitted data in order to reduce energy [2,3] and bandwidth consumption, whereas storage allows one to store data for future use or minimize latency by storing data closer to the receiver as done by caching networks [4]. However, these resources are usually limited and it may not be possible to satisfy the requests of all applications or users[2]. The management and allocation of resources to different applications needs to be planned so that resources are used optimally. Furthermore, when resources are not available, there is a need to first obtain[3] and then optimally allocate the obtained resources for satisfying as

[1]Through paradigms, such as the Internet of Things and Machine-to-Machine Communication.
[2]Throughout the thesis, we use the terms applications and users interchangeably.
[3]Either by purchasing new resources or borrowing from other service providers.

many applications as possible.

Each communication network, in accordance with a single or multiple objectives, intends to optimally allocate resources based on a particular *Service Level Agreement* (SLA) with the applications. For example, in cellular networks, throughput, coverage area and reliability are some of the basic parameters that the SP can optimize to improve user satisfaction and increase the generated revenue. On the other hand, communication links in military networks only support low data transfer rates, i.e., on the order of kilo-bits per second (kbps). However, network robustness, energy efficiency, and latency are the main priorities for military networks. Therefore, the design of network and resource management objectives are highly application dependent.

Reducing network energy consumption is one such objective that has recently attracted a number of researchers [2–10]. It has been reported that the average annual power consumption only for data analytics in communication networks was 25 Giga-Watts in 2013 [9, 11], which is equivalent to operating 23 nuclear reactors. This not only increases the operational costs, but also has an adverse environmental impact. Energy consumption is also a fundamental challenge for many wireless components that operate on limited battery power supply and are usually deployed in remote or inaccessible areas. Similarly, about 5 zetabytes of data passed through the global network in 2017 [12]. This not only requires a huge network bandwidth, but may further increase the energy expenditure, because communicating such a large amount of data incurs tremendous transmission energy costs. These challenges necessitate the need for resource management designs that can enhance the energy efficiency of communication systems with a QoI guarantee. We discuss network energy consumption below.

1.1.1. Network Energy Consumption

Due to the increasing energy cost of operating communication networks, there has been an increase in interest in green networking and green communication where the primary goal is to reduce the energy consumed by the network. Similarly for certain networks, such as sensor networks, minimizing energy consumption is of paramount importance in order to increase the lifetime of the network. A typical communication network consumes energy as follows:

1. Communication Energy: Communication is one of the most energy hungry components of a communication system. The larger the data transmitted, the higher will be the energy consumption. Therefore, reducing the amount of transmitted data can drastically reduce the energy consumed in the network [2].

2. Computation Energy[4]: Computation also incurs energy cost. The higher the data processing, the larger will be the energy cost.

[4]Throughout the course of the thesis, we use computation energy to refer to energy used for data processing.

3. Storage (Caching) Energy: Data caching stores the data to reduce latency and move the data closer to the requesting nodes. However, this also incurs energy costs proportional to the amount of data and time for which the data is cached. This is because the data is usually cached in volatile Random Access Memory (RAM) that incurs low latency [4], but loses data in the absence of power.

To reduce energy consumption in the network, a number of approaches have been proposed in the literature [6–10] ranging from turning the transmitters off when not processing data [2,3]. However, there is no existing work that jointly considers data transmission, data processing and caching energy costs to minimize the overall network energy consumption. Since all these three resources contribute significantly to network energy consumption, we jointly consider them in our novel formulation in Chapter 3 of this thesis and rely on simulations to show the resulting improvement in energy efficiency.

While minimizing energy consumption in particular, or optimal management of resources with a single objective in general, is an important research problem, another significant and relatively novel problem is that of resource sharing and allocation among different service providers in distributed systems, such as cloud or edge computing that have their own objectives and utilities. Particularly, big data and machine learning require a large amount of resources and it may not be possible for a single SP to provide resources to all of its applications at any given time. Therefore, enabling resource sharing among different SPs can help. This is discussed in detail below.

1.1.2. Multi-Objective Resource Sharing and Allocation

So far we primarily discussed the optimal management and utilization of available resources with respect to a single objective. However, service providers also aim to provide resources to as many applications as possible. What happens if new requests for resources arrive and the service provider is operating at maximum operation capability? While different solutions are possible, such as buying or renting extra resources [13,14], or creating a shared resource pool [15–17]; sharing of resources among service providers is considered a promising approach [18]. Let us consider edge computing service providers given in Figure 1.1 as an illustrative example. There are multiple edge SPs that need to provide resources to different applications. In a typical setting, if an SP, say SP 1, does not have enough resources to provide to its native applications, the request is forwarded to the backend cloud through the backhaul and internet [18]. However, this incurs a higher latency. An alternative solution is to create a shared resource pool of SPs[5] or *coalition of SPs* that eliminates the need to use backend cloud resources to satisfy the requests as SPs that have under utilized resources can share their resources with SP 1 to satisfy its applications' requests. Such a resource pool

[5]This thesis does not consider privacy issues with resource sharing. However, SPs can negotiate the terms and conditions for resource sharing. Furthermore, competitors use each other's resources. For example Netflix and Amazon Prime Video services are competing with each other. However, Netflix relies on Amazon's Web Services for its streaming services.

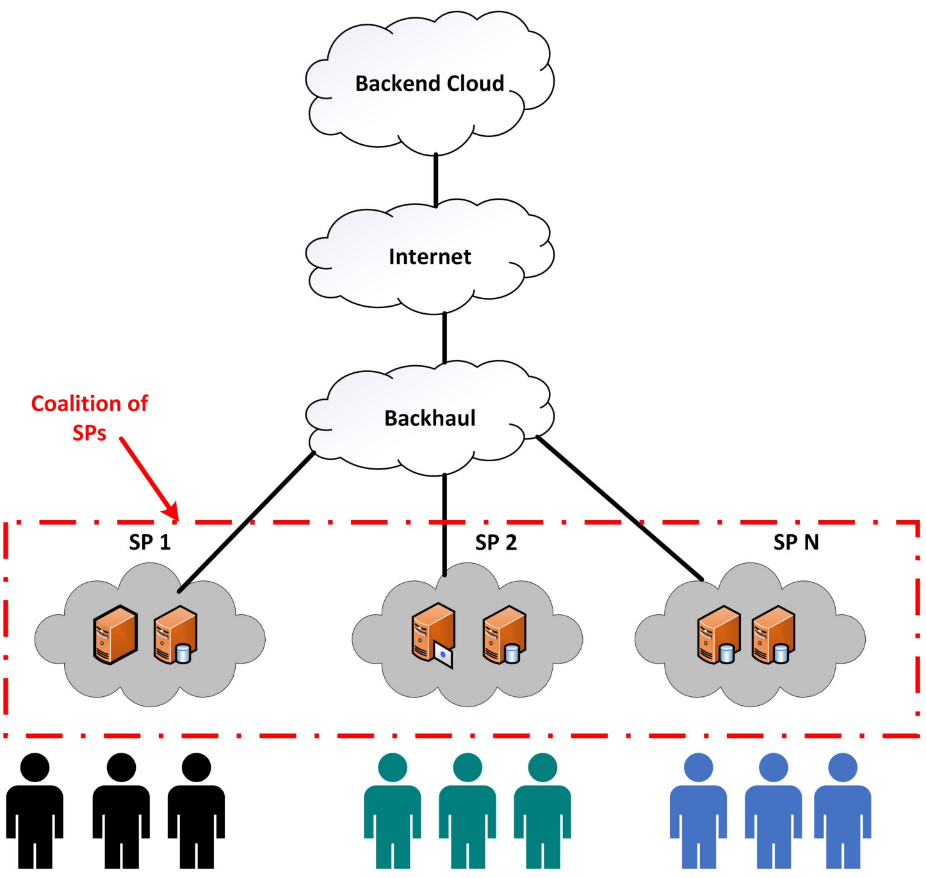

Figure 1.1.: An edge computing setting.

enables SPs to share and use resources whenever needed to meet their dynamic demands. This cooperation and resource sharing among SPs is beneficial, because it is unlikely that resources of different SPs are over-utilized [19]. However, existing work mostly considers the creation of a shared logical pool of resources among the service providers without considering the objective or utilities of individual service providers [15–17]. Furthermore, the emphasis is primarily on a single resource, such as communication resources [20]. Hence, there is lack of a generic framework that considers different resources and the objectives of all the SPs. We bridge this gap in the literature by proposing different multi-objective resource-sharing frameworks based on bargaining theory and cooperative game theory. In particular, through our proposed resource-sharing frameworks, we attempt to answer the following questions:

1. *Should an SP help another SP by sharing resources?*

2. *How should resources be allocated to applications across different SPs, while considering issues, such as communication costs?*

3. *How can SPs share the profits of resource sharing?*

Our proposed frameworks can be used in a number of distributed multi-service provider settings, such as cloud computing, edge computing, etc. However, due to the recent growing interest in edge computing because of its ability to support distributed machine learning among many other applications, we apply our developed multi-objective frameworks to an edge computing setting to highlight their efficacy.

1.2. Outline and Contributions

A summary of each chapter along with the main contributions, is given as follows:

- **Chapter 2: Preliminaries and Related Work** presents the basics of optimization and game theory. Since providing an exhaustive discussion is beyond the scope of this thesis, we primarily emphasize on aspects of optimization and game theory related to the contents of this thesis. Furthermore, the chapter also discusses work related to network energy consumption and resource allocation in the literature.
- **Chapter 3: Optimal Energy Consumption with Communication, Computation, Caching and QoI-Guarantee** considers the single objective optimization problem of energy consumption in a system with communication, computation, and storage resources. The objective is to minimize the energy consumption by finding the optimal data compression rate and caching (storage) location with a QoI guarantee. The formulated optimization problem is proven to be NP-hard. To solve the problem to ε-global optimality, a variant of the Spatial Branch-and-Bound algorithm is proposed. Through simulation results, it is shown that jointly considering communication, computation and caching energy costs can improve energy consumption significantly.
- **Chapter 4: A Distributed Bargaining-Theoretic Approach For Multi-Objective Resource Sharing** moves from the single objective optimization problem of minimizing network energy consumption discussed in the preceding chapter and considers the multi-objective optimization problem of resource-sharing among different service providers. To solve the multi-objective optimization problem, it is modeled as a *bargaining problem* for which a Nash Bargaining Solution (NBS) is used to obtain a fair and Pareto optimal allocation. It is shown that the *strong duality* property holds and a distributed algorithm to compute the NBS is proposed. Through simulation results, it is shown that resource sharing among the SPs can improve the utility as well as application satisfaction.
- **Chapter 5: A Cooperative Game-Theoretic Framework For Resource Sharing: Uniform Priority Case** departs from the preceding chapter's assumption of using concave utilities for service providers and proposes a cooperative game-theoretic framework for resource sharing. The chapter assumes that the SPs have monotone non-decreasing utility functions and it is shown that the resource sharing and allocation problem can be modeled as a cooperative game. General game properties

are discussed and properties, such as superadditivity, *cardinal convexity*, and non-emptiness of the *core* are proven. The framework supports two different strategies that can be employed by the service providers. In the *uniform* priority case, just like Chapter 4, SPs do not differentiate between their native (applications or users belonging to the SP) and non-native applications (applications or users belonging to other SPs). The uniform priority strategy is considered in this chapter. A centralized algorithm, *Game-theoretic Pareto optimal allocation with Uniform Priority strategy* (GPUS), is presented that provides an allocation from the core. Experimental results are provided to show the impact of the proposed game-theoretic framework on utility of SPs, resource utilization and application satisfaction.

- **Chapter 6: A Cooperative Game-Theoretic Framework For Resource Sharing: Non-Uniform Priority Case** extends/modifies the framework proposed in Chapter 5, and considers the setting in which the SPs employ *a* non-uniform priority strategy. SPs first allocate resources to their native applications and then share the remaining resources with applications of resource deficit SPs. It is shown that the resource sharing game with non-uniform priority strategy is also cardinally convex. Two algorithms, *Game-theoretic Pareto optimal allocation* (GPOA) and *Polyandrous-Polygamous Matching based Pareto Optimal Allocation* (PPMPOA) that provide allocations from the core are proposed. Hence, the obtained allocations using GPOA and PPMPOA are Pareto optimal and the grand coalition of all the service providers is stable. Experimental results confirm that the proposed resource sharing framework improves utilities of service providers and application request satisfaction.

- **Chapter 7: Conclusions and Future Work** provides the concluding remarks of the thesis and suggests a number of future research directions.

The frameworks and algorithms proposed in the four core chapters, i.e., Chapter 3, Chapter 4, Chapter 5 and Chapter 6, are evaluated and analyzed through simulations under different settings. Lengthy proofs of the main results are relegated to the appendices to avoid interruptions and improve readability. The system models discussed in Chapters 4, 5, and 6 are almost identical. However, for making each chapter self-sufficient and for sake of completeness, we present the system model in all the aforementioned three chapters. Furthermore, the reader is referred to the table of notations given in each of the four core chapters for a description of symbols and notations used in the corresponding chapters to avoid confusion. The variables, symbols and notations are also defined the first time they are used in each of the chapters.

2. Preliminaries and Related Work

This chapter first discusses the basics of optimization theory, cooperative game theory and Nash bargaining solution. Since providing a detailed discussion on the aforementioned topics is beyond the scope of this thesis, we only discuss them in the context of work done in this thesis. We then present some of the work done in the literature related to network energy consumption, and resource management and sharing in cloud and edge computing. Finally, we also distinguish the work done in Chapters 3, 4, 5 and 6 from the work done in the literature and highlight some of the contributions of this thesis. Detailed information about contributions of each chapter is presented in the corresponding chapters.

2.1. Preliminaries

2.1.1. Optimization Theory

In this section, we present an overview of optimization theory. In particular, we discuss convexity, non-convexity, the Lagrangian function and duality theory. For a detailed discussion, we refer the readers to [21].

Definition 2.1. *Convex Set: A set $\mathcal{Q}$ is considered to be convex if*

- *It is a subset of $\mathbb{R}^n$.*
- *For any $x, y \in \mathcal{Q}$ and $\alpha \in [0,1]$, we have $\alpha x + (1-\alpha)y \in \mathcal{Q}$. This means that all the points on the line segment joining any two points x and y from the set $\mathcal{Q}$ will also be part of the set $\mathcal{Q}$.*

Figure 2.1 shows a convex and a non-convex set. It is evident that any two points in the convex set, i.e., the hexagon, can be joined by a line segment that also lies in the hexagon. However, for the non-convex set, it is evident that the red line segment that joins the two points contains points that do not lie within the set.

Definition 2.2. *Convex Function: A function $f : \mathcal{Q} \to \mathbb{R}$ where $\mathcal{Q}$ is the aforementioned convex set given in Definition 2.1, is a convex function if*

$$f\big(\alpha x + (1-\alpha)y\big) \leq \alpha f(x) + (1-\alpha) f(y), \forall x, y \in \mathcal{Q}, \forall \alpha \in [0,1]. \tag{2.1}$$

This physically means that the line segment between $(x, f(x))$ and $(y, f(y))$ always lies above the graph of f.

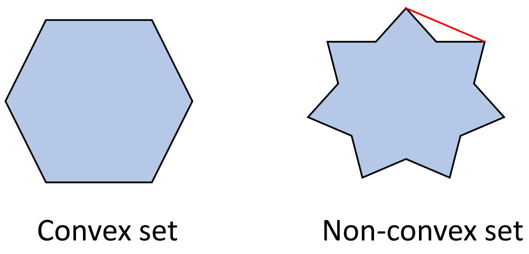

Figure 2.1.: Convex and non-convex set

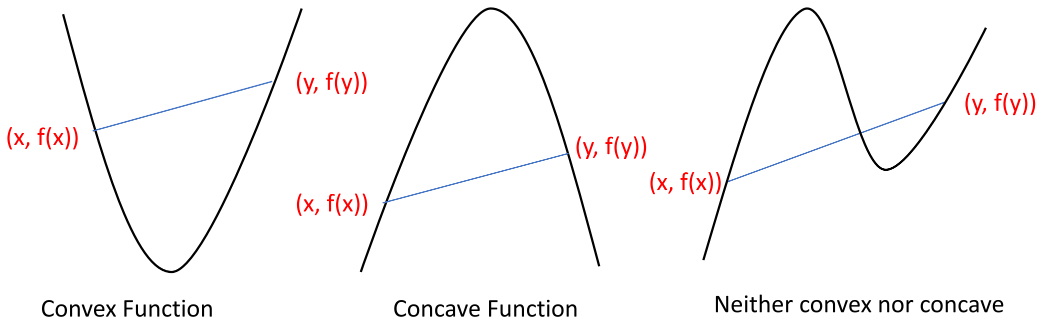

Figure 2.2.: Example of convex, concave and neither convex nor concave function

Definition 2.3. *Concave Function: A function $f : \mathcal{S} \to \mathbb{R}$ where $\mathcal{Q}$ is the aforementioned convex set given in definition 2.1, is a concave function if*

$$f\big(\alpha x + (1-\alpha)y\big) \geq \alpha f(x) + (1-\alpha)f(y), \forall x, y \in \mathcal{Q}, \forall \alpha \in [0,1]. \tag{2.2}$$

This physically means that the line segment between $(x, f(x))$ and $(y, f(y))$ always lies below the graph of f. Figure 2.2 illustrates a convex and concave function along with a function that is neither convex nor concave. It is evident from the definition of convex and concave functions that if f is convex then $-f$ is concave and vice versa.

Definition 2.4. *Strictly convex function: If the inequality given in* (2.1) *is strict, then the function is a strictly convex function.*

Definition 2.5. *Strictly concave function: If the inequality given in* (2.2) *is strict, then the function is a strictly concave function.*

$$\begin{aligned} \min_{x} \quad & f(x), \\ \text{s.t.} \quad & h_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & g_k(x) = 0; \quad k = 1, \ldots, n. \end{aligned} \tag{2.3}$$

The goal of solving an optimization problem given in (2.3) is to obtain a value for the *optimization variable* x that will optimize the objective function $f(x)$ while satisfying the constraints $h_i(x) \leq 0, \quad i = 1, \ldots, m$ and $g_k(x) = 0; \quad k = 1, \ldots, n$. The minimization optimization problem given in (2.3) is a convex optimization problem if the *objective function* $f(x)$ is a convex function and the feasible set (given by the constraints) is a convex set. For a maximization problem, the objective function has to be a concave function.

In general, obtaining the minimum of a convex optimization problem is easier when compared to a non-convex optimization problem. This is because the global and local optimal points for convex optimization problems are the same. However, non-convex optimization problems have local optimal points that are not necessarily global optimal.

Definition 2.6. *ϵ-global optimal: Let $f(x^*)$ be the global optimal value for* (2.3)*. Then $f(\bar{x})$ will be the ϵ-global optimal value if $f(\bar{x}) - f(x^*) \leq \epsilon$ for some $\epsilon \geq 0$, where $\bar{x}$ is the ϵ-global optimal solution.*

An alternative way to solve the optimization problem given in (2.3) is to use *duality theory.* Let us call the problem in (2.3) the *primal* problem. We need to formulate its *dual* problem. To do so, we define the *Lagrangian function.*

$$L(x, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{k=1}^{n} \mu_k g_k(x). \tag{2.4}$$

The Lagrangian function augments the objective function with the weighted sum of constraint functions, whereas the weights are known as *Lagrange multipliers* [21, 22]. λ_i is the Lagrange multiplier associated with the constraint $h_i(x)$ and μ_k is the Lagrange multiplier associated with constraint $g_k(x)$. It is also worth mentioning that the vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are known as the *dual* variables or Lagrange multiplier vectors for the problem in (2.3). Hence, the Lagrangian dual function D is the minimum value of the Lagrangian function or given by

$$D(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{x} L(x, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{x} \Big(f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{k=1}^{n} \mu_k g_k(x) \Big), \tag{2.5}$$

if the Lagrangian function is unbounded below. The inf is the *greatest lower bound.* Since the primal problem in (2.3) is a minimization problem, the resulting dual is a maximization problem. Similarly, if the primal problem is a maximization problem then the dual will be a minimization problem and the inf in (2.5) should be replaced with the *greatest upper bound*, i.e., sup. Note that the dual problem is always convex, even if the primal is non-convex. This is because the dual is the point-wise infimum (or supremum for a maximization primal problem) of the affine combination of the dual variables. Mathematically, the dual for the primal problem in (2.3) is given as

$$\begin{aligned} \max \quad & D(\boldsymbol{\lambda}, \boldsymbol{\mu}), \\ \text{s.t.} \quad & \boldsymbol{\lambda} \geq 0. \end{aligned} \tag{2.6}$$

Definition 2.7. *Weak duality: If p^* is the optimal value for the primal minimization problem and d^* is the optimal for the dual problem, then $p^* \geq d^*$. This is known as* weak duality.

The Lagrangian dual is the lower bound for a primal minimization problem.

Definition 2.8. *Strong duality: If $p^* = d^*$, then the property is known as strong duality.*

For strong duality to hold, certain conditions called *constraint qualifications* need to be satisfied [21].

Definition 2.9. *Duality gap: The difference between the optimal value of the primal and dual problems is known as the duality gap. Mathematically, the duality gap is given by $p^* - d^*$:*

If strong duality holds, then the following condition holds at the optimal point.

$$\sum_{i=1}^{m} \lambda_i^* h_i(x^*) + \sum_{k=1}^{n} \mu_k^* g_k(x^*) = 0. \tag{2.7}$$

This gives rise to *complementary slackness*, which is mathematically expressed as

$$\lambda_i^* > 0 \implies h_i(x^*) = 0. \tag{2.8}$$

This means that λ_i is zero unless the corresponding constraint is active at the optimum.

To solve certain optimization problems in a distributed manner, an iterative method called the gradient method [23] can be used [22]. Gradient method can work only for differentiable objective functions and constraints. At each iteration t of the algorithm, the new value of the optimization variable x is calculated using its current value and the gradient of the Lagrangian function. For a minimization problem, this is given by:

$$x^{t+1} = x^t - \alpha^t D^t \nabla L(x^t), \tag{2.9}$$

where D^t is the positive definite symmetric matrix, α^t is the step size and $\nabla L(x^t)$ is the gradient of the Lagrangian. The term $D^t \nabla L(x^t)$ is known as the direction d^t of the gradient method.

Different variants of gradient method exist based on the direction d^t and step size α^t. In the steepest descent method, D^t is chosen to be an identity matrix. The steepest descent method is less complex, but may be slow to converge. An alternative is Newton's method, where $D^t = (\nabla^2 L(x^t))^{-1}$ and $\nabla^2 L(x^t)$ is a positive definite matrix. Despite a fast convergence rate, Newton's method is complex, because it requires calculating the Hessian at each iteration. Detailed discussion on variants of gradient method based on direction d^t can be found in [23].

Similarly, there are different methods for choosing the step size α^t, such as:

- Constant step size: $\alpha^t = \alpha$ is a positive constant.
- Minimization rule: The step size α^t is chosen such that the cost function is minimized along the direction d^t. Mathematically, the chosen step size α^t should satisfy:

$$f(x^t - \alpha^t d^t) = \min_{\alpha \geq 0} f(x^t - \alpha d^t). \tag{2.10}$$

- Goldstein rule: A fixed scalar $\mu \in (0, 1/2)$ is selected and then the step size α^t is chosen such that:

$$\mu \leq \frac{f(x^t - \alpha^t d^t) - f(x^t)}{\alpha^t \nabla f(x^t)' d^t} \leq 1 - \mu. \tag{2.11}$$

There are various other techniques for choosing a step size details of which can be found in [23]. The constant step size, for a sufficiently small step size, can force the gradient method to converge to a point very close to the optimal solution.

2.1.2. Multi-Objective Optimization

For m inequality constraints and n equality constraints, multi-objective optimization (MOO) identifies a vector $\boldsymbol{x}^* = [x_1^*, x_2^*, \cdots, x_t^*]^T$ that optimizes a vector function

$$\bar{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_N(\boldsymbol{x})]^T, \tag{2.12}$$

such that

$$\begin{aligned} g_i(\boldsymbol{x}) &\geq 0, \;\; i = 1, 2, \cdots, m, \\ h_i(\boldsymbol{x}) &= 0 \;\; i = 1, 2, \cdots, n, \end{aligned} \tag{2.13}$$

where $\boldsymbol{x} = [x_1, x_2, \cdots, x_t]^T$ is a vector of t decision variables and the feasible set is denoted by $\mathcal{F}$. The fundamental difference between a single objective optimization (SOO) and MOO is that MOO involves a vector of objective functions rather than a single objective function. Therefore, in MOO, the optimal solution is not a single point but a *frontier* of solutions known as the *Pareto frontier* or *Pareto boundary* (see [24] for details).

Definition 2.10. *Pareto optimality: For any minimization problem, $\boldsymbol{x}^*$ is Pareto optimal if the following holds for every $\boldsymbol{x} \in \mathcal{F}$,*

$$\bar{f}(\boldsymbol{x}^*) \preceq \bar{f}(\boldsymbol{x}). \tag{2.14}$$

where $\bar{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_N(\boldsymbol{x})]^T$ and $\bar{f}(\boldsymbol{x}^*) = [f_1(\boldsymbol{x}^*), f_2(\boldsymbol{x}^*), \cdots, f_N(\boldsymbol{x}^*)]^T$.

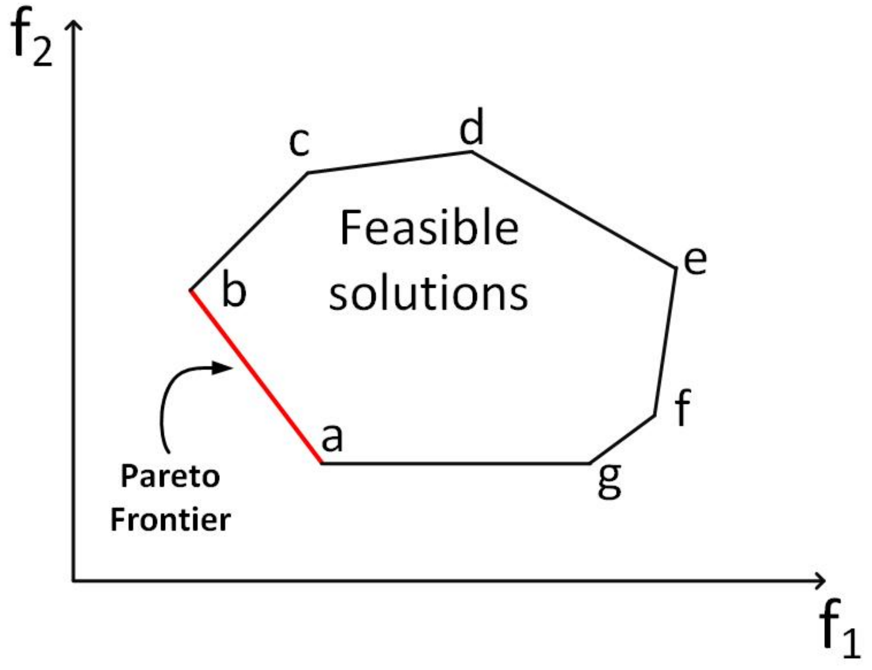

Figure 2.3.: MOO with two objective functions

Figure 2.3 shows an MOO problem (minimization problem) with two objective functions f_1 and f_2. The boundary $\overleftrightarrow{ab}$ consists of all the Pareto optimal solutions and is called the Pareto frontier.

2.1.3. Cooperative Game Theory

Cooperative game theory provides a set of analytical tools that assists in understanding the behavior of rational players in a cooperative setting [25]. Players can have agreements among themselves that affect the strategies as well as obtained utilities of game players.

Definition 2.11. *Strategy: Strategy of the game players is defined as the policy/rule that governs the actions that players take in the game.*

Coalition games are one of the basic types of cooperative games that deal with the formation of coalitions, namely groups of two or more cooperating players. Formally,

Definition 2.12. *Coalition games [25]: Any coalition game with* non-transferable utility *(discussed below) can be represented by the pair $(\mathcal{N}, \mathcal{V})$ where $\mathcal{N}$ is the set of players that play the game, $S \subseteq \mathcal{N}$, and $\mathcal{V}$ is a set of payoff vectors such that [26]:*

1. $\mathcal{V}(S)$ is a closed and convex subset of $\mathbb{R}^S$.
2. $\mathcal{V}(S)$ is comprehensive, i.e., if we are given payoffs $\mathbf{p} \in \mathcal{V}(S)$ and $\mathbf{q} \in \mathbb{R}^S$ where $\mathbf{q} \preceq \mathbf{p}$, then $\mathbf{q} \in \mathcal{V}(S)$. In other words, if the members of coalition S can achieve a payoff allocation $\mathbf{p}$, then the players can change their strategies to achieve an allocation $\mathbf{q}$.

3. The set $\{\mathbf{p}|\mathbf{p} \in \mathcal{V}(S) \text{ and } p_n \geq p_n^{'}, \forall n \in S\}$, with $p_n^{'} = \max\{q_n|\mathbf{q} \in \mathcal{V}(\{n\})\} \leq \infty$, $\forall n \in \mathcal{N}$ is a bounded subset of $\mathbb{R}^S$. In other words, the set of vectors in $\mathcal{V}(S)$ for a coalition S where the coalition members receive a payoff at least as good as working alone (non-cooperatively) is a bounded set.

Definition 2.13. *Value of a coalition: The sum of all players' payoffs from a particular payoff vector for any coalition is known as the value of a coalition.*

It is worth mentioning that $\mathcal{V}$ is the set of payoff vectors, while the value of a coalition, v, is the sum of payoffs that all players get in a particular payoff vector from the set $\mathcal{V}$.

Definition 2.14. *Non-transferable utility (NTU) [25]: If the total utility of any coalition cannot be assigned a single real number or if there is a rigid restriction on how the total utility (payoff) is distributed, then the game has a non-transferable utility.*

Definition 2.15. *Characteristic function [25]: The characteristic function for any coalition game with NTU is a function that assigns a set of payoff vectors, $\mathcal{V}(S) \subseteq \mathbb{R}^S$, where each element of the payoff vector p_n represents a payoff that player $n \in S$ obtains, depending on the selected strategy, within the coalition S.*

Definition 2.16. *Characteristic form coalition games [25]: A coalition game is said to be of characteristic form, if the value of coalition $S \subseteq \mathcal{N}$ depends only on the members of coalition.*

Definition 2.17. *Superadditivity of NTU games [26]: A canonical game with NTU is said to be superadditive if the following property is satisfied.*

$$v(S_1 \cup S_2) \supset \{p \in \mathbb{R}^{S_1 \cup S_2} | (p_n)_{n \in S_1} \in v(S_1), (p_{n'})_{n' \in S_2} \in v(S_2)\} \; \forall S_1 \subset \mathcal{N}, S_2 \subset \mathcal{N}, S_1 \cap S_2 = \emptyset, \quad (2.15)$$

I.e., if any two disjoint coalitions S_1 and S_2 form a large coalition $S_1 \cup S_2$, then the coalition $S_1 \cup S_2$ can always give its members the payoff that they would have received in the disjoint coalition S_1 and S_2.

Definition 2.18. *Canonical game: A coalition game is canonical if it is super-additive and in characteristic form.*

The *core* is a widely used solution concept for canonical games as discussed below.

2.1.4. Core

We first define some terms related to the core [25, 26].

Definition 2.19. *Group rational: A payoff vector $\boldsymbol{p} \in \mathbb{R}^{\mathcal{N}}$ is group-rational if $\sum_{n \in \mathcal{N}} p_n = v(\mathcal{N})$.*

Definition 2.20. *Individually rational: A payoff vector $\boldsymbol{p} \in \mathbb{R}^{\mathcal{N}}$ is individually-rational if every player can obtain a payoff no less than acting alone, i.e., $p_n \geq v(\{n\}), \forall n \in \mathcal{N}$, where $v(\{n\})$ is the payoff a player achieves when working alone.*

Definition 2.21. *Imputation: A payoff vector that is both individually and group rational is an imputation.*

Definition 2.22. *Grand coalition: The coalition formed by all game players in $\mathcal{N}$ is the grand coalition.*

Based on the above definitions, we can now define the core of an NTU canonical coalition game.

Definition 2.23. *Core [25]: For any NTU canonical game $(\mathcal{N}, \mathcal{V})$, the core is the set of imputations in which no coalition $S \subset \mathcal{N}$ has any incentive to reject the proposed payoff allocation and deviate from the grand coalition to form a coalition S instead. This can be mathematically expressed as*

$$\mathcal{C}_{NTU} = \{\mathbf{p} \in \mathcal{V}(\mathcal{N}) | \forall S, \nexists \mathbf{q} \in \mathcal{V}(S), \textit{ such that } q_n > p_n, \forall n \in S\}. \tag{2.16}$$

Remark 2.1. *Any payoff allocation from the core is Pareto-optimal as evident from the definition of the core. Furthermore, the grand coalition formed is stable, i.e., no group of two or more players will have an incentive to leave the grand coalition to form a smaller coalition.*

However, the core is not always guaranteed to exist. Even if the core exists, it may be very large as it is a convex set [27]. Therefore, finding a suitable allocation from the core is challenging.

2.1.5. Nash Bargaining Solution (NBS)

We use a two-player game as an illustrative example to introduce NBS. Consider two players 1 and 2 that need to reach an agreement (e.g., resource sharing and allocation decision) in an outcome space $\mathcal{A} \subseteq \mathbb{R}^2$ [25]. Both players have a utility given by u_1 and u_2 over the space $\mathcal{A} \cup \{D\}$ where D specifies a *disagreement outcome* for players in the event of a disagreement, i.e., when two players cannot reach an agreement. Let $\mathcal{E}$ be the set of all possible utilities that both players can achieve:

$$\mathcal{E} = \{(u_1(a_1), u_2(a_2)) | (a_1, a_2) \in \mathcal{A}\}. \tag{2.17}$$

We also define $d = (d_1, d_2)$, where $d_1 = u_1(D)$ and $d_2 = u_2(D)$, as the payoff each player receives at the disagreement point. We define the bargaining problem as the pair $(\mathcal{E}, d)$ where $\mathcal{E} \subset \mathbb{R}^2$ and $d \in D$ such that

- $\mathcal{E}$ is a convex and compact set;

- There exists $e \in \mathcal{E}$ such that $e > d$.

In NBS, the goal is to obtain a function $f(\mathcal{E}, d)$ that provides a unique outcome in $\mathcal{E}$ for every bargaining problem $(\mathcal{E}, d)$. Nash studied the possible outcomes (agreements) that players can reach whereas the agreements, along with the Pareto optimality, must also satisfy the following set of axioms (also called *fairness* axioms) [25]:

1. **Symmetry:** Let $e_1, e_2 \in \mathcal{E}$, where $e_1 = e_2$ and $d_1 = d_2$, then $f_1(\mathcal{E}, d) = f_2(\mathcal{E}, d)$. This means that the bargaining solution will not discriminate among players if players are indistinguishable, i.e., players have identical utilities.

2. **Invariance to equivalent utility representation:** If a bargaining problem $(\mathcal{E}, d)$ is transformed into another bargaining problem $(\mathcal{E}', d')$ where $\mathcal{E}'_i = \gamma_i \mathcal{E}_i + \zeta_i$ and $d'_i = \gamma_i d_i + \zeta_i$, $\gamma_i > 0$, then $f(\mathcal{E}', d') = \gamma_i f(\mathcal{E}, d) + \zeta_i$.

3. **Independence of irrelevant alternatives:** For any two bargaining problems $(\mathcal{E}, d)$ and $(\mathcal{E}', d)$ where $\mathcal{E}' \subseteq \mathcal{E}$, if $f(\mathcal{E}, d) \in \mathcal{E}'$, then $f(\mathcal{E}', d) = f(\mathcal{E}, d)$.

It is shown in [28] that there is a unique bargaining solution that satisfies the above axioms. We present it in the following theorem.

Theorem 2.1. *[25] There exists a unique solution satisfying the aforementioned axioms and this solution is the pair of utilities $(e_1^*, e_2^*) \in \mathcal{E}$ that solves the following optimization problem:*

$$\max_{e_1, e_2} (e_1 - d_1)(e_2 - d_2), \ s.t. \ (e_1, e_2) \in \mathcal{E}, \ (e_1, e_2) \geq (d_1, d_2). \tag{2.18}$$

The solution of (2.18) is the NBS. The above framework can be extended to N players by allowing $\mathcal{E}$ to be an N-dimensional space [29]. For this case, the bargaining problem $(\mathcal{E}, d)$, with $d = (d_1, d_2, \cdots, d_N)$ as the disagreement point, becomes the unique solution of the optimization problem.

$$\begin{aligned} \max_{e_1, \cdots, e_N} \ & \prod_{n=1}^{N} (e_n - d_n), \\ s.t. \ & (e_1, \cdots, e_N) \in \mathcal{E}, \\ & (e_1, \cdots, e_N) \geq (d_1, \cdots, d_N). \end{aligned} \tag{2.19}$$

Solving (2.18) is easier compared to the $N-$player bargaining problem in (2.19) [25].

Remark 2.2. *Since each player in an N-player bargaining game has a particular objective to optimize, the resulting problem is multi-objective, where the goal is to obtain a Pareto optimal solution. NBS is a fair and Pareto optimal solution for such MOO problems, provided that the fairness axioms are satisfied.*

2.2. Related Work

2.2.1. Energy Consumption

Energy consumption in communication networks, particularly wireless sensor networks, has been extensively studied [30, 31]. However, existing work in wireless sensor networks primarily is concerned with routing [32], MAC protocols [30], and clustering [33]. With the growing deployment of smart sensors in modern systems [3], in-network data processing, such as data aggregation or data compression, has been widely used as a mean of reducing system energy cost by lowering the data volume for transmission. Energy efficient inference in a random fusion network without QoI guarantee was considered in [34]. Network Utility Maximization (NUM) framework was applied in [35] to obtain optimal compression rate for data aggregation as well as optimal locations for performing data compression. The optimal energy allocation between communication and sensing to maximize the total information received at the sink node was studied in [36], but they did not consider data computation. An efficient algorithm for data compression in a data gathering tree was proposed in [37]. A distributed algorithm to minimize overall energy costs in a tree structured network by optimizing the compression factor at each node was presented in [3].

Data Compression: Compression has been supported by many data-parallel programming models [38]. For WSNs, data compression is usually performed over a hierarchical topology to improve communication energy efficiency [39], whereas we focus on energy consumption in a network with communication, computation and caching capabilities in Chapter 3.

Data Caching: Caches play a significant role in many systems with hierarchical topologies, e.g., WSNs, microprocessors, CDNs, etc. There is a rich literature on the performance of caching in terms of designing different caching algorithms, e.g., [40–43], and we do not attempt to provide an overview here. Utility maximization approach has also been studied for cache management [44–46]. However, none of these work considered the costs of caching, which may be significant in some systems [4]. A cooperative caching protocol designed for WSNs was presented in [47]. Two cooperative caching protocols for WSNs that minimized latency and improved the energy efficiency of the WSN were also presented in [48]. A detailed survey on cache based transport protocols for WSNs was presented in [49].

While these work only focus on energy costs of data communication and processing or data communication and caching, we study the energy consumption in data communication, computation and caching with QoI guarantee in Chapter 3. A key focus of the work in Chapter 3 is to demonstrate and validate the joint application of data computation and caching to achieve minimal energy consumption due to data communication, computation and caching with a QoI guarantee. Key decisions for certain communication networks are how much of the computation should be performed at each node and where the data should be cached in the system. The basic building blocks of our model in Chapter 3 are simple and have been studied in various settings. However, to the best of our knowledge, there is no prior work that jointly considers communication, computation and caching costs in data communication networks with a QoI guarantee for end users.

2.2.2. Resource Management

Resource management has been studied in a wide range of systems. However, we primarily focus here on multiple service provider settings such cloud computing and edge computing.

Resource Management in Edge Computing

There have been a number of solutions proposed in the literature related to resource management in edge computing [8, 18, 19, 50–55]. A novel model for allocating resources in an edge computing setting was proposed in [52], where the allocation of distributed edge resources is decoupled from service provisioning management at the service provider side. The authors develop an auction-based resource sharing contract establishment and resource allocation that maximizes the utilities of service providers and edge computing infrastructure providers. The long-held assumption that storage resources are not shareable is relaxed in [18] and the authors study the optimal allocation of both shareable and non-shareable resources in a mobile edge computing setting. The authors consider the joint problem of service placement and request scheduling, and propose a constant-factor approximation algorithm since the problem is proven to be NP-hard. An optimization framework, formulated as a Stackelberg equilibrium, for edge nodes, data service operators and service subscribers that provides optimal resource allocation in a distributed manner is proposed in [53]. A secure caching scheme in heterogeneous networks for multi-homed subscribers is proposed in [54]. The scheme relies on a trust mechanism for verifying the reliability of edge computing-enabled small cell base stations . The authors also propose a Chinese reminder theorem based protocol for preserving privacy. A Stackelberg game is used to model the interaction between the mobile users and the edge cloud and the goal is to maximize utilities of both users and service providers. An auction-based resource allocation scheme for edge service providers that provide resources for blockchains is proposed in [56]. The proposed mechanism maximizes the social welfare and guarantees truthfulness, computational efficiency and individual rationality. The mobility problem in mobile edge clouds is considered in [57] and the authors propose a novel algorithm for selecting communication path and VM placement. The proposed approach relies on predicting user movement that helps in VM placement and accordingly selecting the communication path.

Resource Management in Cloud Computing

A 2-approximation algorithm for network resource allocation in a distributed cloud environment is presented in [58]. The main objective of the system is to minimize the latency. Furthermore, they also propose a similar algorithm that selects the racks and servers where any particular VM requested by the user can be placed. A model for resource allocation in a cloud computing environment is proposed in [59]. The resource allocation problem is modeled in a task-oriented framework where resource allocation task is ranked on the basis of the pairwise comparison among the available resources and user requests. This is achieved using a matrix technique and analytic hierarchy process. An improved differential evolution algorithm for resource allocation in clouds is proposed in [60] that combines differential evolution algorithm with Taguchi method. The system objective is to optimize the total cost and makespan, hence the authors consider a multi-objective problem for a single service provider. A dynamic resource allocation framework that considers different SLA parameters and premptable task execution is proposed in [61]. Simulation results show that the proposed algorithm improves resource utilization when there is fierce competition for limited resources. A system that relies on virtualization technology for allocating resources dynamically to different users and enhances energy efficiency of the system by minimizing the number of servers used is proposed in [62]. The authors rely on the concept of *skewness* to improve resource utilization and prevent system overload. A resource sharing architecture for mobile clouds that relies on service-oriented utility functions is discussed in [63]. The authors primarily consider service latency and rely on a centralized framework.

NBS Based Resource Management

NBS is used to allocate bandwidth for elastic services in high speed networks in [20]. The fairness criteria when allocating resources to different cloud users is considered in [64]. The authors rely on NBS to guarantee fairness. Using dual composition and sub-gradient method, the authors also develop a distributed algorithm. An NBS-based model for cost-effective and dynamic VM allocation with multimedia traffic is proposed in [65] that can reduce the cost of running different servers along with maximizing resource utilization and satisfying the QoS requirements. He et al. [66] study the optimal deployment of content in a cloud assisted video distribution system. NBS is used in [67] for Virtual Machine (VM) migration to maximize the resource utilization in a video streaming data center.

We present a distributed framework based on NBS in Chapter 4 that requires a specific utility function that can be used to model different metrics, such as latency, delay, and numerous other objectives. In contrast with [20, 65–68], to the best of our knowledge, our framework is the first of its kind that uses NBS for resource sharing among SPs with different utilities (objective functions). We show that resource sharing can improve utilities of SPs and enhance application satisfaction. Furthermore, for a particular class of utilities, we have proved that a distributed algorithm exists for obtaining NBS for the resource sharing and allocation problem. Table 2.1 summarizes some other solutions proposed in the literature that use NBS for resource allocation in different systems. Other distributed algorithms proposed in the literature either rely on dual decomposition [64] or gradient projection [20, 68]. However, most of the functions are not dual decomposable and gradient projection is a computationally expensive approach [23], whereas gradient descent is comparatively computationally less expensive, and is widely used particularly in machine and

Table 2.1.: Comparison of our NBS based resource sharing and allocation framework with other NBS based solutions.

Ref.	Objective	Sharing	Distributed Algorithm	
				Technique
[20]	Fair Bandwidth Allocation	×	✓	Gradient Projection
[64]	Fairness	×	✓	Dual decomposition and sub-gradient method are used.
[65]	Cost and Resource Utilization	×	×	N/A
[66]	Cost and User Experience	×	×	N/A
[67]	Resource utilization	×	×	N/A
[68]	Bandwidth allocation	×	✓	Gradient Projection
Chapter 4	SP utility and user satisfaction	✓	✓	Gradient descent based algorithm that works for the class of utilities described in Section 4.2.3.

deep learning.

Game Theory Based Resource Management

The interaction among cloud service operators and edge service owners is modeled as a Stackelberg game for maximizing the utilities of both cloud and edge service providers in [55]. The authors obtain the optimal payment along with the computation offloading strategies. Scheduling in hybrid clouds is modeled as a sequential cooperative game in [69]. The authors proposed a storage and communication aware algorithm that jointly optimizes the execution time and economic cost of scheduling Bag-of-Tasks work flows. A Shapley value based mechanism for on-demand bandwidth allocation between data centers is proposed in [70]. The authors focus on network bandwidth and do not take computing and storage resources into account. An online auction mechanism for dynamically providing virtual clusters in geo-distributed clouds is proposed in [71]. A distributed negotiation mechanism that allows service providers and users to negotiate the contract price, and a decommitment penalty is proposed in [72]. A non-cooperative game theory based approach for resource allocation in the cloud is proposed in [73]. The objective of the algorithm is to maximize fairness among different users. They show that a Nash equilibrium is guaranteed to exist provided that the resource allocation problem has feasible solutions.

Our work in Chapter 5 and Chapter 6 differs from [18,52–55] since we consider the multi-objective nature of the resource sharing problem and allowed different service providers to share resources and improve their utilities, while satisfying the requests of different users. Furthermore, rather than allocating a single resource [70], our framework considers the allocation of a number of different resources that an application requires to perform a certain task. Our framework guarantees Pareto optimality. Every service provider is guaranteed to attain a utility as good as working alone and the grand coalition formed by all service provider is stable. Hence, no service provider has the incentive to leave the resource sharing coalition and are guaranteed a minimum performance. Our cooperative game-theoretic framework allows two different strategies, i.e., *uniform* and *non-uniform* priority. A centralized algorithm that provides an allocation from the *core* is proposed for the uniform priority strategy. Two algorithms that provide allocation from the *core* are proposed for the non-uniform priority strategy.

The algorithms proposed in Chapter 5 and Chapter 6 are designed based on the principles of game theory to ascertain that solutions with desirable characteristics, such as Pareto optimality, individual rationality, group rationality, and stability of coalition are obtained efficiently when compared to using off-the-shelf solvers. For example, an off-the-self solver will require a large (exponential) number of constraints in the optimization problem [25] to guarantee the stability of the coalition. On the other hand, our tailor-made algorithms leverage game-theoretic concepts to eliminate the need for such large number of constraints.

2.3. Summary and Conclusions

This chapter provided the necessary preliminaries for the remainder of this thesis. Basics of optimization theory, multi-objective optimization, cooperative game theory, and bargaining theory were provided. The chapter also discussed some of the work related to resource management in cloud computing, edge computing, use of game theory and bargaining theory for resource management in different settings. Furthermore, we also presented a brief discussion on how the work done in this thesis is different than that done already in the literature. It is evident that none of the existing works have jointly considered communication, computation and caching energy costs when minimizing energy consumption in a communication network. Furthermore, cooperative game theory and NBS have been used for resource allocation in different domains. However, existing literature has not studied the problem of multi-objective resource sharing.

3. Optimal Energy Consumption with Communication, Computation, Caching and QoI-Guarantee

Data transmission is an energy hungry process. Data computation, such as compression, by reducing the number of bits to be transmitted, can reduce energy consumption in the network. However, if the data is compressed beyond a certain threshold, compression can consume more energy than data transmission. Similarly, data caching is primarily used for reducing latency. Additionally, data caching by storing data closer to the requesting node eliminates the need for repeatedly transmitting the data from source node to the requesting node. This can also reduce the communication energy cost. Most existing work has only focused on communication and computation energy costs without accounting for data caching costs. Given the increasing interest in cache networks and energy consumption in a communication network, this is a serious deficiency.

In this chapter, the problem of energy consumption in data communication, compression and caching (C3) with a quality-of-information (QoI) guarantee in a communication network is presented. The goal is to identify the optimal data compression rates and cache placement over the network that minimizes the overall energy consumption in the network. The energy consumption problem is formulated as a *Mixed Integer Non-Linear Programming* (MINLP) problem with non-convex functions, which is shown to be NP-hard. A variant of the Spatial Branch-and-Bound algorithm (V-SBB) is proposed that can provide an ϵ-global optimal solution to the problem. By extensive numerical experiments, it is shown that a C3 optimization framework improves the energy efficiency by up to 88% compared to any optimization that only considers either communication and caching or communication and computation.

3.1. Introduction

The rapid growth of smart environments, and advent of the Internet of Things (IoT) have led to the generation of large amounts of data. However, it is a daunting task to transmit enormous amounts of data through traditional networks due to limited bandwidth and energy limitations [3]. This data needs to be efficiently compressed, transmitted, and cached to satisfy the Quality of Information (QoI) required by end users. In fact, many wireless components operate on a limited battery power supply and are usually deployed

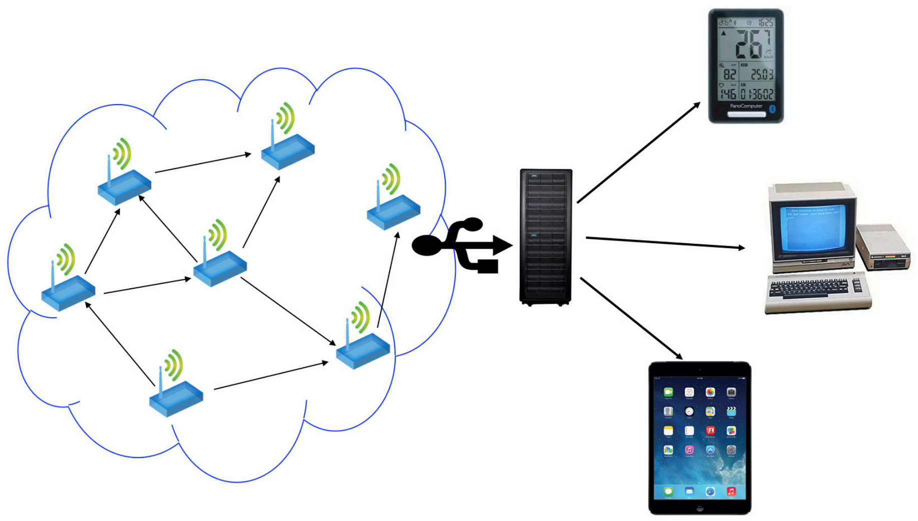

Figure 3.1.: A general wireless sensor network.

in remote or inaccessible areas, which necessitates the need for designs that can enhance the energy efficiency of the system with a QoI guarantee.

A particular example of modern systems that require high energy efficiency is a wireless sensor network (WSN). Consider a WSN with various types of sensors, which can generate enormous amounts of data to serve end applications or users. On one hand, data compression has been adopted to reduce transmission (communication) cost at the expense of computation cost [3]. On the other hand, caches can be used as a mean of reducing transmission costs and access latency, thus enhancing QoI but with the expense of the added caching cost. Hence, there exists a tradeoff in energy consumption due to data communication, computation and caching. This raises the question: *what is the right balance between compression and caching so as to minimize the total energy consumption of the network?* We answer the question in this chapter.

Each node has the ability to compress and cache the data with some finite storage capacity. We focus on wireless sensor networks as our motivating example. In particular, as shown in Figure 3.1, we assume that only edge sensors generate data, and there exists a single sink node that collects and serves the requests for the data generated in this network. The model can be extended to include any arbitrary node that produces data at the expense of added notational complexity.

A common assumption in previous works is that energy required to compress data is smaller than that needed to transmit data. Therefore, data compression was considered a viable technique for reducing energy consumption. However, it has been shown that computational energy cost can be significant and may cause a net-energy increase if data are compressed beyond a certain threshold [74]. Furthermore, the degree of the data aggregation or compression in a system is crucial for QoI. It has been shown that data aggregation can deteriorate QoI in some situations [75]. Hence, it is necessary to consider both transmission and computation costs, and it is important to characterize the trade-off between them [3] along with the impact on QoI.

Caches have been widely used in networks and distributed systems to improve performance by storing information locally, which jointly reduces access latency and bandwidth requirements, and hence improves user experience. Content Distribution Networks (CDNs), Software Defined Networks (SDNs), Named Data Networks (NDNs) and Content Centric Networks (CCNs) are important examples of such systems. The fundamental idea behind caching is to make information available at a location closer to the end-user. Again, most previous work focused on designing caching algorithms to enhance system performance without considering the energy cost of caching. Caching can reduce the transmission energy by storing a local copy of the data at the requesting node (or close by), hence eliminating the need for multiple re-transmission from the source node to the requesting node. However, caching itself can incur significant energy costs [4]. Therefore, analyzing the impact of caching on overall energy consumption in the network (along with data communication and compression) is critical for system design.

We focus on a tree-structured sensor network where each leaf node generates data, and compresses and transmits the data to the sink node in the network, which serves the requests for these data from devices outside this network. Examples of such a setting are military sites, wireless sensors or societal networks, where a large number of devices gather data, and desire to transmit the local information to any device outside this network that requires this information. The objective of our work in this chapter is to obtain optimal data compression rate at each node, and an optimal data placement in the network for minimizing energy consumption with QoI guarantee.

3.1.1. Organization and Main Results

In Section 3.2, we describe our system model in which nodes are logically arranged as a tree. Each node receives and compresses data from its children node(s). The compressed data are transmitted and further compressed towards the sink node. Each node can also cache the compressed data locally. In Section 3.3, we formulate the problem of energy-efficient data compression, communication and caching with QoI constraint as an MINLP problem with non-convex functions, which is shown to be NP-hard. We then show that there exists an equivalent problem obtained through symbolic reformation [1] in Section 3.4, and propose a variant of the Spatial Branch-and-Bound (V-SBB) algorithm to solve it. We show that our proposed algorithm can achieve ϵ-global optimality.

In Section 3.6, we evaluate the performance of our optimization framework and show that the use of caching along with data compression and communication can significantly improve the energy efficiency of a communication network. More importantly, we observe that with the joint optimization of data communication, computation and caching (C3), energy efficiency can be improved by as much as 88% compared to only optimizing communication and computation, or communication and caching (C2). The improvement depends on the values of parameters in the model and the magnitude of improvement varies with different

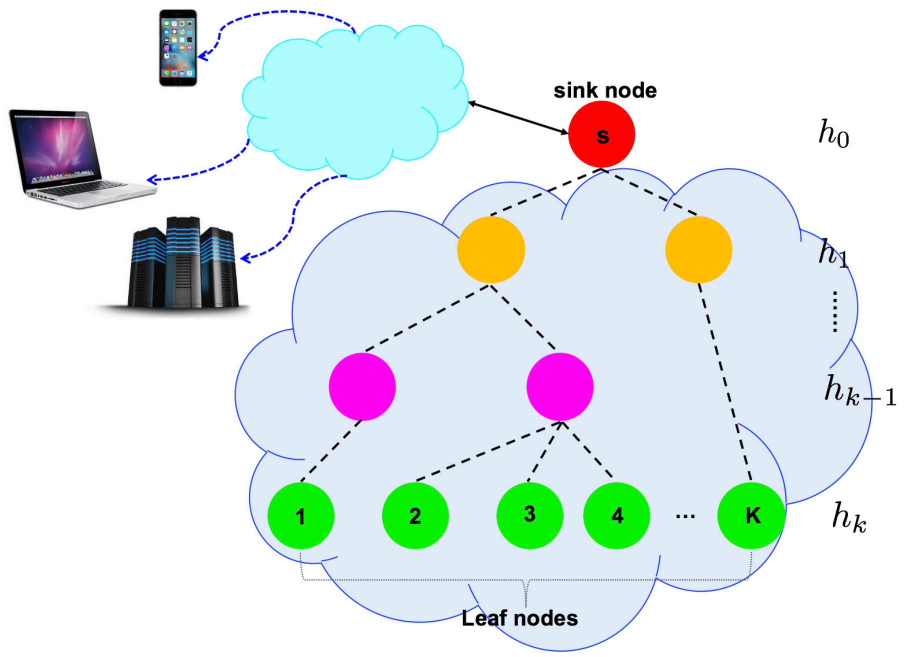

Figure 3.2.: Tree-structured network model.

energy costs of the model. While the improvement in energy efficient is important, our framework helps in characterizing and analyzing the enhancement in energy efficiency for different network settings. We also evaluate the performance of the proposed V-SBB algorithm through extensive numerical studies. In particular, we make a thorough comparison with other MINLP solvers Bonmin [76], NOMAD [77], Matlab's genetic algorithm (GA), Baron [78], SCIP [79] and Antigone [80] under different network scenarios. The results show that our algorithm can achieve ϵ-global optimality, and the achieved objective function value (we achieve a lower objective function value for a minimization problem) is mostly better than stochastic algorithms, such as NOMAD, GA while it performs comparably with deterministic algorithms, such as Baron, Bonmin, SCIP and Antigone. Furthermore, our algorithm provides a solution in varying network situations even when other solvers such as Bonmin, and SCIP are not able to. We provide concluding remarks in Section 3.7.

3.2. Analytical Model

We represent the network as a directed graph $G = (V, E)$. For simplicity, we consider a tree, with $|V|$ nodes, as shown in Figure 3.2. It is possible to generalize our framework to general network topology with arbitrary source nodes, provided that the route between the source and requesting node is known. Node $v \in V$ is capable of storing S_v amount of data. Let $\mathcal{K} \subseteq V$ with $K = |\mathcal{K}|$ be the set of leaf nodes, i.e., $\mathcal{K} = \{1, 2, \dots, K\}$. Time is partitioned in periods of equal length $T > 0$ and data generated in each period are independent. Without loss of generality (W.l.o.g.), we consider one particular period in the remainder of the chapter. We assume that only leaf nodes $k \in \mathcal{K}$ can generate data, and all other nodes in the tree receive and compress data from their children nodes, and either cache or transmit the compressed data to their parent nodes during time T. Arbitrary source nodes can also be incorporated into the model at the cost of added notational and model complexity.

Let y_k be the amount of data generated by leaf node $k \in \mathcal{K}$. The data generated at the leaf nodes are transmitted up the tree to sink node s, which serves requests for data generated in the network. Let $h(k)$ be the depth of node k in the tree. W.l.o.g., we assume that the sink node is located at level $h(s) = 0$. We represent a path from node k to the sink node as the unique path $\mathcal{H}^k$ of length $h(k)$ as a sequence $\{h_0^k, h_1^k, \cdots, h_{h(k)}^k\}$ of nodes $h_j^k \in V$ such that $(h_j^k, h_{j+1}^k) \in E$, where $h_0^k \triangleq s$ (i.e., the sink node) and $h_{h(k)}^k \triangleq k$ (i.e., the node itself).

We denote the per-bit reception, transmission and compression cost of node $v \in V$ as $\varepsilon_{vR}, \varepsilon_{vT}$, and ε_{vC}, respectively. Each node h_i^k along the path $\mathcal{H}^k$ can compress the data generated by leaf node k with a *data reduction rate* $\delta_{k,i}$, where $0 < \delta_{k,i} \leq 1$, $\forall i, k$. The reduction rate characterizes the degree to which a node can compress the received data, which plays an important role for determining the QoI. The higher the value of $\delta_{k,i}$, the lower the compression will be, and vice versa. The higher the degree of data compression, the larger will be the amount of energy consumed by compression. Similarly, caching the data closer to the sink node may reduce the transmission cost for serving the request, however, each node only has finite storage capacity. We study the energy consumed at each node for transmitting, compressing and caching the data.

Denote the total energy consumption at node v as E_v, which consists of reception cost E_{vR}, transmission cost E_{vT}, computation cost E_{vC} and storage (caching) cost E_{vS}; it takes the form

$$\begin{aligned} & E_v = E_{vR} + E_{vT} + E_{vC} + E_{vS}, \\ \text{where} \quad & E_{vR} = y_v \varepsilon_{vR}, \quad E_{vT} = y_v \varepsilon_{vT} \delta_v, \\ & E_{vC} = y_v \varepsilon_{vC} l_v(\delta_v), \quad E_{vS} = w_{ca} y_v T. \end{aligned} \tag{3.1}$$

The above energy consumption models for data transmission, compression and caching have been used in the literature [3,4,35] and are suitable for highlighting the energy consumption in a communication network. However, our formulation can be extended to incorporate various other energy consumption models as well. In (3.1), $l_v(\delta_v)$ captures the computation energy. Since computation energy increases with the degree of compression, we assume that $l_v(\delta_v)$ is a continuous, decreasing and differentiable function of the reduction rate. One candidate function is $l_v(\delta_v) = 1/\delta_v - 1$ [3,35]. Moreover, we consider an energy-proportional model [4] for caching, i.e., $E_{vS} = w_{ca} y_v T$ if the received data y_v is cached for a duration of T where w_{ca} represents the power efficiency of caching, which strongly depends on the storage hardware technology. W.l.o.g., w_{ca} is assumed to be identical for all the nodes. The energy-proportional model is widely used for cache networks [4, 81, 82], since the data is usually cached in volatile Random Access Memory (RAM) in systems, such as WSN. The use of RAMs helps in minimizing access latency[1], because getting data from RAMs is fast. However, the data will be lost if the power supply is lost. For simplicity, denote $f(\delta_v)= \varepsilon_{vR}+\varepsilon_{vT}\delta_v+\varepsilon_{vC}l_v(\delta_v)$ as the sum of per-bit reception, transmission and compression cost at node v per unit time.

During time period T, we assume that there are R_k requests at sink node s for data y_k generated by leaf node k. For simplicity, we assume that the number of requests for the data of a node k is constant. The boolean variable $b_{k,i}$ equals 1 if the data from node k is stored along the path $\mathcal{H}^k$ at node h_i^k, otherwise it equals 0. We allow the data to be cached at only one node along the unique path between the leaf node and root node. For ease of notation, we define $b_{k,h(k)}$ by b_k. Let C_v denote the set of leaf nodes $k \in \mathcal{K}$ that are descendants of node v. We also assume that the energy cost for searching for data at different nodes in the network is negligible [3,43]. For convenience, let $\delta_{k,h(k)} \triangleq \delta_k$. For ease of exposition, the parameters used throughout this chapter are summarized in Table 3.1.

3.3. Energy Optimization

In this section, we first define the cost function in our model and then formulate the optimization problem. Data produced by every leaf node is received, transmitted, and possibly compressed by all nodes in the path from the leaf node to the root node, consuming energy

$$E_k^{\mathrm{C}} = \sum_{i=0}^{h(k)} y_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m}, \tag{3.2}$$

where $\prod_{m=i}^{j} \delta_{k,m} := 1$ if $i \geq j$. (3.2) captures one-time[2] energy cost of receiving, compressing and transmitting data y_k from leaf node (level $h(k)$) to the sink node (level 0). The amount of data received by any node at level i from leaf node k is $y_k \prod_{m=i+1}^{h(k)} \delta_{k,m}$ due to the compression from level $h(k)$ to $i+1$. The term $f(\delta_{k,i})$ captures the reception, transmission and compression energy cost for node at level i along the path from leaf node k to the sink node.

Let E_k^{R} be the total energy consumed in responding to the subsequent (R_k-1) requests. We have

$$E_k^{\mathrm{R}} = \sum_{i=0}^{h(k)} y_k (R_k-1) \left\{ f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} \left(1 - \sum_{j=0}^{i} b_{k,j}\right) + \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) b_{k,i} \left(\frac{w_{ca}T}{R_k-1} + \varepsilon_{kT}\right) \right\}. \tag{3.3}$$

Note that the remaining (R_k-1) requests are either served by the leaf node or a cached copy of data y_k at level i for $i=1,\cdots,h(k)$. W.l.o.g., we consider node $v_{k,i}$ at level i. If

[1]Cache networks are primarily used to minimize access latency.

[2]During every time period T, data is always pushed towards the sink upon the first request.

Table 3.1.: Summary of notations used in Chapter 3

Notation	**Description**
y_k	number of data (bits) generated at node k
$\mathcal{K}$	Set of leaf nodes
$\delta_{k,v}$	reduction rate at node v, is the ratio of amount of output data to input data
γ	the QoI threshold
ε_{vR}	per-bit reception cost of node v
ε_{vT}	per-bit transmission cost of node v
ε_{vC}	per-bit compression cost of node v
$b_{k,v}$	1 if node v caches the data from leaf node k; otherwise 0
S_v	storage capacity of node v
w_{ca}	caching power efficiency
R_k	request rate for data from node k
I_v	set of leaf nodes that are descendants of node v
T	time length that data are cached
ϕ^u	upper bound of the objective function
$\mathcal{L}$	list of regions
$\mathcal{R}$	any sub-region in $\mathcal{L}$
$\phi^{\mathcal{R},u}$	upper bound on the objective function in sub-region $\mathcal{R}$
$\phi^{\mathcal{R},l}$	lower bound on the objective function in sub-region $\mathcal{R}$
ϵ	difference between the upper and lower bound
$w_i^{\mathcal{R},l}$	lower bound on auxiliary variable w_i in sub-region $\mathcal{R}$
$w_i^{\mathcal{R},u}$	upper bound on auxiliary variable w_i in sub-region $\mathcal{R}$
w_{bc}^j	j^{th} candidate variable for branching
w_b	chosen branching variable
w_b^∇	value at which the variable is branched
bt	bilinear terms
lft	linear fractional terms
$\mathcal{T}_{\text{bt}}$	set of bilinear terms (bt)
$\mathcal{T}_{\text{lft}}$	set of linear fractional terms (lft)

data y_k is not cached from $v_{k,i}$ up to the sink node (level 0), i.e., $b_{k,j} = 0$ for $j = 0, \cdots, i$, the cost is incurred due to receiving, transmitting and compressing the data $(R_k - 1)$ times, which is captured by the first term in (3.3), the second term is 0. Otherwise, the $(R_k - 1)$ requests are served by the cached copy at $v_{k,i}$, the corresponding caching and transmission cost serving from $v_{k,i}$ are captured by the second term in (3.3), and the corresponding reception, transmission and compression cost from $v_{k,i-1}$ upto sink node is captured by the first term. Note that the first time cost of reception, transmission and compressing the data from leaf node to $v_{k,i}$ is already captured by (3.2). We present a simple but illustrative example to explain the above equations.

Example 1. We consider a network with one leaf node and one sink node, i.e., $k = 1$ and $h(k) = 1$. Then the cost in (3.2) becomes $E_1^C = y_1 f(\delta_{1,0})\delta_{1,1} + y_1 f(\delta_{1,1})$, where the first and second terms capture the reception, transmission and compression cost for data y_1 at the sink node and the leaf node, respectively. The cost in (3.3) is $E_1^R =$

$$\underbrace{y_1(R_1-1)\left[f(\delta_{1,0})\delta_{1,1}(1-b_{1,0})+\delta_{1,0}\delta_{1,1}b_{1,0}\left(\frac{w_{ca}T}{R_1-1}+\varepsilon_{1T}\right)\right]}_{\text{Term 1}}$$
$$+\underbrace{y_1(R_1-1)\left[f(\delta_{1,1})(1-b_{1,0}-b_{1,1})+\delta_{1,1}b_{1,1}\left(\frac{w_{ca}T}{R_1-1}+\varepsilon_{1T}\right)\right]}_{\text{Term 2}},$$

where Term 1 and Term 2 capture the costs at the sink node and the leaf node, respectively. To be more specific, there are three cases: (i) data y_1 is cached at sink node 0, i.e., $b_{1,0}=1$ and $b_{1,1}=0$ (since we only cache one copy); (ii) data y_1 is cached at leaf node 1, i.e., $b_{1,0}=0$ and $b_{1,1}=1$; and (iii) data y_1 is not cached, i.e., $b_{1,0}=b_{1,1}=0$. We consider these three cases in the following.

Case (i), i.e., $b_{1,0}=1$ and $b_{1,1}=0$, Term 2 becomes 0 and Term 1 reduces to $y_1(R_1-1)\delta_{1,0}\delta_{1,1}b_{1,0}(\frac{w_{ca}T}{R_1-1}+\varepsilon_{1T})$ since all the (R_1-1) requests are served from sink node. This indicates that the total energy cost is due to caching the data for time period T and transmitting it (R_k-1) times from the sink node to users that request it.

Case (ii), i.e., $b_{1,0}=0$ and $b_{1,1}=1$, Term 1 becomes $y_1(R_1-1)f(\delta_{1,0})\delta_{1,1}$, which captures the reception, transmission and compression costs at sink node 0 for serving the (R_1-1) requests. Term 2 becomes $y_1(R_1-1)\delta_{1,1}b_{1,1}\left(\frac{w_{ca}T}{R_1-1}+\varepsilon_{1T}\right)$, which captures the cost of caching data at the leaf node and transmitting the data (R_k-1) times from the cached copy to the sink node. The sum of them is the total cost to serve (R_1-1) requests.

Case (iii), i.e., $b_{1,0}=b_{1,1}=0$, $E_1^R=y_1(R_1-1)f(\delta_{1,0})\delta_{1,1}+y_1(R_1-1)f(\delta_{1,1})$, which captures the reception, transmission and compression costs at sink node 0 and leaf node 1 for serving the (R_1-1) requests since there is no cached copy in the network.

The total energy consumed in the network is E^{total},

$$E^{\text{total}}(\boldsymbol{\delta},\boldsymbol{b}) \triangleq \sum_{k\in\mathcal{K}}\left(E_k^{\text{C}}+E_k^{\text{R}}\right), \tag{3.4}$$

where $\boldsymbol{\delta}=\{\delta_{k,i},\forall k\in\mathcal{K}, i=0,\cdots,h(k)\}$ and $\boldsymbol{b}=\{b_{k,i},\forall k\in\mathcal{K}, i=0,\cdots,h(k)\}$. Our objective is to minimize the total energy consumption of the network with a QoI constraint for end users by choosing the compression ratio vector $\boldsymbol{\delta}$ and caching decision vector $\boldsymbol{b}$ in the network G. Therefore, the optimization problem is,

$$\min_{\boldsymbol{\delta},\boldsymbol{b}} \quad E^{\text{total}}(\boldsymbol{\delta},\boldsymbol{b}) \tag{3.5a}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}} y_k \prod_{i=0}^{h(k)} \delta_{k,i} \geq \gamma, \tag{3.5b}$$

$$\sum_{k \in C_v} b_{k,h(v)} y_k \prod_{j=h(k)}^{h(v)} \delta_{k,j} \leq S_v, \forall\, v \in V, \tag{3.5c}$$

$$\sum_{i=0}^{h(k)} b_{k,i} \leq 1, \forall k \in \mathcal{K}, \tag{3.5d}$$

$$0 < \delta_{k,i} \leq 1, \forall k \in \mathcal{K},\ i = 0, \ldots, h(k), \tag{3.5e}$$

$$b_{k,i} \in \{0, 1\}, \forall k \in \mathcal{K}, i = 0, \ldots, h(k), \tag{3.5f}$$

where $h(v)$ is the depth of node v in the tree. The first constraint is the QoI constraint, i.e., the total data available at the sink node [3]. The second constraint is on total amount of data that can be cached at each node. The third constraint is that at most one copy of the generated data should be cached on the path between the leaf node and the sink node. The fourth constraint indicates that the compression rate is between 0 and 1, whereas the last constraint indicates that our decision (caching) variable $b_{k,i}$ is binary. In this thesis, we define QoI as the total amount of data (bits) that are received at the sink node [2,3]. The implicit assumptions with such a definition of QoI are that the data of all the leaf nodes have equal priority and information depends on the number of bits that are received[3]. However, it is possible that the data of one leaf node may have a higher priority when compared to other leaf nodes. Furthermore, some leaf nodes may produce a small amount of data (bits), but may have more information. For example, a single bit can be used to represent whether a sensor has detected any volcanic activity in an area or not. For such a setting, an alternative approach will be to include separate QoI constraints with different QoI thresholds (γ_k) for each of the leaf node k. This will enable the system to assign different priority to the leaf nodes and ascertain that more valuable information, such as the aforementioned volcanic activity is accounted for, while enabling data compression to reduce energy consumption.

The optimization problem in (3.5) is a non-convex MINLP problem with Θ continuous variables, the $\delta_{k,i}$'s and Θ binary variables, the $b_{k,i}$'s where, $\Theta = \sum_{k \in \mathcal{K}} h(k)$.

3.3.1. Properties

We first analyze the complexity of the problem given in (3.5) and show that it is NP-hard.

Theorem 3.1. *The optimization problem in (3.5) is NP-hard.*

Proof. We prove the hardness by a reduction of any given $0-1$ *knapsack problem* (KP) to a corresponding instance of (3.5). The KP is given as follows:

$$\max_{\mathbf{x}} \quad \sum_{k=1}^{|\mathcal{K}|} v_k x_k$$

[3]The larger the number of bits, the more will be the information.

$$\text{s.t.} \quad \sum_{k=1}^{|\mathcal{K}|} w_k x_k \leq W,$$
$$x_k \in \{0,1\}, \quad \forall k \in \mathcal{K}. \tag{3.6}$$

where $|\mathcal{K}|$ is the number of items, x_k is a boolean variable that shows whether item k is stored in the knapsack or not, w_k is the weight of the item, v_k is the value of the item and W is the knapsack capacity.

In the corresponding instance of (3.5), the processing tree has two levels; that is, the tree has $|\mathcal{K}|$ number of leaf nodes, which can generate data, and the sink node where only the sink node can choose to cache data generated by the leaf nodes. Furthermore, the threshold γ is set to $\gamma = \sum_{k\in\mathcal{K}} y_k$ that enforces all the δ's to be one so that the constraint in (3.5b) can always be satisfied and thus be removed. We also set the cache size at the sink node to be W and the caching decisions of b_k, $\forall k \in \{1, \cdots, |\mathcal{K}|\}$ and the data sizes from all leaf nodes y_k, $\forall k \in \{1, \cdots, |\mathcal{K}|\}$ are one-to-one correspondent with x_k, $\forall k \in \{1, \cdots, |\mathcal{K}|\}$ and w_k, $\forall k \in \{1, \cdots, |\mathcal{K}|\}$ of the knapsack problem, respectively. The sum E_k^C in (3.4) becomes a constant since all δ's have been set to 1. As for the E_k^R in (3.3) can be reduced to:

$$E_k^{\mathrm{R}} = \sum_{i=0}^{1}\Bigg\{y_k(R_k-1)f(1)\Bigg\} - b_k\Bigg(\sum_{i=0}^{1}\Bigg\{y_k(R_k-1)f(1)\Bigg\} - y_k(R_k-1)\left(\frac{w_{ca}T}{R_k-1} + \varepsilon_{kT}\right)\Bigg). \tag{3.7}$$

Note that the first term in the above is a constant and can be removed from the objective function. By setting the caching cost in the instance of (3.5) to zero, (3.7) becomes:

$$E_k^{\mathrm{R}} = -b_k\Bigg(y_k(R_k-1)\Bigg\{\sum_{i=0}^{1} f(1) - \varepsilon_{kT}\Bigg\}\Bigg). \tag{3.8}$$

For the given value of v_k in the knapsack problem, we choose y_k, R_k, ε_{kT} so that (3.8) is given by:

$$E_k^{\mathrm{R}} = -b_k c_k. \tag{3.9}$$

where $c_k = y_k(R_k-1)\Big\{\sum_{i=0}^{1} f(1) - \varepsilon_{kT}\Big\} \geq 0$ as $f(1) \geq \varepsilon_{kT}$. As a result, solving the corresponding instance of (3.5) provides the solution to the knapsack problem. As the latter is NP-hard, so is (3.5). □

Remark 3.1. *The objective function E^{total} defined in* (3.5) *is monotonically increasing in the number of requests R_k for all $k \in \mathcal{K}$ provided that $\boldsymbol{\delta}$ and $\boldsymbol{b}$ are fixed.*

Notice that (3.2) is independent of R_k and (3.3) is linear in R_k, and its multipliers are positive. Hence, for any fixed $\boldsymbol{b}$ and $\boldsymbol{\delta}$, (3.4) increases monotonically with R_k.

Remark 3.2. *Given a fixed network scenario, if we increase the number of requests R_k for the data generated by leaf node k, then these data will be cached closer to the sink node or at the sink node, if there exists enough cache capacity, to reduce the overall energy consumption.*

For fixed $\boldsymbol{\delta}$, observe from (3.3) that energy consumption decreases if the cache is moved closer to the root since the nodes deep in the tree do not need to retransmit.

3.3.2. Relaxation of Assumptions

In our model, we make several assumptions for the sake of simplicity. In the following, we discuss the relaxation of these assumptions.

While we assume that the network is structured as a tree, this assumption can be easily relaxed as long as there exists a simple fixed path from each leaf node to the sink node. The tree structure represents a simple topology that captures the key parameters in the optimization formulation without the complexity introduced by a general network topology. Furthermore, for simplicity, we assume that all parameters across the nodes are identical, which is not necessary as seen from the cost function. We also assume that only leaf nodes generate data. However, our model can be extended to allow intermediate nodes to generate data at the cost of added complexity.

3.4. Variant of Spatial Branch-and-Bound Algorithm

In this section, we present a variant of the Spatial Brand-and-Bound algorithm (V-SBB). Instead of solving the MINLP problem (3.5) directly, we use V-SBB to solve a *standard form* of the original MINLP. We refer the reader to Appendix A.1 for an introduction to the *Symbolic Reformulation* [1] method that reformulates the MINLP (3.5) into a standard form needed by V-SBB.

Definition 3.1. *An MINLP problem is said to be in a* standard form *if it can be written as*

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}_{obj} \\ s.t. \quad & \mathbf{A}\mathbf{w} = \mathbf{b}, \\ & \mathbf{w}^l \leq \mathbf{w} \leq \mathbf{w}^U, \\ & \mathbf{w}_k \equiv \mathbf{w}_i \mathbf{w}_j, \quad \forall (i,j,k) \in \mathcal{T}_{bt}, \\ & \mathbf{w}_k \equiv \mathbf{w}_i / \mathbf{w}_j, \quad \forall (i,j,k) \in \mathcal{T}_{lft}, \end{aligned} \tag{3.10}$$

where the vector of variables $\boldsymbol{w}$ consists of continuous and discrete variables in the original MINLP. The sets $\mathcal{T}_{bt}$ and $\mathcal{T}_{lft}$ contain all relationships that arise in the reformulation. $\boldsymbol{A}$ and $\boldsymbol{b}$ are a matrix and a vector of real coefficients, respectively. The index obj denotes the *position of a single variable corresponding to the objective function value within the vector* $\boldsymbol{w}$.

Theorem 3.2. *The non-convex MINLP problem (3.5) can be transformed into a standard form.*

Due to space constraints, we relegate detailed reformulations (see Appendix A.1 for details of symbolic reformulation) and standard form of (3.5) to Appendix A.2.

Here, we give an example to illustrate the above reformulation process.

Example 2. Consider the same network in Example 1, the non-convex MINLP problem becomes

$$\begin{aligned}
\min_{\boldsymbol{\delta},\boldsymbol{b}} \quad & E^{\text{total}}(\boldsymbol{\delta},\boldsymbol{b}) = E_1^C + E_1^R \\
\text{s.t.} \quad & y_1\delta_{1,0}\delta_{1,1} \geq \gamma, \\
& b_{1,0}, b_{1,1} \in \{0,1\}, \\
& b_{1,0}y_1\delta_{1,0}\delta_{1,1} \leq S_0, \\
& b_{1,1}y_1\delta_{1,1} \leq S_1, \\
& b_{1,0} + b_{1,1} \leq 1.
\end{aligned} \tag{3.11}$$

$\delta_{1,0}\delta_{1,1}$ is a bilinear term. Based on symbolic reformulation rules, a new bilinear auxiliary variable $w_{1,0}^{\text{bt}}$ needs to be added. The first constraint in (3.11) is then transformed into $y_1 w_{1,0}^{\text{bt}} \geq \gamma$, which is linear in auxiliary variable $w_{1,0}^{\text{bt}}$. Similarly, we add $w_{1,0}^{\text{lft}}$ for linear-fractional term $\delta_{1,1}/\delta_{1,0}$ that appears in $f(\cdot)$. $b_{1,0}\delta_{1,0}\delta_{1,1}$ in the third constraint of (3.11) is a tri-linear term. Since $\delta_{1,0}\delta_{1,1}$ is replaced by $w_{1,0}^{\text{bt}}$, we obtain a bilinear term $b_{1,0}w_{1,0}^{\text{bt}}$. Again, based on symbolic reformulation rules, $b_{1,0}w_{1,0}^{\text{bt}}$ is replaced by a new auxiliary variable $\overline{w}_{1,0}^{\text{bt}}$. Similarly we add new auxiliary variables $\tilde{w}_{1,1}^{\text{b}}$, $\tilde{w}_{1,0}^{\text{bt}}$, $\tilde{w}_{1,0}^{\text{lft}}$ and $\tilde{w}_{1,1}^{\text{lft}}$. The objective function in (3.11) can be then expressed as a function of these new auxiliary variables. Therefore, the standard form of (3.11) is

$$\begin{aligned}
\min_{\boldsymbol{\delta},\boldsymbol{b}} \quad & w_{\text{obj}} \\
\text{s.t.} \quad & y_1 w_{1,0}^{\text{bt}} \geq \gamma, \\
& b_{1,0}, b_{1,1} \in \{0,1\}, \\
& y_1\overline{w}_{1,0}^{\text{bt}} \leq S_0, \\
& y_1\tilde{w}_{1,1}^{\text{bt}} \leq S_1, \\
& b_{1,0} + b_{1,1} \leq 1, \\
& w_{1,0}^{\text{bt}} = \delta_{1,1} \times \delta_{1,0}, \\
& w_{1,0}^{\text{lft}} = \delta_{1,1}/\delta_{1,0}, \\
& \overline{w}_{1,0}^{\text{bt}} = b_{1,0} \times w_{1,0}^{b},
\end{aligned}$$

$$
\begin{aligned}
&\tilde{w}^{\text{bt}}_{1,1} = b_{1,1} \times \delta_{1,1}, \\
&\tilde{w}^{\text{bt}}_{1,0} = \delta_{1,1} \times b_{1,0}, \\
&\tilde{w}^{\text{lft}}_{1,0} = b_{1,0}/\delta_{1,1}, \\
&\overline{w}^{\text{lft}}_{1,0} = b_{1,0} w^{\text{lft}}_{1,0}, \\
&\tilde{w}^{\text{lft}}_{1,1} = b_{1,1}/\delta_{1,1}, \\
&w_{\text{obj}} = y_1 \varepsilon_{1R} \delta_{1,1} + \varepsilon_{1T} y_1 w^{\text{bt}}_{1,0} + y_1 \varepsilon_{1C} w^{\text{lft}}_{1,0} - y_1 \varepsilon_{1C} \delta_{1,1} + y_1 \varepsilon_{1R} \varepsilon_{1T} y_1 \delta_{1,1} + y_1 \varepsilon_{1C}/\delta_{1,1} - \\
&y_1 \varepsilon_{1C} + y_1 (R_1 - 1) \Big(\varepsilon_{1R} \delta_{1,1} + \varepsilon_{1T} w^{\text{bt}}_{1,0} - \varepsilon_{1C} \delta_{1,1} + \varepsilon_{1C} w^{\text{lft}}_{1,0} - \varepsilon_{1R} \tilde{w}^{\text{bt}}_{1,0} - \varepsilon_{1T} \overline{w}^{\text{bt}}_{10} + \\
&\varepsilon_{1C} \tilde{w}^{\text{bt}}_{1,0} - \varepsilon_{1C} \overline{w}^{\text{lft}}_{1,0} \Big) + y_1 (R_1 - 1) \varepsilon_{1T} + y_1 w_{ca} T \overline{w}^{\text{bt}}_{1,0} + y_1 (R_1 - 1) \Big(\varepsilon_{1R} - \varepsilon_{1C} + \\
&\varepsilon_{1T} \delta_{1,1} + \varepsilon_{1C}/\delta_{1,1} - \varepsilon_{1R} b_{1,0} - \varepsilon_{1C} b_{1,0} - \varepsilon_{1T} \tilde{w}^{\text{bt}}_{1,0} - \varepsilon_{1C} \tilde{w}^{\text{lft}}_{1,0} - \varepsilon_{1R} b_{1,1} + \varepsilon_{1C} b_{1,1} - \\
&\varepsilon_{1T} \tilde{w}^{\text{bt}}_{1,1} - \varepsilon_{1C} \tilde{w}^{\text{lft}}_{1,1} \Big).
\end{aligned}
\tag{3.12}
$$

Through this reformulation, the non-convex and non-linear terms in the original problem are transformed into bilinear and linear fractional terms that can be easily used to compute the lower bound of each region in V-SBB, which are discussed in details later. This is the reason V-SBB requires reformulating the original problem into a standard form.

Theorem 3.3. *Reformulated problem and the original MINLP are equivalent.*

The proof is available in [1, Section 2, page 460].

Due to the reformulation, the number of variables in the reformulated problem is larger than in the original MINLP. In the following, we show that the number of auxiliary variables that arise from symbolic reformulation is bounded.

Theorem 3.4. *The number of auxiliary variables in the symbolic reformulation is $O(n^2)$, where $n = 2\Theta$ is the number of variables in the original formulation.*

Proof. From [83], a way to transform a general form optimization problem into a standard form (3.10) is through basic arithmetic operations on original variables. To be more specific, any algebraic expression results from the basic operators including the five basic binary operators, i.e., addition, subtraction, multiplication, division and exponentiation, and the unary operators, i.e., logarithms, etc. Therefore, in order to construct a standard problem consisting of simple terms corresponding to these binary or unary operations, new variables need to be added corresponding to these operations. From the symbolic reformulation process [83–85], any added variable results from the basic operations between two (including possibly the same) original variables or added variables. Hence, based on the basic operations, there are at most n^2 combinations of these variables, given that there are n variables in the original problem (3.5). Therefore, the number of added variables in the symbolic reformulation is bounded as $O(n^2)$. □

In the remainder of this section, we present the V-SBB to solve the equivalent problem.

Algorithm 3.1 Variant of Spatial Branch-and-Bound (V-SBB)

```
Step 1: Initialize $\phi^u := \infty$ and $\mathcal{L}$ to a single domain
Step 2: Choose a sub-region $\mathcal{R} \in \mathcal{L}$ using least lower bound rule
if  $\mathcal{L} = \emptyset$  Go to Step 6
end if
if  for chosen region $\mathcal{R}$, $\phi^{\mathcal{R},l}$ is infeasible or $\phi^{\mathcal{R},l} \geq \phi^u - \epsilon$  Go to Step 5
end if
Step 3: Obtain the upper bound $\phi^{\mathcal{R},u}$
if upper bound cannot be obtained or if $\phi^{\mathcal{R},u} > \phi^u$  Go to Step 4
else $\phi^u :=\phi^{\mathcal{R},u}$ and, from the list $\mathcal{L}$, delete all sub-regions $\mathcal{S} \in \mathcal{L}$ such that $\phi^{\mathcal{S},l} \geq \phi^u - \epsilon$
end if
if $\phi^{\mathcal{R},u} - \phi^{\mathcal{R},l} \leq \epsilon$  Go to Step 5
end if
Step 4: Partition $\mathcal{R}$ into new sub-regions $\mathcal{R}_{\text{right}}$ and $\mathcal{R}_{\text{left}}$
Step 5: Delete $\mathcal{R}$ from $\mathcal{L}$ and go to Step 2
Step 6: Terminate Search
if $\phi^u = \infty$  Problem is infeasible
else $\phi^u$ is $\epsilon$-global optimal
end if
```

3.4.1. Our Variant of Spatial Branch-and-Bound

The proposed spatial Branch-and-Bound method is a variant of the method proposed in [83] and is primarily tuned for solving our optimization problem (A.2) that is also the solution of (3.5). Our algorithm is different from [83] because

- We do not use any bounds tightening steps since it does not always guarantee faster convergence [86] and in case of our problem slowed down the process.
- By eliminating the bounds tightening step, we do not need to calculate the lower bound $\phi^{\mathcal{R},l}$ again separately and utilize the lower bound obtained in Step 2 for the chosen region $\mathcal{R}$, hence reducing the computational complexity of the algorithm.

Algorithm 3.1 provides an overview of the steps involved in spatial Branch-and-Bound algorithm. We describe some of the steps in Algorithm 3.1 in detail below.
Step 2: There are a number of approaches that can be used to choose a sub-region $\mathcal{R}$ from $\mathcal{L}$ [87]. Here we use the *least lower bound rule*, i.e., we choose a sub-region $\mathcal{R} \in \mathcal{L}$ that has the lowest lower bound among all the sub-regions, since it is a widely used and well researched method. The lower bound can be obtained by solving a convex relaxation of the problem in (A.2). Since our optimization problem in (3.5) and (A.2) contains only bilinear and linear fractional terms, we use McCormick linear over-estimators and under-estimators [88] (see Appendix A.4) to obtain a convex relaxation of all such terms. The resulting problem is then a Mixed Integer Linear Programming (MILP) problem that we solve using the *SCIP* solver [79]. The SCIP solver is a fast and well known solver for MILP problems. The sub-region with lowest lower bound is then used as the region to explore

for an optimum. The chosen regions' lower bound is used as $\phi^{\mathcal{R},l}$. If the convex relaxation is infeasible or if the obtained lower bound is higher than the existing upper bound ϕ^u of the problem, we *fathom* or delete the current region by moving to step 5.
Step 3*:* In step 3, we calculate the upper bound $\phi^{\mathcal{R},u}$ for the sub-region $\mathcal{R}$ chosen in Step 2. This can be done in a number of ways (see [83]), here we use a local MINLP solver, such as *Bonmin* [76] to obtain a local minimum for the sub-region since it performed better in terms of time than using local non-linear programming optimization with fixed discrete values or added discreteness constraints in our simulation settings. If the upper bound for the region $\phi^{\mathcal{R},u}$ cannot be obtained or if it is greater than ϕ^u then we move to Step 4 to further divide the region and search further for a better solution. Otherwise we set it as the current best solution ϕ^u and delete all the sub-regions whose lower bound is greater than the obtained upper bound since all such regions cannot contain the ϵ-global optimal solution. If the difference between the upper and lower bound for the region is within the ε-tolerance, the current sub-region need not to be searched further, then we delete the current sub-region by going to step 5, otherwise we move to step 4 for further searching in the space.
Step 4*:* Step 4 also known as the branching/partitioning step helps in partitioning/dividing a region to further refine the search for solution. In the branching step, we select a variable for branching/partitioning as well as the value of the variable at which the region is to be divided. There are a number of different rules and techniques that can be used for branching (see [87] for detailed discussion). Here we use the variable selection and value selection rule specified in [84], since it has been found efficient for our problem [84].

We branch on the variable that causes the maximum reduction in the feasibility gap between the solution of the convex relaxation (solution of Step 2) and the exact problem. To do so, the approximation error for the bilinear and linear fractional terms in (A.2) is calculated using (3.13a) and (3.13b), respectively, where $\mathcal{S}_2$ means the value of the variable obtained in Step 2. The variable with the maximum approximation error of all is chosen as the branching variable, because that tightens the gap between the relaxation and the exact problem [84]. This results in two candidate variables for branching i.e., w_i and w_j. If one of the variables is discrete (binary in our case) and the other is continuous then choose the discrete variable since it will result only in finite number of branches. However, if both variables are of the same type (either binary or continuous), then the branching variable is chosen using (3.14), i.e., we choose the variable w_b that has its value $w_b^{\mathcal{R}}$ closer to its range's midpoint. However, we first need to obtain the branching value for the candidate variables w_{bc}^{∇} (the value at which to branch). w_{bc}^{∇} should be between the upper and lower bounds of the variable in the region i.e. $w_{bc}^{\nabla,l} < w_{bc}^{\nabla} < w_{bc}^{\nabla,u}$. The rules for the choice of the branch point have been set in [84], however we restate them here for sake of completeness.

- Set w_{bc}^{∇} to the value obtained in Step 2, i.e., $w_{bc}^{\nabla} := w_{bc}^{\mathcal{S}_2}$.
- If any feasible upper bound $\phi^u = \phi(w_{bc}^*)$ has been obtained and $w_{bc}^{\nabla,l} < w_{bc}^* < w_{bc}^{\nabla,u}$,

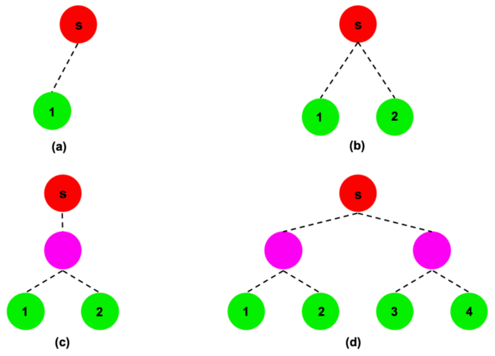

Figure 3.3.: Candidate network topologies used in the experiments: (a) one sink node and one leaf node; (b) one sink node and two leaf nodes; (c) one sink node, one intermediate node and two leaf nodes; and (d) one sink node, two intermediate nodes and four leaf nodes.

then $w_{bc}^{\nabla} := w_{bc}^{*}$ and stop the search for the value.

- If step 4 provided an upper bound $\phi^{\mathcal{R},u}$ for the subregion $\mathcal{R}$, then $w_{bc}^{\nabla} := w_{bc}^{\mathcal{R}}$.

After obtaining the branch point value, we have all the parameters required for (3.14) and can then choose the variable for branching.

$$E_{\text{bt}}^{ijk} = |w_k^{\mathcal{S}_2} - w_i^{\mathcal{S}_2} w_j^{\mathcal{S}_2}| \ \forall (i,j,k) \ \in \ \mathcal{T}_{\text{bt}} \tag{3.13a}$$

$$E_{\text{lft}}^{ijk} = \left| w_k^{\mathcal{S}_2} - \frac{w_i^{\mathcal{S}_2}}{w_j^{\mathcal{S}_2}} \right| \ \forall (i,j,k) \ \in \ \mathcal{T}_{\text{lft}} \tag{3.13b}$$

$$w_b = \text{arg min} \left\{ \left| 0.5 - \left(\frac{w_i^{\nabla} - w_i^{\mathcal{R},l}}{w_i^{\mathcal{R},u} - w_i^{\mathcal{R},l}} \right) \right|, \left| 0.5 - \left(\frac{w_j^{\nabla} - w_j^{\mathcal{R},l}}{w_j^{\mathcal{R},u} - w_j^{\mathcal{R},l}} \right) \right| \right\} \tag{3.14}$$

We partition the sub-region $\mathcal{R}$ into $\mathcal{R}_{\text{right}}$ and $\mathcal{R}_{\text{left}}$ and add $\mathcal{R}_{\text{right}}$, $\mathcal{R}_{\text{left}}$ into our region list $\mathcal{L}$. Then we move to *Step* 5 and delete the sub-region $\mathcal{R}$ from the list $\mathcal{L}$.

3.5. Convergence of Spatial Branch-and-Bound

The spatial branch-and-bound method guarantees convergence to ϵ-global optimality, which has been proven in [86]. However, for sake of completeness, we restate the proof in Appendix A.3.

3.6. Evaluation

We evaluate the performance of our joint communication, compression and caching (C3) optimization framework through a series of experiments on several network topologies as shown in Figure 3.3. Our goal is to study the performance of C3 and assess the improvement in energy efficiency that can be achieved by considering jointly the C3 costs when compared with only optimizing C2 cost. While highlighting the performance gain is valuable, characterizing the performance of C3 for different settings and parameters, and obtaining the optimal caching locations and data compression rates is also of importance. We also compare the performance of our V-SBB algorithm with several existing MINLP solvers (methods).

The highlights of the evaluation results are:

- Our C3 joint optimization framework improves energy efficiency by as much as 88% compared to the C2 optimization over communication and computation, or communication and caching. This reveals the significance of jointly considering the C3 energy costs for networks with limited energy supply.
- The improvement in energy efficiency using the C3 framework increases since the number of requests or the network size increase. Furthermore, as expected, data that has the largest number of requests is cached at the sink node or closer to the sink node in order to reduce communication costs.
- In comparison with different MINLP solvers, the V-SBB algorithm obtained an ϵ-global optimal solution. We vary the network parameters and find that the V-SBB is able to obtain a feasible solution for all settings. Existing solvers, such as SCIP, Baron, Bonmin, and Antigone are faster in obtaining solutions. However, they are either not able to obtain solutions in some of the settings or they provide an objective value higher than that obtained by our algorithm, particularly for low QoI threshold γ.

3.6.1. Methodology

To highlight the improvement in energy consumption by the C3 framework when compared with that in C2, we define energy efficiency as:

$$\Gamma = \frac{E^{\text{total}*}(\text{C2}) - E^{\text{total}*}(\text{C3})}{E^{\text{total}*}(\text{C2})} \times 100\%, \tag{3.15}$$

where $E^{\text{total}*}(\text{C3})$ and $E^{\text{total}*}(\text{C2})$ are the optimal energy costs under the C3 optimization framework in (3.5) and the C2 optimization, respectively. Γ reflects the reduction of energy consumption for the C3 over the C2. While the increase in energy efficiency using C3 framework is noteworthy, characterizing the magnitude of the improvement and the parameters that significantly impact the energy efficiency is important. Such a characterization identifies the *operating regions* for the network and accordingly facilitates development of heuristic algorithms for specific operating regions.

Setup: We implement the V-SBB in Matlab on a Core i7 3.40 GHz CPU with 16 GB RAM. Existing MINLP solvers like Bonmin (version 1.8.4), NOMAD (version 3.8.1), SCIP (version 4.0.0) and GA (version R2018a), are implemented with Opti-Toolbox [90], while Baron

Table 3.2.: Characteristics of the solvers used in this chapter.

Solver	**Characteristics**
Bonmin [76]	A deterministic approach based on Branch-and-Cut method that solves relaxation problem with Interior Point Optimization tool (IPOPT), as well as mixed integer problem with Coin or Branch and Cut (CBC).
NOMAD [77]	A stochastic approach based on Mesh Adaptive Direct Search Algorithm (MADS) that guarantees local optimality. It can be used to solve non-convex MINLP and has a relatively good performance.
GA [89]	A meta-heuristic stochastic approach that can be tuned to solve global optimization problems. We use Matlab *Optimization Toolbox*'s implementation.
SCIP [79]	One of the fastest, non-commercial, deterministic global optimization solver that uses branch-and-bound algorithm for solving MINLP problems.
Baron [78]	A deterministic global solver for MINLP problems that relies on Branch and Cut approach for solving MINLP problems.
Antigone [80]	A deterministic global solver for MINLP problems that relies on special structure of the problem and uses Branch and Cut approach to solve the problem.

Table 3.3.: Parameters used in simulations.

Parameter	**Value**	**Parameter**	**Value (Joules)**
y_k	1000	ε_{vR}	50×10^{-9}
R_k	100	ε_{vT}	200×10^{-9}
w_{ca}	1.88×10^{-6}	ε_{cR}	80×10^{-9}
T	10s	γ	$[1, \sum_{k \in \mathcal{K}} y_k]$

(version 15.6.5) and Antigone (version 1.0) are implemented in GAMS. We summarize the characteristics of these solvers in Table 3.2. Note that these solvers can be applied directly to solve the original optimization problem in (3.5), while our V-SBB solves the equivalent problem obtained through symbolic reformulation. The required reformulation is executed by a Java based module and we derive the bounds on the auxiliary variables. The V-SBB terminates when ϵ-optimality is obtained or a computation timer of 7200 seconds expires. When the timer expires, the last feasible solution is taken as the best solution. We set ϵ=0.001 in our study. Our simulation parameters are provided in Table 3.3, which are the typical values used in the literature [3, 30, 31]. For the results shown in Tables 3.4, 3.5, 3.6, and 3.7, we vary the QoI threshold value of γ in (3.5) between 1%, 25%, 50%, 75% and 100% of the total data produced by all leaf nodes.

3.6.2. Efficacy of the C3 Framework

Figure 3.4 shows that energy consumption increases with the number of requests for differ-

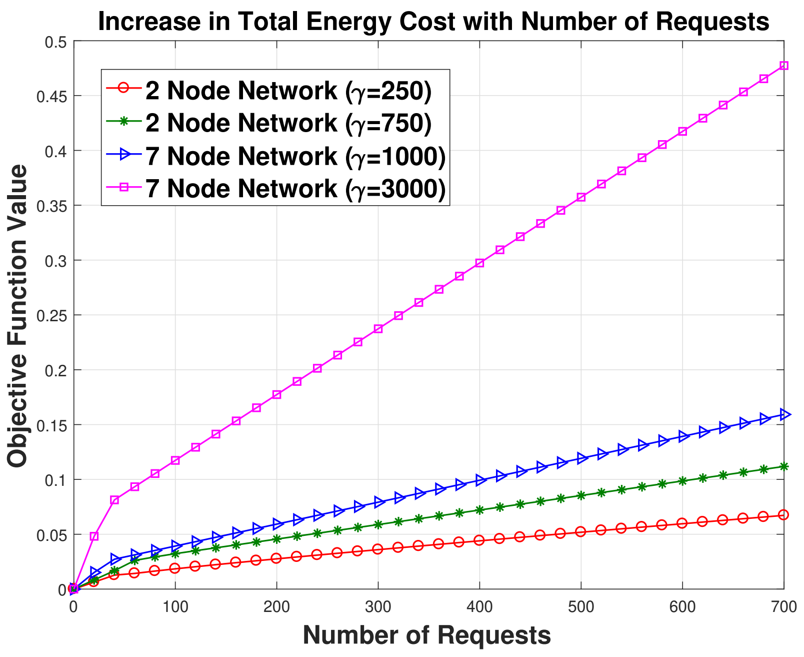

Figure 3.4.: Total energy costs vs. number of requests.

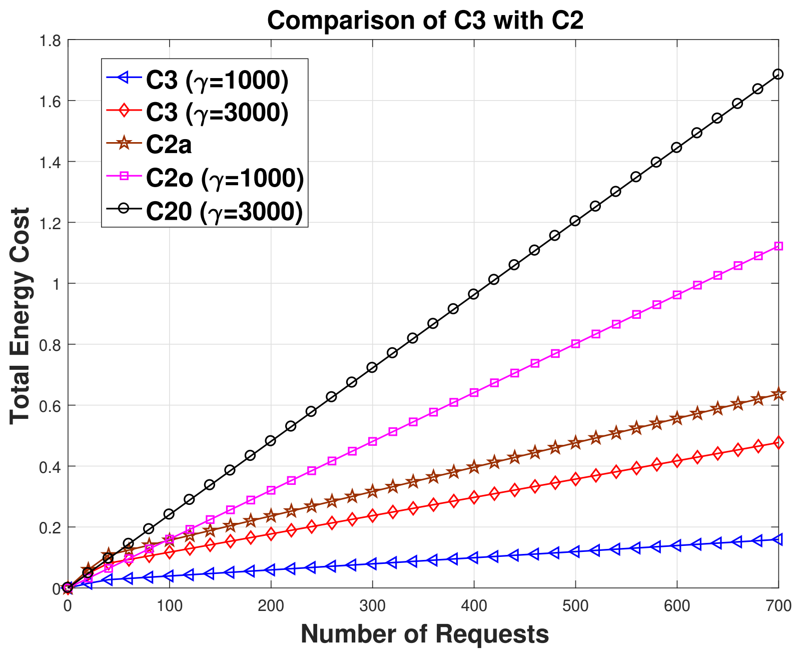

Figure 3.5.: Comparison of C3 and C2 optimization for the seven node network in Figure 3.3.

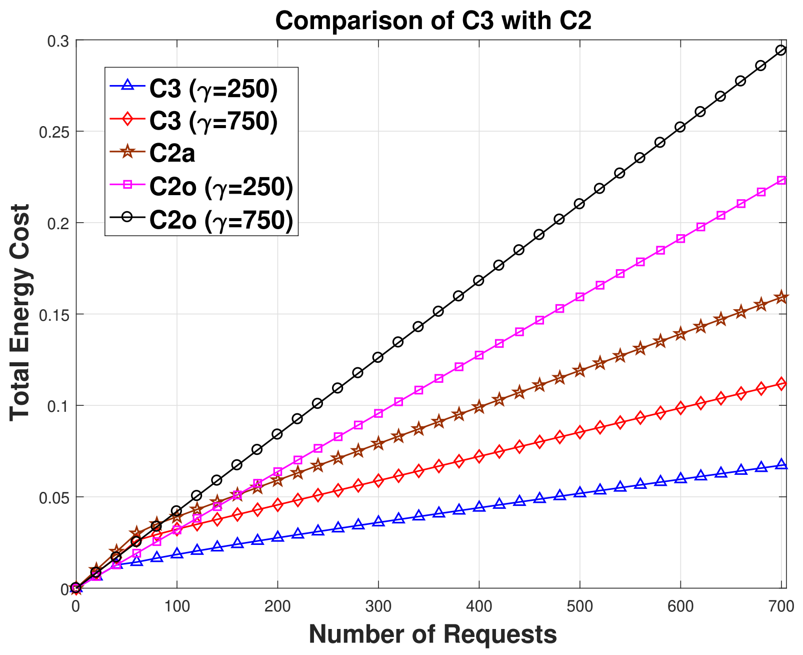

Figure 3.6.: Comparison of C3 and C2 optimization for the two nodes network in Figure 3.3.

ent network and compression settings. We observe that the total energy cost increases, as the number of requests increases, as reflected in Remark 3.1. An important observation is that the increase in the energy cost is large initially. However, when data are cached (i.e., when the number of requests reaches $\approx$ 40), the slope of energy increase decreases. This is because transmission cost is usually much larger than caching cost (using the energy proportional model for caching [4]) and once data are cached, the cached copy is used to satisfy other requests.

We compare the total energy costs under the joint C3 optimization with those under the C2 optimization. We consider two cases for the C2 optimization: (i) C2o (Communication and Computation), where we set $S_v = 0$ for each node to avoid any data caching; (ii) C2a (Communication and Caching), where we set $\gamma = \sum_{k \in \mathcal{K}} y_k$, which is equivalent to $\delta_v = 1$, $\forall v \in V$, i.e., no computation. Comparisons between C3, C2o and C2a are shown in Figure 3.5. For the parameters that we used in the simulation, the energy cost for the C3 joint optimization is lower than that for C2o optimization for the same parameter setting. This highlights the improvement that can be achieved using the C3 optimization. In other words, although C3 incurs caching costs, it may significantly reduce communication and computation energy costs, which in turn brings down total energy cost. Similarly, the C3 optimization outperforms C2a. Energy efficiency ((3.15)) is as much as 88% better for the C3 optimization over C2. These trends are observed for other network topologies. Figure 3.6 shows the improvement of C3 over C2 for a two-node network. Energy efficiency is as much as 70% better for the C3 versus C2. The results for three node and four nodes networks are presented in Tables 3.5 and 3.6.

Table 3.4.: The best solution to the objective function (Obj.) and algorithm run time for two nodes network.

Solver	$\gamma = 1$		$\gamma = 250$		$\gamma = 500$		$\gamma = 750$		$\gamma = 1000$	
	Obj.	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)
Bonmin	100×10^{-4}	0.076	180×10^{-4}	0.07	260×10^{-4}	0.071	320×10^{-4}	0.077	390×10^{-4}	0.102
NOMAD	120×10^{-4}	1.036	380×10^{-4}	0.739	330×10^{-4}	0.640	380×10^{-4}	0.203	390×10^{-4}	0.263
GA	100×10^{-4}	0.286	180×10^{-4}	2.817	260×10^{-4}	7.670	420×10^{-4}	11.020	640×10^{-4}	3.330
SCIP	Infeasible	0.07	120×10^{-5}	0.07	500×10^{-5}	0.05	110×10^{-4}	0.087	390×10^{-4}	0.05
Baron	100×10^{-4}	0.91	18×10^{-3}	0.79	26×10^{-3}	0.77	32×10^{-3}	0.87	39×10^{-3}	0.49
Antigone	10×10^{-3}	0.195	18×10^{-3}	0.18	26×10^{-3}	0.175	32×10^{-3}	0.19	39×10^{-3}	0.2
V-SBB	10×10^{-3}	18.231	18×10^{-3}	17.389	26×10^{-3}	12.278	32×10^{-3}	7.327	39×10^{-3}	19.437

Table 3.5.: The value of objective function (Obj.) and algorithm run time for three node network.

Solver	$\gamma = 1$		$\gamma = 500$		$\gamma = 1000$		$\gamma = 1500$		$\gamma = 2000$	
	Obj.	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)
Bonmin	500×10^{-5}	0.26	100×10^{-4}	0.14	190×10^{-4}	0.10	280×10^{-4}	0.10	383×10^{-4}	5.56
NOMAD	450×10^{-4}	12.42	250×10^{-4}	11.14	330×10^{-4}	9.30	290×10^{-4}	46.41	380×10^{-4}	5.45
GA	500×10^{-5}	0.69	250×10^{-4}	26.11	190×10^{-4}	16.85	340×10^{-4}	40.56	440×10^{-3}	10.34
SCIP	500×10^{-7}	4.96	560×10^{-6}	0.16	540×10^{-6}	0.18	280×10^{-4}	0.07	380×10^{-4}	0.05
Baron	500×10^{-5}	0.1	100×10^{-4}	0.09	190×10^{-4}	0.09	280×10^{-4}	0.1	383×10^{-4}	0.1
Antigone	500×10^{-5}	0.11	100×10^{-4}	0.09	190×10^{-4}	0.08	280×10^{-4}	0.21	380×10^{-4}	1.51
V-SBB	500×10^{-5}	46.1	190×10^{-4}	45.34	190×10^{-4}	8.1	280×10^{-4}	56.3	383×10^{-4}	12.2

Remark 3.3. *Note that the above results are based on parameter values typically used in the literature, as shown in Table 3.3. From our analysis, it is clear that the larger the ratio between ε_{vT} and ε_{vR}, ε_{vC}, the larger the improvement provided by the C3 formulation.*

3.6.3. Comparison with other solution techniques

We compare the performance of our proposed V-SBB with other MINLP solvers with respect to:

Objective Function Value of the Best Solution

We compare the performance of the V-SBB with several solvers in the literature for the networks in Figure 3.3. The results for two, three, four and seven-node networks are presented in Tables 3.4, 3.5, 3.6 and 3.7, respectively. We observe that V-SBB, Bonmin, SCIP, Antigone, and Baron achieve comparable performance for large values of γ, while V-SBB outperforms other algorithms for small values of γ (which will be discussed in detail later). Furthermore, Bonmin and SCIP cannot generate a feasible solution even when it exists for some cases. For the case of Bonmin, there are a number of probable reasons for such a feasibility related problem: a) For MINLP problems with non-convex functions, Bonmin relies on heuristic options and does not guarantee ϵ-global optimality [91]. The heuristics can cause such problems; b)The Branch-and-Cut method, used by Bonmin, is based on the outer-approximation (OA) algorithm [92]. For MINLPs with non-convex functions, OA constraints do not necessarily result in valid inequalities for the problem. Hence the branch-and-cut method in Bonmin sometimes deletes search regions where a better solution exists. NOMAD and GA in general yield a higher objective-function value than V-SBB does. This is because both NOMAD and GA are based on a stochastic approach that does not guarantee obtaining the ϵ-global optimum.

Algorithm Run Time

The time taken to obtain the best solution is important in practice. The time an algorithm requires to obtain its best solution as discussed in Section 3.6.3 is shown in Tables 3.4-3.7. It can be seen that Bonmin, Antigone, Baron and SCIP (when it is able to provide a solution) are the fastest methods. However, Bonmin, and SCIP sometimes cannot find a solution even though it exists. V-SBB takes longer to obtain a better solution, because our reformulation introduces auxiliary variables and additional linear constraints. The reformulation, however, also assists in obtaining an ϵ-global optimal solution.

Sensitivity

As shown in Tables 3.4 to 3.7, Bonmin is faster but does not produce feasible solutions in some cases. Recall from (3.5) that the threshold value of QoI γ determines the acceptable

Table 3.6.: The value of objective function (Obj.) and algorithm run time for four node network.

Solver	$\gamma = 1$		$\gamma = 500$		$\gamma = 1000$		$\gamma = 1500$		$\gamma = 2000$	
	Obj.	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)
Bonmin	200×10^{-5}	0.36	200×10^{-4}	0.11	390×10^{-4}	0.11	600×10^{-4}	0.10	800×10^{-4}	0.16
NOMAD	300×10^{-5}	112.5	230×10^{-4}	97.68	400×10^{-4}	59.86	600×10^{-4}	52.8	100×10^{-3}	2.28
GA	400×10^{-5}	1.01	200×10^{-4}	24.94	400×10^{-4}	13.02	120×10^{-3}	27.7	140×10^{-3}	35.33
SCIP	200×10^{-5}	7.05	200×10^{-4}	1999.4	400×10^{-6}	2.00	900×10^{-5}	0.43	400×10^{-4}	0.16
Baron	204×10^{-5}	0.52	200×10^{-4}	2.69	390×10^{-4}	0.89	600×10^{-4}	0.16	780×10^{-4}	0.1
Antigone	200×10^{-5}	21.2	200×10^{-4}	0.26	420×10^{-4}	0.18	600×10^{-4}	0.1	780×10^{-4}	0.08
V-SBB	200×10^{-5}	452	200×10^{-4}	411	390×10^{-4}	341	600×10^{-4}	445	780×10^{-4}	413

Table 3.7.: The best solution to the objective function (Obj.) and algorithm run time for seven nodes network.

Solver	$\gamma = 1$		$\gamma = 1000$		$\gamma = 2000$		$\gamma = 3000$		$\gamma = 4000$	
	Obj.	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)	**Obj.**	**Time** (s)
Bonmin	200×10^{-6}	0.214	390×10^{-4}	0.164	780×10^{-4}	0.593	117×10^{-3}	0.167	156×10^{-3}	0.212
NOMAD	400×10^{-5}	433.988	121×10^{-3}	381.293	108×10^{-3}	203.696	158×10^{-3}	61.093	181×10^{-3}	26.031
GA	430×10^{-4}	44.538	960×10^{-4}	30.605	164×10^{-3}	44.970	226×10^{-3}	17.307	303×10^{-3}	28.820
SCIP	NS	7200	NS	7200	NS	4829.4	NS	7200	156×10^{-3}	1.37
Baron	200×10^{-6}	0.74	390×10^{-4}	1002.14	780×10^{-4}	7200	117×10^{-3}	3.41	156×10^{-3}	0.15
Antigone	200×10^{-6}	3.57	390×10^{-4}	0.38	810×10^{-4}	0.34	117×10^{-3}	0.32	156×10^{-3}	0.13
V-SBB	100×10^{-6}	1871.403	390×10^{-4}	25.101	780×10^{-4}	30.425	117×10^{-3}	23.706	156×10^{-3}	19.125

Table 3.8.: Comparison between V-SBB and Bonmin for smaller values of γ in seven node network.

Solver	γ =1		γ=3		γ =5		γ =8		γ =50	
	Obj.	**Time (s)**	**Obj.**	**Time (s)**	**Obj.**	**Time (s)**	**Obj.**	**Time (s)**	**Obj.**	**Time (s)**
Bonmin	210×10^{-6}	0.214	300×10^{-6}	0.211	300×10^{-6}	0.224	500×10^{-6}	0.23	210×10^{-5}	0.364
Antigone	201×10^{-6}	3.57	310×10^{-6}	2.47	395×10^{-6}	6.53	510×10^{-6}	15.61	210×10^{-5}	2.71
Baron	200×10^{-6}	0.74	310×10^{-6}	4846	390×10^{-6}	7200	500×10^{-6}	7200	210×10^{-5}	7200
V-SBB	100×10^{-6}	1871	150×10^{-6}	2330	190×10^{-6}	1243	470×10^{-6}	1350	200×10^{-5}	3325

Table 3.9.: Infeasibility of Bonmin for networks in Figure 3.3.

Networks	(a)	(b)	(c)	(d)
# of test values	1000	2000	2000	4000
# of infeasible solutions	0	0	1	216
Infeasibility (%)	0	0	0.05	5.4

degree of data compression. When the γ value is small, the system can compress much less of the data relative to the cases with large values of γ. For this reason, we further examine the sensitivity of Bonmin performance with respect to γ as follows. We fix all other parameters in Table III, except varying the value of γ. Specifically, for each network topology (a) to (d) in Figure 3.3, we choose the maximal value of γ and then run the Bonmin method for all possible integer values of γ from 1 to the maximal value. That is, the number of tests (cases) for the Bonmin method equals the maximal value for each topology. From the results in Table 3.9, we see that as the network size grows, the likelihood that the Bonmin method fails to produce a feasible solution increases. For the topology (d) and γ=4000, the Bonmin method does not yield feasible solutions in 5.4% of the test cases.

Furthermore, from our extensive numerical experiments, Bonmin and Antigone (and Baron to a much lesser extent) can quickly provide feasible solutions for smaller values of γ, but we observe that the solutions by these techniques are not as good as those by the proposed V-SBB in terms of the objective-function value of the solutions. Specifically for the cases in Table 3.8, the energy consumption for the V-SBB outperforms Bonmin, Antigone and Baron by as much as 52.45%, 50%, and 50%, respectively when searching for an ϵ-global optimum. Note that Baron results for γ=5, 8 and 50 in Table 3.8 represent the best solutions obtained so far by Baron when it reaches the run-time limit of 7200s (2 hours).

3.7. Summary and Conclusions

In this chapter, we investigated energy consumption among communication, computation and caching (C3) with QoI guarantees in communication networks. We first formulated an optimization problem that characterizes the energy costs for communication, computation and caching. This optimization problem is an MINLP with non-convex functions, which is hard to solve in general. We then proposed a variant of the spatial Branch-and-Bound (V-SBB) algorithm, which can solve the MINLP with an ϵ-global optimality guarantee. Numerically, we observed that the C3 optimization, which to the best of our knowledge has not been investigated in the literature, leads to energy reduction as large as 88% compared to either of the widely studied C2 optimizations. Furthermore, we compared the performance of the proposed V-SBB with some of the other solvers and show the V-SBB is a viable and robust approach for handling the C3 framework.

4. A Distributed Bargaining-Theoretic Approach For Multi-Objective Resource Sharing

In the previous chapter, we studied the problem of minimizing network energy consumption and showed how our single objective optimization framework can enable optimally choosing caching locations and compression rates to reduce the energy consumption in the network. With the increasing demands for resources particularly due to big data and machine learning, paradigms, such as cloud and edge computing are emerging as viable systems that can provide the required resources. However, resources may belong to different service providers that may have different objectives both in cloud and edge computing settings. Single objective optimization techniques cannot be used for such systems. Therefore, we direct our attention to the multi-objective resource sharing problem. In this chapter, we model the resource sharing problem among different SPs using *Bargaining theory* for a particular class of concave utility functions. We show that we can achieve Pareto optimality and fairness by relying on the Nash Bargaining Solution (NBS). Furthermore, since a distributed solution is highly desirable in such a multi-service provider setting, we show that the strong duality property holds and propose a distributed algorithm to compute the NBS. Our proposed solution can be used in a number of distributed multi-service provider settings, such as cloud and edge computing. However, for the current and remaining chapters in this thesis, we apply our developed multi-objective frameworks to an edge computing setting to highlight their efficacy.

4.1. Introduction

4.1.1. Motivation

Edge computing has received much attention recently since it enables on-demand provisioning of computing resources for different applications and tasks at the network edge [93–95]. One of the fundamental advantages of edge computing is that it can provide resources with low latencies when compared with traditional cloud computing architectures [94]. The demand for edge computing has further increased due to the advent of the Internet of Things (IoT) [96] and wide-scale use of machine and deep learning [97] in different industries since these learning-based models can be trained and run using edge computing nodes with adequate storage and computing power [98].

A typical edge computing system consists of a large number of edge nodes that have different types of resources. An edge service provider earns a *utility* for allocating resources to different applications and guarantees to provide resources according to a *Service Level Agreement* (SLA). However, when compared with cloud computing systems and data centers, resources in an edge setting are limited. Therefore, optimal management and use of these limited resources has been an active area of research. Even when resources are available, allocating them to applications with the goal of maximizing overall edge SP *utility* is a difficult problem. Furthermore, the aforementioned resource intensive paradigms, such as deep learning, and data analytics exacerbate the problem by challenging the scalability of traditional resource allocation techniques.

Edge SPs typically attempt to provide enough resources to different applications at their edge nodes to meet the peak demand. However, it is highly likely that resources of one SP will be over-utilized, while other edge SPs' resources will be under-utilized. For example, an edge SP provisions resources to an application at an edge node that is physically closest to the requesting application. However, if the closest edge node has a resource deficit or is overloaded, the request can be satisfied through edge SP's next closest edge node that may physically be at a distant location or deep in the network, such as at the data center [18]. This incurs high cost and a larger latency that may not be acceptable for delay constrained applications. One possible solution is to create a shared resource pool with other SPs that are physically closer [15–18]. Such a resource pool enables SPs to share and use resources whenever needed to meet their dynamic demands. This cooperation and resource sharing among SPs seem beneficial for them, because it is unlikely that resources of different SPs will be simultaneously over-utilized.

As seen in Figure 4.1, SPs 2 and 3 satisfy all their applications and have resource surpluses whereas SP 1 has resource deficits when working alone. However, through cooperation among these SPs that results in a coalition, SP 1 satisfies its applications by using resources of other coalition partners resulting in an improved utility for all partners. It is evident that if different SPs do not share resources, different application requests cannot be satisfied. Therefore, there is a need for frameworks that allow resource sharing among these SPs. Furthermore, a distributed solution may be favorable when compared with a centralized solution, because

- An attacker can target the central system causing the entire cooperation among SPs to fail.
- SPs need to reach a consensus for choosing the *central* node that runs the resource sharing algorithm.
- A centralized framework requires a large amount of information to be transmitted to the central node.

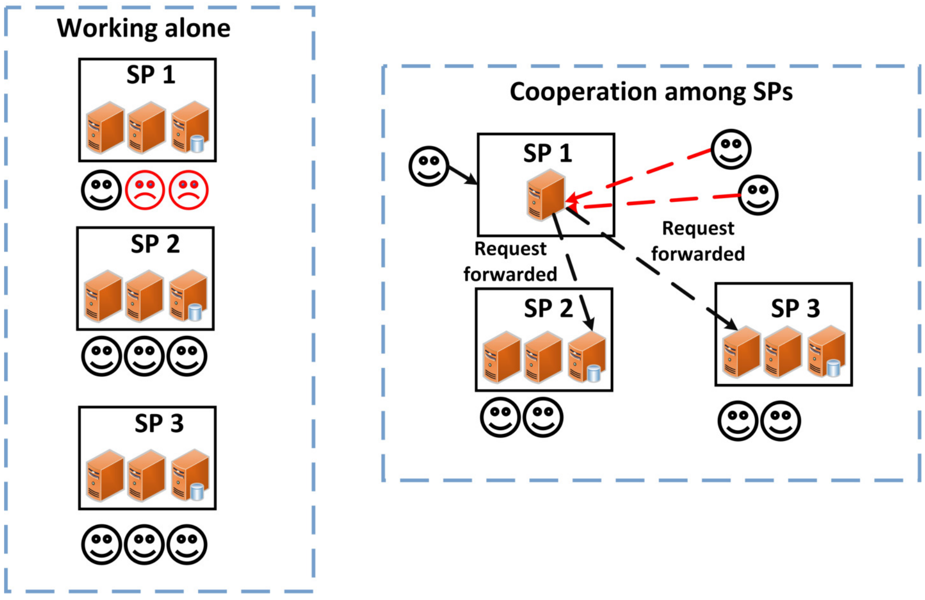

Figure 4.1.: Cooperation among SPs

4.1.2. Methodology and Contributions

We consider a number of edge SPs that share their resources to satisfy resource requests of different applications as shown in Figure 4.1. We formulate such resource sharing and allocation among SPs as a multi-objective optimization (MOO) problem, for which our goal is to achieve a Pareto optimal solution. Furthermore, since Pareto solutions are spread over the *Pareto frontier* [24], choosing a single solution among them is challenging. Since the *Nash Bargaining Solution* (NBS) [28] guarantees the provision of a fair and Pareto optimal solution to such an MOO problem, we develop a distributed NBS-based framework for resource sharing and allocation. *To the best of our knowledge, this is the first generic NBS-based framework for resource sharing and allocation among edge SPs.*

The main contributions of this chapter are:

1. We present an NBS based resource sharing framework for edge SPs with different objectives in allocating resources to meet dynamic demands. Our framework also considers practical engineering constraints, such as communication costs and resource fragmentation[1].

2. We show that SPs can benefit from sharing resources with other SPs, because they can earn a higher profit and improve the average application satisfaction.

3. We show that the *strong-duality* property holds for the formulated problem, which enables us to propose a distributed algorithm to obtain the NBS.

[1]We define resource fragmentation as the scenario in which an application receives resources from multiple SPs rather than a single SP.

4. We evaluate the performance of our algorithms using synthetic and real world traces. Results show that resource sharing improves the utilities of SPs, increases resource utilization and enhances average user (application) satisfaction. Furthermore, our results show that the profit sharing among SPs is also fair.

The rest of this chapter is structured as follows. We present the system model in Section 4.2. In Section 4.3, we describe our proposed NBS framework for resource sharing and allocation. In Section 4.3.2, we present a distributed algorithm for obtaining the NBS. In Section 4.4, we present simulation results of the proposed framework for different settings. We present the summary and conclude the chapter in Section 4.5.

4.2. System Model

Let $\mathcal{N} = \{1, 2, \cdots, N\}$ be the set of all edge SPs. We assume that each provider has a set of $\mathcal{J} = \{1, 2, \cdots, J\}$ types of resources, such as communication, computation and storage resources. $C_n = \{C_{n,1}, \cdots, C_{n,J}\}$, with $C_{n,j}$ denoting the amount of type j resources available at service provider n. Each service provider n has a set of native applications $\mathcal{M}_n = \{1, 2, \cdots, M_n\}$. The set of all applications that request resources from all service providers is given by $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cdots \cup \mathcal{M}_N$, where we assume $\mathcal{M}_n \cap \mathcal{M}_{n'} = \emptyset$, $\forall n \neq n'$, i.e., each application initially demands resources from only its native service provider. We also define $\overline{\mathcal{M}}$ to represent $\mathcal{M} \backslash \mathcal{M}_n$ and $\overline{\mathcal{N}}$ to represent $\mathcal{N} \backslash \{n\}$. Every SP $n \in \mathcal{N}$ has a request (requirement) matrix $\mathbf{R}_n$,

$$\mathbf{R}_n = \begin{bmatrix} \mathbf{r_n^1} \\ \vdots \\ \mathbf{r_n^{M_n}} \end{bmatrix} = \begin{bmatrix} r_{n,1}^1 & \cdots & \cdots & r_{n,J}^1 \\ \vdots & \vdots & \vdots & \vdots \\ r_{n,1}^{M_n} & \cdots & \cdots & r_{n,J}^{M_n} \end{bmatrix}. \tag{4.1}$$

Here $r_{n,j}^m$ is the amount of resource j that application $m \in \mathcal{M}_n$ requires. When an edge SP is working alone (no sharing of its resources with any other service provider), its objective is to maximize its utility by allocating resources to its native applications. A service provider n earns a utility $u^m(x_{n,j}^m)$ by allocating $x_{n,j}^m$ amount of resource j to application $m \in \mathcal{M}_n$, where the vector $\mathbf{x}_n^m = [x_{n,1}^m, x_{n,2}^m, \cdots, x_{n,J}^m]^T$. Table 4.1 contains notations used throughout this chapter. We present the optimization formulation for a single service provider in Section 4.2.1, followed by a formulation for the multiple service provider problem in Section 4.2.2.

4.2.1. Problem Formulation for Single Service Provider

We first present the resource allocation problem for a stand-alone single service provider (i.e., no resource sharing with other service providers). For a single SP $n \in \mathcal{N}$, the allocation

Table 4.1.: List of notations used throughout chapter 4

Notation	**Description**
$\mathcal{N}, N, n$	Set, number and index of SPs
$\mathcal{J}, J, j$	Set, number and index of resources
$\mathcal{M}, M, m$	Set, number and index of applications
$\mathcal{M}_n$	Set of native applications at SP n
C	Capacity vector of all SPs
C_n	Capacity vector of SP n
$C_{n,j}$	Capacity of resource j at SP n
$\mathbf{R}_n$	Request matrix at SP n
$r^m_{n,j}$	Request of application m for resource j from SP n
$x^m_{n,j}$	Allocation decision of resource j for application m at SP n
$\mathbf{x}^m_n$	Allocation decision vector for application m at SP n when working alone, i.e., $\mathbf{x}^m_n = [x^m_{n,1}, \cdots, x^m_{n,J}]^T$
$\mathbf{X}_n$	Allocation decision for SP n in the resource sharing case
$\mathbf{X}$	Allocation decision for the entire set of SPs
$u^m(x^m_{n,j})$	Utility SP n earns by allocating resource j to application m
$u^m(\mathbf{x}^m_n)$	Utility SP n earns by allocating vector of resources to application m

decision consists of vectors $\mathbf{x}^1_n, \cdots, \mathbf{x}^{M_n}_n$. The optimization problem is:

$$\max_{\mathbf{x}^1_n, \cdots, \mathbf{x}^{M_n}_n} \quad \sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} u^m(x^m_{n,j}), \tag{4.2a}$$

$$\text{s.t.} \quad \sum_{m} x^m_{n,j} \leq C_{n,j}, \quad \forall j \in \mathcal{J}, \tag{4.2b}$$

$$x^m_{n,j} \leq r^m_{n,j}, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}, \tag{4.2c}$$

$$x^m_{n,j} \geq 0, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}. \tag{4.2d}$$

The goal of a single service provider in solving this single objective optimization (SOO) problem, as mentioned earlier, is to maximize its utility by appropriately allocating resources. The first constraint (4.2b) indicates that allocated resources cannot exceed capacity. The second constraint (4.2c) reflects that allocated resources should not exceed the requested amounts. The last constraint, (4.2d) says the allocation cannot be negative. However, it is possible that a service provider n may earn a larger utility by providing its resources to applications of other service providers or it may not have sufficient resources to satisfy requests of all its native applications. On the other hand, there may be another service provider $n' \in \mathcal{N} \backslash \{n\}$ that may have a surplus of resources, which can be "rented" by service provider n. Below, we discuss resource sharing among these service providers.

4.2.2. Problem Formulation For Multiple Service Providers

Allowing resource sharing among service providers, while considering their objectives, could improve resource utilization and application satisfaction. By sharing its resources, edge SP

n earns a utility,

$$u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m) - D^m(x_{n,j}^m), \tag{4.3}$$

after allocating $x_{n,j}^m$ for application $m \in \mathcal{M}_{n'}$ where $z_j^m = \sum_{n' \in \overline{\mathcal{N}}} x_{n',j}^m$. The term $u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)$ captures the incremental increase in utility of provider n due to the allocation of additional resource $x_{n,j}^m$. $D^m(x_{n,j}^m)$ represents the communication cost of serving application $m \in \overline{\mathcal{M}}$ by using resource of type j at provider n rather than its native service provider n'. We assume that $u^m(x_{n,j}^m) = u^m(x_{n',j}^m)$ when $x_{n,j}^m = x_{n',j}^m$. We also let

$$\mathbf{x}^m = [\sum_{n \in \mathcal{N}} x_{n,1}^m, \sum_{n \in \mathcal{N}} x_{n,2}^m, \cdots, \sum_{n \in \mathcal{N}} x_{n,J}^m]^T,$$

i.e., the total resource allocated to any application $m \in \mathcal{M}$ is the sum of resources allocated to application m from all service providers. The resource sharing and allocation algorithm, based on resource requests and capacities of service providers, has to make an allocation that optimizes utilities of all service providers $n \in \mathcal{N}$ and satisfy user requests as well. The allocation decision is given by $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$, where $\mathbf{X}_n$, $\forall n \in \mathcal{N}$ is given by:

$$\mathbf{X}_n = \begin{bmatrix} \mathbf{x}_n^1 \\ \vdots \\ \mathbf{x}_n^{|\mathcal{M}|} \end{bmatrix} = \begin{bmatrix} x_{n,1}^1 & \cdots & \cdots & x_{n,J}^1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1}^{|\mathcal{M}|} & \cdots & \cdots & x_{n,J}^{|\mathcal{M}|} \end{bmatrix}. \tag{4.4}$$

Each service provider aims to maximize the sum of utilities by allocating its resources to its native applications, and allocating resources to applications belonging to other service providers. Each provider $n \in \mathcal{N}$ intends to solve the following optimization problem.

$$\begin{aligned} \max_{\mathbf{X}_n} \quad & \sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m) \Big) + \Big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - \\ & D^l(x_{n,j}^l) \big) \Big), && \text{(4.5a)} \\ \text{s.t.} \quad & \sum_m x_{n,j}^m \leq C_{n,j}, \quad \forall\, j \in \mathcal{J}, && \text{(4.5b)} \\ & \sum_{n' \in \mathcal{N}} x_{n',j}^m \leq r_{n,j}^m, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, && \text{(4.5c)} \\ & x_{n,j}^m \geq 0, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, && \text{(4.5d)} \\ & u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l) \geq 0, \forall l \in \overline{\mathcal{M}}, j \in \mathcal{J}. && \text{(4.5e)} \end{aligned}$$

The first part of (4.5a) represents the utility earned by an SP providing resources to the native applications, whereas the second part describes the utility earned by providing resources to non-native applications. Note that constraint (4.5b) indicates that the total allocated resources cannot exceed the resource capacity of the service providers. (4.5c) states that the total amount of resources allocated to any application using the resource-sharing framework cannot exceed the amount of requested resources. (4.5d) says that the resource allocation cannot be negative whereas (4.5e) indicates that the incremental increase in utility earned by providing resources to non-native applications should be non-negative. $\mathbf{z}$ for any SP n in the objective function is fixed, and the objective function is concave only in $\mathbf{x}$. Solving (4.5) is not straightforward since each provider needs to consider the allocations at other SPs.

4.2.3. Assumptions

In our model, we assume that each utility is a concave injective function for which the inverse of the first derivative exists, such as $(1 - e^{-x})$. Strictly speaking, our centralized NBS framework requires only concave injective utility functions [20]. However, the existence of the inverse of the first derivative is required for the distributed NBS (see details in Section 4.3). The communication cost is a convex function, hence the objective function in (4.5) is concave. All resources are fully utilized in the optimal solution. However, there are enough resources to provide a positive utility to all service providers when sharing resources. This is a realistic assumption as the demand for resources is usually more than the supply. Furthermore, we assume that when SPs share resources, there exist solutions that are better than when they are all working alone. This assumption can be relaxed, that is, if certain SPs cannot improve their utility using the bargaining solution, they will not participate in the resource sharing framework. However, the framework can still be used for the remaining SPs.

4.2.4. Choice of Utility Function

While our framework works with any utility that satisfies the conditions in Section 4.2.3, choosing a suitable utility function along with the communication cost can minimize *resource fragmentation*. Generally, a concave utility has a steeper slope at the start that becomes flatter as more resources are allocated, i.e., the rate of increase in the payoff for allocating resources reduces with an increase in the amount of allocated resources as shown in Figure 4.2. It is evident that the utility drastically increases when the allocated resource increases from 0 to 1. However, the rate of increase in the utility decreases as more resources are allocated. This can serve as the impetus for SPs to allocate only a fraction of the requested resources. For example, an SP may have two different applications each asking for 2 units of resources. However, the SP only has two units of resources. Therefore, ideally, the SP should allocate these two units to one of the applications (to reduce resource fragmentation) and borrow resources from other SPs for the other application. However, due to the aforementioned nature of concave utility, the SP may allocate 1 unit to each of the applications in order to maximize its utility. Hence, both the applications do not get all the resources they needed, and have to ask another SP for more resources, resulting in resource fragmentation.

To avoid such problems, we propose that the utility function should consider the mini-

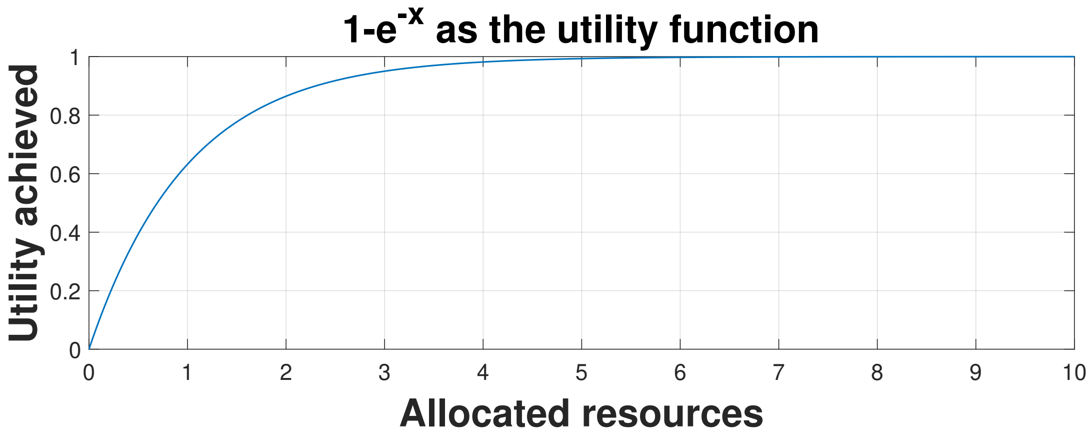

Figure 4.2.: A utility function that does not consider requested resources.

mum acceptable amount of resources. The SPs should earn either zero utility or a negative utility if the resources provided are not within ξ units of the requested resource $r^m_{n,j}$. Rather than using a utility function, such as $1-e^{-(x^m_{n,j})}$ that pays the SP even when a small amount of resources are provided, it is better to use $1-e^{-(x^m_{n,j}-r^m_{n,j}+\xi)}$ that becomes positive only when the allocation $x^m_{n,j}$ is within ξ units of the requested resource $r^m_{n,j}$. If the allocation $x^m_{n,j}$ is not within ξ units of the requested resource $r^m_{n,j}$, the utility can be negative and will serve as a penalty for not providing sufficient resources. Therefore, the SPs will attempt to provide as many resources as possible to the applications to avoid a negative utility. Such a utility along with the communication cost helps minimize the aforementioned fragmentation problem.

4.3. NBS for Resource Sharing among SPs

As mentioned earlier, NBS is a Pareto optimal and fair solution in settings that involve different players (SPs in our case) where each player has an objective to optimize. Therefore, it can be used to solve our MOO problem in (4.5) since we have different SPs that need to optimize their objectives and improve their utilities over what they would receive working alone. We first specify the disagreement point for our bargaining problem. If the SPs cannot come to an agreement, they can all start working alone. Hence the disagreement point is the solution to the SOO problem given in (4.2) for all SPs. Let us denote the solution to the SOO problem by $d^0_n, \forall\ n \in \mathcal{N}$. In the cooperative setting[2], the utility (represented by e_n in (2.19)) for an SP $n \in \mathcal{N}$ is given by:

$$U = \sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}}\Big(u^m(z^m_j + x^m_{n,j}) - u^m(z^m_j)\Big) + \Big(\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}}\big(u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j) - D^l(x^l_{n,j})\big)\Big).$$

Below, we present the centralized NBS algorithm.

[2]Service providers share resources among each other.

4.3.1. Centralized NBS

We first present the optimization problem to obtain NBS for our $N-$ SP bargaining game [29] and then present its equivalent problem [20] that is computationally efficient to solve.

Theorem 4.1. *The NBS for the MOO optimization problem in* (4.5) *can be obtained by solving the following optimization problem:*

$$\max_{\mathbf{X}} \prod_{n=1}^{N}\Big(\sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}}\Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big) + \Big(\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}}\big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)\Big) - d_n^0\Big), \tag{4.6a}$$

$$\text{s.t.} \quad \textit{Constraints in } (4.5\text{b}) - (4.5\text{e}), \forall n \in \mathcal{N},$$

$$\sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}}\Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big) + \Big(\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}}\big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)\Big) > d_n^0, \quad \forall\, n \in \mathcal{N}. \tag{4.6b}$$

Proof. The feasible set for the above optimization problem is convex and compact, since all constraints are convex and the intersection of convex sets is a convex set. Furthermore, based on our assumptions, there exist solutions (allocation and sharing decisions) that provide better utility than the disagreement point $d_n^0, \forall\, n \in \mathcal{N}$. Hence, the solution of the optimization problem in (4.6) is the NBS. □

However, solving (4.6) is computationally complex and the problem is not always convex. Therefore, there is a need for an efficient method to obtain the NBS. Toward this goal, we transform the problem (4.6) into an equivalent problem as proposed in [20].

Corollary 1. The NBS for (4.5) is obtained by solving the following optimization problem:

$$\max_{\mathbf{X}} \sum_{n=1}^{N} \ln\Big(\sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}}\Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big) + \Big(\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}}\big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)\Big) - d_n^0\Big), \tag{4.7}$$

$$\text{s.t.} \quad \textit{Constraints in } (4.5\text{b}) - (4.5\text{e}),\ (4.6\text{b}), \forall n \in \mathcal{N}.$$

Proof. SP utilities are concave and bounded above. Furthermore, the feasible set along with the set of achievable utilities is convex and compact (due to the nature of utilities and constraints). Utilities of SPs are also injective functions of the allocation decision. Hence (4.7) is equivalent to (4.6) [20]. Since the logarithm of any positive real number is concave [21], (4.7) is a convex optimization problem with a unique solution, which is the NBS. □

The centralized Algorithm 4.1 provides the allocation decision using NBS.

Algorithm 4.1 Centralized Algorithm for NBS

```
Input: $C, R$, and vector of utility functions of all SPs $\boldsymbol{u}$
Output: The optimal resource allocation $\mathbf{X}$ and payoffs of all SPs
Step 1:
for $n \in \mathcal{N}$
    $d_n^0 \leftarrow$ Objective function at optimal point in (4.2)
end for
Step 2: $\mathbf{X} \leftarrow$ Solution of the optimization problem in (4.7)
```

$$\mathcal{L}(\mathbf{X},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}) = \sum_{n=1}^{N} \ln\Bigg(\sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}} \Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big) + \Big(\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)\Big) - d_n^0\Bigg) + \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{J}} \gamma_{n,j}^m x_{n,j}^m + \sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{J}} \alpha_{n,j}(C_{n,j} - \sum_{m\in\mathcal{M}} x_{n,j}^m) + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{J}} \beta_j^m (r_j^m - \sum_{n\in\mathcal{N}} x_{n,j}^m) + \sum_{n\in\mathcal{N}} \zeta_n \Bigg(\sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}} \Big(u_n^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big) + \Big(\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)\Big) - d_n^0\Bigg) + \sum_{n\in\mathcal{N}}\sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}} \pi_{n,j}^l \Big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\Big). \tag{4.8}$$

4.3.2. Distributed Algorithm for NBS

While resource allocation and sharing using a central algorithm is feasible, it is desirable to develop low overhead distributed algorithms. To obtain a distributed algorithm, we rely on *Duality Theory* and use the *Lagrangian* function in (4.8). The dual optimization problem is given by

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}} D(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}) = \mathcal{L}\big(\mathbf{X}^*(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}),\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}\big),$$
$$s.t.\ \boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi} \geq 0. \tag{4.9}$$

$$\mathbf{X}^*(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}) = \arg\max_{\mathbf{X}} \mathcal{L}(\mathbf{X},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi}). \tag{4.10}$$

The dual problem can be solved iteratively using steepest gradient descent [23][3] as below:

$$\alpha_{n,j}[t+1] = \alpha_{n,j}[t] - \phi_{n,j}\frac{\partial\mathcal{L}(\mathbf{X},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\gamma},\boldsymbol{\pi})}{\partial\alpha_{n,j}},$$

[3] There are other variants of gradient descent (based on the descent direction), such as Newton's method, modified Newton's method, and Gauss-Newton method. Details of these methods can be found in [23].

$$\beta_j^m[t+1] = \beta_j^m[t] - \eta_j^m \frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \beta_j^m},$$
$$\zeta_n[t+1] = \zeta_n[t] - \omega_n \frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \zeta_n},$$
$$\gamma_{n,j}^m[t+1] = \gamma_{n,j}^m[t] - \theta_{n,j}^m \frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \gamma_{n,j}^m},$$
$$\pi_{n,j}^l[t+1] = \pi_{n,j}^l[t] - \psi_{n,j}^l \frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \pi_{n,j}^l}, \tag{4.11}$$

where $\phi_{n,j}$, η_j^m, ω_n, $\theta_{n,j}^m$ and $\psi_{n,j}^l$ are step sizes. A preliminary discussion on the step size can be found in Chapter 2, whereas a detailed discussion can be found in [23].

Furthermore, the gradients in (4.11) are given below.

$$\frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \alpha_{n,j}} = (C_{n,j} - \sum_{m \in \mathcal{M}} x_{n,j}^m),$$
$$\frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \beta_j^m} = (r_j^m - \sum_{n \in \mathcal{N}} x_{n,j}^m),$$
$$\frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \zeta_n} = \Bigg(\sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m) \Big) + \Big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) -$$
$$u^l(z_j^l) - D^l(x_{n,j}^l) \big) \Big) - d_n^0 \Bigg),$$
$$\frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \gamma_{n,j}^m} = x_{n,j}^m,$$
$$\frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial \pi_{n,j}^l} = \Big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l) \Big). \tag{4.12}$$

Since the objective function in (4.7) is a sum of strictly concave functions and the constraints are partially separable, the dual problem can be solved in a distributed manner [99–101]. However, we need to show that strong duality holds[4] to ascertain that the distributed solution will be the same as the centralized solution. Unless certain conditions, such as Slater's constraint qualification are satisfied, *strong duality* is not guaranteed to hold for our primal and dual problems [21]. However, we rely on Theorems B.1 and B.2 in Appendix B to show that strong duality holds. We present the following theorem about the global optimality of our distributed algorithm given in Algorithm 4.2.

Theorem 4.2. *For any concave injective utility function for which the inverse of the first derivative exists, the iterative distributed algorithm consisting of* (4.10) *and* (4.11) *converges to the global optimal solution.*

Proof. The proof is relegated to Appendix B. □

[4]Strong duality implies that there is no duality gap between the primal and dual problem.

Algorithm 4.2 Distributed Algorithm for NBS

```
Input: $\forall \boldsymbol{\alpha_0}, \boldsymbol{\beta_0}, \boldsymbol{\zeta_0}, \boldsymbol{\gamma_0}, \boldsymbol{\pi_0}, C, R$, vector of utility function of all SPs $\boldsymbol{u}$ and $\mathbf{X}_0$
Output: The optimal resource allocation $\mathbf{X}$ and payoffs of all SPs
Step 0: $t = 0$, $\boldsymbol{\alpha}[t] \leftarrow \boldsymbol{\alpha}_0$, $\boldsymbol{\beta}[t] \leftarrow \boldsymbol{\beta}_0$, $\boldsymbol{\zeta}[t] \leftarrow \boldsymbol{\zeta}_0$, $\boldsymbol{\gamma}[t] \leftarrow \boldsymbol{\gamma}_0$, $\boldsymbol{\pi}[t] \leftarrow \boldsymbol{\pi}_0$, $\mathbf{X}[t] \leftarrow \mathbf{X}_0$
Step 1:
for $n \in \mathcal{N}$
    $d_n^0 \leftarrow$ Objective function at optimal point in (4.2)
end for
Step 2: $t \geq 1$
while First order conditions [102]$\neq$true
    Compute $x_{n,j}^m[t+1]$ for $m \in \mathcal{M}$, $j \in \mathcal{J}$ and $n \in \mathcal{N}$ through (B.3) and (B.4);
    Compute $\boldsymbol{\alpha}[t+1]$, $\boldsymbol{\beta}[t+1]$, $\boldsymbol{\zeta}[t+1]$, $\boldsymbol{\gamma}[t+1]$ and $\boldsymbol{\pi}[t+1]$ through (4.11) given $\mathbf{X}[t+1]$, $\boldsymbol{\alpha}[t]$, $\boldsymbol{\beta}[t]$, $\boldsymbol{\zeta}[t]$, $\boldsymbol{\gamma}[t]$ and $\boldsymbol{\pi}[t]$
end while
```

Protocol for Distributed Execution: All SPs first broadcast information regarding resource requests from their native applications. Then any randomly chosen SP starts allocating resources as specified in Step 2 of Algorithm 4.2 and updates the corresponding Lagrangian multipliers. This SP then passes on the information about its allocation and updated request matrix to the next SP (that has not yet allocated resources in the current round[5]) that repeats the same procedure until all the SPs allocate their resources. This process continues until the first order conditions given in [102] are met.

Algorithm scalability and run-time: To compute the NBS, we need to solve $N+1$ convex optimization problems. In particular, we need to solve (4.2) for each of the N SPs and then solve (4.7). The number of decision variables and the constraints in (4.2) depends on the total number of resource types J and the total number of native applications at SP n given by $M_n = |\mathcal{M}_n|$. Hence, the number of decision variables and constraints in (4.2) is bounded as $\mathcal{O}(M_n J)$. Similarly, the number of decision variables and constraints in (4.7) depends on the total number of SPs N, resource types J, and total applications $M = |\mathcal{M}|$. Mathematically, constraints and decisions variables in (4.7) are bounded as $\mathcal{O}(NMJ)$, where M is large when compared to N and J. The run-time of the algorithm will increase with an increase in these parameters.

4.4. Simulation Results

We evaluate the performance of the proposed NBS framework for resource sharing and allocation across several settings in Table 4.2. Each SP has three different resources, i.e., storage, communication and computation. The model can be extended to include other resources/parameters. We study the proposed framework using both synthetic and real-world data traces [103–105]. For the study with synthetic and real-world data traces,

[5]A single round consists of all the SPs executing Step 2 of Algorithm 4.2 once.

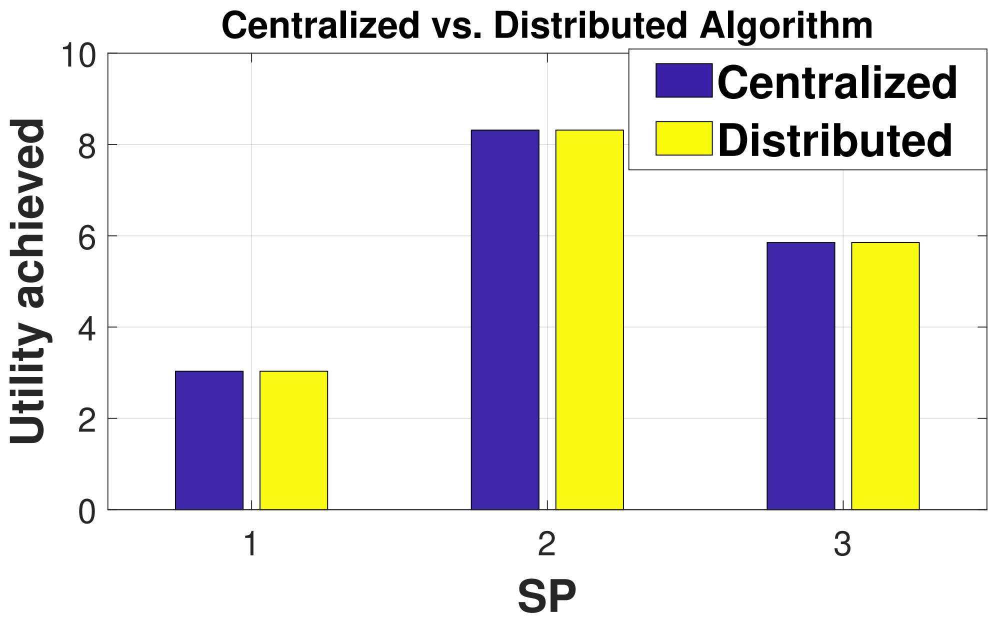

Figure 4.3.: Comparison of centralized and distributed algorithms.

requests R_n and capacities $C_n, \forall n \in \mathcal{N}$ are randomly generated in the range (1, 5) using Matlab's `randi` command, which generates integers from a uniform distribution. The total requests for each individual resource j at every SP is summed and represented by $\Gamma_{n,j}$. For the SPs with resource deficit, the capacity is chosen randomly from a uniform distribution in the range $[0, \Gamma_{n,j})$. For the SPs with resource surplus, the capacity is generated from a uniform distribution in the range $(\Gamma_{n,j}, \kappa_j/N')$, where N' represents the number of SPs with resource surplus and κ_j represents the total amount of resource j deficit at resource deficit SPs. This ensures that all the resources are utilized at the optimal solution.

Simulations were run in `Matlab R2019a` on a `Core-i7` processor with `16 GB RAM`. To solve the optimization problems in (4.2) and (4.7), we use the `OPTI-toolbox` [106]. We primarily used `IPOPT` (version 3.12.7) and `fmincon` (version R2016a) solvers from `OPTI-toolbox`. In certain instances where `IPOPT` and `fmincon` did not converge to the optimal solution, we used `filtersd` (version 1.0) from `OPTI-toolbox`. We evaluate our proposed algorithms from the perspective of service providers that are interested in maximizing their utilities and evaluate the impact of our framework on the applications. We define application *request satisfaction* (RS) as the ratio of allocated resources to requested resources. Mathematically, average RS for an SP n in the resource sharing case is defined as:

$$RS_n = \frac{\sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \left(\frac{\sum_{n' \in \mathcal{N}} x^m_{n',j}}{r^m_{n,j}} \right)}{M_n J} \times 100, \tag{4.13}$$

Table 4.2.: Simulation network settings for NBS based resource sharing framework.

Setting	**Parameters**
1	$N = 3, M_n = 3, \forall n \in \mathcal{N}, J = 3$
2	$N = 3, M_n = 20, \forall n \in \mathcal{N}, J = 3$
3 (data traces)	$N = 3, M_n = 20, \forall n \in \mathcal{N}, J = 3$
4	$N = 6, M_n = 6, \forall n \in \mathcal{N}, J = 3$
5	$N = 6, M_n = 20, \forall n \in \mathcal{N}, J = 3$

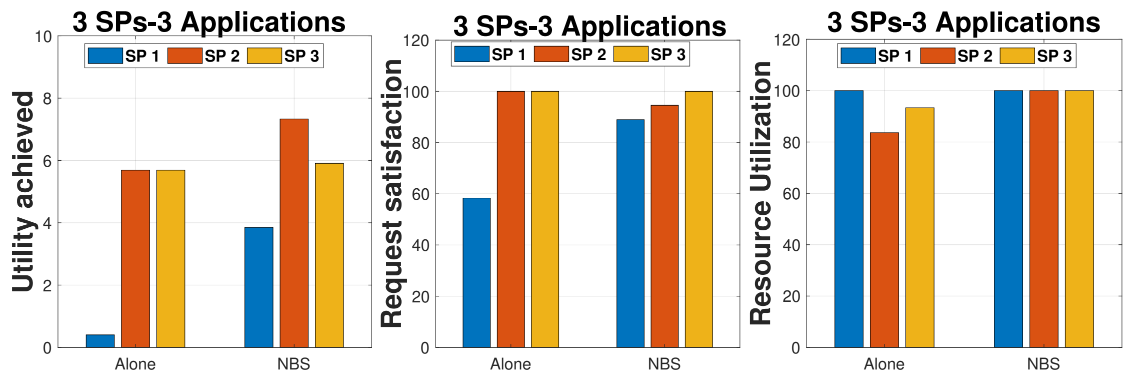

Figure 4.4.: Utility, average request satisfaction and average resource utilization for Setting 1 when SPs are working alone and using our proposed NBS framework.

$$u^m(x^m_{n,j}) = 1 - e^{-(x^m_{n,j} - r^m_{n,j} + \xi)},$$
$$D^m(x^m_{n,j}) = \frac{x^m_{n,j}}{w}. \tag{4.14}$$

The utility and communication cost functions used in simulation are given in (4.14). Here ξ is set to 1 and weight w is randomly chosen from a uniform distribution in the range $(1, 10^4)$.

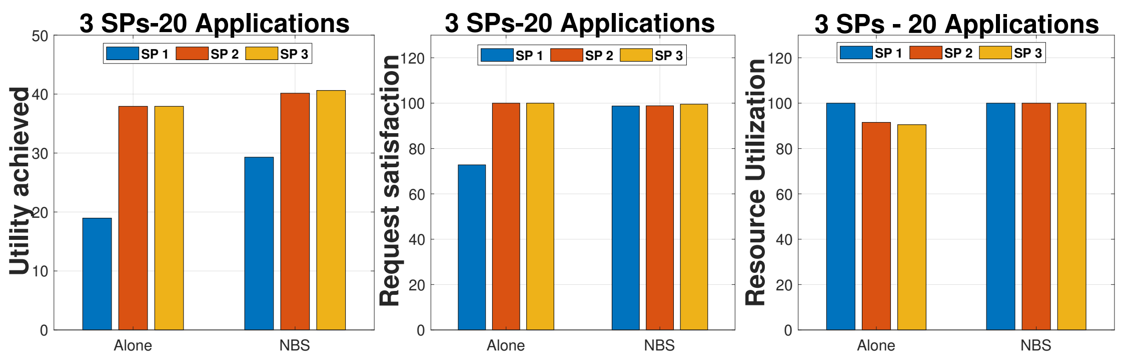

Figure 4.5.: Utility, average request satisfaction and average resource utilization for Setting 2 when SPs are working alone and using our proposed NBS framework.

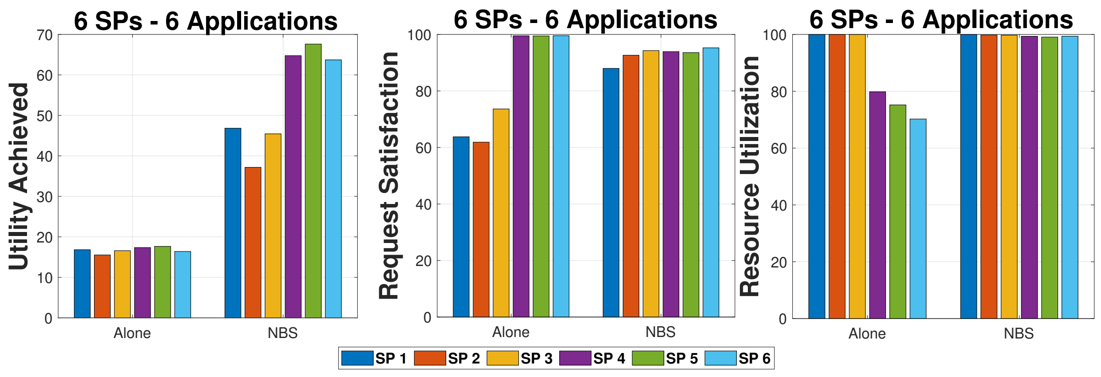

Figure 4.6.: Utility, average request satisfaction and average resource utilization for Setting 4 when SPs are working alone and using our proposed NBS framework.

4.4.1. Simulations Results for Synthetic Data

Evaluation of the Distributed Algorithm

To validate our results related to strong duality and our proposed distributed algorithm, we obtain the solution for Setting 1 in Table 5.2 using both the central and distributed algorithm as shown in Figure 4.3. It is evident that both the algorithms converge to the same solution, hence the duality gap is zero and our proposed distributed algorithm converges to the global optimal.

Utility, Resource utilization, and Request Satisfaction

To highlight the efficacy of our framework, we compare its performance with a setting where SPs work alone (i.e., no resource sharing among edge SPs). In particular, we compare SP utility, the average resource utilization (averaged across all J resources) and the average request satisfaction (averaged across requests of all SPs applications) in Figures 4.4 and 4.5 for settings 1 and 2 given in Table 4.2. When SP 1 works alone, it has a resource deficit (evident from 100% average resource utilization and average request satisfaction of less than 60% and 80% in Figures 4.4 and 4.5, respectively) whereas SPs 2 and 3 have resource surpluses as indicated by less than 100% resource utilization and 100% request satisfaction. The resource deficit results in a lower utility and request satisfaction for SP 1.

On the other hand, both SPs 2 and 3 achieve higher utilities by satisfying all their applications when working alone. However, by using our resource sharing framework, the utilities of all SPs improve since the framework provides optimal resource sharing. For the case with three applications, average request satisfaction improves from 86.11% (working alone) to 94.5% (resource sharing) whereas it improves from 90.9% to 99% for the 20 application case. It is worth noting that request satisfaction for native applications of SPs 2 and 3 reduce, because these SPs allocate their resources to applications of SP 1 for a higher utility. Furthermore, resource utilizations also increase for the SPs with resource surpluses since they share their resources with the SP with a resource deficit. Similar results

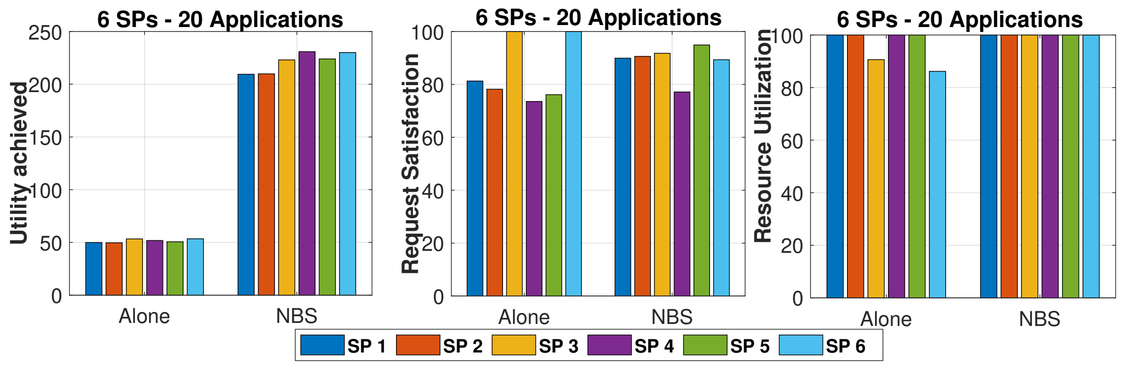

Figure 4.7.: Utility, average request satisfaction and average resource utilization for Setting 5 when SPs are working alone and using our proposed NBS framework.

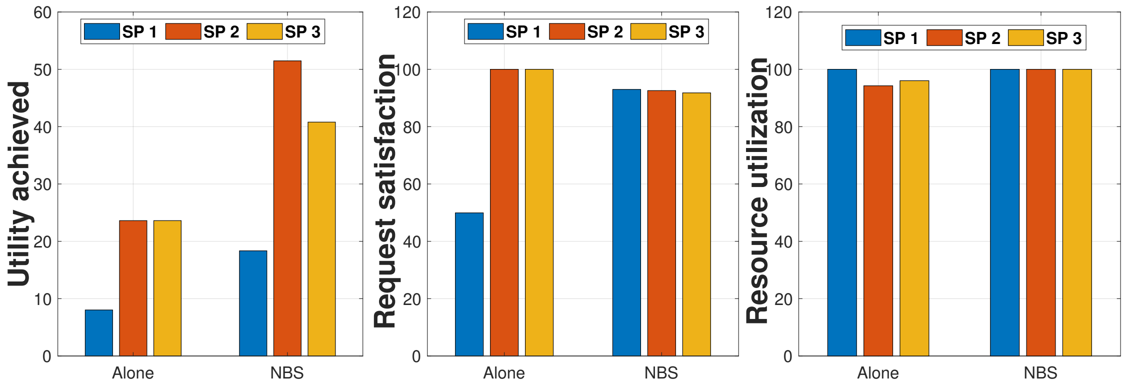

Figure 4.8.: Utility, average request satisfaction and average resource utilization for Setting 3 when SPs are working alone and using our proposed NBS framework.

are obtained for settings 4 and 5 given in Table 4.2. For setting 4, the request satisfaction improves from 82.96% to 92.94% using our proposed NBS framework, whereas request satisfaction improves in setting 5 from 84.85% to 88.94% using our proposed resource sharing framework. Table 4.3 summarizes request satisfactions and resource utilization in different settings using our framework and when working alone.

4.4.2. Results for the Data Traces

We use trace files from *fastStorage*, *Rnd* [103], and *materna* [104, 105]. We simulate a setting with three different SPs and randomly extract the normalized resource request information related to the number of CPU cores, the amount of CPU and memory (RAM) for 20 different resource requests from the *fastStorage*, *Rnd* and *materna* dataset. Since the datasets do not provide the capacities of these service providers, we assign capacities in such a way that fastStorage serves as the SP with resource deficit while the other two have resource surpluses. Figure 4.8 shows SP utilities, request satisfaction and resource utilization based on the data traces. It is evident that SP utilities improve and resource utilizations increase to satisfy more applications by use of the proposed NBS framework.

Table 4.3.: Summary of average request satisfaction and resource utilization in different settings with and without the proposed NBS sharing framework.

Setting	**Request Satisfaction(%)**		**Resource Utilization(%)**	
	Alone	**NBS**	**Alone**	**NBS**
1	91.39	96.90	93.37	100
2	90.93	99.04	94.01	100
3	83.30	92.44	96.76	100
4	82.96	92.94	87.54	100
5	84.85	88.94	96.13	100

The average request satisfaction also increases from 83.3% to 92.44%.

4.4.3. Measure of Fairness

NBS is known for its fairness property [25]. In this section, we show the fairness of our proposed NBS based resource sharing framework for different settings given in Table 5.2. In particular, to measure the degree of fairness of the proposed sharing framework, we calculate *Jain*'s index [107, 108].

Definition 4.1. *Jain's Index (JI) is a number that rates the fairness of utilities assigned to different SPs. Mathematically,*

$$JI(p_1, p_2, \ldots, p_N) = \frac{(\sum_{n=1}^{N} p_n)^2}{N \sum_{n=1}^{N} p_n^2}, \tag{4.15}$$

where p_n is the payoff or utility obtained by SP n. The value of JI ranges from $\frac{1}{N}$ (worst case) and 1 *(best case).*

Figure 4.9 shows Jain's index for different settings given in Table 4.2. The value of Jain's index is larger than 0.95 in all the settings. Especially for scenarios with a large number of applications, these results reveal that our framework enables fair sharing and allocation of resources among SPs, as one would expect from the product-based fairness as offered by the NBS.

As evident from the results above, our proposed resource sharing framework significantly improves the performance of SPs. A higher number of application requests was satisfied by using the available resources. The extent of improvement due to the framework depends on the settings and parameters used.

4.5. Summary and Conclusions

The focus in this chapter was to optimally share and utilize available resources for satisfying a larger number of applications and improving the utility of service providers. Using an edge computing setting as an example, we showed that although edge SPs may have different

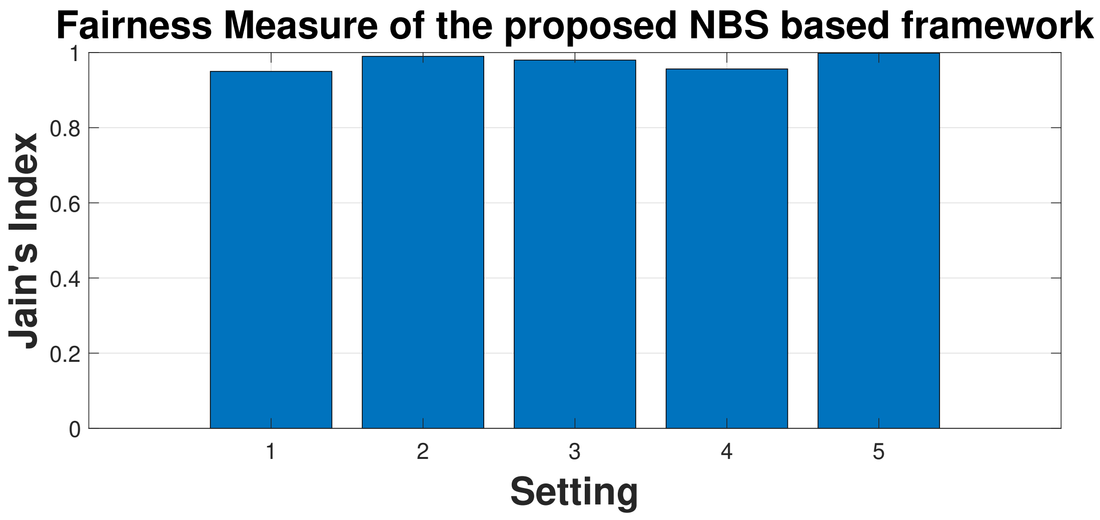

Figure 4.9.: Jain's index in different settings using our proposed NBS based resource sharing framework.

utilities, they should share resources to improve their utilities and enhance application request satisfaction. Resource sharing among SPs was formulated as a bargaining problem and a resource-sharing framework using Nash Bargaining Solution (NBS) was proposed, which was shown to be beneficial for the SPs. Since a centralized solution for obtaining the NBS may not always be desirable, the strong duality property was proven, which enabled us to develop a distributed algorithm for the NBS. Using synthetic and real-world data traces, we demonstrated the effectiveness of the proposed framework. In particular, our results confirmed that SPs with resource deficits and surpluses can improve their utilities as well as application satisfaction by sharing resources.

5. A Cooperative Game-Theoretic Framework For Resource Sharing: Uniform Priority Case

In the previous chapter, we proposed a resource sharing framework in which the service providers had concave utilities and both native and non-native applications had uniform priority. However, the service providers may not always have concave utilities. Furthermore, SPs may employ different resource sharing strategies. In this chapter, we present a cooperative game-theoretic framework for resource sharing that relaxes the assumption of concave utility function and requires service providers to have monotone non-decreasing utility (concave, or non-concave). Hence, the framework is more generic than the NBS based resource sharing framework at the cost of added complexity. Furthermore, the proposed framework can support both uniform and non-uniform priority strategies employed by SPs. However, we consider the uniform priority strategy in this chapter. We show that our proposed resource sharing game is canonical and *cardinally convex*, hence the core is non-empty. We then propose a centralized algorithm, *Game-theoretic Pareto optimal allocation with Uniform Priority* (GPUS), that can provide allocations from the core, hence guaranteeing stability of the *coalition* and Pareto optimality of the solution. Simulation results show that our proposed framework results in higher utilities for the service providers along with an increased application satisfaction.

5.1. Main Contributions

The main contributions of this chapter are:

1. We propose a cooperative game theory (CGT) based multi-objective resource sharing and allocation framework for multiple SPs that guarantees Pareto optimality.

2. We show that the resource sharing and allocation problem can be modeled as a *canonical* game with *non-transferable utility.* Furthermore, when SP objectives are represented by monotone non-decreasing utilities, we show that our canonical game is cardinally convex. Due to the convexity of the game, the *core* is non-empty and the grand coalition formed by all service providers is stable.

3. We address the problem of obtaining an allocation from the core by proposing a centralized algorithm, *Game-theoretic Pareto Optimal Allocation with Uniform Priority*

(GPUS), which can provide allocations from the core.

4. We evaluate the performance of our proposed framework through extensive experiments and show that resource sharing can improve the utility of all service providers and also increase user satisfaction.

The chapter is further structured as: Section 5.2 presents the system model, which is similar to the system model presented in Chapter 4 with minor differences. It also presents the resource sharing and allocation problem. Section 5.3 presents the theoretical properties of the cooperative game used to obtain the Pareto optimal solution. Section 5.4 presents our experimental results. Section 5.5 concludes the chapter.

5.2. System Model

Let $\mathcal{N} = \{1, 2, \cdots, N\}$ be the set of all edge SPs. We assume that each provider has a set of $\mathcal{J} = \{1, 2, \cdots, J\}$ types of resources, such as communication, computation and storage resources. $C_n = \{C_{n,1}, \cdots, C_{n,J}\}$, with $C_{n,j}$ denoting the amount of type j resources available at service provider n. Each service provider n has a set of native applications $\mathcal{M}_n = \{1, 2, \cdots, M_n\}$. The set of all applications that request resources from all service providers is given by $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cdots \cup \mathcal{M}_N$, where we assume $\mathcal{M}_n \cap \mathcal{M}_{n'} = \emptyset$, $\forall n \neq n'$, i.e., each application initially demands resources from only its native service provider. We also define $\overline{\mathcal{M}}$ to represent $\mathcal{M} \backslash \mathcal{M}_n$ and $\overline{\mathcal{N}}$ to represent $\mathcal{N} \backslash \{n\}$. Every SP $n \in \mathcal{N}$ has a request (requirement) matrix $\mathbf{R}_n$,

$$\mathbf{R}_n = \begin{bmatrix} \mathbf{r_n^1} \\ \vdots \\ \mathbf{r_n^{M_n}} \end{bmatrix} = \begin{bmatrix} r_{n,1}^1 & \cdots & \cdots & r_{n,J}^1 \\ \vdots & \vdots & \vdots & \vdots \\ r_{n,1}^{M_n} & \cdots & \cdots & r_{n,J}^{M_n} \end{bmatrix}. \tag{5.1}$$

Here $r_{n,j}^m$ is the amount of resource j that application $m \in \mathcal{M}_n$ requires. When an edge SP is working alone (no sharing of its resources with any other service provider), its objective is to maximize its utility by allocating resources to its native applications. A service provider n earns a utility $u^m(x_{n,j}^m)$ by allocating $x_{n,j}^m$ amount of resource j to application $m \in \mathcal{M}_n$, where the vector $\mathbf{x}_n^m = [x_{n,1}^m, x_{n,2}^m, \cdots, x_{n,J}^m]^T$. Table 5.1 contains notations used throughout the chapter. We present the optimization formulation for a single service provider in Section 5.2.1, followed by a formulation for the multiple service provider problem in Section 5.2.2.

5.2.1. Problem Formulation for Single Service Provider

We first present the resource allocation problem for a single, stand-alone provider (i.e., no resource sharing with other providers). For each provider $n \in \mathcal{N}$, the allocation decisions

Table 5.1.: List of notations used throughout chapter 5

Notation	**Description**
$\mathcal{N}, N, n$	Set, number and index of game players (service providers)
$\mathcal{J}, J, j$	Set, number and index of resources
$\mathcal{M}, M, m$	Set, number and index of applications
$\mathcal{M}_n$	Set of native applications at player (provider) n
S	A coalition of game players (providers), where $S \subseteq \mathcal{N}$
$\mathcal{V}$	Set of payoff vectors
$v(S)$	Value of coalition S
C	Capacity vector of all game players (providers)
C_n	Capacity vector of player (provider)n
$C_{n,j}$	Capacity of resource j at player (provider) n
$\mathbf{R}_n$	Request matrix at player (provider) n
$r^m_{n,j}$	Request of application m for resource j from player (provider) n
$x^m_{n,j}$	Allocation decision of resource j for application m at player (provider) n
$\mathbf{x}^m_n$	Allocation decision vector for application m at player (provider) n when working alone
$\mathbf{X}_n$	Allocation decision for player (provider) n in the coalition
$\mathbf{X}$	Allocation decision for the entire coalition
$u^m(x^m_{n,j})$	Utility player (provider) n earns by allocating resource j to application m
$u^m(\mathbf{x}^m_n)$	Utility player (provider) n earns by allocating vector of resources to application m
GC	Grand coalition
RS	Request satisfiability
NTU	Non-transferable utility
CGT	Cooperative game theory
MOO	Multi-objective optimization
SOO	Single-objective optimization

consist of the vectors $\mathbf{x}^1_n, \cdots, \mathbf{x}^{M_n}_n$. The optimization problem is:

$$\max_{\mathbf{x}^1_n, \cdots, \mathbf{x}^{M_n}_n} \quad \sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} f^m(x^m_{n,j}), \forall n \in \mathcal{N}, \tag{5.2a}$$

$$\text{s.t.} \quad \sum_m x^m_{n,j} \leq C_{n,j}, \quad \forall j \in \mathcal{J}, \tag{5.2b}$$

$$x^m_{n,j} \leq r^m_{n,j}, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}, \tag{5.2c}$$

$$x^m_{n,j} \geq 0, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}. \tag{5.2d}$$

The objective function $f^m(x^m_{n,j})$ is required to be a monotonic, non-negative and non-decreasing function. For illustration purposes in this thesis, we assume that:

$$f^m(x^m_{n,j}) = u^m(x^m_{n,j}) + \frac{x^m_{n,j}}{r^m_{n,j}},$$

where the term $\frac{x^m_{n,j}}{r^m_{n,j}}$ captures the request satisfaction and $u^m(.)$ is the non-decreasing and non-negative utility the service provider earns by providing the resources to application m.

The first constraint (5.2b) indicates that allocated resources cannot exceed capacity. The second constraint (5.2c) reflects that allocated resources should not exceed the requested amounts. The last constraint, (5.2d) says the allocation cannot be negative. However, as mentioned in the preceding chapter, it is possible that a service provider n may earn a larger utility by providing its resources to applications of other service providers or it may not have sufficient resources to satisfy requests of all its native applications. On the other hand, there may be another service provider $n' \in \overline{\mathcal{N}}$ that may have a surplus of resources, which can be "rented" by service provider n. Below, we discuss resource sharing among these service providers.

5.2.2. Problem Formulation for Multiple Service Providers

Let us assume that an edge service provider n' does not have sufficient resources to satisfy all its native applications. Suppose that service provider n has a resource surplus. Allowing resource sharing among such providers can improve resource utilization and application satisfaction. It is worth mentioning here again that in this chapter's formulation, SPs are considered to employ the uniform priority strategy. Hence, an SP does not differentiate between its native and non-native applications. By sharing and allocating its type j resources to a non-native application l, service provider n earns a non-decreasing and non-negative net utility $H^l(x^l_{n,j})$, which in this chapter is assumed to be

$$H^l(x^l_{n,j}) = \Big(u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j) - D^l(x^l_{n,j})\Big), \forall l \in \overline{\mathcal{M}}, n \in \mathcal{N} \backslash n', \tag{5.3}$$

where $z^l_j = \sum_{n'' \in \overline{\mathcal{N}}} x^l_{n'',j}$ is the amount of resource j allocated already to application l by providers other than n. $u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j)$ captures the incremental increase in utility of provider n due to the allocation of additional resource $x^l_{n,j}$. $D^l(x^l_{n,j})$ represents the communication cost of serving application $l \in \overline{\mathcal{M}}$ by using resource of type j at provider n rather than its native service provider n'. We assume that the communication cost is a positive monotonic non-decreasing function.

To consider each provider and its native applications, let $F^m(x^m_{n,j})$ represent the utility service provider n earns by allocating resource j to its native applications along with the increase in request satisfaction due to provision of resources to its applications at different service providers including itself. We assume:

$$F^m(x^m_{n,j}) = u^m(z^m_j + x^m_{n,j}) - u^m(z^m_j) + \frac{\sum_{n' \in \mathcal{N}} x^m_{n',j}}{r^m_{n,j}}. \tag{5.4}$$

Note that F^m is also monotonic non-decreasing and non-negative. Furthermore, we implicitly assume that $u^m(x^m_{n,j}) = u^m(x^m_{n',j})$, if $x^m_{n,j} = x^m_{n',j}$. The latter assumption reflects that different providers earn the same utility when allocating the same amount of resources of the same type to a given application, either locally or remotely. Furthermore,

$$\mathbf{x}^m = [\sum_{n\in\mathcal{N}} x^m_{n,1}, \sum_{n\in\mathcal{N}} x^m_{n,2}, \cdots, \sum_{n\in\mathcal{N}} x^m_{n,J}]^T.$$

That is, the total resources allocated to any application $m \in \mathcal{M}$ is the sum of resources allocated to it across all service providers. The resource sharing and allocation framework, based on resource requests and capacities of service providers, has to make allocations that optimize the utilities of all the service providers $n \in \mathcal{N}$, while satisfying user requests as well. The allocation decision is given by $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$, where $\mathbf{X}_n$, $\forall n \in \mathcal{N}$ is given by:

$$\mathbf{X}_n = \begin{bmatrix} \mathbf{x}^1_n \\ \vdots \\ \mathbf{x}^{|\mathcal{M}|}_n \end{bmatrix} = \begin{bmatrix} x^1_{n,1} & \cdots & \cdots & x^1_{n,J} \\ \vdots & \vdots & \vdots & \vdots \\ x^{|\mathcal{M}|}_{n,1} & \cdots & \cdots & x^{|\mathcal{M}|}_{n,J} \end{bmatrix}. \tag{5.5}$$

When service providers share resources, each service provider aims to maximize the sum of utilities obtained by a) allocating resources to its native applications; b) allocating resources to applications of other service providers; and c) improving the request satisfaction of its applications by using its own resources or borrowing from other service providers (when needed and possible). This is mathematically formulated for each SP n as the following optimization problem:

$$\begin{aligned}
\max_{\mathbf{X}_n} \quad & \sum_{m\in\mathcal{M}_n} \sum_{j\in\mathcal{J}} F^m(x^m_{n,j}) + \sum_{l\in\overline{\mathcal{M}}} \sum_{j\in\mathcal{J}} H^l(x^l_{n,j}), && (5.6a)\\
\text{s.t.} \quad & \sum_{m} x^m_{n,j} \le C_{n,j}, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, && (5.6b)\\
& \sum_{n'\in\mathcal{N}} x^m_{n',j} \le r^m_{n,j}, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, && (5.6c)\\
& x^m_{n,j} \ge 0, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, && (5.6d)\\
& u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j) \ge D^l(x^l_{n,j}), \quad \forall l \in \overline{\mathcal{M}}, j \in \mathcal{J}, && (5.6e)
\end{aligned}$$

where $F^m(x^m_{n,j})$ and $H^l(x^l_{n,j})$ are given in (5.3) and (5.4). Constraints (5.6b)-(5.6d) are similar to those for the formulation in (5.2). (5.6e) indicates that the utility earned by sharing resources must be higher than the communication cost. Since the utility u^j, $\forall n \in \mathcal{N}$ can differ for each application j, (5.6) for all providers n represents a multi-objective optimization problem. However, each SP independently solving (5.6) is not the right approach as the objective function (5.6a) for an SP n also depends on the allocation decisions and strategies of other SPs. Therefore, we solve this problem by a game-theoretic approach as follows.

5.3. Game-Theoretic Solution

In this section, we first present general properties of our game and show that the resource-sharing problem for multiple providers can be modeled as a canonical cooperative game with non-transferable utility. We then show that the core is non-empty, by proving that our canonical game is *cardinally convex* [109]. Finally, we propose our GPUS algorithm in Section 5.3.2 to obtain allocations from the core.

5.3.1. General Game Properties

We model each service provider as a player in our game to obtain the optimal resource sharing/allocation decision. Let $\mathcal{N}$ be the set of players that play the resource sharing and allocation game. The *value of coalition* for the game players $S \subseteq \mathcal{N}$ is given by (5.7), where $\mathcal{F}_S$ is the feasible set for resource sharing and allocation given by, (5.6b)-(5.6e).

$$v(S) = \sum_{n \in S} \Big(\sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m) + \frac{\sum_{n' \in S} x_{n',j}^m}{r_{n,j}^m}\big) + \big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D_n^l(x_{n,j}^l)\big)\big)\Big). \tag{5.7}$$

The value of coalition given in (5.7) is generic and depends on the strategy employed by SPs. In this chapter, as mentioned earlier, SPs employ the uniform priority strategy where the SPs do not differentiate between their native and non-native applications.

Remark 5.1. *A resource allocation and sharing problem (with multiple objectives) for the aforementioned system model where the SPs employ the uniform priority strategy can be modeled as a canonical cooperative game with NTU.*

This is so because the stated problem satisfies the following two conditions.

- **Characteristic form of payoff:** Since the utility function in our formulated resource sharing and allocation problem only relies on service providers that are part of the coalition, the game or payoff function is of characteristic form.
- **Superadditivity:** Let $S_1, S_2 \subseteq \mathcal{N}$ where S_1, S_2 are non-empty and $S_1 \cap S_2 = \emptyset$. Hence, $S_1,\ S_2 \subset (S_1 \cup S_2)$. Superadditivity follows from the monotonic utility functions.

Definition 5.1 (Cardinally convex games [109]). *An NTU game is said to be cardinally convex if* $\forall\ S_1,\ S_2 \subseteq \mathcal{N}$:

$$\mathcal{V}(S_1) \cup \mathcal{V}(S_2) \subseteq \mathcal{V}(S_1 \cup S_2) \cup \mathcal{V}(S_1 \cap S_2). \tag{5.8}$$

Theorem 5.1. *Our canonical game with uniform priority strategy is cardinally convex.*

Proof. Let us consider two coalitions S_1 and S_2 that are non-empty subsets of $\mathcal{N}$. Then from superadditivity of the game,

$$\mathcal{V}(S_1) \cup \mathcal{V}(S_2) \subset \mathcal{V}(S_1 \cup S_2). \tag{5.9}$$

Furthermore, due to the non-negative and non-decreasing nature of the utility used,

$$\mathcal{V}(S_1 \cup S_2) \subseteq \mathcal{V}(S_1 \cup S_2) \cup \mathcal{V}(S_1 \cap S_2). \tag{5.10}$$

The proof follows. □

Remark 5.2. *The core of any convex game $(\mathcal{N}, \mathcal{V})$ is non-empty [110] and large [27].*

Remark 5.3. *Our canonical cooperative game $(\mathcal{N}, \mathcal{V})$ with NTU can be used to obtain the Pareto-optimal solution for the multi-objective resource sharing problem for various service providers that employ a uniform priority strategy.*

The convexity of our game proves that the core is non-empty. However, obtaining an allocation from the core is challenging, which we address in the following subsection.

5.3.2. Centralized GPUS Algorithm for Game-Theoretic Resource Sharing

While the existence of the core, i.e., the core being non-empty, guarantees the grand coalition is stable, finding a suitable allocation from the core is challenging since the core is large. Algorithm 5.1 provides an allocation from the core[1], by solving $|\mathcal{N}|+1$ optimization problems.

$$\begin{aligned}
\max_{\mathbf{X}} \quad & \sum_{n \in \mathcal{N}} \bigg(\sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m) + \frac{\sum_{n' \in \mathcal{N}} x_{n',j}^m}{r_{n,j}^m} \big) + \Big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) \\
& - u^l(z_j^l) - D_n^l(x_{n,j}^l) \big) \Big) \bigg), && \text{(5.11a)} \\
\text{s.t.} \quad & \texttt{constraints in } (5.6b)\text{-}(5.6e), \forall n \in \mathcal{N}, \\
& \bigg(\sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m) + \frac{\sum_{n' \in \mathcal{N}} x_{n',j}^m}{r_{n,j}^m} \big) + \Big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) - \\
& u^l(z_j^l) - D_n^l(x_{n,j}^l) \big) \Big) \bigg) \geq v(\{n\}), \forall n \in \mathcal{N}. && \text{(5.11b)}
\end{aligned}$$

Theorem 5.2. *The allocation decision obtained using Algorithm 5.1 lies in the core.*

Proof. To prove the theorem, we need to show that the allocation decision obtained using Algorithm 5.1: a) is individually rational; b) is group rational; and c) no players have the

[1]Provided that the aforementioned assumptions hold.

Algorithm 5.1 Centralized Game-Theoretic Pareto Optimal Allocation for Uniform Priority Strategy (GPUS)

```
Input: $R, C$, and vector of utility functions of all players $\mathbf{u}$
Output: $\mathbf{X}$, $\mathbf{u}(\mathbf{X})$
Step 1: $\mathbf{u}(\mathbf{X}) \leftarrow 0$, $\mathbf{X} \leftarrow 0$
Step 2:
for $n \in \mathcal{N}$
    $v(\{n\}) \leftarrow$Optimal objective function value in (5.2)
end for
Step 3: $\mathbf{X} \leftarrow$Optimal allocation decision from (5.11)
        $\mathbf{u}(\mathbf{X}) \leftarrow$Payoff vector from (5.11)
```

incentive to leave the grand coalition and form another sub-coalition $S \subset \mathcal{N}$.

Individual Rationality: For each player $n \in \mathcal{N}$, the solution obtained using (5.11) is individually rational due to the constraint in (5.11b). Hence the solution obtained as a result of Algorithm 5.1 is individually rational.

Group Rationality: The value of the grand coalition $v\{\mathcal{N}\}$ as per (5.11a) is the sum of utilities of all players that they achieve from the payoff vector $\mathcal{V}(\mathcal{N}) \subseteq \mathbb{R}^{|\mathcal{N}|}$. Hence the allocation obtained from Algorithm 5.1 is group rational.

Furthermore, due to superadditivity of the game and monotonic non-decreasing nature of the utilities, no subgroup of players have an incentive to form a smaller coalition. Hence Algorithm 5.1 provides a solution from the core. □

In Algorithm 5.1, we first solve the single objective optimization problem (5.2) for all players $n \in \mathcal{N}$. We then solve the problem in (5.11) that provides the allocation from the core. $v(\{n\})$ in (5.11b) is the payoff a player n receives when working alone.

Algorithm scalability and run-time: To obtain an allocation from the core, GPUS solves $N + 1$ optimization problems. In particular, we need to solve (5.2) for each of the N SPs and then solve (5.11). The number of decision variables and the constraints in (5.2) depends on the total number of resource types J and the total number of native applications at SP n given by $M_n = |\mathcal{M}_n|$. Hence, the number of decision variables and constraints in (5.2) is bounded as $\mathcal{O}(M_n J)$. Similarly, the number of decision variables and constraints in (5.11) depends on the total number of SPs N, resource types J, and total applications $M = |\mathcal{M}|$. Mathematically, constraints and decisions variables in (5.11) are bounded as $\mathcal{O}(NMJ)$, where M is large when compared to N and J. The run-time of GPUS will increase with an increase in these parameters.

Remark 5.4. *The optimization problems in* (5.2), *and* (5.11) *are non-convex. The hardness of these problems depend on the parameters, such as the utility functions, requests and capacities. Therefore, it is difficult to evaluate the hardness of these problems, but the instances we consider in Section 5.4 are easily solved using existing solvers. If the optimal solutions are not found for* (5.2), *and* (5.11), *then the allocation will not be from the core.*

Table 5.2.: Simulation network settings for cooperative game-theoretic resource sharing framework with uniform priority strategy.

Setting	**Parameters**
1	$N = 3, M_n = 3, \forall n \in \mathcal{N}, J = 3$
2	$N = 3, M_n = 20, \forall n \in \mathcal{N}, J = 3$
3	$N = 6, M_n = 6, \forall n \in \mathcal{N}, J = 3$
4	$N = 6, M_n = 20, \forall n \in \mathcal{N}, J = 3$

In the next section, we evaluate the performance of our resource sharing and allocation framework.

5.4. Experimental Results

We evaluate the performance of the proposed resource sharing and allocation framework for a number of settings as shown in Table 5.2. Each player has three different types of resources ($J = 3$), i.e., storage, communication and computation. Without loss of generality, the model can be extended to include other type of resources/parameters. We used linear and sigmoidal utilities (see (5.13)) for all the players. However, the results hold for any monotone non-decreasing utility.

$$u^m(x^m_{n,j}) = ax^m_{n,j} + c, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}, m \in \mathcal{M}, \tag{5.12}$$

$$u^m(x^m_{n,j}) = \frac{1}{1 + e^{-\mu(x^m_{n,j} - r^m_{n,j})}}, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}, m \in \mathcal{M}. \tag{5.13}$$

μ is chosen to be 0.01, whereas the constants a and c were randomly generated in the range (1, 5) using Matlab's `randi` command that produces integers from a uniform distribution. Requests R_n and capacities $C_n, \forall n \in \mathcal{N}$ are also generated using Matlab's `randi` command in the range (1, 5). The total requests for each individual resource j at every SP is summed and represented by $\Gamma_{n,j}$. For the SPs with resource deficit, the capacity is chosen randomly from a uniform distribution in the range $[0, \Gamma_{n,j})$, whereas for the SPs with resource surplus, the capacity is randomly generated from a uniform distribution in the range $(\Gamma_{n,j}, 3\Gamma_{n,j}]$.

The simulations were run in `Matlab R2019b` on a `Core-i7` processor with `16 GB RAM`. To solve the optimization problems, we used the `OPTI-toolbox` [106]. We primarily used `IPOPT` (version 3.12.7) and `fmincon` (version R2016a) solvers from `OPTI-toolbox`. In certain instances where `IPOPT` and `fmincon` did not converge to the optimal solution, we used `filtersd` (version 1.0) from `OPTI-toolbox`. Below, we provide detailed experimental results.

5.4.1. Verification of Game-Theoretic Properties

In Table 5.3, we present results for a 3-player 20-application game that verify different game theoretic properties, such as individually rationality, group rationality, super additivity and

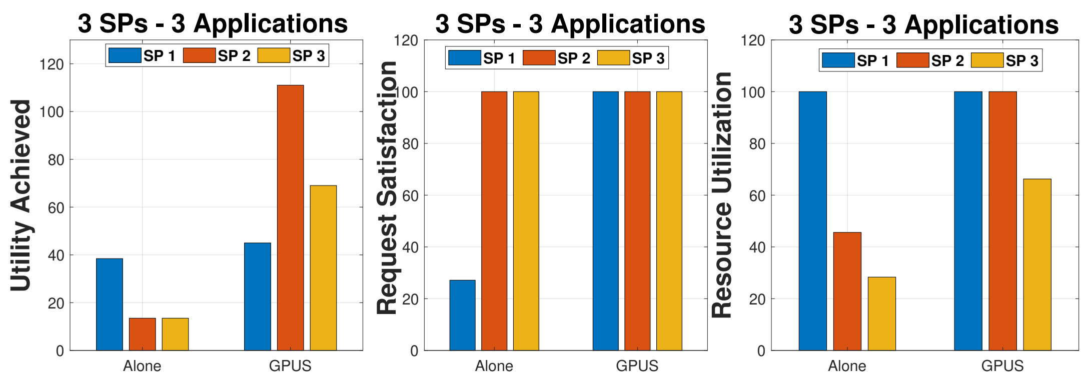

Figure 5.1.: Player utilities working alone and in the grand coalition (GC) with the proposed GPUS algorithm for a 3 SP - 3 Application Settings

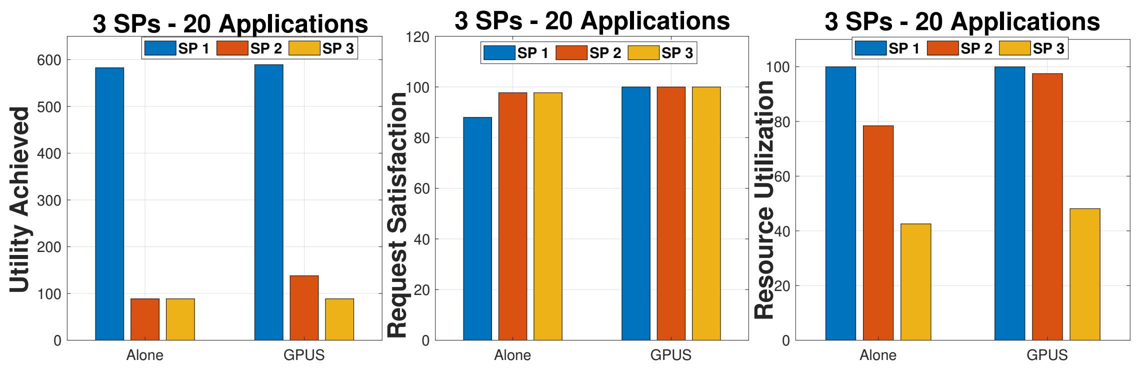

Figure 5.2.: Player utilities working alone and in the grand coalition (GC) with the proposed GPUS algorithm for a 3 SP - 20 Application Settings

show that the obtained allocation is from the core. The payoff all players receive in the grand coalition, i.e., $\{1, 2, 3\}$ is at least as good as players 1, 2 and 3 working alone. This shows that the solution obtained using the GPUS algorithm for the grand coalition is individually rational. Similarly, the value of the coalition is the sum of payoff all players receive, hence our solution is group rational. Furthermore, with an increase in the coalition size, value of coalition also increases. Hence, the grand coalition has the largest value, which shows the super-additive nature of the game. Also, no set of players has any incentive to divert from the grand coalition and form a smaller sub-coalition. Hence, the grand coalition is stable and the allocation we obtain using Algorithm 5.1 is from the core. Similar results can be obtained for other settings given in Table 5.2.

5.4.2. Efficacy of the Resource Sharing Framework

Figure 5.1 shows the utility of SPs, request satisfaction and resource utilization for setting 1 given in Table 5.2 when working alone and in the grand coalition using GPUS algorithm. SP

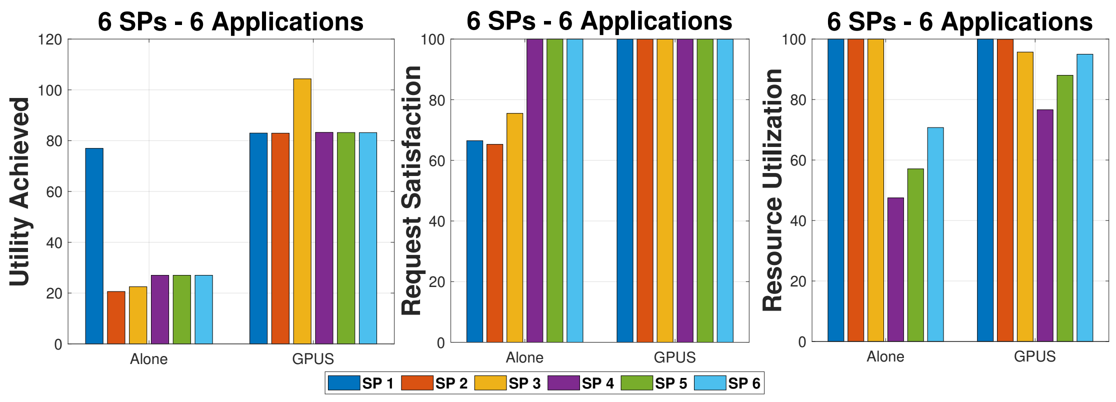

Figure 5.3.: Player utilities working alone and in the grand coalition (GC) with the proposed GPUS algorithm for a 6 SP - 6 Application Settings

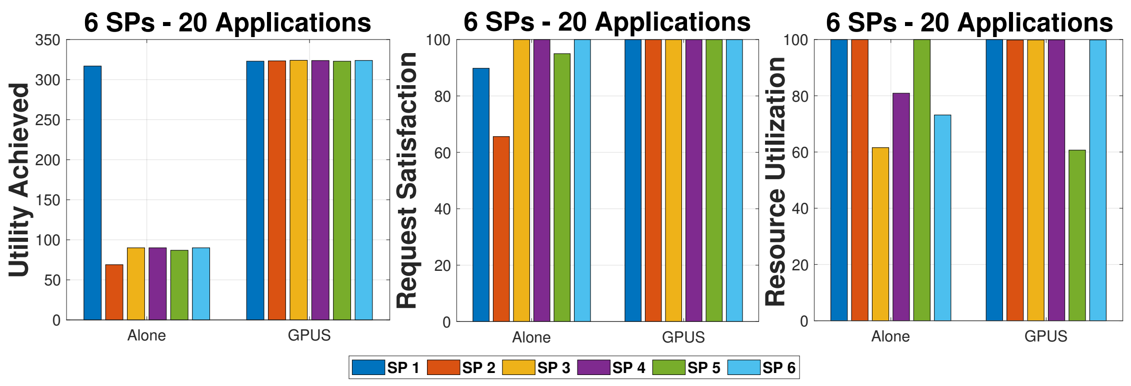

Figure 5.4.: Player utilities working alone and in the grand coalition (GC) with the proposed GPUS algorithm for a 6 SP - 20 Application Settings

1 had resource deficit (100% resource utilization and less than 100% request satisfaction) whereas SPs 2 and 3 had resource surplus, evident from 100% request satisfaction and less than 100% resource utilization when working alone. Since SP 1 has to improve its application request satisfaction by providing resources to as many applications as possible, it forms a coalition with the other 2 SPs, which is beneficial to all the coalition members. SPs 2 and 3 share their resources with applications of SP 1. This results in increased utilities for both SPs 2 and 3 and an improved application request satisfaction for SP 1 when forming the grand coalition using GPUS algorithm. For setting 1 in Table 5.2, the average user satisfaction using GPUS improved from 75.71% (when working alone) to 100%, whereas it improved from 94.4% to 100% in setting 2. Similarly for settings 3 and 4, the average request satisfaction improved from 84.4% to 100% and 91.4% to 100%, respectively. Details of request satisfaction and resource utilization are summarized in Table 5.4. For the communication cost and available resources used in the simulation, request satisfaction for all the SPs is 100% as seen in Figures 5.1, 5.2, 5.3 and 5.4. However, if the communication cost significantly increases, then resource sharing will be constrained due to constraint (5.6e). Therefore, it may not always be possible to achieve 100% request satisfaction even

Table 5.3.: Player payoffs using the proposed GPUS algorithm in different coalitions for a 3 player - 20 application game

Coalition	Player 1	Player 2	Player 3	Value of coalition
{1}	582.74	0.00	0.00	582.74
{2}	0	88.61	0	88.61
{3}	0	0	88.59	88.59
{1, 2}	582.75	217.26	0	800.02
{1, 3}	583.40	0	218.17	801.57
{2, 3}	0	116.80	116.73	238.41
{1, 2, 3}	589.36	137.94	88.59	815.90

Table 5.4.: Summary of average request satisfaction and resource utilization in different settings with and without the proposed GPUS Algorithm.

Setting	Request Satisfaction(%)		Resource Utilization(%)	
	Alone	GPUS	Alone	GPUS
1	75.71	100	57.98	88.75
2	94.46	100	73.66	81.87
3	84.54	100	79.2	92.54
4	91.73	100	85.94	94.82

if resources are available at other SPs due to communication costs.

5.4.3. Measure of Fairness

Figure 5.5 shows Jain's index for different settings given in Table 5.2. It is evident that the proposed framework is fair. For the settings with 3 SPs, Jain's index is lower than for settings with 6 SPs. This can be attributed to the parameter settings as well as larger request and SP diversity in settings 3 and 4.

It is evident from our results that the use of resource sharing and allocation is beneficial for service providers. By allowing different service providers to cooperate, the utility of

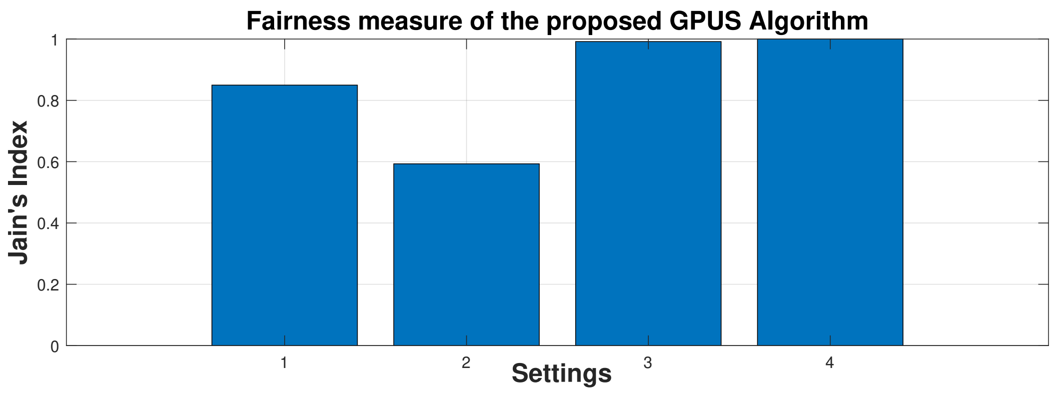

Figure 5.5.: Jain's Index in different settings using our proposed GPUS algorithm.

service providers increases and resources are utilized in an optimal manner. All the service providers in the game have the incentive to work together and use their resource capacities in the best possible way.

5.5. Summary and Conclusions

In this chapter, we proposed a cooperative game theory based framework for resource sharing and allocation among service providers. We showed that for a monotonic non-decreasing utility, resource sharing among multiple service providers can be modeled as a canonical game with non-transferable utility. We proved that the game is convex, hence the core of the game is non-empty. To address the problem of obtaining an allocation from the core for uniform priority strategy, we proposed GPUS, a centralized algorithm, that was proven to provide an allocation decision from the core. Simulation results showed that our proposed framework improves utility of service providers. Furthermore, request satisfaction of users also improved.

6. A Cooperative Game-Theoretic Framework For Resource Sharing: Non-Uniform Priority Case

In the previous chapter, we presented a cooperative game-theoretic framework for resource sharing that only requires service providers to have monotone non-decreasing utility (concave, or non-concave). However, we only explored the uniform priority strategy employed by the SPs. We slightly modify our cooperative game-theoretic framework in this chapter and allow the SPs to employ the non-uniform priority strategy. We show that despite the change in the strategy employed by SPs, the game is canonical, and cardinally convex with a non-empty core. We propose two algorithms, *Game-theoretic Pareto optimal allocation* (GPOA) and *Polyandrous-Polygamous Matching based Pareto Optimal Allocation* (PPMPOA) that provide allocations from the core. Experimental results confirm that our proposed resource sharing framework improves utilities of SPs and application request satisfaction.

6.1. Main Contributions

The main contributions of this chapter are:

1. We modify/extend the cooperative game-theoretic resource sharing framework proposed in Chapter 5 and enable the SPs to employ the non-uniform priority strategy.
2. We prove that the game is super-additive and cardinally convex, hence the *core* is non-empty and the grand coalition of all SPs is stable.
3. We propose a distributed *Game-theoretic Pareto Optimal Allocation* (GPOA) algorithm that provides an allocation from the core.
4. To reduce the number of SPs (with resource surplus) that provide resources to applications of a specific resource deficit SP, we also propose *Polyandrous Polygamous Matching based Pareto Optimal Allocation* (PPMPOA) algorithm that matches resource deficit SPs with resource surplus SPs. Using evaluations we show that PPMPOA:
   - Provides an allocation from the core.

- Results in a *stable* matching [27].
- Reduces the number of SPs that provide resources to applications of resource deficit SPs.

5. We also prove that the proposed algorithms, GPOA and PPMPOA, enforce *truth-telling*, i.e., no service provider has the incentive to misreport its capacity and native application requests.

6. We evaluate the performance of our proposed framework by extensive simulations. We verify the game-theoretic properties of our proposed approach by showing that the game is super-additive and the core is non-empty, i.e., our obtained solutions are from the core and satisfy properties, such as *individual rationality*, *group rationality*, and *stability.* We also show that the resource sharing and allocation framework improves the utilities of game players. Furthermore, our framework also improves application (user) satisfaction.

The rest of the chapter is structured as follows. Section 6.2 describes our system model and presents the resource sharing/allocation optimization problem. Section 6.3 discusses the game-theoretic solution and Section 6.4 presents our experimental results. Section 6.5 concludes the chapter.

6.2. System Model

The basic system model for this chapter is the same as chapter 5. However, the value of a coalition is slightly different. This is because the SPs employ the non-uniform priority strategy where all the SPs allocate resources to native applications and share the surplus resources with resource deficit SPs. For sake of completeness and to make the chapter self-sufficient, we restate the system model below.

Let $\mathcal{N} = \{1, 2, \cdots, N\}$ be the set of all edge SPs. We assume that each provider has a set of $\mathcal{J} = \{1, 2, \cdots, J\}$ types of resources, such as communication, computation and storage resources. $C_n = \{C_{n,1}, \cdots, C_{n,J}\}$, with $C_{n,j}$ denoting the amount of type j resource available at service provider n. Each service provider n has a set of native applications $\mathcal{M}_n = \{1, 2, \cdots, M_n\}$. The set of all applications that request resources from all service providers is given by $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cdots \cup \mathcal{M}_N$, where we assume $\mathcal{M}_n \cap \mathcal{M}_{n'} = \emptyset$, $\forall n \neq n'$, i.e., each application initially demands resources from only its native service provider. We also define $\overline{\mathcal{M}}$ to represent $\mathcal{M} \backslash \mathcal{M}_n$ and $\overline{\mathcal{N}}$ to represent $\mathcal{N} \backslash \{n\}$. Every SP $n \in \mathcal{N}$ has a request (requirement) matrix $\mathbf{R}_n$,

$$\mathbf{R}_n = \begin{bmatrix} \mathbf{r_n^1} \\ \vdots \\ \mathbf{r_n^{M_n}} \end{bmatrix} = \begin{bmatrix} r_{n,1}^1 & \cdots & \cdots & r_{n,J}^1 \\ \vdots & \vdots & \vdots & \vdots \\ r_{n,1}^{M_n} & \cdots & \cdots & r_{n,J}^{M_n} \end{bmatrix}, \tag{6.1}$$

Here $r^m_{n,j}$ is the amount of resource j that application $m \in \mathcal{M}_n$ requires. When the service provider works alone[1], its objective is to maximize its utility by allocating resources to its native applications and improving application satisfaction. A service provider n earns a non-negative *monotonically non-decreasing* utility $u^m(x^m_{n,j})$ by making allocation decision $x^m_{n,j}$, i.e., allocating $x^m_{n,j}$ amount of resource j to application $m \in \mathcal{M}_n$, where $\mathbf{x}^m_n = [x^m_{n,1}, x^m_{n,2}, \cdots, x^m_{n,J}]^T$. Table 6.1 summarizes the notations used throughout the chapter. We present the optimization formulation for a single service provider in Section 6.2.1, followed by the multiple service provider problem in Section 6.2.2.

6.2.1. Problem Formulation for Single Service Provider

We first present the resource allocation problem for a single, stand-alone provider (i.e., no resource sharing with other providers). For each provider $n \in \mathcal{N}$, the allocation decisions consist of the vectors $\mathbf{x}^1_n, \cdots, \mathbf{x}^{M_n}_n$. The optimization problem is:

$$\max_{\mathbf{x}^1_n, \cdots, \mathbf{x}^{M_n}_n} \quad \sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} f^m(x^m_{n,j}), \tag{6.2a}$$

$$\text{s.t.} \quad \sum_m x^m_{n,j} \leq C_{n,j}, \quad \forall j \in \mathcal{J}, \tag{6.2b}$$

$$x^m_{n,j} \leq r^m_{n,j}, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}, \tag{6.2c}$$

$$x^m_{n,j} \geq 0, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}. \tag{6.2d}$$

The objective function $f^m(x^m_{n,j})$ is required to be a monotonic, non-negative and non-decreasing function. For illustration purposes in this thesis, we assume that:

$$f^m(x^m_{n,j}) = u^m(x^m_{n,j}) + \frac{x^m_{n,j}}{r^m_{n,j}},$$

where the term $\frac{x^m_{n,j}}{r^m_{n,j}}$ captures the request satisfaction and $u^m(.)$ is the non-decreasing and non-negative utility the service provider earns by providing the resources to applications. The first constraint in (6.2b) indicates that the allocated resources cannot exceed capacity. The second constraint in (6.2c) indicates that the allocated resources should not exceed the required amounts. The last constraint, (6.2d) requires the allocation to be non-negative.

The goal of this single-objective optimization problem for every provider, as mentioned earlier, is to maximize its utility by allocating available resources to its native applications and improving application satisfaction. We note that this single-provider formulation does not consider the following. For example, it is possible that a service provider n may earn a larger utility by providing its resources to applications of other service providers. Furthermore, provider(s) may not have sufficient resources to satisfy requests of all its native applications, while other providers may have resource surpluses that can be "rented" and utilized by other providers. As mentioned earlier, in this chapter we focus on strategies

[1]Not borrowing from or renting out its resources to any other service provider.

Table 6.1.: List of notations used throughout chapter 6

Notation	**Description**
$\mathcal{N}, N, n$	Set, number and index of game players (service providers)
$\mathcal{J}, J, j$	Set, number and index of resources
$\mathcal{M}, M, m$	Set, number and index of applications
$\mathcal{M}_n$	Set of native applications at player (provider) n
S	A coalition of game players (providers), where $S \subseteq \mathcal{N}$
$\mathcal{V}$	Set of payoff vectors
$v(S)$	Value of coalition S
$\mathcal{G}_1$	Set of players (providers) with resource deficit
$\mathcal{G}_2$	Set of players (providers) with resource surplus
C	Capacity vector of all game players (providers)
C_n	Capacity vector of player (provider)n
$C_{n,j}$	Capacity of resource j at player (provider) n
$\mathbf{R}_n$	Request matrix at player (provider) n
$r_{n,j}^m$	Request of application m for resource j from player (provider) n
$x_{n,j}^m$	Allocation decision of resource j for application m at player (provider) n
$\mathbf{x}_n^m$	Allocation decision vector for application m at player (provider) n when working alone
$\mathbf{X}_n$	Allocation decision for player (provider) n in the coalition
$\mathbf{X}$	Allocation decision for the entire coalition
$\mathcal{X}_n$	Allocation decision for player (provider) n in the coalition excluding its native applications
${}^{n'}\mathbf{X}_n$	Allocation decision for applications of player (provider) n' at player (provider) n in the coalition s
$u^m(x_{n,j}^m)$	Utility player (provider) n earns by allocating resource j to application m
$u^m(\mathbf{x}_n^m)$	Utility player (provider) n earns by allocating vector of resources to application m
GC	Grand coalition
RS	Request satisfiability
TU	Transferable utility
NTU	Non-transferable utility
CGT	Cooperative game theory
MOO	Multi-objective optimization
SOO	Single-objective optimization

where each service provider allocates resources to its native applications and then shares any remaining resources with other service providers who are not able to fully satisfy their native applications. These aspects are considered in the following section.

6.2.2. Multiple Service Provider Setting

Let us assume that a service provider n' does not have sufficient resources to satisfy all its native applications. Suppose that service provider n has a resource surplus, after allocating its resources to its native applications that can be shared with n'. Allowing resource sharing among such providers can improve resource utilization and application satisfaction. By sharing and allocating its type j resources to application l, service provider n earns a non-decreasing and non-negative net utility $H^l(x^l_{n,k})$, which is assumed to be

$$H^l(x^l_{n,j}) = \Big(u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j) - D^l(x^l_{n,j})\Big), \forall l \in \overline{\mathcal{M}}, n \in \mathcal{N} \backslash n', \tag{6.3}$$

where $z^l_j = \sum_{n'' \in \overline{\mathcal{N}}} x^l_{n'',j}$ is the amount of resource j allocated already to application l by providers other than n. $u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j)$ captures the incremental increase in utility of provider n due to the allocation of additional resource $x^l_{n,j}$. $D^l(x^l_{n,j})$ represents the communication cost of serving application $l \in \mathcal{M}_{n'}$ by using resource of type j at provider n rather than its native service provider n'. We assume that the communication cost is a positive monotonic non-decreasing function.

To consider each provider and its native applications, let $F^m(x^m_{n,j})$ represent the utility service provider n earns by allocating resource j to its native application m along with the increase in request satisfaction due to provision of resources to its applications at different service providers including itself. We assume:

$$F^m(x^m_{n,j}) = u^m(x^m_{n,j}) + \frac{\sum_{n' \in \mathcal{N}} x^m_{n',j}}{r^m_{n,j}}. \tag{6.4}$$

Note that F^m is monotonic non-decreasing and non-negative. Furthermore, we implicitly assume that $u^m(x^m_{n,j}) = u^m(x^m_{n',j})$, if $x^m_{n,j} = x^m_{n',j}$. The latter assumption reflects that different providers earn the same utility when allocating the same amount of resources of the same type to a given application, either locally or remotely. Furthermore,

$$\mathbf{x}^m = [\sum_{n \in \mathcal{N}} x^m_{n,1}, \sum_{n \in \mathcal{N}} x^m_{n,2}, \cdots, \sum_{n \in \mathcal{N}} x^m_{n,J}]^T.$$

That is, the total resources allocated to any application $m \in \mathcal{M}$ is the sum of resource allocated to it across all service providers. The resource sharing and allocation framework, based on resource requests and capacities of service providers, has to make allocations that optimize the utilities of all the service providers $n \in \mathcal{N}$ while satisfying user requests as well. The allocation decision is given by $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$, where $\mathbf{X}_n$, $\forall n \in \mathcal{N}$ is given by:

$$\mathbf{X}_n = \begin{bmatrix} \mathbf{x}^1_n \\ \vdots \\ \mathbf{x}^{|\mathcal{M}|}_n \end{bmatrix} = \begin{bmatrix} x^1_{n,1} & \cdots & \cdots & x^1_{n,J} \\ \vdots & \vdots & \vdots & \vdots \\ x^{|\mathcal{M}|}_{n,1} & \cdots & \cdots & x^{|\mathcal{M}|}_{n,J} \end{bmatrix}. \tag{6.5}$$

When service providers share resources, each service provider aims to maximize the sum of utilities obtained by a) allocating resources to its native applications; b) allocating resources (if available) to applications of other service providers; and c) improving the request satisfaction of its applications by using its own resources or borrowing from other service providers (when needed and possible).

$$\max_{\mathbf{X}_n} \quad \sum_{m\in\mathcal{M}_n}\sum_{j\in\mathcal{J}} F^m(x^m_{n,j}) + \sum_{l\in\overline{\mathcal{M}}}\sum_{j\in\mathcal{J}} H^l(x^l_{n,j}), \tag{6.6a}$$

$$\text{s.t.} \quad \sum_{m} x^m_{n,j} \le C_{n,j}, \quad \forall\, m \in \mathcal{M}_n, j \in \mathcal{J}, \tag{6.6b}$$

$$\sum_{m'} x^{m'}_{n,j} \le \max\{C_{n,j} - \sum_{m\in\mathcal{M}_n} r^m_{n,j}, 0\}, \quad \forall\, m' \in \overline{\mathcal{M}}, j \in \mathcal{J}, \tag{6.6c}$$

$$\sum_{n'\in\mathcal{N}} x^m_{n',j} \le r^m_{n,j}, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, \tag{6.6d}$$

$$x^m_{n,j} \ge 0, \quad \forall\, m \in \mathcal{M}, j \in \mathcal{J}, \tag{6.6e}$$

$$u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j) \ge D^l(x^l_{n,j}), \forall l \in \overline{\mathcal{M}}, j \in \mathcal{J}, \tag{6.6f}$$

where $F^j(.)$ and $H^l(.)$ are given in (6.3) and (6.4). Constraint in (6.6b) indicates that resources allocated to native application cannot exceed the capacity. Constraint in (6.6c) ascertains that resources are first allocated to native applications and only surplus resources are shared with non-native applications. It also indicates that resources allocated to non-native applications cannot exceed the surplus capacity (if any). Constraint in (6.6d) indicates that allocated resources cannot exceed the requests whereas constraint in (6.6e) indicates that the allocation decision is non-negative. Constraint in (6.6f) indicates that the utility earned by sharing resources must be higher than the communication cost.

Since the utility u^j, $\forall n \in \mathcal{N}$ can differ for each application j, (6.6) for all providers n represents a multi-objective optimization problem. However, solving (6.6) is not straightforward since the objective function (6.6a) for an SP n also depends on the allocation decisions and strategies of other SPs. Therefore, we solve this problem by a game-theoretic approach as follows.

6.3. Game-Theoretic Solution

In this section, we first present general properties of our game and show that the resource-sharing problem for multiple providers employing the non-uniform priority strategy can be modeled as a canonical cooperative game with non-transferable utility. We then show that the core is non-empty, by proving that our canonical game is *cardinally convex* [109]. Finally, we propose algorithms in Section 6.3.2 to obtain allocations from the core.

6.3.1. General Game Properties

We model each service provider as a player in our game to obtain the optimal resource sharing/allocation decision. Let $\mathcal{N}$ be the set of players that play the resource sharing and allocation game. The *value of coalition* for the game players $S \subseteq \mathcal{N}$ is given by (6.7), where

$\mathcal{F}_S$ is the feasible set for resource sharing and allocation.

$$v(S) = \sum_{\substack{n \in S \\ \mathcal{X} \in \mathcal{F}_S}} \Bigg(\sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \big(u^m(x^m_{n,j}) + \frac{\sum_{n' \in S} x^m_{n',j}}{r^m_{n,j}}\big) + \Big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z^l_j + x^l_{n,j}) - u^l(z^l_j) - D^l_n(x^l_{n,j})\big)\Big)\Bigg). \tag{6.7}$$

Remark 6.1. *A resource allocation and sharing problem (with multiple objectives) for the aforementioned system model where the SPs employ non-uniform priority strategy can be modeled as a canonical cooperative game with NTU.*

This is so because the payoff is of characteristic form and the game is super-additive as given below.

- **Characteristic form of payoff:** Since the utility function in our formulated resource sharing and allocation problem only relies on service providers that are part of the coalition, the game or payoff function is of characteristic form.
- **Superadditivity:** Let $S_1, S_2 \subseteq \mathcal{N}$ where S_1, S_2 are non-empty and $S_1 \cap S_2 = \emptyset$. Hence, $S_1,\ S_2 \subset (S_1 \cup S_2)$. Superadditivity follows from the monotonic utility functions.

Theorem 6.1. *Our canonical game with non-uniform priority is cardinally convex.*

Proof. This can easily be proved using the arguments given in the proof of Theorem 5.1. □

Since the game is cardinally convex, our canonical resource sharing game has a large core.

Remark 6.2. *Our canonical cooperative game $(\mathcal{N}, \mathcal{V})$ with NTU can be used to obtain the Pareto-optimal solutions for multi-objective resource sharing problem among many service providers.*

The convexity of our game proves that the core is non-empty. However, obtaining an allocation from the core is challenging, which we address in the following subsection.

6.3.2. Proposed Algorithms

In the non-uniform priority strategy, as mentioned earlier, all the SPs first allocate resources to their native applications. Once that is done, some of the SP may have resource surpluses while the others may have resource deficits. The goal then for the SPs with resource surplus is to achieve a larger utility for its available resources. The goal for the SPs with resource deficit is to avail as many resources as possible for their applications. Therefore, we propose two different algorithms in this section to achieve allocations from the core. However, the two algorithms have a subtle difference in how they allocate resources of SPs with surpluses to applications of SPs with deficits. We first discuss our proposed *Game-theoretic Pareto Optimal Allocation* (GPOA) that enables an SP with resource surplus to jointly consider all the SPs with resource deficits when allocating resources. This means that GPOA enables SPs with resource surpluses to maximize their utilities by considering all the applications. We then describe our second algorithm, *Polyandrous-Polygamous Matching based Pareto Optimal Allocation* (PPMPOA), that enables an SP with resource surplus to consider SPs with resource deficits one by one when allocating resources. We show that both these algorithms provide allocations from the core. Below we discuss these algorithms in detail.

Game-theoretic Pareto Optimal Allocation

We propose a computationally efficient algorithm, GPOA in Algorithm 6.1, that obtains a Pareto-optimal allocation by solving $N + |\mathcal{G}_2|$ optimization problems, where $\mathcal{G}_2$ represents the set of players with resource surplus. Inputs are available resources/capacities, application requests and utilities of all game players. The algorithm outputs an allocation. In Step 1, utilities of players, the allocation decision, and vectors $\boldsymbol{v}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are initialized. $\boldsymbol{v}$ stores the utility each service provider achieves when working alone. $\boldsymbol{A}$ stores the utility of players in $\mathcal{G}_2$ whereas $\boldsymbol{B}$ stores the utility increase due to request satisfaction of applications in resource deficit group $\mathcal{G}_1$. Step 2 calculates every player's utility in the absence of resource sharing, i.e., the single objective optimization problem in (6.2) is solved by every player. To consider the resources allocated in Step 2, the remaining resource capacities and requests for various applications C, R are updated and stored in C' and R', respectively. Step 4 divides players into two groups $\mathcal{G}_1$ (resource deficit) and $\mathcal{G}_2$ (resource surplus), on the basis of updated capacities and requests. Step 5 allocates the shared resources of players in $\mathcal{G}_2$ to players in $\mathcal{G}_1$. In step 6, the utility reflecting an increase in application satisfaction for native applications of players in $\mathcal{G}_1$ is calculated using the allocation obtained in Step 5. Below, we prove that the GPOA Algorithm given in Algorithm 6.1 provides an allocation from the core.

Theorem 6.2. *The solution obtained using GPOA lies in the* core.

Proof. To prove the theorem, we need to show that a) utilities obtained using GPOA are individually rational and group rational, and b) no group of players has the incentive to leave the grand coalition to form another sub-coalition $S \subset \mathcal{N}$.

Individual Rationality: For each player $n \in \mathcal{N}$, $v(\{n\})$ obtained by solving (6.2) is the utility a player obtains by working alone in the absence of resource sharing. Because the utilities are non-negative, $A_n \geq 0$, $\forall n \in \mathcal{N}$. Furthermore utilities of players in $\mathcal{G}_1$ may increase due to increase in request satisfaction if its applications are provided additional resources by players in $\mathcal{G}_2$ given by $B_{n'}$. Hence GPOA produces an individually rational resource allocation.

Group Rationality: The value of the grand coalition $v\{\mathcal{N}\}$ as per (6.7) is the sum of different utilities. Steps 2, 5 and 6 of GPOA obtain the sum of utilities as well. Hence the

Algorithm 6.1 Game-theoretic Pareto Optimal Allocation (GPOA)

```
Input: $R, C$, and vector of players' utility function $\mathbf{u}$
Output: $\mathbf{X}$
Step 1: $\mathbf{u}(\mathbf{X}) \leftarrow 0$, $\mathcal{X} \leftarrow 0$, $\boldsymbol{v} \leftarrow 0$, $\boldsymbol{A} \leftarrow 0$, $\boldsymbol{B} \leftarrow 0$, $\boldsymbol{z} \leftarrow 0$
Step 2:
for $n \in \mathcal{N}$
    $v(\{n\}) \leftarrow$ Objective function value of (6.2)
    $\mathbf{x}_n^1, \cdots, \mathbf{x}_n^{M_n} \leftarrow$ Allocation decision of (6.2)
    Update $z^m$'s based on the allocation decision of (6.2)
end for
Step 3: Update $C$ and $R$ based on Step 2
        $C' \leftarrow C_{updated}$, $R' \leftarrow R_{updated}$
Step 4: Divide the players into two subsets $\mathcal{G}_1$ and $\mathcal{G}_2$ representing players with resource
deficit and resource surplus
Step 5:
for $n \in \mathcal{G}_2$
    $\mathcal{X}_n = \mathbf{X}_n \backslash \{\mathbf{x}_n^1, \cdots, \mathbf{x}_n^{M_n}\} \leftarrow$ Optimal allocation decision of (6.8) $\forall n' \in \mathcal{G}_1$
    $A_n \leftarrow$ Utility earned by $n \in \mathcal{G}_2$ due to resource allocation $\mathcal{X}_n$
    Update $z_j^m$'s based on $\mathcal{X}_n$
    Update $C'$ and $R'$
end for
Step 6:
for $n' \in \mathcal{G}_1$
    $B_{n'} \leftarrow$ Utility earned by satisfying users $m \in \mathcal{M}_{n'}$ due to allocation
decision in Step 5
end for
```

solution obtained as a result of GPOA is group rational.

Furthermore, due to superadditivity of the game and monotonic non-decreasing nature of the utilities, no group of players has the incentive to form a smaller coalition. Hence GPOA provides a solution from the core. □

Remark 6.3. *GPOA is a distributed algorithm. All players first allocate resources to their native applications and then update their capacities and requests. The updated requests and capacities are broadcasted by all players to obtain $\mathcal{G}_1$ and $\mathcal{G}_2$. The order to execute Step 5 is chosen based on some criteria (see details below). Players in $\mathcal{G}_2$, after allocating resources, send the updated requests to other players in $\mathcal{G}_2$ that have not yet executed Step 5.*

Remark 6.4. *Any payoff allocation from the core generated by GPOA is Pareto-optimal.*

Remark 6.5. *The allocation decision obtained using GPOA always belongs to the core irrespective of the order in which players execute Step 5.*

$$\max_{\mathcal{X}_n} \quad \sum_{n' \in \mathcal{G}_1} \left(\sum_{m \in \mathcal{M}_{n'}} \sum_{j \in \mathcal{J}} \left(\left(u^m({z_j}^m + x_{n,j}^m) - u^m(z_j^m) - D^m(x_{n,j}^m)\right) \right) \right), \ \forall n \in \mathcal{G}_2, \tag{6.8a}$$

$$\text{s.t.} \quad \sum_m x_{n,j}^m \leq C'_{n,j}, \ \forall j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}, \forall n' \in \mathcal{G}_1, \tag{6.8b}$$

$$x^m_{n,j} \leq r'^m_{n',j}, \quad j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}, \forall n' \in \mathcal{G}_1, \tag{6.8c}$$
$$x^m_{n,j} \geq 0, \quad j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}, \forall n' \in \mathcal{G}_1, \tag{6.8d}$$
$$u^m(z^m_j + x^m_{n,j}) - u^m(z^m_j) \geq D^m(x^m_{n,j}),\ j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}, \forall n' \in \mathcal{G}_1. \tag{6.8e}$$

Based on Remark 6.5, it is important to note that GPOA can generate different solutions according to the order in which players execute Step 5. Some candidate ordering schemes are listed below.

1. **Capacity ascending order (CAO):** Players in $\mathcal{G}_2$ are arranged in an ascending order on the basis of the remaining capacities of a resource j, which is then used for executing Step 5.
2. **Capacity descending order (CDO):** Players in $\mathcal{G}_2$ are arranged in a descending order on the basis of the remaining capacities of a resource j, which is then used for executing Step 5.
3. **Random Order:** The order in which players execute Step 5 is random.

While all these variants provide allocations from the core, finding the most beneficial order for the `for-loop` in Step 5 requires evaluating all possible combinations of possible orders for executing the `for-loop`.

Next, we present below a matching-based resource sharing algorithm that intends to reduce the number of SPs that provide resources to applications native to SPs with resource deficits. The advantage of reducing the number of SPs that provide resources to applications of resource deficit SPs is that a) it eliminates the overhead of negotiating and coming to an agreement with different SPs; b) applications receive resources from the minimum number of SPs that reduces resource fragmentation, i.e., the splitting of resources across different SPs.

$$\max_{n'\mathbf{X}_n} \quad \sum_{m \in \mathcal{M}_{n'}} \sum_{j \in \mathcal{J}} \big(u^m({z_j}^m + x^m_{n,j}) - u^m({z_j}^m) - D^m(x^m_{n,j})\big), \forall n' \in \mathcal{G}_1, n \in \mathcal{G}_2 \tag{6.9a}$$
$$\text{s.t.} \quad \sum_m x^m_{n,j} \leq C'_{n,j},\ \forall j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}, \tag{6.9b}$$
$$x^m_{n,j} \leq r'^m_{n',j}, \quad j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}, \tag{6.9c}$$
$$x^m_{n,j} \geq 0, \quad j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'} \tag{6.9d}$$
$$u^m(z^m_j + x^m_{n,j}) - u^m(z^m_j) \geq D^m(x^m_{n,j}),\ j \in \mathcal{J}, \forall m \in \mathcal{M}_{n'}. \tag{6.9e}$$

Polyandrous-Polygamous Matching for Resource Sharing

Our proposed algorithm is based on polyandrous polygamous matching [111] that matches players (service providers) in $\mathcal{G}_1$ to $\mathcal{G}_2$ with the goal of maximizing utility for players in

$\mathcal{G}_2$ and reducing the number of SPs with resource surpluses that provide resources to applications of players in $\mathcal{G}_1$. The idea behind our matching algorithm, inspired from *two-sided markets* [27], is as follows. Players in $\mathcal{G}_1$ have specific preferences over which players in $\mathcal{G}_2$ they would like to obtain resources from on the basis of the amount of resources that players in $\mathcal{G}_2$ can provide. On the other hand, players in $\mathcal{G}_2$ aim to maximize their utilities. Since our framework relies on monotonic non-decreasing utilities, the objectives of players in $\mathcal{G}_1$ and $\mathcal{G}_2$ can be mapped to each other, i.e., the more resources players in $\mathcal{G}_2$ share with players in $\mathcal{G}_1$, the higher will their utilities be and the lower will the number of SPs be with resource surpluses that provide resources to the applications of resource deficit SPs. Therefore, during each round, resources of one player in $\mathcal{G}_2$ are assigned to applications of a player in $\mathcal{G}_1$ that produces the largest increase in its utility. This means that a $\mathcal{G}_2$ player will provide as many resources as possible (increasing its utility) that in turn should reduce number of SPs with resource surpluses that provide resources to applications in $\mathcal{G}_1$. Our scheme will transform into *polygamous*/many-to-one matching [27] or *stable marriage*/one-to-one matching [112] in the best case scenario. Below, we discuss the different steps of the algorithm.

In Algorithm 6.2, all variables are initialized in Step 1. $\boldsymbol{\rho}$ is a vector that stores allocations for every application $m \in \mathcal{M}$ across different players. In Step 2, all game players $n \in \mathcal{N}$ allocate resources to their native applications. Request matrices and capacity vectors are updated in Step 3, and players are divided into two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ in Step 4. Here $\boldsymbol{A}$ and $\boldsymbol{B}$ record utilities of players in $\mathcal{G}_2$ and $\mathcal{G}_1$, respectively. Step 5 is the core of the algorithm. In Step 5a, Algorithm 6.3 is used to construct a matching matrix Υ. The elements of matching matrix Υ (n' rows and n columns) are obtained by solving and obtaining the objective function value of the problem in (6.9) for every possible pair of $(n' \in \mathcal{G}_1, n \in \mathcal{G}_2)$. (6.9) maximizes the incremental utility a player in $\mathcal{G}_2$ earns by sharing its resources with a player in $\mathcal{G}_1$. (6.9b) indicates that the allocated resources cannot exceed the capacity. (6.9c) indicates that the allocation cannot be more than the required resources. (6.9d) shows that the allocation decision is non-negative whereas (6.9e) indicates that the incremental increase in utility cannot be less than the communication cost.

Algorithm 6.3 also provides total amount of resources used for each element of Υ and allocation decision ${}^{n'}X_n$ that captures how resources are allocated to all applications $m \in \mathcal{M}_{n'}$ for $n' \in \mathcal{G}_1$ at player $n \in \mathcal{G}_2$. In 5b, the largest element in Υ, given by $\Upsilon_{n',n}$, is used to match n' to n, i.e., the player n (with resource surpluses) shares its resources with the player n' (with resource deficits). If there are multiple maximum values, then the one that uses smallest number of resources is used to choose the matching. In 5c, applications $m \in \mathcal{M}_{n'}$ are assigned to $n \in \mathcal{G}_2$. In 5d, $\boldsymbol{A}_n$ is updated with the utility of player $n \in \mathcal{G}_2$ based on the assignment in 5c. Similarly, $\boldsymbol{B}_{n'}, \mathcal{X}_n$ and $\boldsymbol{\rho}^m$ are updated to reflect the increase in application satisfaction, resources allocated by n to different applications and resources allocated to $m \in \mathcal{M}_{n'}$, respectively. Requests and capacities are updated in 5e whereas 5f updates elements of $\mathcal{G}_1$ and $\mathcal{G}_2$ if required.

Algorithm 6.2 Polyandrous-Polygamous Matching based Pareto Optimal Allocation (PPMPOA)

```
Input: $R, C$, and vector of players' utility function $\mathbf{u}$
Output: $\mathbf{X}$, $\mathbf{u}(\mathbf{X})$
Step 1: $\mathbf{u}(\mathbf{X}) \leftarrow 0$, $\mathbf{X} \leftarrow 0$, $\boldsymbol{v} \leftarrow 0$, $[\Upsilon]_{|\mathcal{G}_1| \times |\mathcal{G}_2|} \leftarrow \mathbf{0}, \boldsymbol{\rho} \leftarrow \mathbf{0}$
Step 2:
for $n \in \mathcal{N}$
    $v(\{n\}) \leftarrow$Objective function value from (6.2)
    $\mathbf{x}_n^1, \cdots, \mathbf{x}_n^{M_n} \leftarrow$Allocation decision from (6.2)
    Update $z^m$'s based on the allocation decision of (6.2)
end for
Step 3: Update $C$ and $R$ based on Step 2
        $C' \leftarrow C_{updated}$, $R' \leftarrow R_{updated}$
Step 4: Divide the players into two subsets $\mathcal{G}_1$ and $\mathcal{G}_2$ representing players with resource deficit and resource surplus
          $\boldsymbol{A} \leftarrow 0$, $\boldsymbol{B} \leftarrow 0$
Step 5:
while  $\mathcal{G}_1 \neq \emptyset$ || $\mathcal{G}_2 \neq \emptyset$
    Step 5a: $[\Upsilon]_{|\mathcal{G}_1| \times |\mathcal{G}_2|} \leftarrow$ Algorithm 6.3 for constructing matching matrix
    Step 5b:
    if Multiple maximum values in $[\Upsilon]$
        Choose one with the lowest $\mathcal{R}_{n',n}$
    else
        Obtain the row $n'$ and column $n$ of the maximum value in $[\Upsilon]$
    end if
    Step 5c: Assign all $m \in \mathcal{M}_{n'}, n' \in \mathcal{G}_1$ to $n \in \mathcal{G}_2$
    Step 5d: Update $\boldsymbol{A}_n, \boldsymbol{B}_{n'}, \mathcal{X}_n$ and $\boldsymbol{\rho}^m$ using the preceding assignment
    Step 5e: Update $C_n$ and $R_{n'}$
    Step 5f:
    if  $C_n = 0$,
        $\mathcal{G}_2 = \mathcal{G}_2 \backslash n$
    end if
    if  $R_{n'} = 0$,
        $\mathcal{G}_1 = \mathcal{G}_1 \backslash n'$
    end if
end while
```

Definition 6.1. *Objection to a matching [27]: A player $n' \in \mathcal{G}_1$ and $n \in \mathcal{G}_2$ object to a matching Ψ if they both prefer being matched to each other than to whom they are matched by Ψ.*

Definition 6.2. *Stable matching [27]: A matching Ψ is stable if there is no pair $n' \in \mathcal{G}_1$ and $n \in \mathcal{G}_2$ that objects to a matching.*

Theorem 6.3. *The polyandrous-polygamous matching in Algorithm 6.2 is stable.*

Proof. We prove this by contradiction. Assume that the matching Ψ obtained from Algorithm 6.2 is not stable and there exists a pair $(n' \in \mathcal{G}_1, n \in \mathcal{G}_2)$ that objects to the

Algorithm 6.3 Constructing matching matrix Υ

```
**Input**: $R, C$, $\mathbf{u}, \boldsymbol{\rho}$
**Output**: $\Upsilon, \mathcal{R}, {}^{n'}\boldsymbol{X}_n$
**for** $n \in \mathcal{G}_2$
    **for** $n' \in \mathcal{G}_1$
        $\Upsilon_{n',n} \leftarrow$ Payoff for allocation decision from (6.9)
        $\mathcal{R}_{n',n} \leftarrow$ Resources allocated in (6.9)
    **end for**
**end for**
```

matching. Ψ has originally matched $n' \in \mathcal{G}_1$ to $\bar{n}$ and $n \in \mathcal{G}_2$ to n''. The objection by n means that it can achieve a higher utility by being matched with n' when compared with the current match $n'' \in \mathcal{G}_1$. However, this is not possible since n'' was matched with n based on the maximization problem given in (6.9), i.e., n'' provides the largest utility to n for its resources. Hence, our assumption is wrong and the matching is stable. $\square$

Remark 6.6. *PPMPOA given in Algorithm 6.2 provides an allocation from the core.*

Algorithm scalability and run-time: To obtain an allocation from the core, GPOA solves $N + |\mathcal{G}_2|$ optimization problems. In particular, we need to solve (6.2) for each of the N SPs and then solve (6.8) for all the SPs in $\mathcal{G}_2$. The number of decision variables and constraints in (6.2) depends on the total number of resource types J and the total number of native applications at SP n given by $M_n = |\mathcal{M}_n|$. Hence, the number of decision variables and constraints in (6.2) is bounded as $\mathcal{O}(M_n J)$. The number of decision variables and constraints in (6.8) depends on the total number of SPs in $\mathcal{G}_1$, resource types J, and applications in $\mathcal{G}_1$ given by $M_{\mathcal{G}_1}$. Therefore, the number of decision variables and constraints in (6.8) is bounded as $\mathcal{O}(|\mathcal{G}_1| M_{\mathcal{G}_1} J)$, where $M_{\mathcal{G}_1}$ can be large when compared to $|\mathcal{G}_1|$ and J.

PPMPOA solves $N + \nu|\mathcal{G}_1||\mathcal{G}_2|$, where $\nu > 0$ represents the finite number of iterations of Step 5 in Algorithm 6.2. Just like GPOA, PPMPOA needs to first solve (6.2) for each of the N SPs. Then it needs to solve (6.9) $\nu|\mathcal{G}_1||\mathcal{G}_2|$ times. The number of decision variables and constraints in (6.9) is bounded as $\mathcal{O}(M_n J)$.

Remark 6.7. *The optimization problems in* (6.2), (6.8) *and* (6.9) *are non-convex. The hardness of these problems depend on the parameters, such as the utility functions, requests and capacities. Therefore, it is difficult to evaluate the hardness of these problems, but the instances we consider in Section 6.4 are easily solved using existing solvers. In general, certain non-convex optimization problems can usually be solved to optimality by proving that* strong duality *holds [21], and then solving the convex dual of these non-convex problems. Detailed discussions on strong duality can be found in [21] whereas [22] presents the necessary and sufficient conditions for strong duality of non-convex problems. If the optimal solutions are not found for* (6.2), (6.8) *or* (6.9), *then the allocation will not be from the core.*

Lemma 6.1. *Our resource sharing framework improves request satisfaction.*

Proof. Let the average application request satisfaction for the non-resource sharing case be given by $RS_{n,1}$, where $RS_{n,1}$ is obtained following the resource allocation decision in Step 2 of Algorithms 6.1 and 6.2, $\forall n \in \mathcal{N}$. Since players in $\mathcal{G}_2$ share their resources with players in $\mathcal{G}_1$, average user satisfaction as a result of the allocation, given by $RS_{n,2}$ is also positive, i.e., > 0. Hence, user satisfaction achieved using our framework is given by $RS_n = RS_{n,1} + RS_{n,2}$, where $RS_n \geq RS_{n,1}$. □

6.3.3. Strategy-Proof Incentive

In this section, we answer the question: *Is there an incentive for a single player or group of players to cheat and report incorrect capacities and application requests to other players?* That is, will misreporting the capacities or application requests improve the payoff a player or group of players receive? The answer is no as given by the following Theorem.

Theorem 6.4. *When the service providers use Algorithms 6.1 and 6.2 for resource sharing and allocation, no player or group of players has any incentive to cheat and misreport capacities and application requests, i.e., each player $n \in \mathcal{N}$ can achieve the highest payoff by truthfully reporting its capacity and requests.*

Proof. Assume a player $n \in \mathcal{N}$ improves its utility $\mathcal{U}_n$ by misreporting its capacity and application requests, i.e., $\mathcal{U}'_n > \mathcal{U}_n$ where $\mathcal{U}'_n$ is the utility obtained by cheating and $\mathcal{U}_n$ is the utility obtained by truthful reporting. Each player receives its payoff for providing resources to its own applications first and then sharing the remaining resources (if any) with the applications of other service providers. Player payoff improves once resources are provided and user requests are satisfied.

The larger the amount of resources provided and requests satisfied, the higher will be the obtained payoff. There are two cases for cheating:

- **Under-reporting[2] the capacity and application requests:** In this case, the player will not receive the payoff possible by satisfying its applications and fully utilizing its capacity by allocating resources to its own applications and applications of other players. This contradicts the assumption that $\mathcal{U}'_n > \mathcal{U}_n$. Hence, the player has no incentive to under-report its capacity and application requests.

- **Over-reporting[3] the capacity and application requests:** Since the payoff depends on the actual amount of resources provided and requests satisfied rather than the reported capacity and application requests, the highest payoff possible for the player is $\mathcal{U}_n$ contradicting our assumption that $\mathcal{U}'_n > \mathcal{U}_n$.

□

[2]Reporting a lower value than the actual value.
[3]Reporting a larger value than the actual value.

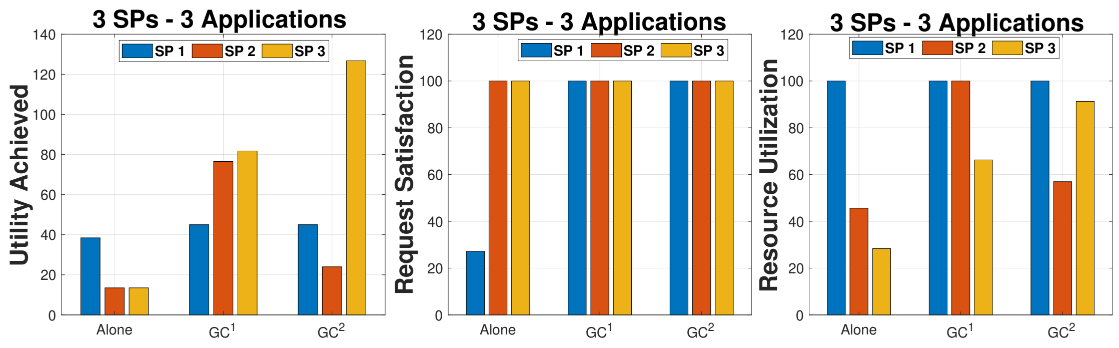

Figure 6.1.: Performance of our framework with GPOA for 3 player - 3 application settings.

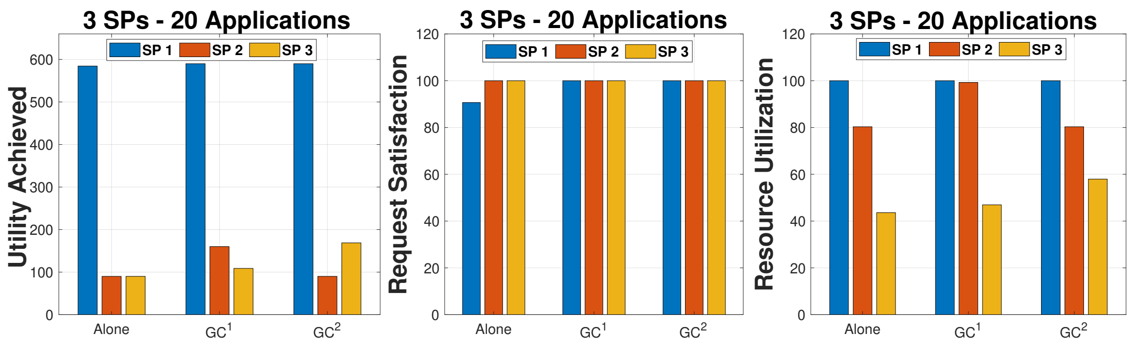

Figure 6.2.: Performance of our framework with GPOA for 3 player - 20 application settings.

6.4. Experimental Results

We evaluate the performance of proposed resource sharing and allocation framework for several parameter settings as shown in Table 5.2, where $\mathcal{G}_1$ and $\mathcal{G}_2$ represent the set of service providers with resource deficits and surpluses respectively. Each player has three different types of resources ($J = 3$), i.e., storage, communication and computation. Without loss of generality, the model can be extended to include other types of resources. We use linear and sigmoidal utilities for all the players given below.

$$u^m(x^m_{n,j}) = a x^m_{n,j} + c, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}, m \in \mathcal{M}, \tag{6.10}$$

$$u^m(x^m_{n,j}) = \frac{1}{1 + e^{-\mu(x^m_{n,j} - r^m_{n,j})}}, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}, m \in \mathcal{M}. \tag{6.11}$$

μ is chosen to be 0.01, whereas the constants a and c were randomly generated in the range (1, 5) using Matlab's `randi` command that produces integers from a uniform distribution. Requests R_n and capacities $C_n, \forall n \in \mathcal{N}$ are also generated using Matlab's `randi` command in the range (1, 5). The total requests for each individual resource j at every SP is added and represented by $\Gamma_{n,j}$. Since we require two different groups $\mathcal{G}_1$ and $\mathcal{G}_2$, the capacities are generated in such a way that for SPs in $\mathcal{G}_1$, the capacity is chosen randomly from a

Table 6.2.: Simulation network settings for cooperative game-theoretic resource sharing framework with non-uniform priority strategy.

Setting	**Parameters**	$\mathcal{G}_1$	$\mathcal{G}_2$
1	$N = 3, M_n = 3, \forall n \in \mathcal{N}, J = 3$	{1}	{2, 3}
2	$N = 3, M_n = 20, \forall n \in \mathcal{N}, J = 3$	{1}	{2, 3}
3	$N = 6, M_n = 6, \forall n \in \mathcal{N}, J = 3$	{1, 2, 3}	{4, 5, 6}
4	$N = 6, M_n = 20, \forall n \in \mathcal{N}, J = 3$	{1, 2, 5}	{3, 4, 6}

uniform distribution in the range $[0, \Gamma_{n,j})$, whereas for SPs in $\mathcal{G}_2$, the capacity is randomly generated from a uniform distribution in the range $(\Gamma_{n,j}, 3\Gamma_{n,j}]$.

Simulations were run in `Matlab R2019b` on a `Core-i7` processor with `16 GB RAM`. To solve the optimization problems, we used the `OPTI-toolbox` [106]. We primarily used `IPOPT` (version 3.12.7) and `fmincon` (version R2016a) solvers from `OPTI-toolbox`. In certain instances where `IPOPT` and `fmincon` did not converge to the optimal solution, we used `filtersd` (version 1.0) from `OPTI-toolbox`. Below, we provide detailed experimental results for both algorithms.

6.4.1. Results for GPOA Algorithm

Verification of game-theoretic properties

In Table 6.3, we present results for a 3-player 20-application game that verify different game theoretic properties, such as individually rationality, group rationality, super additivity and show that the obtained allocation lies in the core. Player 1 has a resource deficit while players 2 and 3 have resource surpluses. The payoffs all players receive in the grand coalition are at least as good as they would receive working alone. This shows that the solution obtained using GPOA for the grand coalition is individually rational. Similarly, the value of coalition is the sum of payoff all players receive, hence our solution is group rational. Furthermore, as the coalition size increases, coalition value increases. Hence, the grand coalition has the largest value, which shows the super additive nature of the game. Also, no set of players has any incentive to divert from the grand coalition and form a smaller coalition. Hence, the grand coalition is stable and the allocation we obtain using GPOA lies in the core. There are two different results for the grand coalition, i.e., GC^1 or $\{1, 2, 3\}$ and GC^2 or $\{1, 3, 2\}$ which shows that changing the order of `for-loop` in Step 5 of GPOA given in Algorithm 6.1 only changes the utility achieved by players in $\mathcal{G}_2$. However, both allocations lie in the core. For $\{1, 2, 3\}$, player 2 precedes player 3 in Step 5 (of GPOA given in Algorithm 6.1) while player 3 precedes player 2 in Step 5 for the grand coalition in $\{1, 3, 2\}$. From the results, it is evident that having player 2 execute Step 5 in GPOA before player 3 is preferable to the order player 3 before player 2. However, it is impossible to know that in advance and we need to try all possible combinations.

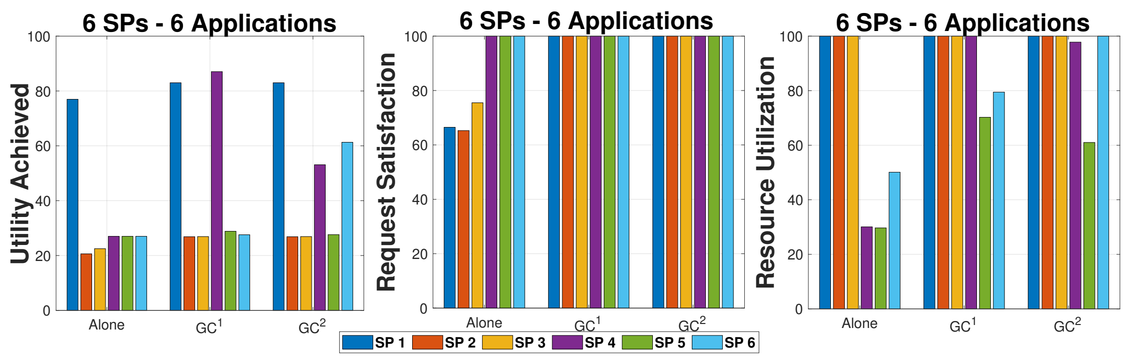

Figure 6.3.: Performance of our framework with GPOA for 6 player - 6 application settings.

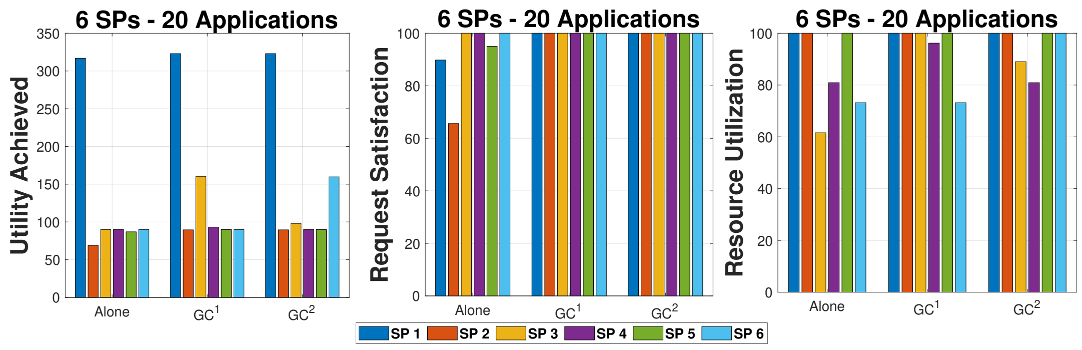

Figure 6.4.: Performance of our framework with GPOA for 6 player - 20 application settings.

Efficacy of the resource sharing framework

Figures 6.1, 6.2, 6.3 and 6.4 show the efficacy of our resource sharing framework using GPOA and compare it with a setting in which the SPs are working alone for different player-application settings given in Table 6.2. We evaluate the impact of the proposed framework on SP utility, request satisfaction, and resource utilization. It is evident that SPs with resource deficits are able to improve their application satisfaction and utilities (due to increase in application satisfaction) whereas SPs with resource surpluses improve their utilities by sharing their resources, which in turn increases resource utilization. For

Table 6.3.: Player payoff in different coalitions for a 3 player - 20 application game

Coalition	**Player 1**	**Player 2**	**Player 3**	**Value of coalition**
{1}	584.40	0.00	0.00	584.40
{2}	0	90	0	90
{3}	0	0	90	90
{1, 2}	588.72	160	0	748.72
{1, 3}	589.99	0	168.75	758.74
{2, 3}	0	90	90	180
{1, 2, 3}^{1}	590	160	108.75	858.75
{1, 3, 2}^{2}	590	90	168.75	848.75

Table 6.4.: Summary of average request satisfaction and resource utilization in different settings with and without the proposed GPOA Algorithm.

Setting	**Request Satisfaction (%)**		**Resource Utilization (%)**	
	Alone	**GC**	**Alone**	**GC**
1	75.71	100	57.98	88.75
2	96.88	100	74.63	82.06
3	84.54	100	68.30	93.13
4	91.73	100	85.94	94.98

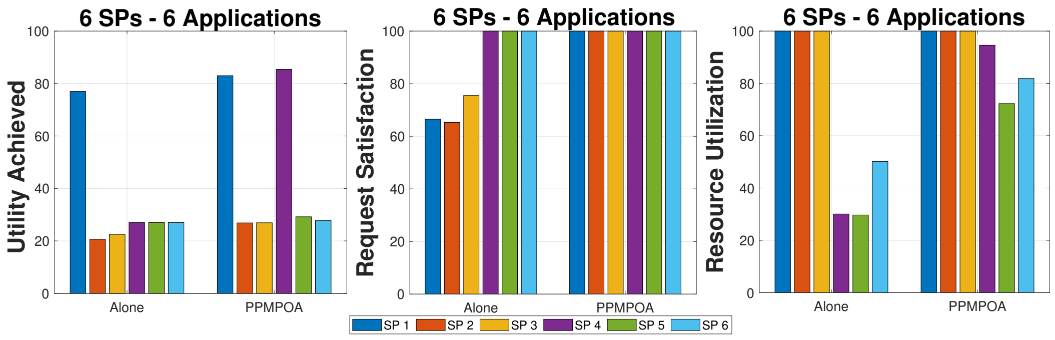

Figure 6.5.: Performance of our framework with PPMPOA for 6 player - 6 application settings.

GC^1 results in the 6 player-6 application setting, player 4 and 5 precede player 6 in Step 5 of Algorithm 6.1 whereas in GC^2, player 6 precedes players 4 and 5. For GC^1 results in the 6 player-20 application setting, players 3 and 4 precede player 6 whereas in GC^2, player 6 precedes players 3 and 4 in step 5 of Algorithm 6.1. The utility function distributions among players vary for GC^1 and GC^2, however, both allocations are in the core and are Pareto optimal.

Table 6.4 summarizes the average request satisfaction and resource utilization with GPOA algorithm and when the SPs are working alone. It is evident that the request satisfaction improves in all settings. Furthermore, resource utilization also increases.

6.4.2. Results for PPMPOA Algorithm

Figure 6.5 presents the results for PPMPOA Algorithm given in Algorithm 6.2. It is evident that Algorithm 6.2 improves the utilities and application satisfaction of different SPs (compared with SPs working alone). A comparison of Figures 6.3 and 6.5 indicates that the distribution of utility among SPs using GPOA and PPMPOA is different. However, both of them result in a Pareto optimal allocation. Furthermore, the value of a coalition achieved using PPMPOA will be either the same or lower, but cannot be larger than GPOA. This is because an SP with resource surplus that uses GPOA jointly considers all the resource deficit SPs when allocating resources whereas in PPMPOA, an SP with resource surplus considers a single resource deficit SP at a time. To highlight the decrease

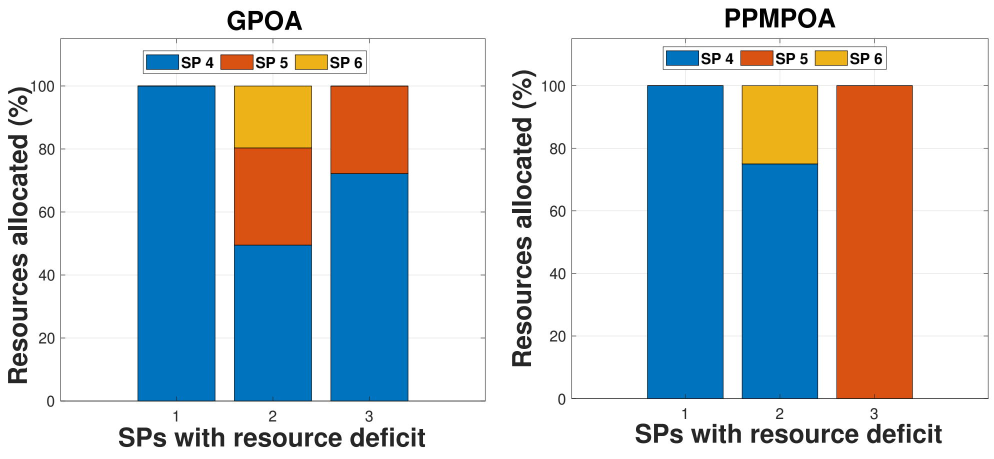

Figure 6.6.: SPs in $\mathcal{G}_2$ that provide resources to SPs in $\mathcal{G}_1$ using GPOA and PPMPOA algorithms in a 6 player - 6 application setting.

in number of SPs that provide resources to applications of SPs in $\mathcal{G}_1$ due to PPMPOA, we compare it with GPOA. Figure 6.6 shows the percentage of resources allocated to applications of different SPs in $\mathcal{G}_1$ by SPs in $\mathcal{G}_2$ using GPOA and PPMPOA.

Using GPOA for resource sharing in the 6 player-6 application setting (GC^1), applications of SP 1 are allocated resources at SP 4, whereas applications of SP 2 receive resources from SPs 4, 5 and 6. SP 3 receive resources from SPs 4, and 5. For the same setting and using PPMPOA algorithm, resources to applications of SP 2 are provided by SP 4 and 6. Resources to applications of SP 1 and 3 are provided by SP 4 and 5, respectively. This shows that PPMPOA reduces the number of SPs in $\mathcal{G}_2$ that provide resources to applications of SPs in $\mathcal{G}_1$ when compared with GPOA algorithm at the cost of solving large number of optimization problems.

6.4.3. Measure of Fairness

Figure 6.7 shows Jain's index for different settings given in Table 6.2 using GPOA algorithm for both GC^1 and GC^2. It is evident that GPOA achieves a high value of Jain's index in all settings, highlighting its fairness property. It is also worth mentioning that for the 6 player and 6 application setting, PPMPOA resulted in a Jain's index value of 0.75. These results show that the proposed GPOA and PPMPOA algorithms are also fair.

6.5. Summary and Conclusion

In this chapter, we extended our cooperative game-theoretic framework given in Chapter 5 by allowing the SPs to employ the non-uniform priority as a strategy. We showed that for monotonic, non-decreasing and non-negative utilities, resource sharing among SPs under the non-uniform priority strategy can be modeled as a cardinally convex canonical game. We proved that the core exists and proposed two efficient algorithms that provide allocation

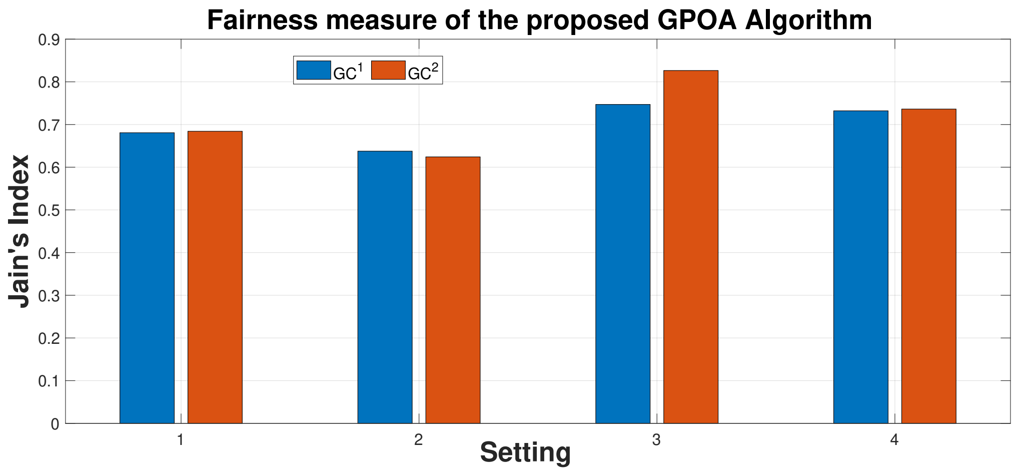

Figure 6.7.: Jain's index for different settings of Table 6.2 using GPOA algorithm.

from the core. Hence, the obtained solutions are Pareto optimal and the grand coalition of the service providers is stable. Furthermore, we reduced the number of SPs with resource surplus that provide resources to applications of resource deficit SPs using the matching based algorithm that also provides an allocation in the core. Experimental results show that utilities of all service providers and user satisfaction are improved, when compared with edge clouds working alone, using our framework.

7. Conclusions and Future Work

This thesis has addressed the following single and multi-objective problems for resource management in communication networks:

1. Managing caching, computational and communication resources to minimize network energy consumption while providing Quality of Information (QoI) guarantees.
2. Multi-objective resource sharing among different service providers.

We have started by considering the problem of minimizing energy consumption in a communication network in Chapter 3. Each node in the network has the ability to transmit, compress and store data. The goal is to find the optimal compression rate and caching location to minimize the overall energy consumed in the network subject to the QoI constraint. We have formulated the problem as a non-convex MINLP and showed that it is NP-hard. We then propose a variant of Spatial Branch-and-Bound Algorithm that provides an ϵ-global optimal solution. Through simulations, we show that jointly considering communication, compression and caching can improve energy efficiency by about 88% when compared with only considering communication and compression or communication and caching.

In Chapter 4, we propose a multi-objective framework based on Nash Bargaining Solution (NBS) to enable different service providers with varying objectives or utilities to share resources. Our proposed NBS based framework provides Pareto optimal allocation of resources and is fair to all the service providers. Furthermore, we have also shown that strong duality holds for a particular class of utility functions and proposed a distributed algorithm to obtain NBS. Through simulation using synthetic data and trace files, we showed that the proposed framework improves the utility of service providers, increases resource utilization and average application satisfaction.

Since the NBS-based resource-sharing framework requires a concave utility function, we have developed a cooperative game-theoretic framework in Chapter 5 for resource sharing that only requires a monotonically non-decreasing utility function. The service providers employ the uniform priority strategy where native and non-native applications are given equal priority in receiving resources. We have shown that the resource sharing among service providers can be modeled as a canonical cooperative game with non-transferable utility. We show that the game is convex, hence the core is non-empty. To address the problem of obtaining a resource allocation from the core, we have proposed the centralized *GPUS* algorithm. Through simulations, we showed that the proposed algorithm provides allocations from the core. Furthermore, it has been shown that the proposed resource sharing framework improves the utility for service providers, application satisfaction and increases resource utilization.

In Chapter 6, we have further extended our game-theoretic framework to cases where the service providers employ a strategy in which they prioritize their native applications over non-native applications in resource allocation. We have shown that even with the non-uniform priority strategy, the game is canonical and convex with a non-empty core. To address the problem of getting a resource allocation from the core, we have proposed two algorithms, Game-theoretic Pareto optimal allocation (GPOA) and Polyandrous-Polygamous Matching based Pareto Optimal Allocation (PPMPOA). Our simulation results confirm that both algorithms can provide allocations from the core. Furthermore, the proposed algorithms improve utility of the service providers, application satisfaction and increase resource utilization. PPMPOA further reduces the number of service providers from which a resource deficit provider can borrow resources in order to satisfy its applications, thus avoiding resource fragmentation when possible. However, PPMPOA requires solving a larger number of optimization problems when compared with GPOA.

In terms of future work, several research directions can be explored as follows.

- **Reducing the complexity of symbolic reformulation:** Symbolic reformulation reduces the problem complexity for non-convex MINLPs by removing the non-convex terms from the objective function. However, the addition of auxiliary variables due to symbolic reformulation is problematic and makes it suitable only for small size problems [87]. Therefore, there is a need to modify and enhance the symbolic reformulation rules to reduce the number of auxiliary variables.

- **Single-Agent reinforcement learning for minimizing network energy consumption:** Due to the recent advances in deep learning and reinforcement learning, it will be interesting to train an agent for the energy minimization problem tackled in Chapter 3. Using existing reinforcement learning algorithms and fine tuning them to the energy minimization problem can help develop a caching and data compression policy that will choose a suitable compression rate and caching location for different network settings.

- **Multi-Agent reinforcement learning (MARL) for resource sharing:** To automate the resource sharing process, *MARL with partial or no observations* [113] can be explored as a viable alternative. While MARL may not provide any performance guarantee, it can help in obtaining a resource sharing policy for service providers. Each provider can have an agent that has access to the capacity as well as resource request information at the SP. An agent affiliated with each provider can learn a resource sharing policy in a distributed manner that will then eliminate the need for solving optimization problems every single time when new requests arrive or the capacities of service providers change. To the best of our knowledge, there is no existing fully distributed MARL algorithm in which the agents have no access to the observations of other agents throughout training and testing.

Bibliography

[1] E. M. Smith and C. C. Pantelides, "Global Optimisation of General Process Models," in *Glo. Opt. Eng. Des.* Springer, 1996, pp. 355–386.

[2] S. Nazemi Gelyan, "Distributed Optimisation Framework for In-Network Data Processing," Ph.D. dissertation, Imperial College London, 2016.

[3] S. Nazemi, K. K. Leung, and A. Swami, "QoI-aware Tradeoff Between Communication and Computation in Wireless Ad-hoc Networks," in *Proc. IEEE PIMRC*, 2016.

[4] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, "In-Network Caching Effect on Optimal Energy Consumption in Content-Centric Networking," in *Proc. IEEE ICC*, 2012.

[5] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, "Optimal Energy Tradeoff among Communication, Computation and Caching with QoI-Guarantee," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 151–162, 2019.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[7] A. El Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi, "Energy-Efficient Scheduling of Packet Transmissions over Wireless Networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1773–1782.

[8] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[9] T. E. Klein and P. Richard, "ICT Energy Challenges, Impact and Solutions," in *19th International ICIN Conference-Innovations in Clouds, Internet and Networks*, 2016.

[10] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[11] https://www.theregister.co.uk/2013/08/16/it_electricity_use_worse_than_you_thought/.

[12] Nokia Bell Labs, https://www.bell-labs.com/var/articles/gwtt/.

[13] S. Taherizadeh and V. Stankovski, "Auto-scaling Applications in Edge Computing: Taxonomy and Challenges," in *Proceedings of the International Conference on Big Data and Internet of Thing*, 2017, pp. 158–163.

[14] N. Wang, M. Matthaiou, D. S. Nikolopoulos, and B. Varghese, "DYVERSE: DYnamic VERtical Scaling in Multi-Tenant Edge Environments," *arXiv preprint arXiv:1810.04608*, 2018.

[15] "ETSI ISG on Multi-access Edge Computing (mec)," http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing, [Online; accessed 05-May-2020].

[16] "Open Edge Computing," http://openedgecomputing.org/, [Online; accessed 05-May-2020].

[17] "OpenFog Consortium," https://www.openfogconsortium.org/, [Online; accessed 05-May-2020].

[18] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It?s Hard to Share: Joint Service Placement and Request Scheduling in Edge Clouds with Sharable and Non-sharable Resources," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 365–375.

[19] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, "A Game-Theoretic Approach to Multi-Objective Resource Sharing and Allocation in Mobile Edge," in *Proceedings of the 2018 on Technologies for the Wireless Edge Workshop*. ACM, 2018, pp. 9–13.

[20] H. Yaïche, R. R. Mazumdar, and C. Rosenberg, "A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 5, pp. 667–678, 2000.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[22] G. Tychogiorgos, A. Gkelias, and K. K. Leung, "A Non-Convex Distributed Optimization Framework and its Application to Wireless Ad-hoc Networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4286–4296, 2013.

[23] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific Belmont, MA, 1998.

[24] J.-H. Cho, Y. Wang, R. Chen, K. S. Chan, and A. Swami, "A Survey on Modeling and Optimizing Multi-Objective Systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1867–1901, 2017.

[25] Z. Han, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications.* Cambridge University Press, 2012.

[26] R. B. Myerson, *Game Theory.* Harvard University Press, 2013.

[27] M. Maschler, E. Solan, and S. Zamir, *Game theory.* Cambridge University Press, 2013.

[28] J. F. Nash Jr, "The Bargaining Problem," *Econometrica: Journal of the Econometric Society*, pp. 155–162, 1950.

[29] J. C. Harsanyi, "A Simplified Bargaining Model for the N-Person Cooperative Game," *International Economic Review*, vol. 4, no. 2, pp. 194–220, 1963.

[30] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *System sciences*, 2000.

[31] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. of IEEE INFOCOM*, 2002.

[32] A. Manjeshwar and D. P. Agrawal, "TEEN: a Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks," in *IPDPS*, 2001.

[33] M. Ye, C. Li, G. Chen, and J. Wu, "EECS: an Energy Efficient Clustering Scheme in Wireless Sensor Networks," in *Proc. of IEEE IPCCC*, 2005.

[34] A. Anandkumar, J. E. Yukich, L. Tong, and A. Swami, "Energy Scaling Laws for Distributed Inference in Random Fusion Networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 7, 2009.

[35] S. Eswaran, J. Edwards, A. Misra, and T. F. L. Porta, "Adaptive In-Network Processing for Bandwidth and Energy Constrained Mission-Oriented Multihop Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 9, pp. 1484–1498, Sept 2012.

[36] C. Zhang, J. Kurose, Y. Liu, D. Towsley, and M. Zink, "A Distributed Algorithm for Joint Sensing and Routing in Wireless Networks with Non-steerable Directional Antennas," in *Proc. of IEEE ICNP*, 2006.

[37] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Data Gathering with Tunable Compression in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 2, pp. 276–287, 2008.

[38] O. Boykin, S. Ritchie, I. O'Connell, and J. Lin, "Summingbird: A Framework for Integrating Batch and Online Mapreduce Computations," *Proc. of VLDB*, 2014.

[39] R. Rajagopalan and P. K. Varshney, "Data Aggregation Techniques in Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 48?–63, 2006.

[40] M. Garetto, E. Leonardi, and V. Martina, "A Unified Approach to the Performance Analysis of Caching Systems," *ACM TOMPECS*, vol. 1, no. 3, p. 12, 2016.

[41] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of TTL-based Cache Networks," in *Proc. of VALUETOOLS*, 2012.

[42] J. Li, S. Shakkottai, J. C. S. Lui, and V. Subramanian, "Accurate Learning or Fast Mixing? Dynamic Adaptability of Caching Algorithms," *IEEE Journal on Selected Areas in Communications*, 2018.

[43] S. Ioannidis and E. Yeh, "Adaptive Caching Networks with Optimality Guarantees," in *Proc. of ACM SIGMETRICS*, 2016.

[44] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. Tay, "A Utility Optimization Approach to Network Cache Design," in *Proc. of IEEE INFOCOM*, 2016.

[45] N. K. Panigrahy, J. Li, and D. Towsley, "Hit Rate vs. Hit Probability Based Cache Utility Maximization," in *Proc. of ACM MAMA*, 2017.

[46] N. K. Panigrahy, J. Li, F. Zafari, D. Towsley, and P. Yu, "Optimizing Timer-based Policies for General Cache Networks," *Arxiv preprint arXiv:1711.03941*, 2017.

[47] N. Dimokas, D. Katsaros, and Y. Manolopoulos, "Cooperative Caching in Wireless Multimedia Sensor Networks," *Mobile Networks and Applications*, vol. 13, no. 3-4, pp. 337–356, 2008.

[48] N. Dimokas, D. Katsaros, L. Tassiulas, and Y. Manolopoulos, "High Performance, Low Complexity Cooperative Caching for Wireless Sensor Networks," *Wireless Networks*, vol. 17, no. 3, pp. 717–737, 2011.

[49] M. Alipio, N. M. Tiglao, A. Grilo, F. Bokhari, U. Chaudhry, and S. Qureshi, "Cache-based Transport Protocols in Wireless Sensor Networks: A Survey and Future Directions," *Journal of Network and Computer Applications*, 2017.

[50] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation Offloading and Resource Allocation in Wireless Cellular Networks with Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[51] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[52] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-Aware Resource Allocation for Edge Computing," in *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2017, pp. 47–54.

[53] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing Resource Allocation in Three-Tier IoT Fog Networks: A Joint Optimization Approach Combining Stackelberg Game and Matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[54] Q. Xu, Z. Su, Q. Zheng, M. Luo, B. Dong, and K. Zhang, "Game Theoretical Secure Caching Scheme in Multi-Homing Edge Computing-enabled Heterogeneous Networks," *IEEE Internet of Things Journal*, 2018.

[55] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive Mechanism for Computation Offloading using Edge Computing: A Stackelberg Game Approach," *Computer Networks*, vol. 129, pp. 399–409, 2017.

[56] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social Welfare Maximization Auction in Edge Computing Resource Allocation for Mobile Blockchain," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[57] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic Resource Allocation Exploiting Mobility Prediction in Mobile Edge Computing," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2016, pp. 1–6.

[58] M. Alicherry and T. Lakshman, "Network Aware Resource Allocation in Distributed Clouds," in *Infocom, 2012 proceedings IEEE*. IEEE, 2012, pp. 963–971.

[59] D. Ergu, G. Kou, Y. Peng, Y. Shi, and Y. Shi, "The Analytic Hierarchy Process: Task Scheduling and Resource Allocation in Cloud Computing Environment," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 835–848, 2013.

[60] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized Task Scheduling and Resource Allocation on Cloud Computing Environment using Improved Differential Evolution Algorithm," *Computers & Operations Research*, vol. 40, no. 12, pp. 3045–3055, 2013.

[61] C. S. Pawar and R. B. Wagh, "Priority Based Dynamic Resource Allocation in Cloud Computing with Modified Waiting Queue," in *Intelligent Systems and Signal Processing (ISSP), 2013 International Conference on*. IEEE, 2013, pp. 311–316.

[62] Z. Xiao, W. Song, Q. Chen *et al.*, "Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2012.

[63] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-Oriented Heterogeneous Resource Sharing for Optimizing Service Latency in Mobile Cloud," in *Proceedings of the First International Workshop on Mobile Cloud Computing & Networking.* ACM, 2013, pp. 19–26.

[64] H. Xu and B. Li, "A General and Practical Datacenter Selection Framework for Cloud Services," in *2012 IEEE Fifth International Conference on Cloud Computing.* IEEE, 2012, pp. 9–16.

[65] M. M. Hassan and A. Alamri, "Virtual Machine Resource Allocation for Multimedia Cloud: A Nash Bargaining Approach," *Procedia Computer Science*, vol. 34, pp. 571–576, 2014.

[66] J. He, D. Wu, Y. Zeng, X. Hei, and Y. Wen, "Toward Optimal Deployment of Cloud-Assisted Video Distribution Services," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 10, pp. 1717–1728, 2013.

[67] Y. Feng, B. Li, and B. Li, "Bargaining Towards Maximized Resource Utilization in Video Streaming Datacenters," in *INFOCOM, 2012 Proceedings IEEE.* IEEE, 2012, pp. 1134–1142.

[68] J. Guo, F. Liu, D. Zeng, J. C. Lui, and H. Jin, "A Cooperative Game Based Allocation for Sharing Data Center Networks," in *INFOCOM, 2013 Proceedings IEEE.* IEEE, 2013, pp. 2139–2147.

[69] R. Duan, R. Prodan, and X. Li, "Multi-Objective Game Theoretic Scheduling of Bag-of-Tasks Workflows on Hybrid Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 29–42, 2014.

[70] W. Shi, C. Wu, and Z. Li, "A Shapley-Value Mechanism for Bandwidth on Demand between Datacenters," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 19–32, 2018.

[71] ——, "An Online Auction Mechanism for Dynamic Virtual Cluster Provisioning in Geo-Distributed Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 677–688, 2017.

[72] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1.* International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 981–988.

[73] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A Game-Theoretic Method of Fair Resource Allocation for Cloud Computing Services," *The journal of supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.

[74] K. C. Barr and K. Asanović, "Energy-Aware Lossless Data Compression," *ACM Transactions on Computer Systems*, 2006.

[75] S. A. Ehikioya, "A Characterization of Information Quality Using Fuzzy Logic," in *NAFIPS*, 1999.

[76] P. Bonami *et al.*, "An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs," *Disc. Opt.*, vol. 5, no. 2, pp. 186–204, 2008.

[77] S. Le Digabel, "Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm," *ACM TOMS*, vol. 37, no. 4, p. 44, 2011.

[78] M. Tawarmalani and N. V. Sahinidis, "A Polyhedral Branch-and-Cut Approach to Global Optimization," *Mathematical Programming*, vol. 103, no. 2, pp. 225–249, 2005.

[79] T. Achterberg, "SCIP: Solving Constraint Integer Programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[80] R. Misener and C. A. Floudas, "ANTIGONE: Algorithms for Continuous/Integer Global Optimization of Nonlinear Equations," *Journal of Global Optimization*, vol. 59, no. 2-3, pp. 503–526, 2014.

[81] C. Fan, T. Zhang, Z. Zeng, and Y. Chen, "Energy Efficiency Analysis of Cache-Enabled Cellular Networks with Limited Backhaul," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[82] K. Guan, G. Atkinson, D. C. Kilper, and E. Gulsen, "On the Energy Efficiency of Content Delivery Architectures," in *2011 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2011, pp. 1–6.

[83] E. M. Smith and C. C. Pantelides, "A Symbolic Reformulation/Spatial Branch-and-Bound Algorithm for the Global Optimisation of Nonconvex MINLPs," *Comp. & Chem. Eng.*, vol. 23, no. 4, pp. 457–478, 1999.

[84] E. M. Smith, "On the Optimal Design of Continuous Processes," Ph.D. dissertation, Imperial College London (University of London), 1996.

[85] L. Liberti, "Reformulation and Convex Relaxation Techniques for Global Optimization," *4OR: A Quarterly Journal of Operations Research*, vol. 2, no. 3, pp. 255–258, 2004.

[86] ——, "Reformulation and Convex Relaxation Techniques for Global Optimization," Ph.D. dissertation, Imperial College London, 2004.

[87] C. A. Floudas, *Deterministic Global Optimization: Theory, Methods and Applications.* Springer Science & Business Media, 2013, vol. 37.

[88] G. P. McCormick, "Computability of Global Solutions to Factorable Nonconvex Programs: Part I?Convex Underestimating Problems," *Mathematical Programming*, vol. 10, no. 1, pp. 147–175, 1976.

[89] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[90] OPTI Toolbox, "A Free Matlab Toolbox for Optimization," https://www.inverseproblem.co.nz/OPTI/index.php/Main/HomePage, [Online; accessed 28-Jun-2017].

[91] A. Fiat and P. Sanders, "Algorithms-esa 2009," *Lecture Notes in Computer Science*, vol. 5757, 2009.

[92] P. Bonami and J. Lee, "BONMIN Users' Manual," https://projects.coin-or.org/Bonmin/browser/stable/1.5/Bonmin/doc/BONMIN_UsersManual.pdf?format=raw, 2011.

[93] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1. ACM, 2014, pp. 71–83.

[94] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[95] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[96] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[97] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[98] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When Edge Meets Learning: Adaptive Control for Resource-Constrained Distributed Machine Learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 63–71.

[99] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "A Dual Approach for Optimal Algorithms in Distributed Optimization Over Networks," *Optimization Methods and Software*, pp. 1–40, 2020.

[100] I. Notarnicola and G. Notarstefano, "A Duality-Based Approach for Distributed Optimization with Coupling Constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 326–14 331, 2017.

[101] J. Lei, H.-F. Chen, and H.-T. Fang, "Primal-Dual Algorithm for Distributed Constrained Optimization," *Systems & Control Letters*, vol. 96, pp. 110–117, 2016.

[102] J. Wilde, "Constrained Optimization," http://www.columbia.edu/~md3405/Constrained_Optimization.pdf, 2013, [Online; accessed 12-July-2018].

[103] S. Shen, V. van Beek, and A. Iosup, "Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* IEEE, 2015, pp. 465–474.

[104] A. Kohne, M. Spohr, L. Nagel, and O. Spinczyk, "FederatedCloudSim: A SLA-Aware Federated Cloud Simulation Framework," in *Proceedings of the 2nd International Workshop on CrossCloud Systems.* ACM, 2014, p. 3.

[105] A. Kohne, D. Pasternak, L. Nagel, and O. Spinczyk, "Evaluation of SLA-based Decision Strategies for VM Scheduling in Cloud Data Centers," in *Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms.* ACM, 2016, p. 6.

[106] J. Currie, D. I. Wilson, N. Sahinidis, and J. Pinto, "OPTI: Lowering the Barrier between Open Source Optimizers and the Industrial MATLAB User," *Foundations of computer-aided process operations*, vol. 24, p. 32, 2012.

[107] R. Jain, A. Durresi, and G. Babic, "Throughput Fairness Index: An Explanation," in *ATM Forum contribution*, vol. 99, no. 45, 1999.

[108] R. Jain, D.-M. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems," Digital Equipment Corporation," Technical Report DEC-TR-301, Tech. Rep., 1984.

[109] W. W. Sharkey, "Convex Games without Side Payments," *International Journal of Game Theory*, vol. 10, no. 2, pp. 101–106, 1981.

[110] ——, "Cooperative Games with Large Cores," *International Journal of Game Theory*, vol. 11, no. 3-4, pp. 175–182, 1982.

[111] M. Baıou and M. Balinski, "Many-to-Many Matching: Stable Polyandrous Polygamy (or Polygamous Polyandry)," *Discrete Applied Mathematics*, vol. 101, no. 1-3, pp. 1–12, 2000.

[112] D. Gale and L. S. Shapley, "College Admissions and the Stability of Marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[113] L. Bu, R. Babu, B. De Schutter *et al.*, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

A. Supplementary Content for Chapter 3

A.1. Symbolic Reformulation

The first step of the symbolic reformulation is to represent the algebraic expression (objective function and constraints) using a binary tree as shown in Figure A.1. Symbolic reformulation transforms the algebraic expression represented as a binary tree into a set of linear constraints that might involve some newly introduced variables. Since our optimization problem (3.5) contains bilinear and linear fractional terms, the newly introduced auxiliary variables are therefore either products or ratios of other variables i.e. $w_i \equiv w_j w_k$ and $w_i \equiv \frac{w_j}{w_k}$. The rules for efficiently[1] achieving such transformation are presented in [1], part of which we restate in the Table A.1. We create a binary tree for representing the algebraic expressions and assign the leaf nodes a class that can be either a constant (C), an expression (X), or a variable (V). If we are at some intermediate node that represents a multiplication operation, and both its right and left child nodes are of class expression (X), then the reformulation would require us to introduce two linear constraints (for both right and left node explained) as well as introduce new bilinear auxiliary variable.

$$
\begin{aligned}
\min_{\boldsymbol{w}} \quad & \boldsymbol{w}_f \\
\text{s.t.} \quad & \mathbf{A}\mathbf{w} = \mathbf{b}, \\
& \mathbf{w}^l \leq \mathbf{w} \leq \mathbf{w}^U, \\
& \mathbf{w}_k \equiv \mathbf{w}_i \mathbf{w}_j, \ \forall (i,j,k) \ \in \ \mathcal{T}_{\text{bt}}, \\
& \mathbf{w}_k \equiv \mathbf{w}_i / \mathbf{w}_j, \ \forall (i,j,k) \ \in \ \mathcal{T}_{\text{lft}}.
\end{aligned} \tag{A.1}
$$

All the linear constraints are added into the constraint $Aw = b$ in (A.1) while the variables introduced are added into the vector $\mathbf{w}$ and depending on its type (either bilinear or linear fractional) its definition is added into either $\mathcal{T}_{\text{bt}}$ or $\mathcal{T}_{\text{lft}}$. After such reformulation, we obtain (A.1). The new variable vector $\mathbf{w}$ consists of continuous and discrete variables in the original MINLP, as well as other auxiliary variables introduced as a result of reformulation. The objective function w_f is a single auxiliary variable. This reformulation ensures that the new objective function and first constraint in (A.1) are linear, and all non-convexities and non-linearities in the original MINLP are absorbed by the sets $\mathcal{T}_{bt}$ and $\mathcal{T}_{lft}$.

[1]Keeping the number of newly introduced variables to minimum

Table A.1.: Symbolic Reformulation Rules defined in [1] where X stands for expression, C stands for Constant and V stands for variable

Left Subtree Class	**Right Subtree Class**	**Binary Operator**	**New Variable Definition**	**New Linear Constraint**	**Binary Tree Class**
C	C	$\pm$			C
		$\times$			C
		$\div$			C
V	C	$\pm$			X
		$\times$			X
		$\div$			X
X	C	$\pm$			X
		$\times$			X
		$\div$			X
C	V	$\pm$			X
		$\times$			X
		$\div$	Linear Fractional		V
V	V	$\pm$			X
		$\times$	Bilinear		V
		$\div$	Linear Fractional		V
X	V	$\pm$			X
		$\times$	Bilinear	Left	X
		$\div$	Linear Fractional	Left	X
C	X	$\pm$			X
		$\times$			X
		$\div$	Linear Fractional	Right	V
V	X	$\pm$			X
		$\times$	Bilinear	Right	V
		$\div$	Linear Fractional	Right	V
X	X	$\pm$			X
		$\times$	Bilinear	Left, Right	V
		$\div$	Linear Fractional	Left, Right	V

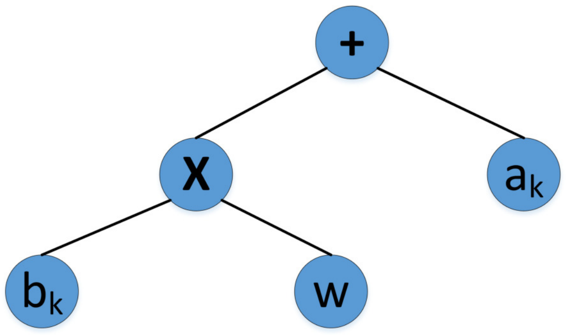

Figure A.1.: Binary Tree representation for algebraic expression $b_k w + a_k$

A.1.1. Linear Constraint and Variable Creation

As seen in Table A.1, certain arithmetic operations during the symbolic reformulation require creation of new linear constraints and introduction of new variables. This can be easily explained by an example. Let the parent node (any intermediate node that has 2 child nodes) represent a multiplication operation, the left subtree (child) be abc (expression where a,b are constants and c is a variable) and the right subtree be d (variable), then using the rules in the Table A.1, we need to introduce a new linear constraint (for the left subtree) and then a bilinear variable. So the linear constraint would be $abc - w(i) = 0$ where $w(i)$ is the i^{th} auxiliary variable introduced. The bilinear variable that has to be introduced would be $w(i+1) \equiv w(i)d$ where d is the variable in the original right subtree. The linear constraint will become part of of $Aw = b$ in (A.1) while $w(i+1)$ will be part of the set of binary terms $\mathcal{T}_{\text{bt}}$. If the intermediate node was a division operation, we introduce the linear constraint just like we did for the multiplication operation, however we follow that by adding a linear fractional term $w(i+1) \equiv \frac{w(i)}{d}$. This is a recursive process and is repeated until all the terms in our objective function as well constraints are reformulated. Note that symbolic reformulation does not affect the linear terms in the original problem (3.5). Using this process, we reformulate (3.5) into (A.2) that can then be used with V-SBB.

A.2. Standard Form of the Optimization Problem in (3.5)

$$
\begin{aligned}
\min_{w} \quad & w_f \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}} y_k \overline{w}^{C_1}_{k,j} \geq \gamma, \\
& \sum_{k\in C_v} y_k \overline{w}^{C_2}_{k,i} \leq S_v, \forall\, v\in V, \\
& \sum_{i=0}^{h(k)} b_{k,i} \leq 1, \forall k\in\mathcal{K}, \\
& b_{k,i} \in \{0,1\}, \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \mathbf{w}^l \leq \mathbf{w} \leq \mathbf{w}^U,\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& w^b_{k,i} = \delta_{k,i}\times \overline{w}_{k,a},\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& w^f_{k,i} = \frac{\overline{w}_{k,a}}{\delta_{k,i}},\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& y_k b_{k,i} - \overline{w}^{C_2''}_{k,i} = 0,\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \overline{w}^{C_2}_{k,i} = \overline{w}^{C_2''}_{k,i} \times \overline{w}^{C_2'}_{k,\beta},\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \sum_{j=0}^{i-1} b_{k,j} - \widetilde{w}_{k,i} = 0,\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \overline{\overline{w}}_{k,i} = \overline{w}_{k,a}\times \widetilde{w}_{k,i},\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \overline{w}^b_{k,i} = w^b_{k,i}\times \widetilde{w}_{k,i},\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \overline{w}^f_{k,i} = w^f_{k,i}\times \widetilde{w}_{k,i},\ \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\
& \prod_{m=i+1}^{h(k)} \delta_{k,m} = \overline{w}_{k,\underbrace{h(k)-2-i}_{a}} = \begin{cases} \delta_{k,h(k)}\times\delta_{k,h(k)-1}, & \forall a=0 \\ \overline{w}_{k,a-1}\times\delta_{k,m}, & m+a=h(k)-1 \\ \delta_{k,h(k)}, & \forall i=h(k)-1, \end{cases} \\
& \prod_{i=0}^{h(k)} \delta_{k,i} = \overline{w}^{C_1}_{k,\underbrace{h(k)-1}_{j}} = \begin{cases} \delta_{k,h(k)}\times\delta_{k,h(k)-1}, & \forall j=0 \\ \overline{w}^{C_1}_{k,j-1}\times\delta_{k,j+1}, & \forall j=1\cdots h(k)-1 \\ \delta_{k,h(k)}, & \forall i=h(k), \end{cases} \\
& \prod_{j=h(k)}^{h(v)} \delta_{k,j} = \overline{w}^{C_2'}_{k,\underbrace{\tau-2}_{\beta}} = \begin{cases} \delta_{k,h(k)}\times\delta_{k,h(k)-1}, & \forall \beta=0 \\ \overline{w}^{C_1}_{k,j-1}\times\delta_{k,j+1}, & \forall \beta>0 \\ \delta_{k,h(k)}, & \forall \beta<0, \end{cases} \\
& w_f = \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k \left(\left(\varepsilon_{kR}\overline{w}_{k,a} + \varepsilon_{kT} w^b_{k,i} + \varepsilon_{kC} w^f_{k,i} - \varepsilon_{kc}\overline{w}_{ka}\right) + A + B\right), \\
& A = \varepsilon_{kR}R_k\overline{w}_{k,a} + \varepsilon_{kT}w^b_{k,i} + \varepsilon_{kc}R_k w^f_{k,i} - \varepsilon_{kC}R_k\overline{w}_{k,a} - \varepsilon_{kR}\overline{w}_{k,a} - \varepsilon_{kT}w^b_{k,i} - \varepsilon_{kC}w^f_{k,i} + \varepsilon_{kC}\overline{w}_{k,a}, \\
& B = -\varepsilon_{kR}R_k\overline{\overline{w}}_{k,i} - \varepsilon_{kT}R_k\overline{w}^b_{k,i} - \varepsilon_{kc}R_k\overline{w}^f_{k,i} + \varepsilon_{kC}R_k\overline{\overline{w}}_{k,i} + \varepsilon_{kR}\overline{\overline{w}}_{k,i} + \varepsilon_{kT}\overline{w}^b_{k,i} + \varepsilon_{kC}\overline{w}^f_{k,i} - \varepsilon_{kC}\overline{\overline{w}}_{k,i}.
\end{aligned}
\tag{A.2}
$$

A.3. Convergence Proof for the Spatial Branch and Bound Algorithm

For completeness, we present the following proofs for the convergence of spatial branch-and-bound [86], which work for our V-SBB.

Definition A.1. *Let $\Omega \subseteq \mathbb{R}^n$. A finite family of sets $\mathcal{S}$ is a net for Ω if it is pairwise disjoint and it covers Ω.*

Definition A.2. *A net $\mathcal{S}'$ is a refinement of the net $\mathcal{S}$ if there are finitely many pairwise disjoint $s'_i \in \mathcal{S}'$ such that $s = \bigcup_i s'_i \in \mathcal{S}$ and $s \notin S$.*

Definition A.3. *Let $\mathcal{M}_n$ be an infinite sequence of subsets of x such that $\mathcal{M}_i \in \mathcal{S}_i$. $\mathcal{M}_n$ is a* filter *for $\mathcal{S}_n$ if $\forall i \in \mathbb{N}\ \mathcal{M}_i \subseteq \mathcal{M}_{i-1}$ where $M_\infty = \bigcap_{i \in N} M_i$ be the limit of the filter.*

Definition A.4. *Let $x \subseteq \mathbb{R}^n$ and $f(x)$ be the objective function of an MINLP problem then a spatial branch-and-bound algorithm would be convergent if $\gamma^* = \inf\ f(x) = \lim\limits_{k \to \infty} \gamma_k$.*

Definition A.5. *A selection rule is* exact *if*

1. *The infinimum objective function value of any region that remains qualified during the whole solution process is greater than or equal to the globally optimal objective function value, i.e.,*
$$\forall M \in \bigcap_{k=1}^{\infty} \mathcal{R}_k(\inf\ f(x \cap M) \geq \gamma^*)$$
2. *The limit $\mathbb{M}_\infty$ of any filter M_k is such that* $\inf f(\Omega \cap M) \geq \gamma^*$ *where Ω is the feasible set.*

Theorem A.1. *A Spatial branch-and-bound algorithm using an exact selection rule converges.*

Proof. Proof by contradiction:

Let there be $x \in \Omega$ with $f(x) < \gamma^*$. Let $x \in M$ with $M \in \mathbb{R}_n$ for some $n \in \mathbb{N}$. Because of the first condition of exactness of selection rule, the filter M cannot remain qualified forever. Furthermore, unqualified regions may not, by hypothesis, include points with better objective function values than the current incumbent γ_k. Hence M must necessarily be split at some iteration $n' > n$ so x belongs to every $\mathcal{M}_n$ in some filter $\{\mathcal{M}_n\}$, thus $x \in \Omega \cap \mathbb{M}_\infty$. By condition 2 of exactness of selection rule, $f(x) \geq f(\Omega \cap M_\infty) \geq \gamma^*$. The result follows. □

A.4. McCormick Over and Under Estimators

A.4.1. Bilinear Terms

The McCormick linear over estimator and under estimator for bilinear terms with form $w_k \equiv w_i w_j$ are given by (A.3) and (A.4), respectively.

$$\begin{aligned} w_k &\leq w_i^l w_j + w_j^u w_i - w_i^l w_j^u, \\ w_k &\leq w_i^u w_j + w_j^l w_i - w_i^u w_j^l. \end{aligned} \tag{A.3}$$

$$\begin{aligned} w_k &\geq w_i^l w_j + w_j^l w_i - w_i^l w_j^l, \\ w_k &\geq w_i^u w_j + w_j^u w_i - w_i^u w_j^u. \end{aligned} \tag{A.4}$$

A.4.2. Linear Fractional Terms

The linear over estimator and under estimator for a linear fractional term with form $w_k \equiv \frac{w_i}{w_j}$ are similar to the over estimator and under estimator of bilinear terms given in (A.3) and (A.4). We first transform the linear fractional term into bilinear term, i.e., $w_i \equiv w_k w_j$ and then we can use (A.5) and (A.6) for the the linear over estimator and under estimator, respectively.

$$\begin{aligned} w_i &\leq w_k^l w_j + w_j^u w_k - w_k^l w_j^u, \\ w_i &\leq w_k^u w_j + w_j^l w_k - w_k^u w_j^l. \end{aligned} \tag{A.5}$$

$$\begin{aligned} w_i &\geq w_k^l w_j + w_j^l w_k - w_k^l w_j^l, \\ w_i &\geq w_k^u w_j + w_j^u w_k - w_k^u w_j^u. \end{aligned} \tag{A.6}$$

The advantage of such linear understimator and overestimator is that even if the original problem is a non-convex MINLP, the relaxed problem will be an MILP which is comparatively easy to solve.

B. Supplementary Content for Chapter 4

Theorem B.1 (Sufficient Condition [22]). *If the price based function* $\mathbf{X}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})$ *is continuous at one or more of the optimal Lagrange multipliers, the iterative algorithm consisting of* (4.10) *and* (4.11) *will converge to the global optimal solution.*

Theorem B.2 (Necessary Condition [22]). *The condition in Theorem B.1 is also necessary if at least one of the constraints in* (4.7) *is active (binding) at the optimal solution.*

Lemma B.1. *At the optimal solution, the optimal Lagrange multiplier vector corresponding to the capacity constraint is non-zero, i.e.,* $\boldsymbol{\alpha_j^*} > 0$.

Proof. At the optimal point, all resources are fully utilized, i.e., capacity constraints are active. From *complementary slackness* [21], we know that

$$\alpha_{n,j}^* \Big(\sum_{m \in \mathcal{M}} x_{n,j}^{*m} - C_{n,j} \Big) = 0, \forall n \in \mathcal{N}. \tag{B.1}$$

$\sum_{m \in \mathcal{M}} x_{n,j}^{*m} = C_{n,j}$, i.e., the constraint is active, which implies that $\boldsymbol{\alpha_j^*} > 0$. □

B.1. Proof of Theorem 4.2

Proof. We show that the price function obtained by $\frac{\partial \mathcal{L}(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}, \boldsymbol{\pi})}{\partial x_{n,j}^m} = 0$ is continuous at one or more of the optimal Lagrange multipliers for (4.8).

Let $u^{'-m}(.)$ and $f^{'-l}(.)$ represent the inverses of the first derivative of $\Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big)$ and $\big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)$, respectively. $\alpha_{n,j}$, β_j^m, ζ_n, $\gamma_{n,j}^m$ and $\pi_{n,j}^l$ are the *Lagrange* multipliers whereas:

$$\Delta_n = \sum_{m \in \mathcal{M}_n} \sum_{j \in \mathcal{J}} \Big(u^m(z_j^m + x_{n,j}^m) - u^m(z_j^m)\Big) + \Big(\sum_{l \in \overline{\mathcal{M}}} \sum_{j \in \mathcal{J}} \big(u^l(z_j^l + x_{n,j}^l) - u^l(z_j^l) - D^l(x_{n,j}^l)\big)\Big) - d_n^0. \tag{B.2}$$

We consider two different cases

$m \in \mathcal{M}_n$**:** $\frac{\partial \mathcal{L}}{\partial x^m_{n,j}} = 0$

$$\begin{aligned}
&u'^m(z^m_j + x^m_{n,j})(1+\Delta_n\zeta_n) - \Delta_n(\alpha_{n,j} + \beta^m_j - \gamma^m_{n,j}) = 0, \\
&\implies u'^m(z^m_j + x^m_{n,j}) = \frac{\Delta_n(\alpha_{n,j} + \beta^m_j - \gamma^m_{n,j})}{1+\Delta_n\zeta_n}, \\
&x^m_{n,j} = u'^{-m}\left(\frac{\Delta_n(\alpha_{n,j} + \beta^m_j - \gamma^m_{n,j})}{1+\Delta_n\zeta_n}\right) - z^m_j.
\end{aligned} \tag{B.3}$$

We prove that (B.3) is continuous at the optimal point by showing that the numerator $\Delta_n(\alpha_{n,j} + \beta^m_j - \gamma^m_{n,j})$ and denominator $1+\Delta_n\zeta_n$ are positive[1]. At the optimal point, the denominator $(1+\Delta^*_n\zeta^*_n)$ is positive as $\Delta^*_n > 0$ (from Section 4.2.3) and $\zeta^*_n = 0$ (from complementary slackness). Similarly, in the numerator, $(\alpha^*_{n,j} + \beta^{*m}_j - \gamma^{*m}_{n,j}) > 0$, as $\alpha^*_{n,j} > 0$ (from Lemma B.1) and $\alpha^*_{n,j} + \beta^{*m}_j > \gamma^{*m}_{n,j}$ (from *sensitivity analysis* [21]). Hence, (B.3) is continuous at the optimal point.

$l \in \{\mathcal{M}\backslash\mathcal{M}_n\}$**:** $\frac{\partial \mathcal{L}}{\partial x^l_{n,j}} = 0$

$$\begin{aligned}
&f'^l(z^l_j + x^l_{n,j})(1+\Delta_n\zeta_n + \Delta_n\pi^l_{n,j}) - \Delta_n(\alpha_{n,j} + \beta^l_j - \gamma^l_{n,j}) = 0, \\
&\implies f'^l(z^l_j + x^l_{n,j}) = \frac{\Delta_n(\alpha_{n,j} + \beta^l_j - \gamma^l_{n,j})}{1+\Delta_n\zeta_n + \Delta_n\pi^l_{n,j}}. \\
&x^l_{n,j} = f'^{-l}\left(\frac{\Delta_n(\alpha_{n,j} + \beta^l_j - \gamma^l_{n,j})}{1+\Delta_n\zeta_n + \Delta_n\pi^l_{n,j}}\right) - z^l_j.
\end{aligned} \tag{B.4}$$

The continuity at optimal point can be established using arguments similar to that for $m \in \mathcal{M}_n$ case. The proof of zero-duality gap follows from Theorem B.1. Hence, strong duality holds and our proposed distributed algorithm 4.2 converges to the NBS. □

[1]Positive numerator and denominator are required if u'^{-m} is log. For other cases, it will suffice to prove that the denominator is non-zero.