Communication-efficient Distributed Multi-resource Allocation

Syed Eqbal Alam

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Information and Systems Engineering) at

Concordia University

Montréal, Québec, Canada

November 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Syed Eqbal Alam**

Entitled: **Communication-efficient Distributed Multi-resource Allocation**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. John Xiupu Zhang

_____ External Examiner
Dr. George Atia

_____ Examiner to Program
Dr. Luis Rodrigues

_____ Examiner
Dr. Chadi Assi

_____ Examiner
Dr. Walter Lucia

_____ Supervisor
Dr. Jia Yuan Yu

_____ Co-supervisor
Dr. Robert Shorten

Approved by _____

Dr. Abdessamad Ben Hamza, Chair
Concordia Institute for Information Systems Engineering

_____October 5, 2021

_____

Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Communication-efficient Distributed Multi-resource Allocation

Syed Eqbal Alam, Ph.D.

Concordia University, 2021

Distributed resource allocation arises in many application domains such as smart cities, intelligent transportation systems, sharing economy, cloud computing, edge-computing, power systems, etcetera. In several scenarios, the agents such as Internet of Things (IoT) devices may require multiple shared resources to achieve social optimum values; furthermore, they may have heterogeneous resource demands. Such distributed resource allocation problems are challenging to solve, especially when the agents are constrained through communication infrastructure, computational capabilities or do not wish to communicate with other agents in the network due to privacy reasons. Additionally, when the cost functions of agents are non-separable and are coupled through the allocation of multiple resources, in such cases, the single resource allocation algorithms are not efficient and provide suboptimal solutions.

In the available distributed solutions for multiple resources, best to my knowledge, agents exchange their information with at least one neighboring agent that may incur communication overhead or compromise agents' privacy. We develop several solutions to solve such problems for multiple divisible and multiple indivisible resources wherein no inter-agent communication is required. Moreover, we assume that each agent has private cost functions coupled with multiple resources; these functions are strictly convex, twice continuously differentiable, and increasing in each variable. Our first contribution is the stochastic distributed algorithm that solves multi-resource allocation problems with no direct agent-to-agent communication for divisible resources; moreover, it achieves social-optimum values. In the algorithms, each agent decides its resource demands locally, and an agent is unaware of the resource allocations of other agents. We solve the divisible multi-resource allocation problem by extending the additive-increase multiplicative-decrease

(AIMD) algorithm for single resource allocation by Wirth and co-authors. In the algorithm, the agents keep increasing the demands for a resource linearly until they receive a one-bit signal from a central agency. The central agency broadcasts the signal whenever one of the allocated resources reaches its capacity. Agents then respond to this signal in a probabilistic manner to decrease the resource demand. By doing so, the social optimum is achieved in long-term averages.

Our second contribution is a derandomized AIMD algorithm to solve a class of distributed optimization problems for multiple divisible shared resources. The algorithm is a derandomized version of the stochastic additive-increase and multiplicative-decrease (AIMD) algorithm. The developed solution uses a one-bit feedback signal for each resource between the system and the agents, and it does not require inter-agent communication. We show empirically that the long-term average allocations of multiple shared resources converge to optimal allocations, and the system achieves minimum social cost. Furthermore, we show that the derandomized AIMD algorithm converges faster than the stochastic AIMD algorithm, and both approaches provide approximately the same solutions.

Our third contribution is the stochastic algorithm for multiple indivisible (unit-demand) resources, inspired by classical stochastic approximation techniques. Each agent's consumption is modeled as a Bernoulli random variable in the solution, and no inter-agent communication is required. Moreover, we provide fundamental guarantees of convergence. Additionally, we present an example illustrating the performance of the algorithm.

Finally, we study the development of *Internet-of-Things* (IoT) enabled sharing economy applications. In many sharing economy scenarios, agents both produce as well as consume a resource; we call them *prosumers*. A community of prosumers agrees to sell excess resources to another community in a prosumer market. We propose a stochastic algorithm to regulate the number of prosumers in a prosumer community; each prosumer has a cost function coupled through its time-averaged production and consumption of the resource. Furthermore, each prosumer runs its distributed algorithm and takes (binary) decisions in a probabilistic way, whether to produce one unit of the resource or not and to consume one resource unit or not. In the developed approach, prosumers do not explicitly exchange information with each other due to privacy reasons but little with a central agency. Additionally, prosumers achieve the optimal values asymptotically.

Dedication

I dedicate this thesis to my parents Dr. Syed Izhar Alam and Mrs. Razia Khatoon.

Acknowledgments

For the last several years, many people have supported me in my academic journey to a Ph.D. They made this journey exciting and enjoyable; I am indebted for their support.

I would first like to express my deepest gratitude to my supervisor Dr. Jia Yuan Yu, and co-supervisor Dr. Robert Shorten, for their guidance and encouragement. They provided me the opportunity to explore several research directions during my Ph.D. program. In addition, their insightful discussions and feedback were invaluable and helped me complete this thesis. I am thankful to Bob for hosting me several times at Imperial College London, UK, and University College Dublin, Ireland.

I would also like to thank Dr. Fabian Wirth for his guidance and support during my Ph.D. journey. I also thank him for hosting me at the University of Passau, Germany. It was great working with you, Fabian. I spent wonderful days in Passau.

I express my heartfelt gratitude to Dr. George Atia for serving as an external examiner on my thesis committee. I am also grateful to Dr. Chadi Assi, Dr. Luis Rodrigues, and Dr. Walter Lucia for serving on my thesis committee. I want to thank them for their insightful suggestions and feedback to improve the thesis.

Furthermore, I would like to thank my master's thesis supervisor at IIIT-Bangalore, Dr. Shrisha Rao, for his guidance and encouragement to continue my research journey towards a Ph.D. I would also like to thank my collaborators Dr. Jakub Marecek and Ramen Ghosh, for their discussions and feedback. Thanks also to my friends and colleagues Dr. Rashid Hussain Khokhar, Dr. Nadeem Ashraf, Dr. Hesham Maghrabie, and Dr. Soroosh Shahtalebi, for brainstorming over coffee. I thank Innovai group members at Concordia University for arranging reading sessions. Particularly Dr.

Shaui Ma, Hamid Nabati, Viral Thakar, Denis Ergashbaev, Farshid Faal, Victor Deleau, and Mehdi Merai. I also thank Roman Overko at University College Dublin, Dr. Pietro Ferraro and Andrew Cullen at Imperial College London, and Dr. Andrii Mironchenko at the University of Passau, for their help and logistics support during my visits.

I acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC), Mitacs, Concordia University, Capbeast, Dataperformers, and Science Foundation Ireland, for their financial assistance in completing part of the work.

I express my heartfelt gratitude to my family members and friends for their support and encouragement during this challenging journey. Finally and most importantly, I thank my wife and soul-mate, Asfia Fathima, and daughter Hafsa for their unconditional love and support during this journey. Without your support, I would not have made it this far.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Distributed multi-resource allocation is a fundamental and important problem that arises in many application domains such as smart cities, intelligent transportation systems, sharing economy, cloud computing, edge-computing, peer-to-peer energy trading, to name a few. In many scenarios, agents such as cameras, wearable devices, energy consumers or producers, cars, etcetera in a network may require multiple shared divisible or indivisible resources to complete their tasks. The divisible resources are the resources that can be allocated as a fraction, such as random access memory, disk storage, network bandwidth, CPU cycles, etc. Furthermore, the indivisible resources or unit-demand resources are the resources that can either be allocated one unit or zero units, for example, parking slots. Moreover, the agents may have heterogeneous resource demands and may aim to minimize the social cost over the network. However, such distributed multi-resource allocation problems are challenging to solve than the single distributed resource allocation problems. Mainly when the cost of an agent is coupled through the allocation of multiple resources or the agents are constrained through communication infrastructure, computational capabilities. In addition, they may demand the resources based on local computation, and they do not want to exchange information with other agents in the network due to privacy reasons.

Distributed multi-resource allocation problems are studied in two research directions; the first is the computational social choice theory, wherein a group of agents makes collective decisions to maximize the sum of their utilities and achieve the group's social welfare. Therein fair allocation of resources is studied, such as envy-freeness, maximin, proportionality, strategy-proofness,

Pareto-optimality, sharing incentive, etc., (Endriss, 2014), (W. Wang, Liang, & Li, 2015), (Conitzer, Freeman, Shah, & Vaughan, 2019), (Benadè, Procaccia, & Qiao, 2019), (Benabbou, Chakraborty, Elkind, & Zick, 2019), (Aziz & Rey, 2020), (Bei, Li, Liu, Liu, & Lu, 2021), (Aziz, Caragiannis, Igarashi, & Walsh, 2022), (Halpern & Shah, 2021). In the second direction, the resource allocation problems are formulated as distributed optimization problems wherein agents in a network collaborate to access multiple shared resources and aim to minimize the sum of agents' cost functions to obtain social optimum value. In this thesis, we formulate multi-resource allocation problems as distributed optimization problems. Moreover, we develop several distributed iterative algorithms to solve the resource allocation problems for multiple-divisible and multiple-indivisible resources wherein several agents in a network collaborate to access the shared resources and minimize the total cost over the network. Roughly speaking, in the existing literature, the agents achieve optimal allocation of resources through regular communication by exchanging their gradients of the cost functions (Nedic & Ozdaglar, 2009), (Lee, Nedic, & Raginsky, 2018), (J. Zhang, Uribe, Mokhtari, & Jadbabaie, 2019), (Pu, Shi, Xu, & Nedic, 2021), (Pu & Nedic, 2021) or sharing the Lagrange multipliers (Boyd, Parikh, Chu, Peleato, & Eckstein, 2011), (Carli & Dotoli, 2020), (Shang et al., 2021) with at least one of their neighbors, which may lead the agents to reveal their private information. Moreover, sharing information may also increase the communication overhead on the system. As noted in (G. Lan, Lee, & Zhou, 2018), (Léauté & Faltings, 2013), (B. Li, Cen, Chen, & Chi, 2020), (Elgabli et al., 2021), communication and privacy are two of the major issues in distributed optimization.

It is challenging to solve distributed optimization problems for multiple resources for many reasons. The first is when the agents demand resources based on local computation and do not want to communicate with other agents in the network due to privacy reasons. Second, such systems can be dynamic, and the number of participating agents may change over time. Third, the agents may not know the capacity constraints and the total number of agents in the network. Finally, when the allocation of one resource depends on other resources, and the agents use non-separable cost functions coupled through the allocation of multiple resources. We call a function non-separable if it can not be split into independent cost functions, as in the case of separable cost functions. Moreover, the single resource allocation algorithms are not efficient for such scenarios and provide

suboptimal solutions. Thus, we need algorithms that are efficient and provide optimal solutions for such scenarios and incur little communication overhead. In this thesis, we develop distributed iterative algorithms that address these.

1.1 Problem formulation

Let us consider that n agents in a network collaborate to access m shared resources. Let $\mathcal{C}_1 \geq 0$, $\mathcal{C}_2 \geq 0$, and $\mathcal{C}_m \geq 0$ be the capacities of resources, respectively. Moreover, agent i has allocation states $x_{ji} \in [0, \mathcal{C}_j]$ for resource j; they represent the amount of resource j allocated to agent i, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. We assume that agent i has a strictly convex, continuously differentiable cost function $f_i : \mathbb{R}_+^m \to \mathbb{R}_+$, for $i = 1, 2, \ldots, n$. Let $t_0 < t_1 < t_2 < \ldots$, be discrete time instants. Furthermore, let $x_{ji}(t_k) \in [0, \mathcal{C}_j]$ denote the amount of resource j allocated to agent i at time instant t_k, for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and $k \in \mathbb{N}$. We formulate the multi-resource allocation problems as the following optimization problem and aim to solve it in a distributed way:

$$\begin{aligned} \min_{x_{11},\ldots,x_{mn}} \quad & \sum_{i=1}^{n} f_i(x_{1i}, x_{2i}, \ldots, x_{mi}), \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{1i} = \mathcal{C}_1, \\ & \vdots \\ & \sum_{i=1}^{n} x_{mi} = \mathcal{C}_m, \\ & x_{ji} \geq 0, \text{ for } i = 1, \ldots, n \text{ and } j = 1, 2, \ldots, m. \end{aligned} \tag{1.1}$$

We define the time-averaged allocation $\overline{x}_{ji}(t_k) \triangleq \frac{1}{k+1} \sum_{\ell=0}^{k} x_{ji}(t_\ell)$ of agent i for resource j, for $i = 1, \ldots, n$ and $j = 1, 2, \ldots, m$. Let $\mathbf{x}^* = (x_{11}^*, \ldots, x_{mn}^*) \in (\mathbb{R}_+^n)^m$ be the optimal point.

We aim to develop distributed iterative algorithms that determine the values of $x_{ji}(t_k)$ at each time instant t_k with no inter-agent communication; however, a little with a central server, and obtain optimal allocations in long-term averages:

$$\lim_{k \to \infty} \overline{x}_{ji}(t_k) = x_{ji}^*, \text{ for } i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, m. \tag{1.2}$$

Note that we use *central server*, *central agent*, or *control unit* interchangeably.

For the indivisible resource allocation cases, the allocation states of agent i is updated with $x_{ji} \in \{0, 1\}$ for resource j, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. Additionally, the amount of resource j allocated to agent i at time instant t_k is updated with $x_{ji}(t_k) \in \{0, 1\}$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

1.2 A motivating example

Suppose that a smart city deploys a *multi-camera coordination system*, in which several cameras work together for the surveillance of the city; the system diagram is illustrated in Figure 1.1. Moreover, suppose that the cameras are deployed at different locations. If a camera observes any unusual activities, it should demand the required amount of resources with a higher probability than other cameras to notify the observed activities immediately. Suppose that there are servers set up by the city agency, which store and process the videos sent by all cameras. One of the servers acts as a central server. The central server keeps track of the aggregate demand of the resources at each time step and sends a notification in the network when the aggregate demand exceeds resource capacity. Each camera requires a different amount of network bandwidth, CPU cycles, memory (RAM), and storage to transmit, process, and store the videos on the servers. Each camera has a private cost function which is coupled through the allocation of multiple resources. Allocating resources to cameras incurs some cost captured by the camera's cost function. Assume that a camera decides its demands based on its cost function and its average allocations. Moreover, the cameras should receive optimal allocations, and the network achieves social optimum cost over time.

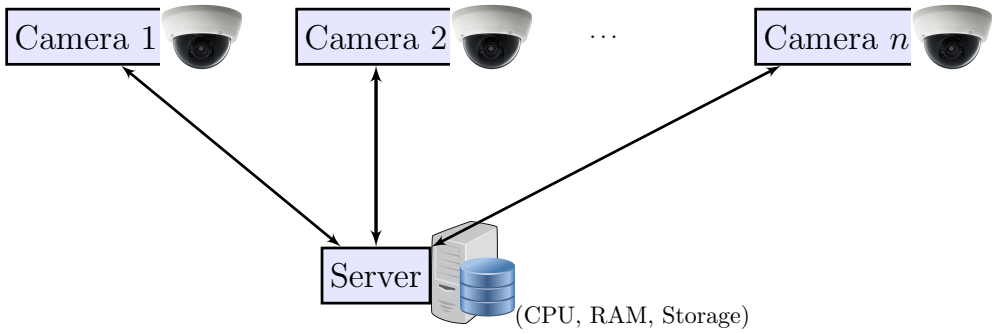


Figure 1.1: Multi-camera coordination system.

1.3 Objectives

This thesis aims to develop distributed algorithms to solve multi-resource allocation problems with no inter-agent communication but a little with a central server that keeps track of the aggregate demands. Precisely, my goal is to achieve the following objectives.

(1.) To extend the distributed iterative single resource allocation algorithms (Wirth, Stüdli, Yu, Corless, & Shorten, 2019) to multi-resource allocation algorithms with no inter-agent communication.

(2.) To extend the distributed iterative algorithm to regulate the number of consumers (Griggs, Yu, Wirth, Hausler, & Shorten, 2016) to an algorithm that regulates the number of prosumers (prosumers are agents that act as both consumers and producers) in a sharing economy setting with no prosumer to prosumer communication.

(3.) To show the convergence of the proposed multi-resource allocation algorithms.

To achieve these objectives, we developed several distributed iterative algorithms for multiple-divisible, and multiple-indivisible resources wherein agents in a network collaborate to access the shared resources and minimize the total cost over the network. The cost functions of the agents are coupled through the allocation of multiple resources. We consider a central server that works as a coordinator—aggregates resource demands by agents and occasionally sends feedback signals

in the network. Agents in the network perform local computation and demand resources; however, they do not share their cost function, partial derivatives of the cost function, or allocation history with other agents or the central server. Figure 1.2 presents the system diagram. Therein, agents such as mobile phones, surveillance cameras, smartwatches perform the local computation to demand the resources. The central server aggregates the demands. If the demands exceed the capacity of the resource, it sends a capacity constraint notification to agents to reduce their resource demands in the next step.

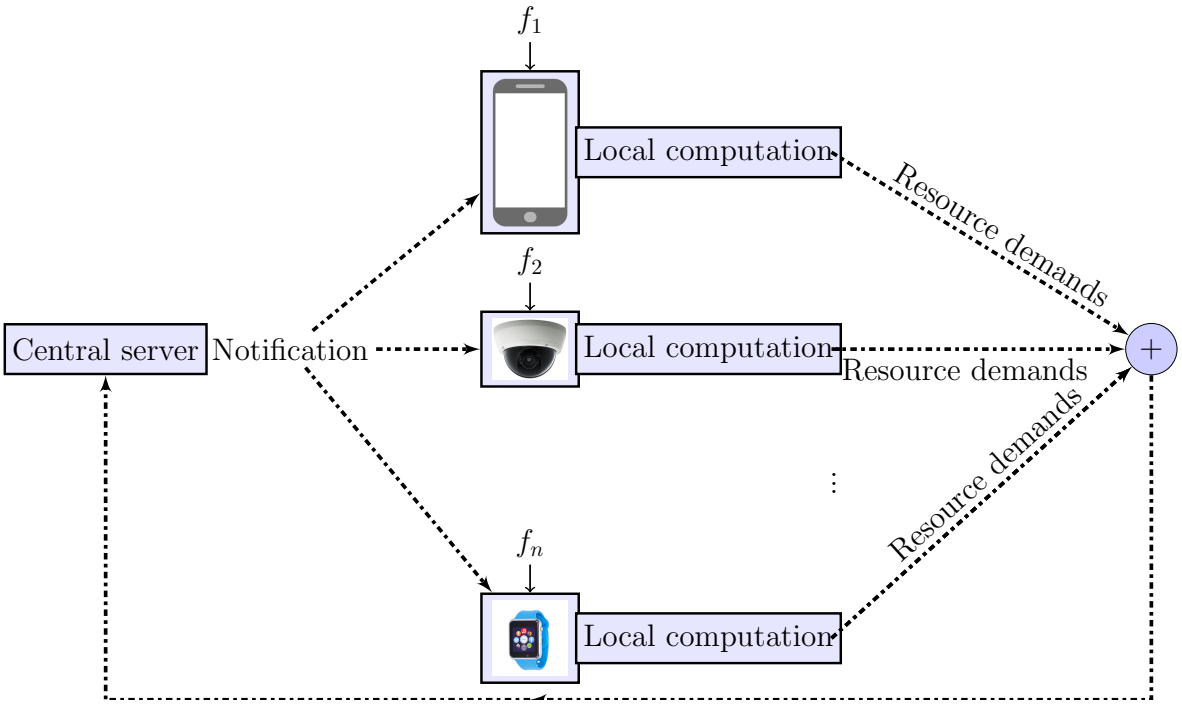


Figure 1.2: System diagram: Distributed multi-resource allocation with no inter-agent communication; the agents demand the resources based on local computation. The central server keeps track of aggregate resource demands and occasionally sends capacity constraints notifications to agents in the network.

Recently, such settings have been extensively studied by the machine learning community as federated learning. *Federated learning* is a distributed machine learning technique in which several agents (clients) collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global

model (McMahan, Moore, Ramage, Hampson, & Arcas, 2017), (Konecny, McMahan, Ramage, & Richtarik, 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny, McMahan, & Ramage, 2015), (J. Wang, Liu, Liang, Joshi, & Poor, 2020).

Our algorithms for divisible multi-resource allocations are based on the additive increase multiplicative decrease (AIMD) algorithm, discussed briefly in Section 1.3.1. Furthermore, the indivisible multi-resource allocation algorithms are an extension of Griggs and co-authors' single resource allocation algorithm (Griggs et al., 2016), based on the ideas from stochastic approximation techniques (Robbins & Monro, 1951), (Borkar, 2008).

1.3.1 AIMD based distributed multi-resource allocation for divisible resources

The additive increase multiplicative decrease (AIMD) algorithm was developed for congestion avoidance in transmission control protocol (TCP) (Chiu & Jain, 1989), (Jacobson, 1988), which is studied extensively in many areas (Wirth et al., 2019), (Crisostomi, Shorten, Studli, & Wirth, 2017), (Avrachenkov, Borkar, & Pattathil, 2017), (Shah, Incremona, Bolzern, & Colaneri, 2019), (Leung, Lai, & Ding, 2021). We now present the simplest version of the AIMD algorithm. Interested readers can refer to (Corless, King, Shorten, & Wirth, 2016) for a detailed discussion of the AIMD algorithms. Briefly, the AIMD algorithm has two phases: *Additive increase* and *multiplicative decrease*. In the *additive increase* (AI) phase, an agent keeps increasing its resource demands linearly by a constant value $\alpha_i > 0$ until it receives a one-bit signal called *capacity event* from the central server. The central server broadcasts the one-bit capacity event signal when the aggregate resource demand reaches the capacity of the resource. After receiving the capacity event signal, agents respond and reduce their demands multiplicatively by a constant $0 \leq \beta_i \leq 1$; this phase is called the *multiplicative decrease* (MD). After the MD phase, agents enter the AI phase again until they receive the next capacity event signal. This process repeats over time.

Consider n agents collaborate to access a shared resource, and let $x_i(t) \geq 0$ denote the amount of the resource demand by agent i at time t. Let $\mathcal{C}$ denote the capacity of the resource. Let $t_k, k \in \mathbb{N}$ denote the time at which capacity events occur, that is, the time at which the total demand reaches the capacity, $\sum_{i=1}^{n} x_i(t_k) = C$. When capacity is reached, agents decrease their resource demands.

The instantaneous decrease of the resource demand by agent i is defined as:

$$x_i(t_k^+) \triangleq \beta_i x_i(t_k). \tag{1.3}$$

Moreover, in the AI phase, agents increase their shares at a constant rate, defined as:

$$x_i(t) = \beta_i x_i(t_k) + \alpha_i(t - t_k), \quad t_k < t \leq t_{k+1}. \tag{1.4}$$

Wirth and co-authors (Wirth et al., 2019) introduced probabilistic intent with which an agent responds to the capacity event to solve the distributed optimization problem in a stochastic and distributed way for a single resource. We extend this to develop a stochastic distributed algorithm for divisible multi-resource allocation in Chapter 2. Additionally, in Chapter 3, we proposed a deterministic distributed AIMD algorithm wherein agents demand the resources in a deterministic manner to achieve a social optimum cost for the network. The algorithms incur very little communication overhead. Chapter 2 achieves Objectives 1 and 3, and Chapter 3 achieves Objective 1.

1.3.2 Distributed multi-resource allocation for indivisible resources

We develop a stochastic distributed algorithm in Chapter 4 wherein agents in a network collaborate to access multiple indivisible shared resources. It extends the single indivisible resource allocation algorithm by Griggs and co-authors (Griggs et al., 2016) to multiple indivisible resources. Chapter 4 achieves Objectives 1 and 3. Furthermore, Chapter 5 presents stochastic distributed algorithms for community-based prosumer markets; *prosumers* are the agents that both produce and consume a resource. It regulates the number of prosumers participating in the prosumer community to achieve social optimum cost. Chapter 5 achieves Objective 2.

We discuss the contributions of the thesis in detail in Section 1.5. Moreover, the following section presents the background and literature review relevant to the thesis.

1.4 Background and literature review

This section briefly presents the background and related work on distributed resource allocation (as a computational social choice), distributed optimization, federated optimization, and sharing economy. We also discuss optimality conditions for multiple resources.

1.4.1 Computational social choice theory

In one research direction, distributed resource allocation problems are studied as a computational social choice problem, wherein multiple agents coordinate to maximize social welfare. It has attracted significant attention from artificial intelligence researchers (Endriss, 2014), (Brandt, Conitzer, Endriss, Lang, & Procaccia, 2016), (Freeman, Zahedi, & Conitzer, 2017), (Benadè et al., 2019), (Bei et al., 2021), (Aziz et al., 2022), (Halpern & Shah, 2021). In this setting, the notion of fairness—such as envy freeness, Pareto optimality, strategy proofness, etcetera, are studied.

There are two types of resources—divisible and indivisible. *Divisible resources* can be allocated as a fraction, for example, CPU time, memory, storage, network bandwidth, etcetera. Furthermore, an *indivisible resource* can either be allocated one unit to an agent or not allocated. Single resource allocation models are studied extensively, but multi-resource allocation is not well studied. In many cases, single resource allocation strategies are applied to allocate multiple resources, which may be inefficient. In the resource allocation literature, generally, fairness and efficiency of allocations are studied (Ghodsi et al., 2011), (Joe-Wong, Sen, Lan, & Chiang, 2013). Two popular multi-resource allocation strategies are maximin and proportional fairness (Poullie, Bocek, & Stiller, 2018). Generally speaking, social welfare is classified into the following three categories based on the utility functions used.

(i) Utilitarian social welfare: In this model, the sum of utilities of all the agents in the group is maximized.

(ii) Egalitarian social welfare: In this model, the utility of the agent, which receives the minimum allocation (worst-off agent), is maximized.

(iii) Nash product social welfare: In this model, the product of utilities of all the agents is maximized.

Further details of these classes and utility functions can be found at (Chevaleyre et al., 2006), (T. T. Nguyen, Roos, & Rothe, 2013), and (Poullie et al., 2018). In this thesis, we take the utilitarian viewpoint (Boutilier et al., 2012) in which multiple agents coordinate to minimize the sum of their cost functions subject to capacity constraints of the resources.

Divisible resources

Divisible resource allocations are studied by Ghodsi et al. (Ghodsi et al., 2011), they propose a dominant resource fairness algorithm for multiple divisible resource allocations. Recently, Nguyen et al. (D. T. Nguyen, Le, & Bhargava, 2019) proposed a market-based algorithm to allocate multiple resources in fog computing. Furthermore, Fossati et al. (Fossati, Moretti, Perny, & Secci, 2020) proposed a multi-resource allocation for network slicing in the context of 5G networks. The distributed resource allocation for 5G-vehicle to vehicle communication was proposed by Alperen et al. (Gündoğan, Gürsu, Pauli, & Kellerer, 2020); and an envy-free fair allocation mechanism is proposed in (Khamse-Ashari, Lambadaris, Kesidis, Urgaonkar, & Zhao, 2019) for allocating multiple computing resources of servers. A survey on the allocation of multiple resources can be found at (Poullie et al., 2018).

Indivisible resources

Matt and Toni in (Matt & Toni, 2006) developed a mechanism to compute egalitarian allocations of indivisible resources. Furthermore, group fairness for indivisible goods is studied in (Aziz & Rey, 2020), (Conitzer et al., 2019), (Freeman, Sikdar, Vaish, & Xia, 2019). In (Aziz et al., 2022), the authors propose fair allocation algorithms for a general setting wherein agents may have positive utility (as in goods) or negative utility (as in chores) for each item. A fair allocation mechanism for multiple indivisible goods is proposed in (Dogan, 2021). Moreover, interested readers can refer to (Ohseto, 2021), (Chakraborty, Igarashi, Suksompong, & Zick, 2021), (Kawase & Sumita, 2020), (Murhekar & Garg, 2021), (Baklanov, Garimidi, Gkatzelis, & Schoepflin, 2021), (Oh, Procaccia, &

Suksompong, 2021) for some of the recent works on fair allocation of indivisible goods.

In a recent work (Bei et al., 2021), Bei and co-authors presented mechanisms for fair allocation of mixed resources, divisible and indivisible. In addition, optimality in computational social choice is studied in (Boutilier et al., 2015), (Elkind, Faliszewski, & Slinko, 2009). We can think of our optimization problems as *budgeted social choice* problems (T. Lu & Boutilier, 2011), where the capacity of a resource represents the budget.

1.4.2 Distributed optimization

Tsitsiklis and co-authors proposed the distributed optimization problem in their seminal work (Tsitsiklis, Bertsekas, & Athans, 1986; Tsitsiklis, 1984). We discuss some of the existing distributed optimization methods here.

Consensus-based methods

Let us consider n agents in a network, let $\mathbf{x} \in \mathbb{R}^n$ be the decision vector, and $f_i : \mathbb{R}^n \to \mathbb{R}$ be convex and smooth cost functions. Tsitsiklis et al. (Tsitsiklis et al., 1986; Tsitsiklis, 1984) in their seminal work, proposed a consensus-based distributed iterative computational model to minimize the global objective function $f(\mathbf{x})$. The optimization problem is formulated as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}).$$

To solve this unconstrained optimization problem, the decision vector $\mathbf{x}$ is distributed among n agents; the agents exchange their local information with their neighbors to find an approximate solution. After this seminal work, distributed optimization is studied extensively. We present some of the consensus-based distributed optimization approaches as follows.

(i) **Sub-gradient methods**: Nedic and Ozdaglar (Nedic & Ozdaglar, 2009) proposed a distributed sub-gradient method to solve an unconstrained distributed optimization problem in which they minimize the sum of convex cost functions $\sum_{i=1}^{n} f_i(\mathbf{x})$ of the agents in the network, where $f_i(\cdot)$ is the private cost function of the agent i, for $i = 1, 2, \ldots, n$. In this method, an agent updates its information state at a discrete-time based on the sub-gradient of its cost

function and exchanges it with its neighbors. The communication is asynchronous; that is, the agents are not required to communicate at the same time. Additionally, the number of agents in the network may vary with time. Following this, information states of agents make consensus, and the agents achieve social optimal value. Yuan et al. (K. Yuan, Ling, & Yin, 2016) propose a distributed model to solve the unconstrained optimization problem based on a sub-gradient method with a constant step-size, in which an agent communicates with its neighbors to update its local variables. Interested readers can refer to (Romao, Margellos, Notarstefano, & Papachristodoulou, 2021), (Liu, Qiu, & Xie, 2017), (Nedic, Ozdaglar, & Parrilo, 2010), (C. Li, Chen, Li, & Wang, 2019) for other works based on sub-gradient methods.

(ii) **Gossip algorithm for distributed averaging**: The aim here is to solve the following optimization problem.

$$\min_{\mathbf{x}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}), \quad \text{where } \mathbf{x} \in \mathbb{R}^n.$$

In this method, each agent exchanges its information with a randomly chosen neighbor at each iteration. Because the neighbors are chosen randomly, the algorithm is called a *gossip* algorithm (Boyd, Ghosh, Prabhakar, & Shah, 2006). The authors in (Boyd et al., 2006) consider a network in which agents can join and leave any time. They calculate the optimal averaging based on semidefinite programs, which is further solved by the sub-gradient method to obtain an approximately optimal solution without the need for complete knowledge of the network's topology, which is generally needed to solve the semidefinite programs. Gradient-based methods for distributed optimization were developed in (Pu et al., 2021), (Pu & Nedic, 2021), wherein an agent maintains an estimate of decision variables as well as an estimate of the gradients of the cost functions. The estimate of the gradients is pushed to the neighboring agents, whereas the estimates of the decision variables is pulled from the neighboring agents. Additionally, the settings work for synchronous and asynchronous random gossip.

(iii) **Dual averaging**: Suppose there are n agents in a network. Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be a convex cost function of the agent i and $\mathbf{x}$ be in the compact set X. Duchi et al. (Duchi, Agarwal, &

Wainwright, 2012) proposed a dual averaging sub-gradient algorithm to solve the following constrained optimization problem.

$$\min_{\mathbf{x}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}), \quad \text{subject to } \mathbf{x} \in X.$$

In this algorithm, at each iteration $k = 0, 1, 2, \ldots$, the agent i keeps track of two parameters $x_i(k)$ and $y_i(k)$, where $x_i(k)$ is the estimate of its local decision variable and $y_i(k)$ is the dual parameter. Dual parameter $y_i(k)$ is updated using the weighted or probabilistic estimates of the neighbors of agent i and the sub-gradient of the cost function f_i. Here, the neighbors of agent i are the agents directly connected to the agent i. After the dual update, agent i updates its local parameter $x_i(k)$ based on a projection criteria which uses dual parameter $y_i(k)$ and $\gamma(k) > 0$ a non-increasing step size. They show that the time-averaged sequence $\left\{\overline{x}_i(k) \triangleq \frac{1}{k+1}\sum_{\ell=0}^{k} x_i(\ell)\right\}$ converges to optimal value asymptotically. Tsianos et al. (Tsianos, Lawlor, & Rabbat, 2012) extended this work to propose Push-Sum distributed dual averaging and provide the guaranteed convergence to optimal values. Furthermore, Chang et al. (Chang, Nedic, & Scaglione, 2014) propose a consensus-based distributed primal-dual perturbation algorithm to minimize the sum of objective functions of agents, subject to inequality constraint. In a recent work, a privacy-preserving dual averaging algorithm for distributed optimization is proposed in (D. Han, Liu, Sandberg, Chai, & Xia, 2021). Additionally, (Uribe, Lee, Gasnikov, & Nedic, 2021) studied dual-based methods for distributed optimization. Interested readers can refer to (Lee et al., 2018), (Liu, Chen, & Hero, 2018) and the papers cited there for dual-averaging based distributed optimization techniques.

(iv) **Broadcast-based approach**: Nedic (Nedic, 2011) proposed a broadcast-based algorithm. They consider asynchronous communication between agents; in their solution, the failure of a link does not affect the solution. Let n be the number of agents in a network. Let $f_i : \mathbb{R}^m \to \mathbb{R}$ be a convex private cost function of the agent i and $\mathbf{x}$ be in the compact set X.

The optimization problem is formulated as follows.

$$\min_{\mathbf{x}} \quad \sum_{i=1}^{n} f_i(\mathbf{x}) \quad \text{subject to } \mathbf{x} \in X.$$

Furthermore, each agent has a clock that is independent of the clocks of other agents, the clock of an agent ticks based on the Poisson rate of 1. When the clock of an agent ticks, then the agent becomes active and broadcasts its estimated parameter to its neighbors. If a neighbor receives the broadcast, it updates its parameter using the received parameter, the sub-gradient of its cost function, and step size. Because the communication link may fail, some of the neighbors may not receive the broadcast message. The authors consider two step-sizes; they show almost sure convergence for decreasing step-size and convergence in attraction for constant step-size. Interested readers can refer to broadcast-based gossip algorithms at (Aysal, Yildiz, Sarwate, & Scaglione, 2009; Lee & Nedic, 2016; Salehisadaghiani & Pavel, 2016; Silvestre, Hespanha, & Silvestre, 2019).

Berahas et al. (Berahas, Bollapragada, Keskar, & Wei, 2019) propose an adaptive framework for unconstrained distributed optimization problems that aim to minimize the sum of cost functions of agents in a network. The framework balances the communication and computation cost of an algorithm depending upon the application considered. Koloskova et al. (Koloskova, Stich, & Jaggi, 2019) propose algorithms to solve unconstrained distributed convex optimization problems based on the gossip algorithm. Using gradient compression, they reduce the communication overhead by two orders of magnitude. Furthermore, Lan et al. (G. Lan et al., 2018) propose primal–dual type distributed algorithms which skip some of the inter-agent communication rounds in solving primal subproblems to reduce communication overhead. Recently, privacy-preserving distributed optimization has been studied in several papers; a few are listed as (D. Han et al., 2021), Xiong2020, (Huo & Liu, 2022). Moreover, in (Tallapragada & Cortes, 2020), the authors proposed distributed algorithms for coordinating vehicular traffic at an intersection to minimize travel time and energy consumption as an application to intelligent transportation systems.

Alternating direction method of multipliers

One of the most famous approaches to solve distributed convex optimization problems is the *alternating direction method of multipliers (ADMM)* (Boyd et al., 2011). The ADMM algorithm decomposes a problem into two subproblems and then sequentially solves them. At each iteration, the Lagrangian multipliers are updated.

In this algorithm also, agents coordinate to solve a large global problem. Now, let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$ be the convex cost functions, and let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$ be the decision variables. Additionally, let $A \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{p\times m}$ be the matrices and $c \in \mathbb{R}^p$ be a constant. Then, we write the following optimization problem

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \quad \text{subject to } A\mathbf{x} + B\mathbf{z} = c. \tag{1.5}$$

Let $\delta > 0$ be a constant called penalty parameter; furthermore, let $\mathcal{L}_\delta : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ be the Lagrangian and $\mu \in \mathbb{R}^p$ be the Lagrangian multipliers, then the augmented Lagrangian of (1.5) is obtained as follows,

$$\mathcal{L}_\delta(\mathbf{x}, \mathbf{z}, \mu) = f(\mathbf{x}) + g(\mathbf{z}) + \mu^\top (A\mathbf{x} + B\mathbf{z} - c) + (\delta/2) \left\| A\mathbf{x} + B\mathbf{z} - c \right\|_2^2 .$$

Then by fixing one of $\mathbf{x}(k+1)$ and $\mathbf{z}(k+1)$, the minimum value of $\mathcal{L}_\delta(\cdot)$ is found and value of the other is updated. Further, using these values, $\mu(k+1)$ is updated.

In this solution, the regularization variables $\|x_i - \mathbf{z}(k)\|_2^2$ are updated in such a way that the social optimum value is achieved (Boyd, Parikh, Chu, Peleato, & Eckstein, 2010). Wei et al. (Wei & Ozdaglar, 2012) extend ADMM in which each agent in the network solves a subproblem using its local cost functions and information exchanged with its neighbors. Zhang and Kwok (R. Zhang & Kwok, 2014) propose an asynchronous distributed ADMM algorithm; furthermore, a broadcast-based distributed ADMM algorithm is proposed in (Makhdoumi & Ozdaglar, 2014). Recently, distributed ADMM is studied in (Carli & Dotoli, 2020), and a quantized group ADMM is studied in (Elgabli et al., 2021) wherein an agent communicates with its two neighbors to reduce communication overhead. Additionally, ADMM is used in many application areas, for example, electric

vehicle charging (Vaya, Andersson, & Boyd, 2014), (Rivera, Goebel, & Jacobsen, 2017), smart grid (Liu & Han, 2015), machine learning (Boyd et al., 2011), (Shang et al., 2021), to name a few. A detailed survey on distributed optimization can be found at (Nedic, Pang, Scutari, & Sun, 2018), (Yang et al., 2019).

In all the above methods, the agents communicate with at least one neighboring agent. However, in our developed algorithms, the agents do not exchange information with other agents in the network; however, a little with a central server that keeps track of the aggregate demands and broadcast feedback signals in the network.

1.4.3 Federated optimization

Federated learning is a distributed machine learning technique in which several agents (clients) collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020).

One of the most popular and widely used federated optimization techniques is FederatedAveraging (FedAvg) by McMahan and co-authors (McMahan et al., 2017). It is based on stochastic gradient descent. In the algorithm, a fixed number of agents are selected randomly from agents in the network at each iteration. Each selected agent performs the stochastic gradient descent on its local data using the global model for a certain number of epochs. Moreover, each agent performs the local updates for the same number of epochs. Finally, it communicates the averaged gradients to the central server. The central server then takes a weighted average of the gradients by agents and updates the global model. The process is repeated until the model is trained.

Another federated optimization technique is FedProx (T. Li et al., 2020), which is a generalization of the FedAvg. In FedProx, the number of epochs is not fixed for each iteration, as in FedAvg; however, it varies, and partial updates by agents are averaged to update the global model. It is

proposed for heterogeneous devices and datasets, data that are not independent and identically distributed (non-IID). In addition, the authors provide convergence guarantees. Sattler and co-authors (Sattler, Wiedemann, Muller, & Samek, 2020) also proposed a federated optimization technique for non-IID data. We list a few other federated optimization techniques such as FedNova (J. Wang et al., 2020), SCAFFOLD (Karimireddy et al., 2020), Overlap-FedAvg (Zhou, Ye, & Lv, 2022), federated composite optimization (H. Yuan, Zaheer, & Reddi, 2021), federated adaptive optimization (Reddi et al., 2021). Besides, interested readers can refer to (Kairouz et al., 2021), (Yin, Zhu, & Hu, 2021) and the papers cited therein for a detailed discussion on advances in federated learning and future research directions.

1.4.4 Sharing economy

Recently, consumers in several sectors have adopted shared ownership of resources and services with guaranteed access. They want to do so due to the need to reduce wastage, environmental concerns, or monetary benefits.

The analytics work on sharing economy has primarily followed two directions; (i) market design and equilibria, and (ii) optimal allocation of resources. Here, we give a very brief picture of some of the most recent work. In (Georgiadis, Iosifidis, & Tassiulas, 2020), Georgiadis et al. propose three types of resource allocation mechanisms; centralized, coalition-based (game-theoretic approach), and peer-to-peer. In (Gkatzikis, Iosifidis, Koutsopoulos, & Tassiulas, 2014), Gkatzikis et al. propose a collaborative consumption mechanism to minimize the electricity cost of a community, and in (Tushar et al., 2018), Tushar et al. developed a game-theoretic model for peer-to-peer energy trading. Furthermore, in (Moret & Pinson, 2018), Moret et al. design a community-based distributed energy collective market model that helps energy prosumers to optimize their energy resources and to achieve the social welfare of the community. Iosifidis and Tassiulas (Iosifidis & Tassiulas, 2017) propose optimization techniques for resource exchange and production scheduling for cooperative systems. Courcoubetis and Weber (Courcoubetis & Weber, 2012) propose mechanisms for the optimal allocation of shared computing resources. In (Hasan, Hentenryck, Budak, Chen, & Chaudhry, 2018), Hasan et al. propose a community-based car-sharing model to maximize trip sharing. To do so, they use mixed-integer programming, graph theory, and clustering

techniques. In (C. Zhang, Wu, Zhou, Cheng, & Long, 2018), Zhang et al. propose a game-theoretic peer-to-peer energy trading model between local prosumers and distributed energy resources. They show that the model gives rise to an equilibrium between energy production and consumption. In addition to these works, a peer-to-peer equilibrium model for collaborative consumption is proposed by Benjaafar et al. in (Benjaafar, Kong, Li, & Courcoubetis, 2018). Furthermore, in (D. H. Nguyen, 2021), authors propose decentralized optimization techniques based on alternating direction method of multipliers (ADMM) for peer-to-peer energy trading.

Recently, peer-to-peer energy systems for connected communities have been studied in (Tushar et al., 2021). We refer interested readers to a recent review paper, which covers market design, optimization techniques, and interesting future directions at (Sousa et al., 2019). Interested readers can also refer to (Tushar, Saha, Yuen, Smith, & Poor, 2020), (Sampath, Paudel, Nguyen, Foo, & Gooi, 2021), (Umer, Huang, Khorasany, Afzal, & Amin, 2021) for works on peer-to-peer energy trading.

1.4.5 Optimality conditions for multiple resources

Let us consider n agents that collaborate to access two shared resources. We consider two resources here; however, it can easily be extended for more than two resources. Let $\mathcal{C}_1 \geq 0$ and $\mathcal{C}_2 \geq 0$ be the capacities of resource 1 and resource 2, respectively. Let the cost function $f_i : \mathbb{R}_+^2 \to \mathbb{R}_+$ be strictly convex, increasing in each variable, and continuously differentiable, for $i = 1, 2, \ldots, n$. We formulate the following optimization problem (cf. (1.1)).

$$\begin{aligned} \min_{x_{11},\ldots,x_{1n},x_{21},\ldots,x_{2n}} \quad & \sum_{i=1}^{n} f_i(x_{1i}, x_{2i}), \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{1i} = \mathcal{C}_1, \\ & \sum_{i=1}^{n} x_{2i} = \mathcal{C}_2, \\ & x_{1i} \geq 0, \quad x_{2i} \geq 0, \text{ for } i = 1, \ldots, n. \end{aligned} \tag{1.6}$$

Let $(x_{1i}^*, x_{2i}^*) \in \mathbb{R}_+^2$ be the optimal point, for $i = 1, \ldots, n$. Let $\mathbf{x}_1 = (x_{11}, \ldots, x_{1n}) \in \mathbb{R}_+^n$ and $\mathbf{x}_2 = (x_{21}, \ldots, x_{2n}) \in \mathbb{R}_+^n$. Let the vectors $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{R}^2$, and $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{R}^n$ and $\mathbf{r} = (r_1, r_2, \ldots, r_n) \in \mathbb{R}^n$ be Lagrange multipliers of optimization problem (1.6) corresponding to the equalities and the inequalities, respectively. Let us now consider the Lagrangian of (1.6), we obtain

$$\begin{aligned} H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) = \sum_{i=1}^{n} f_i(x_{1i}, x_{2i}) + \mu_1 \sum_{i=1}^{n} (x_{1i} - \mathcal{C}_1) \\ + \mu_2 \sum_{i=1}^{n} (x_{2i} - \mathcal{C}_2) - \sum_{i=1}^{n} s_i x_{1i} - \sum_{i=1}^{n} r_i x_{2i}. \end{aligned} \tag{1.7}$$

Let $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*) \in (\mathbb{R}_+^n)^2$ minimize $H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r})$. We write the KKT conditions (Chapter 5.5.3) (Boyd & Vandenberghe, 2004) of Problem 1.6 as:

(i) Stationary: We have

$$\frac{\partial}{\partial x_{1i}} H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r})\Big|_{\mathbf{x}=\mathbf{x}^*} = 0, \text{ for } i = 1, 2, \ldots, n. \tag{1.8}$$

Analogously, we have

$$\frac{\partial}{\partial x_{2i}} H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r})\Big|_{\mathbf{x}=\mathbf{x}^*} = 0, \text{ for } i = 1, 2, \ldots, n. \tag{1.9}$$

(ii) Complementary slackness:

$$s_i x_{1i} = 0, \text{ and } r_i x_{2i} = 0, \text{ for } i = 1, 2, \ldots, n. \tag{1.10}$$

(iii) Primal feasibility:

$$x_{ji}^* \geq 0, \text{ for } i = 1, 2, \ldots, n, j = 1, 2, \text{ and } \sum_{i=1}^{n} x_{ji}^* - \mathcal{C}_j = 0, \text{ for } j = 1, 2.$$

(iv) Dual feasibility:

$$s_i \geq 0, \text{ and } r_i \geq 0, \text{ for } i = 1, 2, \ldots, n.$$

Let solution $\mathbf{x}^* \in (\mathbb{R}^n_+)^2$ to problem (1.6) be strictly positive (each entry of the vector is strictly positive). From Equation (1.10), we obtain $s_i = 0$ and $r_i = 0$, for $i = 1, 2, \ldots, n$. Then the last term of Equation (2.13) is not active; thus, we obtain

$$\frac{\partial}{\partial x_{1i}} H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) = \frac{\partial}{\partial x_{1i}} f_i(x_{1i}, x_{2i})\Big|_{\mathbf{x}=\mathbf{x}^*} + \mu_1, \text{ for } i = 1, 2, \ldots, n. \tag{1.11}$$

And

$$\frac{\partial}{\partial x_{2i}} H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) = \frac{\partial}{\partial x_{2i}} f_i(x_{1i}, x_{2i})\Big|_{\mathbf{x}=\mathbf{x}^*} + \mu_2, \text{ for } i = 1, 2, \ldots, n. \tag{1.12}$$

Thus, for strictly positive optimal solution $\mathbf{x}^* \in (\mathbb{R}^n_+)^2$ of problem (1.6), the Karush-Kuhn-Tucker (KKT) conditions yield the following conditions for optimality:

$$\frac{\partial}{\partial x_{1i}} f_i(x_{1i}, x_{2i}) = \frac{\partial}{\partial x_{1u}} f_u(x_{1u}, x_{2u})\Big|_{\mathbf{x}=\mathbf{x}^*}, \text{ for } i, u = 1, 2, \ldots, n. \tag{1.13}$$

Analogously, we obtain

$$\frac{\partial}{\partial x_{2i}} f_i(x_{1i}, x_{2i}) = \frac{\partial}{\partial x_{2u}} f_u(x_{1u}, x_{2u})\Big|_{\mathbf{x}=\mathbf{x}^*} \text{ for } i, u = 1, 2, \ldots, n. \tag{1.14}$$

The partial derivatives of cost functions of all agents competing for a particular resource should be in consensus to achieve the necessary and sufficient conditions of optimality.

Thus, the KKT conditions are satisfied by the consensus of derivatives of the cost functions that are necessary and sufficient conditions of optimality of the optimization Problem (1.6); readers can find further details of the KKT conditions in Chapter 5.5.3 (Boyd & Vandenberghe, 2004). We use this principle to show that the proposed algorithms reach optimal values asymptotically. Note that for ease of readability of the chapters, we presented the optimality results there also.

1.5 Contributions

This thesis aims to develop distributed algorithms to solve multi-resource allocation problems with no inter-agent communication but a little with a central server that keeps track of the aggregate

demands. We now briefly present the contributions of this thesis.

1.5.1 Stochastic algorithm for distributed multi-resource allocation for divisible resources

We develop a distributed stochastic algorithm for divisible multi-resource allocation, described in Chapter 2. It achieves Objectives 1 and 3. The algorithm is based on the additive-increase multiplicative-decrease (AIMD) algorithm. Our contribution in this chapter is to extend the single resource allocation algorithm by Wirth and co-authors (Wirth et al., 2019) to a much broader and useful class of distributed optimization problems that involve multiple resources. The developed algorithm can be useful in smart cities, smart grids, edge computing, and other application domains. The algorithm uses probabilistic responses of agents that depend on average allocations of multiple divisible resources; the cost functions are coupled through the average allocations of these shared resources. Briefly, the developed algorithm works as follows: An agent continuously demands an increasing amount of a shared divisible resource until it receives a one-bit capacity event signal from the controller or central server; this phase is called the *additive increase* phase. The central server broadcasts the one-bit capacity event signal when the aggregate resource demand of agents reaches the capacity of the resource. After receiving this signal, agents respond in a probabilistic manner to reduce their demands; this phase is called the *multiplicative decrease* phase. We introduced a probabilistic way in which agents respond to the capacity events to solve a portfolio of optimization problems in a stochastic and distributed manner for multiple resources.

Furthermore, we present AIMD matrices for multiple resources. We model the system as a non-homogeneous Markov chain with place-dependent probabilities. Then we present the results on a deterministic system with fixed probabilities, and we present a perturbation analysis to develop the main result on almost sure convergence of the accumulative average (long-term average) allocation to the KKT point.

Additionally, we present the numerical results and compare the optimal values obtained by the developed algorithm and the optimization problem solved in a centralized way (by a solver). We observe that the results of the distributed algorithms are very close to the solver's optimal values. We also present an analysis on the number of iterations required for the convergence that depends on

the number of agents, their characteristic cost functions, additive-increase parameters. Furthermore, we compare the experimental results obtained by the single resource allocation algorithm and the proposed multi-resource allocation algorithm with separable and non-separable cost functions. We observe that for the separable cost function, the sum of the total costs obtained by the single resource algorithms and the total cost by the multi-resource algorithm converges to the same value. However, with non-separable cost functions, the results with the multi-resource allocation algorithms provide close to optimal values. In contrast, the single resource allocation algorithms provide suboptimal solutions.

1.5.2 Deterministic algorithm for distributed multi-resource allocation for divisible resources

We develop a deterministic AIMD based distributed divisible multi-resource allocation algorithm, described in Chapter 3. Wherein agents demand the resources in a deterministic manner to achieve a social optimum cost over the network. It achieves Objective 1. The algorithm is a derandomized version of the stochastic AIMD algorithm. Similar to the stochastic AIMD algorithm, the deterministic algorithm also has two phases: *Additive increase and multiplicative decrease*. The algorithm uses a deterministic approach, and the agents respond to the capacity event signal in a deterministic manner. In brief, the additive increase phase is the same as the stochastic AIMD algorithm, wherein an agent continuously demands an increasing amount of a shared divisible resource until it receives a one-bit capacity event signal from the central server. The central server broadcasts the one-bit capacity event signal when the aggregate resource demand reaches the capacity of the resource. After receiving this signal, that is, in the *multiplicative decrease* phase, agents multiplicatively decrease their demands in a deterministic way. The deterministic response depends on an agent's average resource allocation and the derivative of its cost functions. Moreover, the agent's cost function is coupled through its average allocations of multiple divisible shared resources.

In addition, we present a use case of a tourist attraction center, where we assume heterogeneous IoT devices, such as a set of wearable devices, surveillance cameras, etcetera. These IoT devices receive on-demand shared resources from the Cloudlets for task offloading. Such facilities can be helpful in realizing the potential of IoT devices in emerging applications such as assisting tourists

with a physical disability, uploading pictures of scenes on social media, getting detailed real-time information about a painting in a museum, real-time foreign language translation, etcetera. Finally, we present numerical results to verify the algorithm's efficacy; we also compare the results with the stochastic AIMD algorithm.

1.5.3 Stochastic algorithm for distributed multi-resource allocation for indivisible resources

We develop a stochastic algorithm for indivisible multi-resource allocation, described in Chapter 4. The cost functions of agents are coupled through the average allocations of multiple unit-demand (indivisible) shared resources. The algorithm extends the single unit-demand resource allocation algorithm by Griggs et al. (Griggs et al., 2016) to multiple unit-demand resources. The chapter achieves Objectives 1 and 3. Each agent in the algorithm demands the unit-demand (indivisible) shared resources in a probabilistic way based on its cost function and its average allocations. Moreover, each agent's consumption is modeled as a Bernoulli random variable. We consider a central agency (server) that keeps track of the aggregate demand of resources. Based on the aggregate demand and resource capacity, the central agency calculates and updates a feedback signal and broadcasts it in the network at each time step. After receiving this signal, agents respond in a probabilistic way whether to demand one unit of the resource or not. By doing so, the agents receive optimal allocations in the long-term averages.

In addition, we derive the results on convergence for networks with a single resource based on the classical results of stochastic approximation techniques. We extend the results to multiple resources. Finally, to check the efficacy of our algorithm, we present a use case for regulating the number of electric vehicles that share a limited number of level 1 and level 2 charging points.

1.5.4 Stochastic algorithm for regulating prosumers in a prosumer market

We develop distributed control algorithms to solve regulation problems with optimality constraints for the community-based prosumer market, described in Chapter 5. It achieves Objective 2. Recall that *prosumers* are agents that both produce and consume a resource. The algorithm is

based on ideas from stochastic approximation but formulated in a control-theoretic setting. The algorithm reaches optimality asymptotically while simultaneously regulating instantaneous capacity constraints. To do so, the algorithm does not require communication between prosumers but little communication with the sharing platform. The algorithm is suitable for situations where prosumer communities come together to purchase and sell related commodities. Such systems arise, for example, in energy systems where agents (prosumers) both produce and consume energy. Specifically, the distributed algorithm can be deployed on IoT platforms and can be used to support distributed community-wide buying and selling of resources. In sharing economy settings, a user may not know how many prosumers are participating in the sharing scheme, whether prosumers can or are willing to communicate with each other due to privacy considerations, and whether enough computational power is available to the whole network to allocate resources in real-time optimally. The developed algorithm places only modest demands on infrastructure and can be used to implement complex policies in the face of the uncertainties mentioned above.

Additionally, we present two use cases—community-based car-sharing and collaborative energy storage for prosumer markets. Finally, we present numerical results to check the efficacy of the algorithms.

Notice that because our models consider a central server that keeps track of aggregate demands and sends feedback signals in the network, the system may fail if the central server stops working. However, we can fix this by adding a backup server that keeps all the information as the central server and receives feedback signals from the central server. When the central server stops sending the feedback signals, the backup server takes charge and works as the central server until the central server comes live. As described in federated optimization Section 1.4.3, such settings are explored extensively by the machine learning community as federated learning. Wherein several agents collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020). Because all the algorithms involve a central server that keeps track

of aggregate demands and broadcasts feedback signals, for clarity, we restate the above statement in each chapter in the thesis.

1.6 Contribution as publications

(i) S. E. Alam, R. Shorten, F. Wirth, and J. Y. Yu, "Distributed algorithms for Internet-of-things enabled prosumer markets: A control-theoretic perspective," Analytics for the Sharing Economy: Mathematics, Engineering and Business perspectives, Springer, pages 125–149, March 2020, ISBN-978-3030350314. Chapter 5 is based on this work.

(ii) S. E. Alam, R. Shorten, F. Wirth, and J. Y. Yu, "Derandomized distributed multi-resource allocation with little communication overhead," Allerton Conference on Communication, Control, and Computing, Urbana, pp. 84–91, October 2018. Chapter 3 is based on this work.

(iii) S. E. Alam, R. Shorten, F. Wirth, and J. Y. Yu, "Communication-efficient distributed multi-resource allocation," IEEE International Smart Cities Conference (ISC2 2018), Kansas City, pp. 1–8, September 2018, (finalist for the best paper award). A part of Chapter 2 is based on this work.

Working/under-review articles:

(i) S. E. Alam, F. Wirth, J. Y. Yu, R. Ghosh, J. Marecek, and R. Shorten, "Multi-resource allocation for federated settings: A non-homogeneous Markov chain model," September 2021, (a journal article, we plan to submit it to Automatica). A part of Chapter 2 is based on this work.

(ii) S. E. Alam, R. Shorten, F. Wirth, and J. Y. Yu, "On the control of agents coupled through shared unit-demand resources," arXiv:1803.10386 [cs.SY], October 2019, (we plan to submit it to IEEE Systems journal or IEEE Transactions on Control of Network Systems). Chapter 4 is based on this work.

(iii) S. E. Alam, J. Y. Yu, R. Shorten, and S. Rao, "Local differential privacy for distributed optimization", November 2021, (we plan to submit it to IEEE Access).

(iv) "On unique ergodicity of coupled AIMD flows," with F. Wirth, J. Marecek, J. Y. Yu, P. Ferraro, R. Ghosh, R. Shorten, and *S. E. Alam*, 2021, (a journal article, co-authors are listed in the order of first names).

(v) S. E. Alam, F. Wirth, J. Y. Yu, and R. Shorten, The convergence of finite-averaging of AIMD for distributed heterogeneous resource allocations, arXiv:2001.08083 [math.OC], January, 2020.

1.7 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 presents the distributed stochastic algorithm for divisible multi-resource allocation. The algorithm is based on the additive-increase multiplicative-decrease (AIMD) algorithm. Next, Chapter 3 introduces distributed de-randomized algorithm for divisible multi-resource allocation. Again, the algorithm is based on the AIMD algorithm; however, agents demand the resources in a deterministic manner to achieve a social optimum cost for the network. Furthermore, Chapter 4 presents a distributed stochastic algorithm for multiple indivisible resource allocations. Chapter 5 presents stochastic distributed algorithms for community-based prosumer markets. Finally, Chapter 6 concludes the thesis and also discusses the future research directions. The thesis structure is illustrated in Figure 1.3.

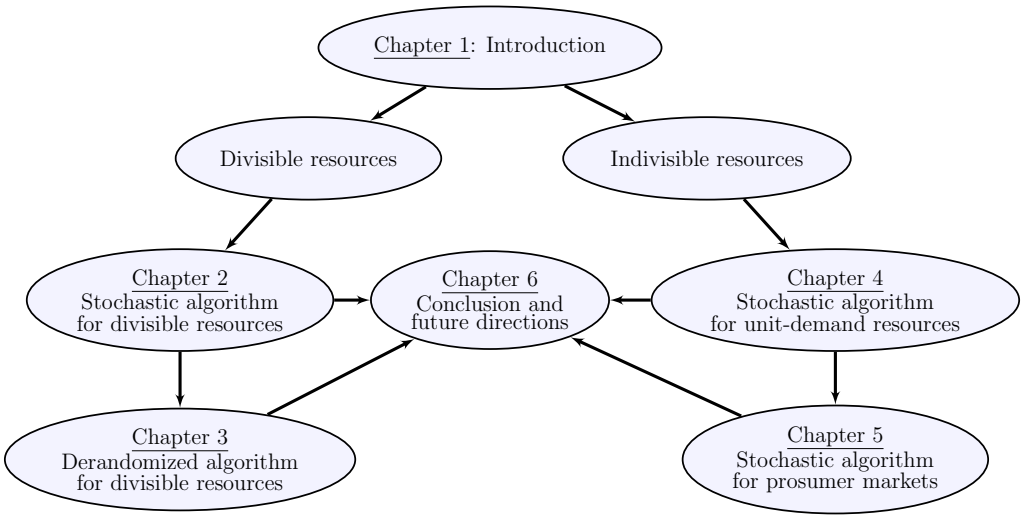


Figure 1.3: Structure of the thesis.

Chapter 2

Stochastic Distributed Algorithm for Divisible Multi-resource Allocation

In this chapter, we extend the AIMD-based single resource allocation algorithm by Wirth and co-authors (Wirth et al., 2019) to multi-resource allocation. The chapter addresses Objectives 1 and 3. Wirth and his co-authors describe how the basic *additive-increase multiplicative-decrease* (AIMD) algorithm can be modified in a straightforward manner to solve a class of distributed optimization problems for a single shared resource with no inter-agent communication. The AIMD algorithm is one of the most successful distributed resource allocation algorithms currently deployed in practice. It is best known as the backbone of the Internet and is also widely explored in other application areas. We extend the single-resource algorithm to multiple heterogeneous shared resources that emerge in smart cities, sharing economy, and many other applications. Our main results show the convergence of the average allocations to the optimal values. We model the system as a non-homogeneous Markov chain with place-dependent probabilities. Furthermore, numerical results are presented to demonstrate the efficacy of the algorithms and to highlight the main features of our analysis.

2.1 Introduction

Smart cities are built on smart infrastructures like intelligent transportation systems, smart security systems, smart grids, smart hospitals, smart waste management systems, etc., (Harrison et al., 2010; Zanella, Bui, Castellani, Vangelista, & Zorzi, 2014), (Alam, Shorten, Wirth, & Yu, 2018a). Internet of things (IoT) are the essential building blocks to develop such smart infrastructures (Hernandez-Munoz et al., 2011; Mohanty, Choppali, & Kougianos, 2016), we call these devices as *Internet of things (IoT)* devices. In this chapter, we use the words agents and IoT-devices interchangeably. In several smart city applications, multiple shared resources must be allocated among agents that are coupled through multiple resources. Generally speaking, such problems are more difficult to solve than those with a single resource. This is particularly true when agents are constrained—either through limitations of communication infrastructure or due to privacy considerations. The recent literature is rich with algorithms that are designed for distributed control and optimization applications. While this body of work is too numerous to enumerate, we point the interested readers to (Pu et al., 2021), (Uribe et al., 2021), (Kia, Cortes, & Martinez, 2015), (Tallapragada & Cortes, 2020), (J. Zhang et al., 2019), (S. Han, Topcu, & Pappas, 2017), (Wirth et al., 2019), for recent contributions. Interested readers can also refer to survey articles (Nedic et al., 2018), (Yang et al., 2019) and the papers cited therein.

In many instances in smart cities and other areas, a network of agents achieve optimal allocation of resources through regular communication with each other and/or with a central agent. Motivated by such scenarios, we develop an algorithm that is tailored for these but does not require inter-agent communication due to privacy considerations. The developed solution is based on the generalization of stochastic *additive-increase and multiplicative-decrease (AIMD)* algorithm (Wirth et al., 2019). By way of background, the AIMD algorithm was developed in the context of congestion avoidance in transmission control protocol (TCP) (Chiu & Jain, 1989). The AIMD algorithm is further explored and used in several application domains for example, (Leung et al., 2021), micro-grids (Crisostomi, Liu, Raugi, & Shorten, 2014), multimedia (Cai, Shen, Pan, & Mark, 2005), electric vehicle (EV) charging (Studli, Crisostomi, Middleton, & Shorten, 2012), (Shah et al., 2019), resource allocation (Avrachenkov et al., 2017), etc. Interested readers can refer to the book (Corless et al.,

2016) for an overview of some of the applications. The authors of (Wirth et al., 2019) demonstrate that simple algorithms from Internet congestion control can be used to solve certain optimization problems. Wirth et al. (Wirth et al., 2019) modified the basic AIMD algorithm to solve a class of optimization problems for a single shared resource with no inter-agent communication but a little with a central agent; the central agent keeps track of the aggregate allocations.

Recently, such settings have been extensively studied by the machine learning community as federated learning. *Federated learning* is a distributed machine learning technique in which several agents (clients) collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server or central agent. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020).

Roughly speaking, in (Wirth et al., 2019), the iterative distributed optimization algorithm works as follows. Agents continuously acquire an increasing share of the shared resource; this phase is called the *additive increase* phase. When the aggregate resource demand of agents exceeds the total capacity of the resource, then the central agent broadcasts a one-bit *capacity event* notification to all competing agents. The agents respond in a probabilistic manner to reduce the demand; this phase is called the *multiplicative decrease* phase. By judiciously selecting the probabilistic manner in which agents respond, a portfolio of optimization problems can be solved in a stochastic and distributed way.

Our contribution in this chapter is to demonstrate that the ideas of the single resource allocation of (Wirth et al., 2019) extend to a much broader and more useful class of optimization problems that can be used in many application domains of smart cities and other areas. Our algorithm builds on the choice of probabilistic response strategies described therein but is different in the sense that we generalize the approach to deal with *multiple resource constraints* and the cost functions are coupled through multiple resources. Furthermore, we show that the optimal value obtained by the algorithm is the same as if the optimization problem is solved in a centralized way. We model the system as a non-homogeneous Markov chain with place-dependent probabilities. Our algorithm for

multiple heterogeneous resources allows for the almost sure convergence to optimal allocations. For ease of discussion, we first present the single resource allocation model of Wirth et al. (Wirth et al., 2019) followed by the model for two resources in the following subsection.

2.1.1 Optimzation problem formulations

Consider a network of n agents that collaborate to access a single shared resource. Agent i has a state $x_i \geq 0$, for $i = 1, 2, \ldots, n$, representing the amount of the allocated resource. The allocations are updated at discrete time instants $t_0 < t_1 < t_2 < \ldots$. The agents keep track of their average allocations

$$\overline{x}_i(t_k) \triangleq \frac{1}{k+1} \sum_{\ell=0}^{k} x_i(t_\ell)\,. \tag{2.1}$$

For convenience, we index time instants t_k using $k = 0, 1, 2, \ldots$; for example, we denote $\overline{x}_i(t_k)$ by $\overline{x}_i(k)$ and $x_i(t_k)$ by $x_i(k)$. We assume that the resource has capacity $\mathcal{C} \geq 0$, referred to as a *capacity constraint*. We also assume that agent i has a continuously differentiable cost function $f_i : [0, \mathcal{C}] \to \mathbb{R}_+$, for $i = 1, 2, \ldots, n$. In this simple model, the problem of network-wide optimal allocation is stated as

$$\begin{aligned} \min_{x_1,\ldots,x_n} \quad & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = \mathcal{C}, \quad x_i \geq 0, \text{ for } i = 1, \ldots, n. \end{aligned} \tag{2.2}$$

Under strict convexity assumptions, this optimization problem is known to have a unique optimal point $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*) \in \mathrm{ri}\Sigma$, and we are looking for a distributed algorithm where the average values $\overline{x}_i(k)$ converge to this point, that is

$$\lim_{k\to\infty} \overline{x}_i(k) = x_i^*, \quad \text{for } i = 1, \ldots, n.$$

Wirth et al. (Wirth et al., 2019) show that this convergence can be achieved using the AIMD algorithm.

Let us now consider a more complicated example, where n agents collaborate to access two shared resources. Let $\mathcal{C}_1 \geq 0$ and $\mathcal{C}_2 \geq 0$ be the capacities of resource 1 and resource 2, respectively. Furthermore, let $x_{1i}(k) \in [0, \mathcal{C}_1]$ denote the amount of resource 1 allocated to agent i at time instant k; analogously, $x_{2i}(k) \in [0, \mathcal{C}_2]$ is defined, for $i = 1, 2, \ldots, n$. Let the cost function $f_i : \mathbb{R}_+^2 \to \mathbb{R}_+$ be continuously differentiable, for $i = 1, 2, \ldots, n$. We wish to solve the following optimization problem in a distributed way:

$$\begin{aligned} \min_{x_{11},\ldots,x_{1n},x_{21},\ldots,x_{2n}} \quad & \sum_{i=1}^{n} f_i(x_{1i}, x_{2i}), \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{1i} = \mathcal{C}_1, \\ & \sum_{i=1}^{n} x_{2i} = \mathcal{C}_2, \\ & x_{1i} \geq 0, \quad x_{2i} \geq 0, \text{ for } i = 1, \ldots, n. \end{aligned} \tag{2.3}$$

To formulate this as a long-term requirement, we define the time-averaged allocation $\overline{x}_{1i}(k)$ of resource 1 as in (2.1); analogously, we define $\overline{x}_{2i}(k)$ of resource 2. Let $x_{1i}^*, x_{2i}^* > 0$ be the optimal point, for $i = 1, \ldots, n$. Under strict convexity assumptions, we wish to develop an iterative and distributed algorithm determining values of $x_{1i}(k)$ and $x_{2i}(k)$ at each time instant k such that for the long-term (accumulative) averages, for $i = 1, 2, \ldots, n$, we obtain:

$$\lim_{k\to\infty} \overline{x}_{1i}(k) = x_{1i}^*; \quad \lim_{k\to\infty} \overline{x}_{2i}(k) = x_{2i}^*. \tag{2.4}$$

The AIMD based solution to these problems is practically important for several reasons. First, agents are not required to reveal their cost functions or their current allocations of the resource. Second, inter-agent communication is not necessary. Finally, agents can infer that the capacity constraint is violated by information made available intermittently by an entity that monitors the overall resource consumption; we call this entity the *central agent*. Note that we use *central agent*, *control unit*, or *central server* interchangeably. Together, these properties make AIMD based optimization an extremely useful tool to realize a distributed optimization algorithm for problems in which the decision variables are coordinate-wise strictly positive.

2.1.2 Contributions and the structure of the chapter

In Section 2.2, we introduce the AIMD algorithms and briefly describe AIMD based optimization for a single resource and two resources. Then in Section 2.3, we present the AIMD matrix model for a single divisible resource and introduce a novel matrix model for multiple heterogeneous divisible resources in Subsection 2.3.2. In Section 2.4, we present the convergence results on accumulative averaging of allocations, we also call it *long-term average allocations*. In this section, we model the system as a non-homogeneous Markov chain with place-dependent probabilities. Furthermore, we present the results on an approximate system with fixed probabilities; using these results, we present the perturbed version of the system to develop the main result on almost sure convergence of the accumulative average to a unique fixed point. Section 2.5 corroborates our analysis with numerical results. Finally, Section 2.6 concludes the chapter.

2.2 AIMD based optimization

We present a brief overview of the AIMD algorithm that is necessary for the discussion in this chapter. The AIMD algorithm is described in great detail in many publications (Wirth et al., 2019), (Corless et al., 2016).

2.2.1 AIMD based optimization for a single resource

In the AIMD algorithm, agents probe for an available resource at a rate that increases linearly by $\alpha \in (0, \mathcal{C}]$, the *additive-increase parameter*. When the total requested resource demand exceeds the capacity, a signal is sent to all or some of the agents to request that agents reduce the amount of resource demanded. We call these signals *capacity events*. Agents then respond by scaling back their request by a multiplicative factor $\beta \in [0, 1)$, the *multiplicative-decrease parameter*. We assume that an agent updates its AIMD algorithm every φ seconds. We denote these time instances by t_ν, where $\nu \in \mathbb{N}$ indexes the ν'th time instant. For $i = 1, 2, \ldots, n$, let λ_i be a probabilistic rule determining whether or not an agent responds to a capacity event. By designing these functions, we

shall see that the AIMD algorithm can be re-purposed to implement different optimization policies. Moreover, a parameter Γ is broadcast by the central agent to all agents in the network which ensures that $0 \leq \lambda_i \leq 1$ (defined in (2.9)).

The intuition underpinning AIMD based optimization is straightforward and is described in (Wirth et al., 2019), (Corless et al., 2016). For convenience, we describe the basic ideas here again.

Lemma 2.2.1. *(Wirth et al., 2019, Lemma 3.1) For $i = 1, 2, \ldots, n$, let $f_i : [0, \mathbb{C}] \to \mathbb{R}_+$ be strictly convex and continuously differentiable, let f_i' denote the derivative of f_i. Let there exists a constant $\Gamma > 0$ such that for all $x_i \in [0, \mathcal{C}]$ and $i = 1, 2, \ldots, n$, we have*

$$0 \leq \Gamma \frac{f_i'(x_i)}{x_i} \leq 1. \tag{2.5}$$

For $x_i = 0$, Equation (2.5) holds for the continuous extension of $\frac{f_i'(x_i)}{x_i}$ at $x_i = 0$, $i = 1, 2, \ldots, n$. Then

(i) There exists a unique optimal point $\mathbf{x}^* = (x_1^*, \ldots, x_n^*) \in \mathbb{R}_+^n$ for the optimization problem (2.2).

(ii) The optimal point x_i^* is strictly positive, for $i = 1, 2, \ldots, n$.

□

The following is an excerpt from (Wirth et al., 2019). Let $\mathbf{x} = (x_1, \ldots, x_n)$. We introduce the Lagrange multiplier $\mu \in \mathbb{R}$ and consider the Lagrangian of (2.2) as:

$$H(\mathbf{x}, \mu) = \sum_{i=1}^{n} f_i(x_i) - \mu \left(\sum_{i=1}^{n} x_i - C \right). \tag{2.6}$$

Under the assumption of strict convexity, from the Karush-Kuhn-Tucker (KKT) conditions (Boyd & Vandenberghe, 2004, Section 5.5.3), the necessary and sufficient conditions for optimality are obtained by setting all partial derivatives to zero. As the optimal point $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ has strictly positive entries; thus, the inequality constraints x_i are not active. So under the assumption $\mu^* \in \mathbb{R}$ and strict positivity of the optimal point $\mathbf{x}^*$, we have

$$\mu^* = f_i'(x_i)\big|_{\mathbf{x}=\mathbf{x}^*}, \quad \text{for } i = 1, \ldots, n. \tag{2.7}$$

In other words, the system is at optimality when the derivatives of the cost functions are in consensus. For Markovian models of the AIMD, weak assumptions guarantee ergodicity (Shorten, King, Wirth, & Leith, 2007). It is then known that almost surely, we have

$$\lim_{k\to\infty} \frac{1}{k+1} \sum_{\ell=0}^{k} x_i(\ell) = \frac{\Theta}{\lambda_i}, \tag{2.8}$$

where this average is calculated over all capacity events; moreover, Θ is a network-specific constant, and λ_i is the steady-state probability that the i'th agent responds to a notification of a capacity event. Thus, we can now aim to choose the probability λ_i so that the steady-state behavior (2.8) is equivalent to the KKT condition (2.7). With this in mind, suppose each agent responds to a capacity event with probability

$$\lambda_i(\overline{x}_i(k)) \;=\; \Gamma\, \frac{f_i'(\overline{x}_i(k))}{\overline{x}_i(k)}, \tag{2.9}$$

where k denotes the k'th capacity event.

Here, Γ is a network-wide constant chosen to ensure that $0 < \lambda_i \leq 1$. Suppose now that for large k, we have $\overline{x}_i(k) \approx x_i^*$. Then we obtain $\lambda_i(\overline{x}_i(k)) \approx \lambda_i^*$. Provided that for this choice, (2.8) holds, we obtain

$$\lambda_i(\overline{x}_i(k)) \;\approx\; \Gamma \frac{f_i'(\overline{x}_i(k))}{\Theta} \lambda_i(\overline{x}_i(k)), \tag{2.10}$$

and so $f_i'(\overline{x}_i(k)) \approx \Theta/\Gamma \approx f_u'(\overline{x}_u(k))$, for $i, u = 1, 2, \ldots, n$ and large k.

Let $S(k) \in \{0, 1\}$ denote the capacity event signal. When a capacity event occurs then it is updated with $S(k) = 1$. The algorithm of a central agent is presented in Algorithm 1. Additionally,

the algorithm for agent i for a single resource is presented in Algorithm 2.

Algorithm 1: Algorithm of central agent

Input: Capacity $\mathcal{C}$.

Output: Capacity event signal $S(k)$, for $k \in \mathbb{N}$. Broadcast Γ for the agents in the network. It is used by agent i to ensure that the response probability $0 \leq \lambda_i \leq 1$, for all i.

Initialization: $S(0) \leftarrow 0$, the counter for capacity event $k \leftarrow 0$;

while *time step* $\nu \in \mathbb{N}$ **do**

  if $\sum_{i=1}^{n} x_i(\nu) \geq \mathcal{C}$ **then**

    $k \leftarrow k + 1$;

    $S(k) \leftarrow 1$;

    broadcast $S(k)$;

  end

end

Algorithm 2: The AIMD algorithm to demand a single shared resource.

Input: Capacity event signal $S(k)$ and Γ received from the central agent.

Initialization: Agent i sets its state $x_i(0)$.We can initialize $x_i(0)$ with any value in $[0, \mathcal{C}]$ such that $\sum_{i=1}^{n} x_i(0) \leq \mathcal{C}$;

while *time step* $\nu \in \mathbb{N}$ **do**

  if $S(k) = 1$ **then**

    $$x_i(\nu + 1) \leftarrow \begin{cases} \beta x_i(\nu) & \text{w. p. } \lambda_i(\overline{x}_i(k)) \\ x_i(\nu) & \text{w. p. } 1 - \lambda_i(\overline{x}_i(k)) \end{cases} ;$$

  else

    $x_i(\nu + 1) \leftarrow x_i(\nu) + \alpha$;

  end

end

2.2.2 AIMD-based optimization for multiple resources

For the sake of exposition, we consider two resources here. Moreover, we state the following result on strict positivity of the optimal point.

Lemma 2.2.2. *For $i = 1, 2, \ldots, n$, let $f_i : \mathbb{R}_+^2 \to \mathbb{R}_+$ be strictly convex and continuously differentiable, let $\frac{\partial}{\partial_{x_{1i}}} f_i(x_{1i}, x_{2i})$, $\frac{\partial}{\partial_{x_{2i}}} f_i(x_{1i}, x_{2i})$ be the partial derivatives of $f_i(x_{1i}, x_{2i})$. Let there exists constants $\Gamma_1, \Gamma_2 > 0$ such that for all $x_{1i} \in [0, \mathcal{C}_1]$ and $x_{2i} \in [0, \mathcal{C}_2]$, $i = 1, 2, \ldots, n$, we have*

$$0 \leq \Gamma_1 \frac{\frac{\partial}{\partial_{x_{1i}}} f_i(x_{1i}, x_{2i})}{x_{1i}} \leq 1. \tag{2.11}$$

And

$$0 \leq \Gamma_2 \frac{\frac{\partial}{\partial_{x_{2i}}} f_i(x_{1i}, x_{2i})}{x_{2i}} \leq 1. \tag{2.12}$$

For $x_{1i} = 0$, Equation (2.11) holds for the continuous extension of $\frac{\frac{\partial}{\partial_{x_{1i}}} f_i(x_{1i}, x_{2i})}{x_{1i}}$ at $x_{1i} = 0$, and for $x_{2i} = 0$, Equation (2.12) holds for the continuous extension of $\frac{\frac{\partial}{\partial_{x_{2i}}} f_i(x_{1i}, x_{2i})}{x_{2i}}$ at $x_{2i} = 0$, $i = 1, 2, \ldots, n$. Then

(i) There exists a unique optimal point $\mathbf{x}^* = (x_{11}^*, \ldots, x_{2n}^*) \in (\mathbb{R}_+^n)^2$ for the optimization problem (2.3).

(ii) The optimal point x_{ji}^* is strictly positive, for $i = 1, 2, \ldots, n$, $j = 1, 2$.

□

Proof. It is similar to Lemma 3.1 of (Wirth et al., 2019). □

Let $\mathbf{x}_1 = (x_{11}, \ldots, x_{1n}) \in \mathbb{R}_+^n$ and $\mathbf{x}_2 = (x_{21}, \ldots, x_{2n}) \in \mathbb{R}_+^n$. Let the vectors $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{R}^2$, and $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{R}^n$ and $\mathbf{r} = (r_1, r_2, \ldots, r_n) \in \mathbb{R}^n$ be Lagrange multipliers of optimization problem (2.3) corresponding to the equalities and the inequalities, respectively. We

obtain the Lagrangian of (2.3) as follows,

$$H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) = \sum_{i=1}^{n} f_i(x_{1i}, x_{2i}) + \mu_1 \sum_{i=1}^{n} (x_{1i} - \mathcal{C}_1) + \mu_2 \sum_{i=1}^{n} (x_{2i} - \mathcal{C}_2) - \sum_{i=1}^{n} s_i x_{1i} - \sum_{i=1}^{n} r_i x_{2i}. \tag{2.13}$$

For strictly positive $\mathbf{x}_1^* = (x_{11}^*, \ldots, x_{1n}^*), \mathbf{x}_2^* = (x_{21}^*, \ldots, x_{2n}^*)$, the KKT conditions yield the following conditions for optimality:

$$\frac{\partial}{\partial x_{1i}} f_i(x_{1i}, x_{2i}) = \frac{\partial}{\partial x_{1u}} f_u(x_{1u}, x_{2u})\big|_{\mathbf{x}_1 = \mathbf{x}_1^*, \mathbf{x}_2 = \mathbf{x}_2^*} \text{ for } i, u = 1, 2, \ldots, n. \tag{2.14}$$

Analogously, we obtain

$$\frac{\partial}{\partial x_{2i}} f_i(x_{1i}, x_{2i}) = \frac{\partial}{\partial x_{2u}} f_u(x_{1u}, x_{2u})\big|_{\mathbf{x}_1 = \mathbf{x}_1^*, \mathbf{x}_2 = \mathbf{x}_2^*}, \text{ for } i, u = 1, 2, \ldots, n. \tag{2.15}$$

Thus, the partial derivatives of cost functions of all agents competing for a particular resource should be in consensus to achieve the necessary and sufficient conditions of optimality. For ease of readability of the chapter, we presented the optimality results; it is also found in Chapter 1.4.5.

To realize an algorithm to solve this optimization problem, we shall again use the AIMD algorithm. This time, in contrast to the single resource case, we have two sets of capacity events, one associated with resource 1 and another with resource 2. We index these events $k_1 = 1, 2, \ldots$, and $k_2 = 1, 2, \ldots$, respectively. We also denote the times associated with these events as t_{k_1} and t_{k_2}. As before, each agent keeps track of the allocation of each resource averaged over the respective set of capacity events, denoted by $\overline{x}_{1i}(k_1)$ up to capacity event k_1 of resource 1; analogously, by $\overline{x}_{2i}(k_2)$ up to capacity event k_2 of resource 2.

Similar to (2.1), the average $\overline{x}_{1i}(k_1)$ up to capacity event k_1 is defined as:

$$\overline{x}_{1i}(k_1) \triangleq \frac{1}{k_1 + 1} \sum_{\ell=0}^{k_1} x_{1i}(\ell). \tag{2.16}$$

In the sequel, $x_{1i}(\ell)$ means evaluation of resource 1 at the ℓ'th capacity event. Analogously, we define $\overline{x}_{2i}(k_2)$. Given this setting, we shall show that convergence to the optimum solution is guaranteed asymptotically, provided each agent responds to a capacity event as follows.

- Suppose that the k_1'th capacity event associated with the resource 1 is communicated to all agents. Then agent i responds to this notification by reducing the demand for resource 1, with probability:

$$\lambda_{1i}(t_{k_1}) = \Gamma_1 \frac{1}{\overline{x}_{1i}(t_{k_1})} \frac{\partial}{\partial x}\bigg|_{x=\overline{x}_{1i}(t_{k_1})} f_i(\overline{x}_{1i}(t_{k_1}), \overline{x}_{2i}(t_{k_1})), \tag{2.17}$$

  where $\overline{x}_{2i}(t_{k_1})$ denotes the average allocation of agent i over the capacity events of resource 2 up to time instant t_{k_1}.

- Suppose furthermore that the k_2'th capacity event associated with the resource 2 is communicated to all agents. Then agent i responds to this notification by reducing the demand for resource 2, with probability:

$$\lambda_{2i}(t_{k_2}) = \Gamma_2 \frac{1}{\overline{x}_{2i}(t_{k_2})} \frac{\partial}{\partial y}\bigg|_{y=\overline{x}_{2i}(t_{k_2})} f_i(\overline{x}_{1i}(t_{k_2}), \overline{x}_{2i}(t_{k_2})), \tag{2.18}$$

  where $\overline{x}_{1i}(t_{k_2})$ denotes the average allocation of agent i over the capacity events of resource 1 up to time instant t_{k_2}.

Notice that for notational simplicity, we drop $\overline{x}_{1i}$ and $\overline{x}_{2i}$ in the definitions of λ_{1i} and λ_{2i}. As before, Γ_1, Γ_2 are positive constants that are not dependent on network topology. They are chosen to ensure that the probabilities λ_{1i}, λ_{2i} are in the valid range; however, the smaller the values of Γ_1, Γ_2 are, the more number of iterations are required for the convergence.

Each agent runs its algorithm to demand resources. In the algorithm, we consider that each agent updates its resource demand at discrete time steps denoted by $\nu \in \mathbb{N}$. Let $S_1(k_1) \in \{0, 1\}$ and $S_2(k_2) \in \{0, 1\}$ denote the capacity event signals, for resource 1 and 2, respectively. When a

capacity event occurs then S_j is updated with $S_j(k_j) = 1, j = 1, 2$. The algorithm of a central agent is presented in Algorithm 3. Additionally, the algorithm for agent i for two resources is presented in Algorithm 4.

Algorithm 3: Algorithm of central agent

Input: Capacities $\mathcal{C}_1$ and $\mathcal{C}_2$.

Output: Capacity event signals $S_1(k_1)$ and $S_2(k_2)$, for $k_1, k_2 \in \mathbb{N}$. Broadcast Γ_1 and Γ_2 for the agents in the network. They are used by agent i to ensure that the response probabilities $0 \leq \lambda_{ji} \leq 1$, for all i and $j = 1, 2$.

Initialization: $S_1(0) \leftarrow 0$ and $S_2(0) \leftarrow 0$, the counter for capacity events $k_1 \leftarrow 0$ and $k_2 \leftarrow 0$;

while *time step* $\nu \in \mathbb{N}$ **do**

  if $\sum_{i=1}^{n} x_{1i}(\nu) \geq \mathcal{C}_1$ **then**

    $k_1 \leftarrow k_1 + 1$;

    $S_1(k_1) \leftarrow 1$;

    broadcast $S_1(k_1)$;

  end

  if $\sum_{i=1}^{n} x_{2i}(\nu) \geq \mathcal{C}_2$ **then**

    $k_2 \leftarrow k_2 + 1$;

    $S_2(k_2) \leftarrow 1$;

    broadcast $S_2(k_2)$;

  end

end

Algorithm 4: The algorithm of agent i to demand two shared resources.

Input: Capacity event signals $S_1(k_1)$ and $S_2(k_2)$ and Γ_1 and Γ_2 received from the central agent.

Output: Resource demands $x_{1i}(\nu)$ and $x_{2i}(\nu)$, for $\nu \in \mathbb{N}$.

Initialization: Agent i sets its states $x_{1i}(0)$ and $x_{2i}(0)$. We can initialize $x_{ji}(0)$ with any value in $[0, \mathcal{C}_j]$ such that $\sum_{i=1}^{n} x_{ji}(0) \leq \mathcal{C}_j$, for $j = 1, 2$.

while *time step* $\nu \in \mathbb{N}$ **do**

  if $S_1(k_1) = 1$ **then**

$$x_{1i}(\nu+1) \leftarrow \begin{cases} \beta_1 x_{1i}(\nu) & \text{w. p. } \lambda_{1i}(k_1) \\ x_{1i}(\nu) & \text{w. p. } 1 - \lambda_{1i}(k_1) \end{cases} ;$$

  else

$$x_{1i}(\nu+1) \leftarrow x_{1i}(\nu) + \alpha_1 ;$$

  end

  if $S_2(k_2) = 1$ **then**

$$x_{2i}(\nu+1) \leftarrow \begin{cases} \beta_2 x_{2i}(\nu) & \text{w. p. } \lambda_{2i}(k_2) \\ x_{2i}(\nu) & \text{w. p. } 1 - \lambda_{2i}(k_2) \end{cases} ;$$

  else

$$x_{2i}(\nu+1) \leftarrow x_{2i}(\nu) + \alpha_2 ;$$

  end

end

Remark 2.2.3 (Communication overhead)**.** *For $\nu \in \mathbb{N}$ and $j = 1, 2, \ldots, m$, let the capacity event signal be denoted by $S_j(\nu) \in \{0, 1\}$. Moreover, if $\sum_{i=1}^{n} x_{ji}(\nu) \geq \mathcal{C}_j$ then $S_j(\nu) = 1$. Then, in a network with m resources, the communication overhead will be $\sum_{j=1}^{m} S_j(\nu)$ bits at $\nu \in \mathbb{N}$ time step. In the worst case scenario, the communication overhead will be m bits per time step, which is quite low. Additionally, the communication complexity is independent of the number of participating agents in the network.*

As we stated in Chapter 1, we restate that because our models consider a central server that keeps track of aggregate demands and sends feedback signals in the network, the system may fail if

the central server stops working. However, we can fix this by adding a backup server that keeps all the information as the central server and receives feedback signals from the central server. When the central server stops sending the feedback signals, the backup server takes charge and works as the central server until the central server comes live. Such settings are explored extensively by the machine learning community as federated learning. Wherein several agents collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020). For clarity, we restate it in each chapter in the thesis.

2.2.3 Notations and conventions

The vector space of real column vectors with n entries is denoted by $\mathbb{R}^n$ with elements $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^\top$, where $\mathbf{x}^\top$ denotes the transpose of $\mathbf{x}$. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we write $\mathbf{x} \gg (\geq) \mathbf{y}$ if $x_\nu > (\geq) y_\nu$ for all $\nu = 1, \dots, n$. The positive orthant is $\mathbb{R}^n_+ \triangleq \{\mathbf{x} \in \mathbb{R}^n; \mathbf{x} \geq 0\}$. We denote $\mathbf{e} := [1 \quad 1 \quad \dots \quad 1]^\top \in \mathbb{R}^n$ and the ℓ'th standard basis vector by $\mathbf{e}_\ell$. The Hadamard product (or componentwise product) on $\mathbb{R}^n$ is defined as

$$\mathbf{x} \odot \mathbf{y} \triangleq \begin{bmatrix} x_1 y_1 & \dots & x_n y_n \end{bmatrix}^\top, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \tag{2.19}$$

Moreover, the space of $n \times n$ matrices is denoted by $\mathbb{R}^{n \times n}$, and $\mathbb{R}^{n \times n}_+$ is the set of non-negative matrices. The identity matrix is denoted by $I \in \mathbb{R}^{n \times n}$, and to specify the dimension we denote it by $I^a \in \mathbb{R}^{a \times a}$, $a \in \mathbb{N}$. For $X, Y \in \mathbb{R}^{n \times n}$, the diagonal matrix $\begin{bmatrix} X & 0; & 0 & Y \end{bmatrix}$ is denoted by $\operatorname{diag}(X, Y)$. We denote the product of finite number of matrices $Y \in \mathbb{R}^{n \times n}$ by Π_Y. The convex hull of a set $X \subset \mathbb{R}^n$ is denoted by $\operatorname{conv} X$; it may be defined as the smallest convex set containing X. The norm we use on $\mathbb{R}^n$ is the 1-norm, defined by $\|\mathbf{x}\|_1 \triangleq \sum_{i=1}^n |x_i|$. For a non-empty set

$X \subset \mathbb{R}^n$, the distance of a point v to X with respect to the 1-norm is defined as

$$d_1(v, X) \triangleq \inf \Big\{ \|v - x\|_1 : x \in X \Big\}. \tag{2.20}$$

Let the capacity of the resource be $\mathcal{C} = 1$. The standard simplex Σ in $\mathbb{R}^n$ is defined by

$$\Sigma \triangleq \left\{ \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n_+ \mid \sum_{i=1}^{n} x_i = 1 \right\}. \tag{2.21}$$

We will write Σ_n if we want to make the dimension of the simplex explicit. The relative interior, see (Rockafellar, 2015), of Σ is given by $\mathrm{ri}\Sigma = \{\mathbf{x} \in \Sigma : \mathbf{x} \gg 0\}$. The relative interior becomes a complete metric space, if it is endowed with the Hilbert metric d_H, see (Hartfiel, 2002), defined by

$$d_H(\mathbf{x}, \mathbf{y}) \triangleq \max_{i=1,\ldots,n} \log \left[\frac{x_i}{y_i} \right] - \min_{u=1,\ldots,n} \log \left[\frac{x_u}{y_u} \right], \quad \mathbf{x}, \mathbf{y} \in \mathrm{ri}\Sigma. \tag{2.22}$$

Associated with this metric, it is sometimes convenient to consider

$$\exp(d_H(\mathbf{x}, \mathbf{y})) \triangleq \frac{\max\limits_{i=1,\ldots,n} \left[\frac{x_i}{y_i} \right]}{\min\limits_{u=1,\ldots,n} \left[\frac{x_u}{y_u} \right]}, \quad \mathbf{x}, \mathbf{y} \in \mathrm{ri}\Sigma. \tag{2.23}$$

Note that $\log$ denotes the natural logarithm in the chapter. Moreover, we denote the closed 1-norm ball of radius δ and center 0 by $\overline{B}_1(0, \delta)$, and the closed ball with respect to the Hilbert metric by $B_H(\mathbf{x}, \delta)$, for $\mathbf{x} \in \Sigma$.

In the sequel, we will frequently deal with product spaces of the form $(\mathbb{R}^n)^m$, Σ^m, $(\mathrm{ri}\Sigma)^m$. The elements of such spaces are denoted by $\mathbf{z} = \left[\mathbf{z}_1{}^\top \ \mathbf{z}_2{}^\top \ \ldots \ \mathbf{z}_m{}^\top \right]^\top$ with $\mathbf{z}_\ell \in \mathbb{R}^n, \ell = 1, \ldots, m$, etc. Note the distinction between bold entries in a vector (which are then subvectors of equal size) and non-bold entries as in $\mathbf{x} = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$; in the latter case, the entries are real numbers. Note also that $\mathrm{ri}(\Sigma^m) = (\mathrm{ri}\Sigma)^m$.

Given our norm on $\mathbb{R}^n$, resp. the Hilbert metric on $\mathrm{ri}\Sigma$; for $\nu \in \mathbb{N}$, $j = 1, \ldots, m$, let $\mathbf{z}_j \in (\mathbb{R}^n)^\nu$ and $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_m) = (\mathbf{z}_{11}, \ldots, \mathbf{z}_{1\nu}, \ldots, \mathbf{z}_{m1}, \ldots, \mathbf{z}_{m\nu})$. We will use the product structure to

define a norm/metric on the product space via

$$\|\mathbf{z}_j\| \triangleq \max_{1\le\ell\le\nu} \|\mathbf{z}_{j\ell}\|_1 \text{, resp.} \quad D_H(\mathbf{x},\mathbf{y}) \triangleq \max_{1\le\ell\le m} d_H(\mathbf{x}_\ell,\mathbf{y}_\ell). \tag{2.24}$$

2.3 AIMD matrix model

Much of what follows is based on a matrix representation of AIMD network dynamics (Wirth et al., 2019), (Corless et al., 2016), and here we recall some important results from these references. While the following material is known, the discussion of the allocation of multiple heterogeneous resources introduces some notational issues. Thus, to aid our discussion, we introduce new notations to make the dependence on each resource clear.

2.3.1 AIMD matrix model for a single resource

We will first describe the evolution of the agents' allocations of one resource and thus drop the index j for the moment. There are two parameters associated with a resource, $\alpha \in (0, \mathcal{C}]$, the additive-increase parameter and $0 \le \beta < 1$, the multiplicative decrease parameter. At a capacity event, that is, at a time t_k at which the total demand equals the available capacity $\mathcal{C}$, agents reduce their demand to $x_i(t_k^+) = \beta x_i(t_k)$ or do not change their demands. The choice is determined for each agent in a probabilistic manner as described in (2.9). After this, each agent increases demand with the constant rate α until the next capacity event occurs at time t_{k+1}. We can compute the time from t_k to the next capacity event as

$$t_{k+1} - t_k = \frac{\mathcal{C} - \sum_{i=1}^n \boldsymbol{\beta}_i(k) x_i(t_k)}{n\alpha}, \tag{2.25}$$

where $\boldsymbol{\beta}_i(k) = \beta$ if agent i responds to the k'th capacity event, and $\boldsymbol{\beta}_i(k) = 1$ if it does not respond to the capacity event. The evolution of the capacity usage of agent i is then

$$x_i(t_{k+1}) = \boldsymbol{\beta}_i(k) x_i(t_k) + (t_{k+1} - t_k)\alpha. \tag{2.26}$$

Denoting $\boldsymbol{\beta}(k) := \begin{bmatrix} \boldsymbol{\beta}_1(k) & \dots & \boldsymbol{\beta}_n(k) \end{bmatrix}^\top$, and using elementary arguments (see (Corless et al., 2016)), it can be shown that the evolution from capacity event k to $k+1$ is linear and of the form

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k), \quad k \in \mathbb{N} \tag{2.27}$$

where

$$\mathbf{A}(k) := \text{diag } (\boldsymbol{\beta}(k)) + \frac{1}{n}\mathbf{e}\left(\mathbf{e}^\top - \boldsymbol{\beta}(k)^\top\right). \tag{2.28}$$

It follows that the matrix $\mathbf{A}(k) \in \mathbb{R}^{n\times n}$ is drawn from a finite set of column stochastic matrices determined by the manner in which agents respond to a capacity event. We denote this set of the AIMD matrices by $\mathcal{A}$. For n agents in the network, the set $\mathcal{A}$ consists of 2^n AIMD matrices. Thus, $\mathcal{A} \triangleq \{A_1, \dots, A_{2^n}\}$. Note that the set $\mathcal{A}$ includes the identity matrix, i.e. the possibility that no agent responds to an event. This is to simplify the discussion. From a practical point, this would just result in an immediate repeat of the capacity signal. The matrix corresponding to all agents responding to a capacity event plays a special role in our discussion. We denote this matrix by A_1, where:

$$A_1 = \text{diag } (\beta_1, \dots, \beta_n) + \frac{1}{n}\mathbf{e}\left(\mathbf{e}^\top - [\beta_1 \ \dots \ \beta_n]\right). \tag{2.29}$$

Clearly, in (2.28), we have $\mathbf{A}(k) \in \mathcal{A}$, for all capacity events k.

For the ease of reading, we now present a brief background on Perron eigenvalue and eigenvector. Let $X \in \mathbb{R}_+^{n\times n}$ be a strictly positive matrix then Perron–Frobenius theorem states that there exists a real and positive eigenvalue τ of X with strictly positive eigenvectors $\mathbf{w}_p$; that is, $X\mathbf{w}_p = \tau\mathbf{w}_p$. Furthermore, the magnitude of eigenvalue τ is the largest of all other eigenvalues of X. The eigenvalue τ is called *Perron eigenvalue* and the eigenvector is called *Perron eigenvector* (Corless et al., 2016). Moreover, if matrix X is strictly positive and column stochastic then Perron eigenvalue $\tau = 1$; thus, $X\mathbf{w}_p = \mathbf{w}_p$, where $\mathbf{e}^\top\mathbf{w}_p = 1$.

We now recall two elementary results from (Wirth et al., 2019). If the response of agents to capacity events is mutually independent and independent of the state, in an IID fashion, with response probabilities $0 < \lambda_i \leq 1$, $i = 1, \ldots, n$, then for all choices $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n)$, where $\boldsymbol{\beta}_i \in \{1, \beta\}$, we have

$$\mathbb{P}\left(\mathbf{A}(k) = \text{diag }(\boldsymbol{\beta}) + \frac{1}{n}\mathbf{e}\left(\mathbf{e}^\top - \boldsymbol{\beta}^\top\right)\right) = \prod_{\boldsymbol{\beta}_i=\beta} \lambda_i \prod_{\boldsymbol{\beta}_i=1} (1-\lambda_i) \triangleq p_\beta. \tag{2.30}$$

Moreover, the system in (2.27) with fixed probability in (2.30) constitute a Markov chain. For the fixed response probability $\lambda_i > 0$, for $i = 1, 2, \ldots, n$, (Wirth, Stanojevic, Shorten, & Leith, 2006), (Corless et al., 2016), (Wirth et al., 2019) obtained that there exists a unique, invariant measure on Σ for the Markov chain.

As $\mathbf{A}(k)$ and $\mathbf{x}(k)$ are independent, it follows that:

$$\mathbb{E}[\mathbf{x}(k+1)] = \mathbb{E}[\mathbf{A}(k)\mathbf{x}(k)] = \mathbb{E}[\mathbf{A}(k)]\mathbb{E}[\mathbf{x}(k)].$$

Here $\mathbb{E}[\mathbf{A}(k)]$ is a column stochastic matrix that under mild assumptions is strictly positive with right Perron eigenvector $\mathbf{w}_p$, given by

$$\mathbf{w}_p = \Big(\sum_{i=1}^{n} \frac{1}{\lambda_i}\Big)^{-1}\Big[\frac{1}{\lambda_1} \;\; \frac{1}{\lambda_2} \;\; \ldots \;\; \frac{1}{\lambda_n}\Big]^\top, \tag{2.31}$$

where $\mathbf{e}^\top \mathbf{w}_p = 1$.

2.3.2 AIMD matrix model for multiple resources

We now extend the matrix formulation of AIMD to the case of multiple heterogeneous resources. The index indicating resource type is $j = 1, 2, \ldots, m$. We assume independent AIMD algorithms for the resources, but eventually, we are interested in a coupling of the dynamics. This coupling will occur by the response probability of the agents, which will depend, for each agent, on the vector of its resource consumption.

To each resource, we associate AIMD parameters $\alpha_j, \beta_j, j = 1, \ldots, m$, and define the corresponding AIMD matrices, as in (2.28). For example, the set $\mathcal{A}_j$ denotes a set of AIMD matrices

associated with resource j, with $A_{j,q} \in \mathcal{A}_j$ denoting the q'th matrix in this set. Again, the matrix $A_{j,1} \in \mathcal{A}_j$ as in (2.29) is the matrix where all agents respond to a capacity event for resource j.

For the sake of simplicity, let us assume that there are two resources in the network, that is, $m = 2$. The following arguments easily extended to the case $m > 2$. We denote by $K = \{t_k | k \in \mathbb{N}\} \subset \mathbb{R}_+$ the ordered set of all capacity events and the subsets $K_j \subset K, j = 1, 2$, of capacity events associated to resource j. Define the index map

$$\phi : \mathbb{N} \to \{1, 2\}, \tag{2.32}$$

where $\phi(k) = j$, if $t_k \in K_j$. We note that once the agents' responses are fixed at a capacity event, then the time to the next capacity event is determined by (2.34). In particular, $\phi(k)$ is a random variable, which is also determined by the probabilities of choosing the matrices $A_{j,q}$.

For $k \in \mathbb{N}$ and $j \in \{1, 2\}$, we define the random variables counting the number of capacity events for each resource by

$$k_j := \#\{0 \leq \ell \leq k | \phi(\ell) = j\}. \tag{2.33}$$

For the sake of exposition, we state the following for resource j, $j = 1, 2, \ldots, m$. As in (2.25), for resource j, we obtain the time difference between the k_j'th capacity event and the $k_j + 1$'th capacity event, as follows

$$t_{k_j+1} - t_{k_j} = \frac{\mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j})}{n\alpha_j}, \tag{2.34}$$

where $\boldsymbol{\beta}_{ji}(k_j) = \beta_j$ if agent i responds to the k_j'th capacity event, and $\boldsymbol{\beta}_{ji}(k_j) = 1$ if it does not respond to the capacity event, for $j = 1, 2, \ldots, m$.

Furthermore, as in (2.26), for resource j, we obtain the instantaneous demands of agent i at the $k_j + 1$'th capacity event as follows

$$x_{ji}(t_{k_j+1}) = \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j}) + (t_{k_j+1} - t_{k_j})\alpha_j. \tag{2.35}$$

For resource j, denoting $\boldsymbol{\beta}_j(k_j) \triangleq \begin{bmatrix} \boldsymbol{\beta}_{j1}(k_j) & \ldots & \boldsymbol{\beta}_{jn}(k_j) \end{bmatrix}^\top$, we have the following AIMD

matrix (cf. (2.28)):

$$\mathbf{A}_j(k_j) = \text{diag }(\boldsymbol{\beta}_j(k_j)) + \frac{1}{n}\mathbf{e}\left(\mathbf{e}^\top - \boldsymbol{\beta}_j(k_j)^\top\right)$$
$$= \begin{bmatrix} \boldsymbol{\beta}_{j1}(k_j) & 0 & \dots & 0 \\ 0 & \boldsymbol{\beta}_{j2}(k_j) & 0 \dots & 0 \\ \vdots & & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & & 0 & \boldsymbol{\beta}_{jn}(k_j) \end{bmatrix} + \frac{1}{n}\mathbf{e}\begin{bmatrix} 1-\boldsymbol{\beta}_{j1}(k_j) & \dots & 1-\boldsymbol{\beta}_{jn}(k_j) \end{bmatrix}. \tag{2.36}$$

Now, to formulate the joint evolution of the resource allocation, we define

$$\mathbf{A}(k) \triangleq \begin{cases} \text{diag }(\mathbf{A}_1(k_1), I) & \text{if } \phi(k) = 1, \\ \text{diag }(I, \mathbf{A}_2(k_2)) & \text{if } \phi(k) = 2. \end{cases} \tag{2.37}$$

Note that $\mathbf{A}_j(k_j)$ in (2.37) denotes the AIMD matrix of resource j over resource j's capacity events up to time instant t_k, for $j = 1, 2$.

Thus, for $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \Sigma^2$, we have the following time-varying dynamics for the overall system:

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k). \tag{2.38}$$

We briefly discuss the easiest case, in which each agent responds to each capacity event with an independent choice according to fixed response probabilities $\lambda_{ji} > 0$, $i = 1, 2, \dots, n, j = 1, 2, \dots, m$.

We now extend the results of (2.30) and (2.31) obtained for the single resource to multiple resources. For resource $j = 1, 2, \dots, m$, we have $\mathbf{A}_j(k_j) \in \mathcal{A}_j$, for all capacity events k_j. Similar to the single resource case, if we consider that the response of agents to capacity events is mutually independent and independent of the state, in an IID fashion, with response probabilities $0 < \lambda_{ji} \leq 1$, $i = 1, \dots, n$ and $j = 1, 2, \dots, m$. Then for a fixed j, and for all choices $\boldsymbol{\beta}_j = (\boldsymbol{\beta}_{j1}, \dots, \boldsymbol{\beta}_{jn})$,

where $\boldsymbol{\beta}_{ji} \in \{1, \beta_j\}$, we obtain

$$\mathbb{P}\left(\mathbf{A}_j(k_j) = \text{diag}\,(\boldsymbol{\beta}_j) + \frac{1}{n}\mathbf{e}\left(\mathbf{e}^\top - \boldsymbol{\beta}_j^\top\right)\right) = \prod_{\boldsymbol{\beta}_{ji}=\beta_j} \lambda_{ji} \prod_{\boldsymbol{\beta}_{ji}=1} (1-\lambda_{ji}) \triangleq p_{\beta_j}. \tag{2.39}$$

Now, for a fixed j, $\mathbf{A}_j(k_j)$ and $\mathbf{x}_j(k_j)$ are independent and it follows that:

$$\mathbb{E}[\mathbf{x}_j(k_j+1)] = \mathbb{E}[\mathbf{A}_j(k_j)\mathbf{x}_j(k_j)] = \mathbb{E}[\mathbf{A}_j(k_j)]\mathbb{E}[\mathbf{x}_j(k_j)].$$

Furthermore, $\mathbb{E}[\mathbf{A}_j(k_j)]$ is a column stochastic matrix that under mild assumptions is strictly positive with right Perron eigenvector $\mathbf{w}_{p_j}$, given by

$$\mathbf{w}_{p_j} = \Big(\sum_{i=1}^{n} \frac{1}{\lambda_{ji}}\Big)^{-1}\Big[\frac{1}{\lambda_{j1}}\ \frac{1}{\lambda_{j2}}\ \cdots\ \frac{1}{\lambda_{jn}}\Big]^\top, \tag{2.40}$$

where $\mathbf{e}^\top \mathbf{w}_{p_j} = 1$.

Define

$$\overline{\mathbf{S}}_j(k_j) \triangleq \frac{1}{k_j+1}\left(\sum_{\ell=1}^{k_j} \mathbf{A}_j(\ell-1)\ldots\mathbf{A}_j(0) + I\right). \tag{2.41}$$

Moreover, (for $m=2$), let $\overline{\mathbf{S}}(k)$ denote the diagonal matrix:

$$\overline{\mathbf{S}}(k) \triangleq \text{diag}\,(\overline{\mathbf{S}}_1(k_1), \overline{\mathbf{S}}_2(k_2)). \tag{2.42}$$

Then, for $\mathbf{x}(0) \in \Sigma^2$, we obtain

$$\overline{\mathbf{x}}(k) = \overline{\mathbf{S}}(k)\mathbf{x}(0). \tag{2.43}$$

Let $\|\cdot\|_1$ be the max column sum norm which for $X \in \mathbb{R}^{n\times n}$ is easily seen to satisfy

$$\|X\|_1 = \max_{\mathbf{x}\in\Sigma} \|X\mathbf{x}\|_1 = \max_{\ell=1,2,\ldots,n} \|X\mathbf{e}_\ell\|_1\,. \tag{2.44}$$

Also, for $j = 1, 2, \ldots, m$, let $X_j \in \mathbb{R}^{n \times n}$, and $Z \triangleq \operatorname{diag}(X_1, \ldots, X_m)$, we have

$$\|Z\|_1 = \max_{j=1,2,\ldots,m} \|X_j\|_1 . \tag{2.45}$$

Lemma 2.3.1. *Let the probabilities $\boldsymbol{\lambda} = (\lambda_{11}, \ldots, \lambda_{mn}) \in ((0,1]^n)^m$ be fixed. Let $\overline{\mathbf{S}}(k)$ be given by (2.42) and the sequence of random variables $\{\mathbf{A}(k)\}_{k \in \mathbb{N}}$ be IID. Let $\mathbb{P}_{\boldsymbol{\lambda}}$ be the probability measure. Also, let $\mathbf{w}_{p_j}$ be right Perron eigenvector as defined in (2.40) for resource j. Moreover, for $m = 2$, let $\mathbf{w}_p \triangleq [\mathbf{w}_{p_1}^\top \ \mathbf{w}_{p_2}^\top]^\top$, and $\mathrm{w} \triangleq \operatorname{diag}\left(\mathbf{w}_{p_1}\mathbf{e}^\top, \mathbf{w}_{p_2}\mathbf{e}^\top\right)$. Then for every $\epsilon, \delta > 0$, there exists k_0 such that for all $k \geq k_0$, we have*

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(k) - \mathrm{w}\right\|_1 > \delta\right) < \epsilon. \tag{2.46}$$

□

Proof. The proof is similar to (Wirth et al., 2019, Lemma 2.2). For the Markov chain defined in (2.38), there exists a unique invariant measure on Σ^m. For a fixed j, from (Wirth et al., 2006), (Wirth et al., 2019), we know that $\lim_{k\to\infty} \overline{\mathbf{x}}_j(k) = \mathbf{w}_{p_j}$ almost surely. Furthermore, almost sure convergence implies convergence in probability. That is, for all $\delta > 0$ and $\mathbf{x}_j(0) \in \Sigma$, there exist k_0 and $\epsilon' > 0$ such that for all $k_j \geq k_0$, we have

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{x}}_j(k_j) - \mathbf{w}_{p_j}\right\|_1 > \delta\right) < \epsilon'. \tag{2.47}$$

For $\mathbf{x}_j(0) \in \Sigma$, as $\mathbf{e}^\top \mathbf{x}_j(0) = 1$, then from (2.43) and (2.47), we obtain

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\left(\overline{\mathbf{S}}_j(k_j) - \mathbf{w}_{p_j}\mathbf{e}^\top\right)\mathbf{x}_j(0)\right\|_1 > \delta\right) < \epsilon'.$$

Recall that for $\ell = 1, 2, \ldots, n$, $\mathbf{e}_\ell \in \mathbb{R}^n$ is the ℓ'th standard basis vector; now, let us choose $k_{j\ell}$ such that for all $k_j \geq k_{j\ell}$ and $\epsilon'_{j\ell} > 0$. For a fixed ϵ, let $\sum_{j=1}^m \sum_{\ell=1}^n \epsilon'_{j\ell} = \epsilon$. Thus, from (2.44), we obtain

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\left(\overline{\mathbf{S}}_j(k_j) - \mathbf{w}_{p_j}\mathbf{e}^\top\right)\mathbf{e}_\ell\right\|_1 > \delta\right) < \epsilon'_{j\ell}.$$

Notice that $\left\| \left(\overline{\mathbf{S}}_j(k_j) - \mathbf{w}_{p_j} \mathbf{e}^\top \right) \mathbf{e}_\ell \right\|_1$ is the 1-norm of the ℓ'th column of the matrix $\left(\overline{\mathbf{S}}_j(k_j) - \mathbf{w}_{p_j} \mathbf{e}^\top \right)$. For $\ell = 1, \ldots, mn$, let X_ℓ be an event; we know $\mathbb{P}(\cup_{\ell=1}^{mn} X_\ell) \leq \sum_{\ell=1}^{mn} \mathbb{P}(X_\ell)$.

Furthermore, let $k_0 = \max\limits_{j=1,\ldots,m;\ell=1,\ldots,n} k_{j\ell}$; then, for all $k \geq k_0$, we obtain:

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{\lambda}} \left(\left\| \overline{\mathbf{S}}(k) - \mathrm{w} \right\|_1 > \delta \right) &= \mathbb{P}_{\boldsymbol{\lambda}} \left(\max_{j=1,\ldots,m;\ell=1,\ldots,n} \left\| \left(\overline{\mathbf{S}}_j(k_j) - \mathbf{w}_{p_j} \mathbf{e}^\top \right) \mathbf{e}_\ell \right\|_1 > \delta \right) \\
&\leq \sum_{j=1}^{m} \sum_{\ell=1}^{n} \mathbb{P}_{\boldsymbol{\lambda}} \left(\left\| \left(\overline{\mathbf{S}}_j(k_j) - \mathbf{w}_{p_j} \mathbf{e}^\top \right) \mathbf{e}_\ell \right\|_1 > \delta \right) \\
&< \sum_{j=1}^{m} \sum_{\ell=1}^{n} \epsilon'_{j\ell} = \epsilon. \qquad \square
\end{aligned}$$

For simplicity of notation, for m resources in the network, we define

$$[\mathbf{x}]_i \triangleq (x_{1i}, \ldots, x_{mi}), \quad i = 1, 2, \ldots, n. \tag{2.48}$$

Lemma 2.3.1 considers the case of fixed probabilities. We now turn to the question of state-dependent probabilities and consider the map $\boldsymbol{\lambda} : \Sigma^m \to ([0,1]^n)^m$ (with component functions $\lambda_{ji}(\cdot)$) which associates to each state in Σ^m the probabilities with which agents respond to capacity events for the different resources. The following assumption will be crucial:

Assumption 2.3.2. *(i) The map $\boldsymbol{\lambda} : \Sigma^m \to ([0,1]^n)^m$ is continuous.*

(ii) The map $R : \Sigma^m \to ([0,1]^n)^m$, $R(\mathbf{x}) = \mathbf{x} \odot \boldsymbol{\lambda}(\mathbf{x})$ is the gradient of a strictly convex function $Q : \Sigma^m \to \mathbb{R}$.

(iii) There exist $\lambda_{j,\min} > 0$, $j = 1, \ldots, m$ such that $\lambda_{ji}(\cdot) \geq \lambda_{j,\min}$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

2.4 Convergence of accumulative averaging

In this section, we present the main result on the convergence of accumulative or long-term average allocations. The probabilities for choosing the AIMD matrices depend on the accumulative

averages of the realization. We model the system as a non-homogeneous Markov chain with place-dependent probabilities. To prove the convergence of accumulative average allocations, first we present the results for an approximate system with fixed probabilities; we call it the *deterministic system*, described in Section 2.4.1. Furthermore, a perturbed version of the system is presented in Section 2.4.2 that uses results on the deterministic system to prove the main theorem on almost sure convergence of the accumulative average to the unique fixed point. For $k \in \mathbb{N}$, let $\boldsymbol{\xi}(k)$ be the state vector for the resources, defined as

$$\boldsymbol{\xi}(k) \triangleq \left[\mathbf{x}_1(k)^\top \; \overline{\mathbf{x}}_1(k)^\top \; \ldots \; \mathbf{x}_m(k)^\top \quad \overline{\mathbf{x}}_m(k)^\top\right]^\top. \tag{2.49}$$

Note that $\mathbf{x}_j(k)$ denotes the state vector of resource j at time instant t_k, for $j = 1, 2, \ldots, m$. Similarly, $\overline{\mathbf{x}}_j(k)$ denotes the average allocation over the capacity events of resource j up to time instant t_k, for $j = 1, 2, \ldots, m$.

For a fixed j, we reformulate (2.16) as

$$\overline{\mathbf{x}}_j(k_j + 1) = \frac{1}{k_j + 2}\mathbf{x}_j(k_j + 1) + \frac{k_j + 1}{k_j + 2}\overline{\mathbf{x}}_j(k_j). \tag{2.50}$$

Thus, we obtain

$$\begin{bmatrix} \mathbf{x}_j(k_j + 1) \\ \overline{\mathbf{x}}_j(k_j + 1) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_j(k_j) & 0 \\ \frac{1}{k_j+2}\mathbf{A}_j(k_j) & \frac{k_j+1}{k_j+2}I \end{bmatrix} \begin{bmatrix} \mathbf{x}_j(k_j) \\ \overline{\mathbf{x}}_j(k_j) \end{bmatrix}. \tag{2.51}$$

For the sake of simplicity, we consider two resources here. Recall that we denote by $K = \{t_k | k \in \mathbb{N}\} \subset \mathbb{R}_+$ the ordered set of all capacity events and the subsets $K_j \subset K, j = 1, 2$ of the capacity events of resource j. Additionally, $\phi : \mathbb{N} \to \{1, 2\}$ is the index map, where $\phi(k) = j$, if $t_k \in K_j$.

Moreover, let $\mathbf{V}(k) \in \mathbb{R}^{4n\times 4n}$ denote the following matrix

$$\mathbf{V}(k) \triangleq \begin{cases} \operatorname{diag}\left(\begin{bmatrix} \mathbf{A}_1(k_1) & 0 \\ \frac{1}{k_1+2}\mathbf{A}_1(k_1) & \frac{k_1+1}{k_1+2}I \end{bmatrix}, I^{2n}\right) & \text{if } \phi(k)=1, \\ \operatorname{diag}\left(I^{2n}, \begin{bmatrix} \mathbf{A}_2(k_2) & 0 \\ \frac{1}{k_2+2}\mathbf{A}_2(k_2) & \frac{k_2+1}{k_2+2}I \end{bmatrix}\right) & \text{if } \phi(k)=2. \end{cases} \tag{2.52}$$

Based on the occurrence of the capacity event of a particular resource, the matrix $\mathbf{V}(k)$ is chosen, and the corresponding state vectors are updated, but the state vectors of other resources remain unchanged. Moreover, for $A_{j,q} \in \mathcal{A}_j$, let V_k denote the following matrix,

$$V_k = \begin{cases} \operatorname{diag}\left(\begin{bmatrix} A_{1,q} & 0 \\ \frac{1}{k_1+2}A_{1,q} & \frac{k_1+1}{k_1+2}I \end{bmatrix}, I^{2n}\right) & \text{if } \phi(k)=1, \\ \operatorname{diag}\left(I^{2n}, \begin{bmatrix} A_{2,q} & 0 \\ \frac{1}{k_2+2}A_{2,q} & \frac{k_2+1}{k_2+2}I \end{bmatrix}\right) & \text{if } \phi(k)=2. \end{cases} \tag{2.53}$$

Let $\mathcal{G}(k)$ represent the set of all V_k matrices, for $k \in \mathbb{N}$. We obtain the following non-homogeneous Markov chain

$$\boldsymbol{\xi}(k+1) = \mathbf{V}(k)\boldsymbol{\xi}(k), \quad \text{for } k \in \mathbb{N}, \tag{2.54}$$

with place-dependent probabilities

$$\mathbb{P}\left(\mathbf{V}(k) = V_k \mid \overline{\mathbf{x}}_1(t_k) = \mathbf{y}_1, \overline{\mathbf{x}}_2(t_k) = \mathbf{y}_2\right) = p_V(\mathbf{y}_1, \mathbf{y}_2).$$

Recall that $\overline{\mathbf{x}}_j(t_k)$ denotes the average allocation over the capacity events of resource j up to time instant t_k, for $j = 1, 2$. We now state the main result on the convergence of average allocations as follows.

Theorem 2.4.1 (Convergence of average allocations)**.** *Suppose that Assumption 2.3.2 holds. Let*

$\mathbf{x}^* \in \mathrm{ri}\Sigma^m$ *be the KKT point of Problem* (2.3) *as in Lemma 2.4.2. Let us consider the non-homogeneous Markov chain* (2.54)*. Then, for an initial value* $\boldsymbol{\xi}(0) \in \Sigma^m \times \Sigma^m$*, the following holds almost surely,*

$$\overline{\mathbf{x}}(k) \to \mathbf{x}^*, \text{ when } k \to \infty.$$

□

The proof of Theorem 2.4.1 is presented in Subsection 2.4.3.

Let $W \in \mathbb{N}$ denote a fixed time window and $\mathrm{v} \in \mathbb{N}$ denote an averaging period; let v be very small as compared to W. Our main intuition here is that in the long run, the accumulative average $\overline{\mathbf{x}}(W+\ell)$ is almost a constant for ℓ in the interval $[W, W+\mathrm{v}]$. Thus, for $j = 1, \ldots, m$, the probability of choosing an AIMD matrix $A_{j,q} \in \mathcal{A}$ is also almost a constant in this interval, here $\mathcal{A} \triangleq \cup_{j=1}^{m} \mathcal{A}_j$. We obtain

$$\overline{\mathbf{x}}(W+\mathrm{v}) = \frac{W+1}{W+\mathrm{v}+1}\overline{\mathbf{x}}(W) + \frac{\mathrm{v}}{W+\mathrm{v}+1}\left(\frac{1}{\mathrm{v}}\sum_{\ell=1}^{\mathrm{v}}\mathbf{x}(W+\ell)\right). \tag{2.55}$$

Using the results on AIMD with constant probabilities, we approximate the dynamics of the above system. Let ϵ_{v} denote $\frac{\mathrm{v}}{W+\mathrm{v}+1}$; we reformulate (2.55) to obtain

$$\overline{\mathbf{x}}(W+\mathrm{v}) = (1-\epsilon_{\mathrm{v}})\overline{\mathbf{x}}(W) + \epsilon_{\mathrm{v}}\left(\frac{1}{\mathrm{v}}\sum_{\ell=1}^{\mathrm{v}}\mathbf{x}(W+\ell)\right). \tag{2.56}$$

Let $P : \Sigma^m \to \Sigma^m$, and $P = (P_1, \ldots, P_m)$. For $\mathbf{y} \in \Sigma^m$, $P_j(\mathbf{y})$ denote the expectation of the invariant measure of the AIMD model with fixed probabilities $\boldsymbol{\lambda}_j(\mathbf{y}) = (\lambda_{j1}([\mathbf{y}]_1), \ldots, \lambda_{jn}([\mathbf{y}]_n))$. From (2.40), we obtain

$$P_j(\mathbf{y}) \triangleq \mathbf{w}_{p_j}(\mathbf{y}) = \left(\sum_{i=1}^{n}\frac{1}{\lambda_{ji}([\mathbf{y}]_i)}\right)^{-1}\left[\frac{1}{\lambda_{j1}([\mathbf{y}]_1)} \ \cdots \ \frac{1}{\lambda_{jn}([\mathbf{y}]_n)}\right]^{\top}. \tag{2.57}$$

Thus, we have

$$P(\mathbf{y}) = \left[P_1(\mathbf{y})^{\top} \ P_2(\mathbf{y})^{\top} \ \ldots \ P_m(\mathbf{y})^{\top}\right]^{\top}. \tag{2.58}$$

Let the perturbation term associated with $\mathbf{x}_j$ (see (2.56)) be denoted by $\Delta_j \in \mathbb{R}^n$, and let $\Delta = \begin{bmatrix}\Delta_1^\top & \dots & \Delta_m^\top\end{bmatrix}^\top$. For a time window W, we denote the vector of the perturbation terms $\Delta(W)$ by

$$\Delta(W) = \begin{bmatrix}\Delta_1(W)^\top & \dots & \Delta_m(W)^\top\end{bmatrix}^\top. \tag{2.59}$$

We reformulate (2.56) to obtain

$$\overline{\mathbf{x}}(W + \mathrm{v}) = (1 - \epsilon_\mathrm{v})\overline{\mathbf{x}}(W) + \epsilon_\mathrm{v}\left(P(\overline{\mathbf{x}}(W)) + \Delta(W)\right). \tag{2.60}$$

To understand the dynamics of the system presented in (2.60), we analyze the following system and describe (2.60) as its perturbed version.

$$\overline{\mathbf{x}}(W + \mathrm{v}) = (1 - \epsilon_\mathrm{v})\overline{\mathbf{x}}(W) + \epsilon_\mathrm{v}\left(P(\overline{\mathbf{x}}(W))\right). \tag{2.61}$$

In the following subsection, we analyze the system in (2.61) to obtain characterizations of its unique fixed point; we call this system, a deterministic system. The system in (2.60) is the perturbed version of (2.61).

2.4.1 Results on deterministic systems

Here, we discuss the discrete-time system

$$\mathbf{x}(k+1) = P(\mathbf{x}(k)), \tag{2.62}$$

where $P : \Sigma^m \to \Sigma^m$ is given by (2.58). From Assumption 2.3.2, for $i = 1, \dots, n$ and $j = 1, \dots, m$, as λ_{ji} is continuous; therefore, $P(\Sigma^m) \subset \mathrm{ri}\Sigma^m$ is compact. We choose a constant $\delta^- > 0$ sufficiently small such that the following holds:

$$P(\Sigma^m) + \overline{B}_1(0, \delta^-) \subset \mathrm{conv}\, P(\Sigma^m) + \overline{B}_1(0, 2\delta^-) \subset \mathrm{ri}\Sigma^m. \tag{2.63}$$

For $k \in \mathbb{N}$, let $0 < \epsilon_k < 1$, we consider the following system that performs successive convex combination of $\mathbf{x}$ and $P(\mathbf{x})$:

$$\mathbf{x}(k+1) = (1-\epsilon_k)\mathbf{x}(k) + \epsilon_k P(\mathbf{x}(k)). \tag{2.64}$$

For the sake of simplicity of notation, let $R_\epsilon : \Sigma^m \to \Sigma^m$, defined as

$$R_\epsilon(\mathbf{x}) \triangleq (1-\epsilon)\mathbf{x} + \epsilon P(\mathbf{x}), \text{ for } 0 \leq \epsilon \leq 1. \tag{2.65}$$

Furthermore, for $\delta > 0$, let $P_{\text{co}}(\delta)$ be defined as

$$P_{\text{co}}(\delta) \triangleq \text{conv}\, P(\Sigma^m) + \overline{B}_1(0, \delta). \tag{2.66}$$

Let δ^+ be the constant defined as

$$\delta^+ \triangleq \max_{\mathbf{y} \in \Sigma^m} \{d_1(\mathbf{y}, P(\Sigma^m))\}. \tag{2.67}$$

Lemma 2.4.2 and its proof is by Fabian Wirth. The iteration in (2.64) has the following properties.

Lemma 2.4.2 (Existence and uniqueness of fixed points)**.** *Suppose that Assumption 2.3.2 holds. Let $P : \Sigma^m \to \Sigma^m$ be given by (2.58). Then*

(i) $P(\cdot)$ has a unique fixed point $\mathbf{x}^* \in \Sigma^m$ such that, for $i, u = 1, 2, \ldots, n$ and $j = 1, \ldots, m$, we have

$$\lambda_{ji}([\mathbf{x}]_i^*) x_{ji}^* = \lambda_{ju}([\mathbf{x}]_u^*) x_{ju}^* \triangleq \gamma_{jF}. \tag{2.68}$$

(ii) For each $0 < \epsilon \leq 1$, the fixed point $\mathbf{x}^* \in \Sigma^m$ is the unique fixed point of the map

$$\mathbf{x} \mapsto (1-\epsilon)\mathbf{x} + \epsilon P(\mathbf{x}).$$

(iii) For every initial state $\mathbf{x}(0) \in \Sigma^m$ *and every sequence* $\{\epsilon_k\}_{k\in\mathbb{N}} \subset (0,1)$*, the solution of*

$$\mathbf{x}(k+1) = (1-\epsilon_k)\mathbf{x}(k) + \epsilon_k P(\mathbf{x}(k)),$$

is strictly positive for all $k \geq 1$.

□

Proof. (i) As Σ^m is compact and convex, and $P : \Sigma^m \to \Sigma^m$ is a continuous map, it follows from Brouwer's fixed point theorem that P has a fixed point $\mathbf{x}^* \in \Sigma^m$. By the definition of P, any fixed point $\mathbf{x}^* \in \Sigma^m$ satisfies the following:

$$P_j(\mathbf{x}^*)_i = \left(\sum_{\ell=1}^{n} \frac{1}{\lambda_{j\ell}([\mathbf{x}]^*_\ell)}\right)^{-1} \frac{1}{\lambda_{ji}([\mathbf{x}]^*_i)} = x^*_{ji},$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Thus, we obtain

$$\left(\sum_{\ell=1}^{n} \frac{1}{\lambda_{j\ell}([\mathbf{x}]^*_\ell)}\right)^{-1} = \lambda_{ji}([\mathbf{x}]^*_i) x^*_{ji}.$$

As the term on the left only depends on j, we obtain (2.68). Assume that P has two fixed points $\mathbf{x}, \mathbf{y} \in \Sigma^m$, $\mathbf{x} \neq \mathbf{y}$. Consider the map

$$\alpha \mapsto \mathbf{x}_\alpha := \alpha\mathbf{x} + (1-\alpha)\mathbf{y}, \quad \alpha \in [0,1].$$

As $\mathbf{x} \odot \boldsymbol{\lambda}(\mathbf{x})$ is the gradient of a strictly convex map Q, we have that $Q_2 : \alpha \mapsto Q(\mathbf{x}_\alpha)$ is strictly convex with derivative

$$\begin{aligned}\frac{d}{d\alpha} Q_2(\alpha) &= \langle \mathbf{x}_\alpha \odot \boldsymbol{\lambda}(\mathbf{x}_\alpha), \mathbf{x} - \mathbf{y}\rangle \\ &= \sum_{j=1}^{m} \langle \mathbf{x}_{\alpha j} \odot \boldsymbol{\lambda}_j(\mathbf{x}_\alpha), \mathbf{x}_j - \mathbf{y}_j\rangle.\end{aligned}$$

By strict convexity, this derivative is strictly increasing in α. On the other hand, we have from (2.68) for the two fixed points $\mathbf{x}, \mathbf{y}$ and for all $j = 1, \ldots, m$, that there are suitable constants

c_j, d_j such that

$$\mathbf{x}_j \odot \boldsymbol{\lambda}_j(\mathbf{x}) = c_j \mathbf{e}, \quad \text{and} \quad \mathbf{y}_j \odot \boldsymbol{\lambda}_j(\mathbf{y}) = d_j \mathbf{e}. \tag{2.69}$$

Recall that $\mathbf{e} \in \mathbb{R}^n$ is a vector of all ones. As $\mathbf{x}_j, \mathbf{y}_j \in \Sigma$, it follows that

$$\begin{aligned} \frac{d}{d\alpha} Q_2(1) &= \sum_{j=1}^{m} \langle \mathbf{x}_j \odot \boldsymbol{\lambda}_j(\mathbf{x}), \mathbf{x}_j - \mathbf{y}_j \rangle \\ &= \sum_{j=1}^{m} c_j \langle \mathbf{e}, \mathbf{x}_j - \mathbf{y}_j \rangle = 0 \\ &= \sum_{j=1}^{m} d_j \langle \mathbf{e}, \mathbf{x}_j - \mathbf{y}_j \rangle = \frac{d}{d\alpha} Q_2(0). \end{aligned}$$

This equation contradicts our previous observation that $\frac{dQ_2}{d\alpha}$ is strictly increasing. This shows the uniqueness of the fixed point.

(ii) As $\epsilon > 0$, if $\mathbf{x} = (1-\varepsilon)\mathbf{x} + \varepsilon P(\mathbf{x})$, it follows that $\mathbf{x} = P(\mathbf{x})$. By (i), $\mathbf{x}$ is equal to the unique fixed point $\mathbf{x}^*$ of P.

(iii) As $\lambda_{ji} \geq \lambda_{j,\min} > 0$, for $i = 1, \ldots, n$, $j = 1, \ldots, m$, we have $P(\Sigma^m) \subset \mathrm{ri}\Sigma^m$. Also, as $\mathbf{x}(k)$ is the convex combination of a point in Σ^m and a point in $\mathrm{ri}\Sigma^m$, it follows that $\mathbf{x}(k) \in \mathrm{ri}\Sigma^m$, for $k \in \mathbb{N}$. □

Lemma 2.4.3. *Let $\mathbf{x} \in \Sigma^m$, and let $\delta^- > 0$ be a constant that satisfies (2.63). For all $\epsilon \in (0, 1]$ and $\delta \in (0, \delta^-)$:*

(i) For $j = 1, 2, \ldots, m$, let $\Delta_j \in \mathbb{R}^n$ such that $\mathbf{e}^\top \Delta_j = 0$; moreover, let $\Delta = [\Delta_1^\top \ \ldots \ \Delta_m^\top]^\top$, then for $\|\Delta_j\|_1 \leq \delta$, we have

$$d_1(R_\epsilon(\mathbf{x}) + \epsilon\Delta, P_{\mathrm{co}}(\delta)) \leq (1-\epsilon) d_1(\mathbf{x}, P_{\mathrm{co}}(\delta)).$$

(ii) Let δ^+ be as in (2.67), then for all $\mathbf{y} \in \Sigma^m$, we have

$$d_1((1-\epsilon)\mathbf{x} + \epsilon\mathbf{y}, P_{\mathrm{co}}(\delta)) \leq (1-\epsilon) d_1(\mathbf{x}, P_{\mathrm{co}}(\delta)) + \epsilon\delta^+.$$

(iii) Let there exist $C_{\overline{\delta}}$ such that for all $\epsilon \in (0,1)$ and $\delta \in (0, \delta^-)$, if $d_1(\mathbf{x}, P_{\text{co}}(\delta)) > \overline{\delta}$, then we have

$$d_1((1-\epsilon)\mathbf{x} + \epsilon\mathbf{y}, P_{\text{co}}(\delta)) \leq (1 + \epsilon C_{\overline{\delta}}) d_1(\mathbf{x}, P_{\text{co}}(\delta)).$$

□

Proof. Similar to the proof of (Wirth et al., 2019, Lemma B.2). □

Lemma 2.4.4. *Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2. For a fixed j, and every $\zeta_j > 0$, there exist constants r_j, R_j and ϵ_j, with $0 < r_j < 1 < R_j$ and $\epsilon_j \in (0,1)$ such that, for all $\epsilon \in (0, \epsilon_j)$ if we have $d_H(\mathbf{x}_j, \mathbf{x}_j^*) > \zeta_j$, and i', u' are such that*

$$\exp(d_H(R_\epsilon(\mathbf{x})_j, \mathbf{x}_j^*)) = \frac{R_\epsilon(\mathbf{x})_{ji'} / x^*_{ji'}}{R_\epsilon(\mathbf{x})_{ju'} / x^*_{ju'}}, \tag{2.70}$$

then $x_{ji'} > R_j x^*_{ji'}$ and $x_{ju'} < r_j x^*_{ju'}$.

Proof. Similar to the proof of (Wirth et al., 2019, Lemma B.3). □

Theorem 2.4.5. *Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \in \Sigma^m$. Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2. Suppose that Assumption 2.3.2 holds. For every $\zeta > 0$, there exists $0 < \epsilon_0 < 1$ such that for all $0 < \epsilon < \epsilon_0$, we have*

$$D_H(\mathbf{x}, \mathbf{x}^*) > \zeta \implies D_H(R_\epsilon(\mathbf{x}), \mathbf{x}^*) < D_H(\mathbf{x}, \mathbf{x}^*).$$

□

Corollary 2.4.6. *(Wirth et al., 2019, Corollary B.5) Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2. Furthermore, let $L_{1j} < \gamma_{jF} < L_{2j}$ and $C_{\zeta_j} > 0$ as in (2.71), then for each $\zeta > 0$ there exists $0 < \epsilon_0 < 1$ such that for all $0 < \epsilon < \epsilon_0$, we have*

$$\begin{aligned} &d_H(\mathbf{x}_j, \mathbf{x}_j^*) \geq \zeta \\ &\implies \exp(d_H(R_\epsilon(\mathbf{x})_j, \mathbf{x}_j^*)) < (1 - \epsilon C_{\zeta_j}) \exp(d_H(\mathbf{x}_j, \mathbf{x}_j^*)), \end{aligned}$$

where

$$C_{\zeta_j} = \min_{\epsilon\in[0,1]} \frac{\frac{1}{L_{1j}} - \frac{1}{L_{2j}}}{\left((1-\epsilon)\sum_{\ell=1}^{n} \frac{1}{\lambda_{j\ell}([\mathbf{x}]_\ell)} + \frac{\epsilon}{L_{1j}}\right)}. \tag{2.71}$$

□

Corollary 2.4.7. *Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2. Let $L_{1j} < \gamma_{jF} < L_{2j}$, and let $C_{\zeta_j} > 0$ as in (2.71). Furthermore, let for each $\zeta > 0$ there exist $0 < \epsilon_0 < 1$ and a constant $C_\zeta > 0$ such that for all $0 < \epsilon < \epsilon_0$, we have*

$$\begin{aligned} &D_H(\mathbf{x}, \mathbf{x}^*) \geq \zeta \\ &\implies \exp(D_H(R_\epsilon(\mathbf{x}), \mathbf{x}^*)) < (1 - \epsilon C_\zeta) \exp(D_H(\mathbf{x}, \mathbf{x}^*)). \end{aligned} \tag{2.72}$$

□

Proof. From Corollary 2.4.6, we get the desired result:

$$\begin{aligned} \max_{j=1,\dots,m} \left\{\exp(d_H(R_\epsilon(\mathbf{x})_j, \mathbf{x}_j^*))\right\} &< \max_{j=1,\dots,m} \left\{\left(1 - \epsilon C_{\zeta_j}\right) \exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right)\right\} \\ &= \max_{j=1,\dots,m} \left\{\left(1 - \epsilon C_{\zeta_j}\right)\right\} \max_{j=1,\dots,m} \left\{\exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right)\right\} \\ &= (1 - \epsilon C_\zeta) \exp\left(D_H(\mathbf{x}, \mathbf{x}^*)\right), \end{aligned}$$

where

$$C_\zeta \triangleq \min_{j=1,\dots,m} \min_{\epsilon\in[0,1]} \frac{\frac{1}{L_{1j}} - \frac{1}{L_{2j}}}{\left((1-\epsilon)\sum_{\ell=1}^{n} \frac{1}{\lambda_{j\ell}([\mathbf{x}]_\ell)} + \frac{\epsilon}{L_{1j}}\right)} > 0. \tag{2.73}$$

□

We now assume that $0 < \delta < \delta^-$; thus, we have

$$\operatorname{conv} P(\Sigma^m) + \overline{B}_1(0, 2\delta) \subset \operatorname{ri}\Sigma^m.$$

Moreover, for a fixed j, and $i, u = 1, 2, \dots, n$, let us choose agents' indexes (i', u') such that the

following holds:

$$\frac{\max_i\{(R_\epsilon(\mathbf{x})_{ji}+\epsilon\Delta_{ji})/x^*_{ji}\}}{\min_u\{(R_\epsilon(\mathbf{x})_{ju}+\epsilon\Delta_{ju})/x^*_{ju}\}}-\frac{\max_i\{R_\epsilon(\mathbf{x})_{ji}/x^*_{ji}\}}{\min_u\{R_\epsilon(\mathbf{x})_{ju}/x^*_{ju}\}}$$
$$\leq \epsilon\delta\frac{(R_\epsilon(\mathbf{x})_{ji'}+R_\epsilon(\mathbf{x})_{ju'})/x^*_{ji'}}{(R_\epsilon(\mathbf{x})_{ju'}-\epsilon\delta)R_\epsilon(\mathbf{x})_{ju'}/x^*_{ju'}}. \tag{2.74}$$

We obtain the following results.

Lemma 2.4.8. *Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2. Let $\delta^- > 0$ satisfy Equation (2.63). Then, for a fixed j, there exists $\varpi_j > 0$ such that for all $\delta \in (0, \delta^-)$, $\epsilon \in (0,1)$, $\mathbf{x} \in \operatorname{conv} P(\Sigma^m) + \overline{B}_1(0,\delta)$, $\Delta_j \in \mathbb{R}^n$ with $\mathbf{e}^\top\Delta_j = 0$ and $\|\Delta_j\|_1 \leq \delta$, we have the following robustness result*

$$\exp(d_H(R_\epsilon(\mathbf{x})_j + \epsilon\Delta_j, \mathbf{x}^*_j)) - \exp(d_H(R_\epsilon(\mathbf{x})_j, \mathbf{x}^*_j)) \leq \epsilon\delta\varpi_j.$$

□

Proof. By assumption, $\operatorname{conv} P(\Sigma^m) + \overline{B}_1(0,\delta)$ is a compact subset of $\operatorname{ri}\Sigma^m$; therefore, each entry of $\mathbf{x}$ and $R_\epsilon(\mathbf{x})$ is bounded away from 0. For a fixed resource j and the agent's index u', from (Wirth et al., 2019, Lemma B.6), we know that $(R_\epsilon(\mathbf{x})_{ju'} - \epsilon\delta)$ is bounded away from 0. Thus, as the denominator of the right-hand side of (2.74) is bounded away from 0, it may be bounded by a constant, ϖ_j. □

Lemma 2.4.9. *Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2. Let $0 < \epsilon < 1$, and let $\delta^- > 0$ satisfy (2.63). For all j, let $\Delta_j \in \mathbb{R}^n$ be the perturbation term such that $\mathbf{e}^\top\Delta_j = 0$. For $\delta > 0$, let $\|\Delta_j\|_1 \leq \delta$. Furthermore, let $\Delta = [{\Delta_1}^\top \ \dots \ {\Delta_m}^\top]^\top$. Also, let there exist a constant $\varpi > 0$ such that for all $0 < \delta < \delta^-$ and $\mathbf{x} \in \operatorname{conv} P(\Sigma^m) + \overline{B}_1(0,\delta)$, we have*

$$\exp(D_H(R_\epsilon(\mathbf{x}) + \epsilon\Delta, \mathbf{x}^*)) - \exp(D_H(R_\epsilon(\mathbf{x}), \mathbf{x}^*)) \leq \epsilon\delta\varpi. \tag{2.75}$$

□

Proof. By Definition (2.23), we obtain

$$\begin{aligned}
&\exp(D_H(R_\epsilon(\mathbf{x}) + \epsilon\Delta, \mathbf{x}^*)) - \exp(D_H(R_\epsilon(\mathbf{x}), \mathbf{x}^*)) \\
&= \max_{j=1,\dots,m} \Big\{ \exp(d_H(R_\epsilon(\mathbf{x})_j + \epsilon\Delta_j, \mathbf{x}_j^*)) - \exp(d_H(R_\epsilon(\mathbf{x})_j, \mathbf{x}_j^*)) \Big\}.
\end{aligned}$$

Let us choose agents' indexes (i_j, u_j) such that (2.74) holds for resource j, for $j = 1, 2, \dots, m$. Let j' be a specific resource and (i', u') be chosen for j' such that the maximum is attained for (2.74) over all resources. Thus, for $j = 1, \dots, m$, from Lemma 2.4.8, we obtain:

$$\begin{aligned}
&\max_j \left\{ \exp(d_H(R_\epsilon(\mathbf{x})_j + \epsilon\Delta_j, \mathbf{x}_j^*)) - \exp(d_H(R_\epsilon(\mathbf{x})_j, \mathbf{x}_j^*)) \right\} \\
&\leq \epsilon\delta \max_j \left\{ \frac{(R_\epsilon(\mathbf{x})_{ji_j} + R_\epsilon(\mathbf{x})_{ju_j})/x^*_{ji_j}}{(R_\epsilon(\mathbf{x})_{ju_j} - \epsilon\delta)R_\epsilon(\mathbf{x})_{ju_j}/x^*_{ju_j}} \right\} \\
&\leq \epsilon\delta \frac{(R_\epsilon(\mathbf{x})_{j'i'} + R_\epsilon(\mathbf{x})_{j'u'})/x^*_{j'i'}}{(R_\epsilon(\mathbf{x})_{j'u'} - \epsilon\delta)R_\epsilon(\mathbf{x})_{j'u'}/x^*_{j'u'}} = \epsilon\delta\varpi,
\end{aligned}$$

where

$$\varpi = \frac{(R_\epsilon(\mathbf{x})_{j'i'} + R_\epsilon(\mathbf{x})_{j'u'})/x^*_{j'i'}}{(R_\epsilon(\mathbf{x})_{j'u'} - \epsilon\delta)R_\epsilon(\mathbf{x})_{j'u'}/x^*_{j'u'}}. \tag{2.76}$$

□

Lemma 2.4.10. *(Wirth et al., 2019, Lemma B.8) For a fixed j, let $\mathbf{x}_j^* \in \Sigma$ be the unique fixed point of P. For every $\mathbf{x}_j \in \mathrm{ri}\Sigma$, $\mathbf{y}_j \in \Sigma$, let there exist $x^*_{j,\min}$ such that $\max_{i=1,\dots,n} \left\{1/x^*_{ji}\right\} = 1/x^*_{j,\min}$, then for each $0 \leq \epsilon < 1$, we have*

$$\exp(d_H((1-\epsilon)\mathbf{x}_j + \epsilon\mathbf{y}_j, \mathbf{x}_j^*)) \leq \exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right) \left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{j,\min}}\right).$$

□

Corollary 2.4.11. *Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as in Lemma 2.4.2. For each $\zeta > 0$, there exists $0 < \epsilon_0 < 1$ and the constant $C_\zeta > 0$ as in (2.73) such that for all $0 < \epsilon < \epsilon_0$, Equation (2.72) holds. Let $\delta^- > 0$ satisfy (2.63) and $\varpi > 0$ be the constant as in Lemma 2.4.9. Furthermore,*

let

$$\delta^* \triangleq \min\left\{\frac{C_\zeta \exp(\zeta)}{2\varpi}, \delta^-\right\}. \tag{2.77}$$

For all j, let $\Delta_j \in \mathbb{R}^n$ be the perturbation term such that $\mathbf{e}^\top \Delta_j = 0$. Also, let $\Delta = [{\Delta_1}^\top \ \dots \ {\Delta_m}^\top]^\top$, and let $P_{\text{co}}(\delta)$ be defined as in (2.66). Then for each $0 < \delta < \delta^*$; $0 < \epsilon < \epsilon_0$; $\mathbf{x} \in \text{conv}\, P(\Sigma^m) + \overline{B}_1(0, \delta)$; and $\|\Delta_j\|_1 \leq \delta$, we have

$$R_\epsilon(\mathbf{x}) + \epsilon\Delta \in P_{\text{co}}(\delta), \tag{2.78}$$

and

$$\begin{aligned} &D_H(\mathbf{x}, \mathbf{x}^*) \geq \zeta \\ &\implies \exp\left(D_H(R_\epsilon(\mathbf{x}) + \epsilon\Delta, \mathbf{x}^*)\right) < \left(1 - \epsilon\frac{C_\zeta}{2}\right)\exp\left(D_H(\mathbf{x}, \mathbf{x}^*)\right). \end{aligned} \tag{2.79}$$

□

Proof. Similar to the proof of (Wirth et al., 2019, Corollary B.7). □

Lemma 2.4.12. *Let $\mathbf{x}^* \in \Sigma^m$ be the unique fixed point of P as described in Lemma 2.4.2, and let there exist $x^*_{\min}$ such that $\max\limits_{j=1,\dots,m}\left\{\frac{1}{x^*_{j,\min}}\right\} = \frac{1}{x^*_{\min}}$, then for every $\mathbf{x} \in \text{ri}\Sigma^m$, $\mathbf{y} \in \Sigma^m$ and for all $\epsilon \in (0, 1)$, we have*

$$\exp(D_H((1-\epsilon)\mathbf{x} + \epsilon\mathbf{y}, \mathbf{x}^*)) \leq \exp\left(D_H(\mathbf{x}, \mathbf{x}^*)\right)\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{\min}}\right). \tag{2.80}$$

We can also express it as

$$D_H((1-\epsilon)\mathbf{x} + \epsilon\mathbf{y}, \mathbf{x}^*) \leq D_H(\mathbf{x}, \mathbf{x}^*) + \log\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{\min}}\right).$$

Specifically, for all $\epsilon_0 \in (0, 1)$, there exists a constant C_0 such that for all $\mathbf{x} \in \text{ri}\Sigma^m$, $\mathbf{y} \in \Sigma^m$, and

$0 < \epsilon < \epsilon_0$, *we have*

$$D_H((1-\epsilon)\mathbf{x} + \epsilon\mathbf{y}, \mathbf{x}^*) \leq D_H(\mathbf{x}, \mathbf{x}^*) + \epsilon C_0. \tag{2.81}$$

□

Proof. From Lemma 2.4.10, for a fixed j, we have

$$\exp(d_H((1-\epsilon)\mathbf{x}_j + \epsilon\mathbf{y}_j, \mathbf{x}_j^*)) \leq \exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right)\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{j,\min}}\right).$$

Thus, for $j = 1, 2, \ldots, m$, we obtain

$$\max_j \left\{\exp(d_H((1-\epsilon)\mathbf{x}_j + \epsilon\mathbf{y}_j, \mathbf{x}_j^*))\right\} \leq \max_j \left\{\exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right)\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{j,\min}}\right)\right\}.$$

By Definition (2.23), for $j = 1, 2, \ldots, m$, we obtain (2.80) as follows:

$$\begin{aligned}
\exp(D_H((1-\epsilon)\mathbf{x} + \epsilon\mathbf{y}, \mathbf{x}^*)) &\leq \max_j \left\{\exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right)\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{j,\min}}\right)\right\} \\
&= \max_j\{\exp\left(d_H(\mathbf{x}_j, \mathbf{x}_j^*)\right)\} \max_j \left\{\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{j,\min}}\right)\right\} \\
&= \exp\left(D_H(\mathbf{x}, \mathbf{x}^*)\right)\left(1 + \frac{\epsilon}{1-\epsilon}\frac{1}{x^*_{\min}}\right).
\end{aligned}$$

□

2.4.2 Results on stochastic systems

We now use the results on deterministic systems to prove almost sure convergence of the long-term average allocation to the unique fixed point. In this direction, we present the results on the length of the averaging period, followed by the proof of the main theorem (Theorem 2.4.1) in several steps.

As before, for the ease of discussion, we consider $m = 2$ resources; however, it is easily extended for $m > 2$. For $\mathbf{y} \in \Sigma^2$, recall that $P(\mathbf{y}) = [P_1(\mathbf{y})^\top \; P_2(\mathbf{y})^\top]^\top$ denotes the expectation

of the invariant measure of the AIMD model with fixed probability $\boldsymbol{\lambda}(\mathbf{y})$. We define

$$P_e(\mathbf{y}) \triangleq \text{diag}\left(P_1(\mathbf{y})\mathbf{e}^\top, P_2(\mathbf{y})\mathbf{e}^\top\right). \tag{2.82}$$

We denote (c.f. Definition (2.42))

$$\overline{\mathbf{S}}(m\mathrm{v}) = \text{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v}), \ldots, \overline{\mathbf{S}}_m(\mathrm{v})\right). \tag{2.83}$$

For $j = 1, 2, \ldots, m$, by (2.41), we obtain

$$\overline{\mathbf{S}}_j(\mathrm{v})\mathbf{x}_j(W) = \frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}} \mathbf{x}_j(W+\ell). \tag{2.84}$$

Thus, we obtain

$$\begin{aligned}
\overline{\mathbf{S}}(m\mathrm{v})\mathbf{x}(W) &= \text{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v}), \ldots, \overline{\mathbf{S}}_m(\mathrm{v})\right) \begin{bmatrix} \mathbf{x}_1(W) \\ \vdots \\ \mathbf{x}_m(W) \end{bmatrix} \\
&= \begin{bmatrix} \overline{\mathbf{S}}_1(\mathrm{v})\mathbf{x}_1(W) \\ \vdots \\ \overline{\mathbf{S}}_m(\mathrm{v})\mathbf{x}_m(W) \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}} \mathbf{x}_1(W+\ell) \\ \vdots \\ \frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}} \mathbf{x}_m(W+\ell) \end{bmatrix} \\
&= \frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}} \mathbf{x}(W+\ell).
\end{aligned} \tag{2.85}$$

We now extend Lemma 2.3.1 to obtain the following continuity result for the Markov chain (2.38) with fixed probabilities that determines the length of the averaging period.

Lemma 2.4.13 (Length of averaging period)**.** *Suppose that Assumption 2.3.2 holds. Let $\mathbf{y} \in \Sigma^m$, and let us consider the family of Markov chains in (2.38), with fixed probability $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{y})$. Then for all $\overline{\delta} > 0$ and $0 < \theta \leq 1$, let there exist $\mathrm{v} \in \mathbb{N}$ such that for all $\mathbf{y} \in \Sigma^m$ and for all m resources,*

we have

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(m\mathrm{v}) - P_e(\mathbf{y})\right\|_1 > \overline{\delta}\right) < \theta. \tag{2.86}$$

□

Proof. For a fixed $\hat{\mathbf{y}} \in \Sigma^m$ and $\delta, \epsilon > 0$, let there exist $\hat{\mathrm{v}} \in \mathbb{N}$, for m resources in the network. Then from Lemma 2.3.1, we have

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(m\hat{\mathrm{v}}) - P_e(\hat{\mathbf{y}})\right\|_1 > \delta\right) < \epsilon. \tag{2.87}$$

By Assumption 2.3.2, the map P is continuous; thus,

$$\mathbf{y} \mapsto \mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(m\hat{\mathrm{v}}) - P_e(\hat{\mathbf{y}})\right\|_1 > \delta\right),$$

is continuous. Hence, we have

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(m\hat{\mathrm{v}}) - P_e(\mathbf{y})\right\|_1 > 2\delta\right) < 2\epsilon. \tag{2.88}$$

Note that it holds on the neighborhood of $\hat{\mathbf{y}}$. Thus, there exists a finite number of such neighborhoods, and so finitely many $\hat{\mathrm{v}}$ exist. For $\mathbf{y} \in \Sigma^m$ and $\hat{\mathrm{v}} \in \mathbb{N}$, let the random variable $\mathbf{M}(m\hat{\mathrm{v}})$ be defined as

$$\mathbf{M}(m\hat{\mathrm{v}}) \triangleq \left\|\overline{\mathbf{S}}(m\hat{\mathrm{v}}) - P_e(\mathbf{y})\right\|_1. \tag{2.89}$$

For the sake of exposition, we consider $m = 2$ here. By Definition (2.45), we obtain

$$\begin{aligned}\mathbf{M}(2\hat{\mathrm{v}}) &= \left\|\mathrm{diag}\left(\overline{\mathbf{S}}_1(\hat{\mathrm{v}}) - P_1(\mathbf{y})\mathbf{e}^\top, \overline{\mathbf{S}}_2(\hat{\mathrm{v}}) - P_2(\mathbf{y})\mathbf{e}^\top\right)\right\|_1 \\ &= \max_{j=1,2}\left\|\overline{\mathbf{S}}_j(\hat{\mathrm{v}}) - P_j(\mathbf{y})\mathbf{e}^\top\right\|_1. \end{aligned} \tag{2.90}$$

As $\overline{\mathbf{S}}_j(\hat{\mathrm{v}})$ and $P_j(\mathbf{y})\mathbf{e}^\top$ are column stochastic matrices, $\overline{\mathbf{S}}(m\hat{\mathrm{v}})$ and $P_e(\mathbf{y})$ will also be column stochastic matrices. Hence, we have $0 \leq \mathbf{M}(m\hat{\mathrm{v}}) \leq 2$ and $0 \leq \mathbb{E}[\mathbf{M}(m\hat{\mathrm{v}})^2] \leq 4$. Moreover, from

(2.87), we obtain

$$0 \leq \mathbb{E}\left[\mathbf{M}(m\hat{\mathrm{v}})\right] \leq 2\delta + 2 \times 2\epsilon = 2\delta + 4\epsilon. \tag{2.91}$$

Let σ^2 denote the variance of $\mathbf{M}(m\hat{\mathrm{v}})$. For a fixed ϵ and δ, the variance σ^2 is finite. Thus, for $m = 2$, and $L \in \mathbb{N}$, for multiples of $\hat{\mathrm{v}}$, we obtain

$$\overline{\mathbf{S}}(2L\hat{\mathrm{v}}) = \mathrm{diag}\left(\overline{\mathbf{S}}_1(L\hat{\mathrm{v}}), \overline{\mathbf{S}}_2(L\hat{\mathrm{v}})\right).$$

Additionally, for a fixed j, by the definition of (2.41), we obtain

$$\overline{\mathbf{S}}_j(L\hat{\mathrm{v}}) = \frac{1}{L\hat{\mathrm{v}}+1}\left(\sum_{\ell=1}^{L\hat{\mathrm{v}}} \mathbf{A}_j(\ell-1)\ldots\mathbf{A}_j(0) + I\right).$$

Let $\mathbf{A}_j(\ell-1)\ldots\mathbf{A}_j(0)$ be denoted by $\Pi_{A_j}(\ell)$. Then, we obtain

$$\overline{\mathbf{S}}_j(L\hat{\mathrm{v}}) = \frac{1}{L\hat{\mathrm{v}}+1}\left(\sum_{\ell=0}^{L-1}\sum_{t=0}^{\hat{\mathrm{v}}-1} \mathbf{A}_j(\ell\hat{\mathrm{v}}+t)\ldots\mathbf{A}_j(\ell\hat{\mathrm{v}})\Pi_{A_j}(\ell\hat{\mathrm{v}}) + I\right).$$

Following the steps as in the proof of (Wirth et al., 2019, Lemma C.2), we obtain

$$\left\|\overline{\mathbf{S}}_j(L\hat{\mathrm{v}}) - P_j(\mathbf{y})\mathbf{e}^\top\right\|_1 \leq \frac{1}{L}\sum_{\ell=0}^{L-1}\left\|\overline{\mathbf{S}}_j(\ell\hat{\mathrm{v}}) - P_j(\mathbf{y})\mathbf{e}^\top\right\|_1. \tag{2.92}$$

From (2.90) and (2.92), we obtain

$$\mathbf{M}(2L\hat{\mathrm{v}}) = \max_{j=1,2}\left\|\overline{\mathbf{S}}_j(L\hat{\mathrm{v}}) - P_j(\mathbf{y})\mathbf{e}^\top\right\|_1 \leq \max_{j=1,2}\frac{1}{L}\sum_{\ell=0}^{L-1}\left\|\overline{\mathbf{S}}_j(\ell\hat{\mathrm{v}}) - P_j(\mathbf{y})\mathbf{e}^\top\right\|_1. \tag{2.93}$$

For $\ell \in \mathbb{N}$, let $X_\ell, Y_\ell \in \mathbb{R}$. We know that

$$\max\left\{\sum_{\ell=1}^{L} X_\ell, \sum_{\ell=1}^{L} Y_\ell\right\} \leq \sum_{\ell=1}^{L}\max\left\{X_\ell, Y_\ell\right\}, \text{ for } L \in \mathbb{N}.$$

Thus, from (2.93), we obtain

$$\mathbf{M}(2L\hat{\mathrm{v}}) \leq \frac{1}{L}\sum_{\ell=0}^{L-1}\max_{j=1,2}\left\|\overline{\mathbf{S}}_j(\ell\hat{\mathrm{v}}) - P_j(\mathbf{y})\mathbf{e}^\top\right\|_1 = \frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}}).$$

Furthermore, we obtain

$$\begin{aligned}\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(2L\hat{\mathrm{v}}) - P_e(\mathbf{y})\right\|_1 > \overline{\delta}\right) &= \mathbb{P}_{\boldsymbol{\lambda}}\left(\mathbf{M}(2L\hat{\mathrm{v}}) > \overline{\delta}\right)\\ &\leq \mathbb{P}_{\boldsymbol{\lambda}}\left(\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}}) > \overline{\delta}\right).\end{aligned} \tag{2.94}$$

As $0 \leq \mathbb{E}\left[\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}})\right] = \mathbb{E}\left[\mathbf{M}(2\hat{\mathrm{v}})\right] \leq 2\delta + 4\epsilon$, we choose ϵ and δ such that $0 \leq \mathbb{E}\left[\mathbf{M}(2\hat{\mathrm{v}})\right] < \frac{\overline{\delta}}{2}$. Thus, we obtain

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}}) > \overline{\delta}\right) \leq \mathbb{P}_{\boldsymbol{\lambda}}\left(\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}}) > \mathbb{E}\left[\mathbf{M}(2\hat{\mathrm{v}})\right] + \frac{\overline{\delta}}{2}\right).$$

Recall that σ^2 denotes the variance of $\mathbf{M}(2\hat{\mathrm{v}})$, consider $m = 2$; thus, the variance of $\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}}) = \frac{L\sigma^2}{L} = \sigma^2$. As σ^2 is finite, from Chebyshev's inequality, we obtain

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\frac{1}{L}\sum_{\ell=0}^{L-1}\mathbf{M}(2\ell\hat{\mathrm{v}}) - \mathbb{E}\left[\mathbf{M}(2\hat{\mathrm{v}})\right] > \frac{\overline{\delta}}{2}\right) < \frac{\sigma^2}{(\frac{\overline{\delta}}{2})^2} = \frac{4\sigma^2}{(\overline{\delta})^2}. \tag{2.95}$$

Let $\frac{4\sigma^2}{(\overline{\delta})^2}$ be denoted by θ. Thus, from (2.94) and (2.95), we obtain

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(2L\hat{\mathrm{v}}) - P_e(\mathbf{y})\right\|_1 > \overline{\delta}\right) < \theta.$$

As there exists a finite number of $\hat{\mathrm{v}}$, we choose v greater than the least common multiple of $\hat{\mathrm{v}}$ to obtain (2.86). □

Corollary 2.4.14. *Suppose that Assumption 2.3.2 holds. Let $\mathbf{x}, \mathbf{y} \in \Sigma^m$. Let us consider the family of Markov chains in (2.38) with fixed probability $\boldsymbol{\lambda}(\mathbf{y})$. Then for all $\overline{\delta} > 0$ and $\theta \in (0, 1]$, let there*

exist $\mathrm{v} \in \mathbb{N}$ *such that for all* $W \in \mathbb{N}$*, we have*

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}(W+\ell)-P(\mathbf{y})\right\|>\overline{\delta}\right)<\theta. \tag{2.96}$$

□

Proof. For $j=1,2,\ldots,m$, we have $\mathbf{e}^{\top}\mathbf{x}_j(W)=1$, and we also have $\|\mathbf{x}(W)\|_1=1$. Also we use the result (2.85); we obtain

$$\begin{aligned}
&\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}(W+\ell)-P(\mathbf{y})\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\begin{bmatrix}\frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}_1(W+\ell)\\ \vdots\\ \frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}_m(W+\ell)\end{bmatrix}-\begin{bmatrix}P_1(\mathbf{y})\\ \vdots\\ P_m(\mathbf{y})\end{bmatrix}\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\begin{bmatrix}\frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}_1(W+\ell)-P_1(\mathbf{y})\\ \vdots\\ \frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}_m(W+\ell)-P_m(\mathbf{y})\end{bmatrix}\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\begin{bmatrix}\overline{\mathbf{S}}_1(\mathrm{v})\mathbf{x}_1(W)-P_1(\mathbf{y})\\ \vdots\\ \overline{\mathbf{S}}_m(\mathrm{v})\mathbf{x}_m(W)-P_m(\mathbf{y})\end{bmatrix}\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\begin{bmatrix}\overline{\mathbf{S}}_1(\mathrm{v})\mathbf{x}_1(W)-P_1(\mathbf{y})\mathbf{e}^{\top}\mathbf{x}_1(W)\\ \vdots\\ \overline{\mathbf{S}}_m(\mathrm{v})\mathbf{x}_m(W)-P_m(\mathbf{y})\mathbf{e}^{\top}\mathbf{x}_m(W)\end{bmatrix}\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\mathrm{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v})-P_1(\mathbf{y})\mathbf{e}^{\top},\ldots,\overline{\mathbf{S}}_m(\mathrm{v})-P_m(\mathbf{y})\mathbf{e}^{\top}\right)\mathbf{x}(W)\right\|_1>\overline{\delta}\right)\\
&\leq\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\mathrm{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v})-P_1(\mathbf{y})\mathbf{e}^{\top},\ldots,\overline{\mathbf{S}}_m(\mathrm{v})-P_m(\mathbf{y})\mathbf{e}^{\top}\right)\right\|_1\|\mathbf{x}(W)\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\mathrm{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v})-P_1(\mathbf{y})\mathbf{e}^{\top},\ldots,\overline{\mathbf{S}}_m(\mathrm{v})-P_m(\mathbf{y})\mathbf{e}^{\top}\right)\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\mathrm{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v}),\ldots,\overline{\mathbf{S}}_m(\mathrm{v})\right)-\mathrm{diag}\left(P_1(\mathbf{y})\mathbf{e}^{\top},\ldots,P_m(\mathbf{y})\mathbf{e}^{\top}\right)\right\|_1>\overline{\delta}\right)\\
&=\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(m\mathrm{v})-P_e(\mathbf{y})\right\|_1>\overline{\delta}\right).
\end{aligned} \tag{2.97}$$

From Lemma 2.4.13, we have

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\overline{\mathbf{S}}(m\mathrm{v}) - P_e(\mathbf{y})\right\|_1 > \overline{\delta}\right) < \theta. \tag{2.98}$$

Hence, from (2.97) and (2.98), we obtain the desired result:

$$\mathbb{P}_{\boldsymbol{\lambda}}\left(\left\|\frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}(W+\ell) - P(\mathbf{y})\right\|_1 > \overline{\delta}\right) < \theta.$$

□

Let us now consider a small averaging period $\mathrm{v} \in \mathbb{N}$ compared to the time-window $W \in \mathbb{N}$ and use the intuition that with increasing W, the accumulative average $\overline{\mathbf{x}}(W+\ell)$ is almost constant for ℓ in the interval $[W, W+\mathrm{v}]$ (discussed previously in (2.55)—(2.61)). Hence, the response probabilities over the interval $[W, W+\mathrm{v}]$ are also almost constant, denoted by the vector $\boldsymbol{\lambda}(\overline{\mathbf{x}}(W))$. The expectation of the invariant measure of the AIMD model with fixed probabilities $\boldsymbol{\lambda}(\overline{\mathbf{x}}(W))$ is denoted by $P(\overline{\mathbf{x}}(W))$, (refer (2.58)). For initial value $\mathbf{x}_0 \in \Sigma^m$, let $\mathbb{P}_{\mathbf{x}_0}$ denote the probability measure with fixed probability $\boldsymbol{\lambda}(\mathbf{x}_0)$.

We choose $W_0 \in \mathbb{N}$ such that for all $W \geq W_0$, we have $\frac{\mathrm{v}}{W+\mathrm{v}+1} < \epsilon_0$; and for $\theta \in (0,1)$, we have (2.99), which is the perturbed version of (2.96):

$$\mathbb{P}_{\mathbf{x}_0}\left(\left\|\frac{1}{\mathrm{v}+1}\sum_{\ell=0}^{\mathrm{v}}\mathbf{x}(W+\ell) - P(\overline{\mathbf{x}}(W))\right\|_1 > \overline{\delta}\right) < \theta. \tag{2.99}$$

Given a small averaging period $\mathrm{v} \in \mathbb{N}$, compared to time-window W; as $W \to \infty$, the place-dependent probabilities of the non-homogeneous Markov chain (2.54) over the interval $[W, W+\mathrm{v}]$ converge to the fixed probabilities $\boldsymbol{\lambda}(\overline{\mathbf{x}}(W))$, and as λ_{ji} is continuous by assumption, so, for $W_0 \in \mathbb{N}$ and all $W \geq W_0$, Equation (2.99) holds for the non-homogeneous Markov chain (2.54).

Now, for $k_0 \in \mathbb{N}$ and $k \geq k_0$, we analyze the evolution of the following map

$$\overline{\mathbf{x}}(W+k\mathrm{v}) \mapsto \overline{\mathbf{x}}(W+(k+1)\mathrm{v}).$$

For convenience, let us define

$$\tau(k) \triangleq W + k\mathrm{v}; \tag{2.100}$$

$$\epsilon_{k_j} \triangleq \frac{\mathrm{v}}{W + (k_j + 1)\mathrm{v} + 1}; \tag{2.101}$$

$$\epsilon_k \triangleq \begin{bmatrix} \epsilon_{k_1} \\ \epsilon_{k_2} \end{bmatrix}. \tag{2.102}$$

Thus, similar to (2.55), we obtain

$$\overline{\mathbf{x}}(\tau(k+1)) = \begin{bmatrix} (1-\epsilon_{k_1})\,\overline{\mathbf{x}}_1(\tau(k_1)) + \epsilon_{k_1}\left(\frac{1}{\mathrm{v}}\sum_{\ell=1}^{\mathrm{v}} \mathbf{x}_1(\tau(k_1)+\ell)\right) \\ (1-\epsilon_{k_2})\,\overline{\mathbf{x}}_2(\tau(k_2)) + \epsilon_{k_2}\left(\frac{1}{\mathrm{v}}\sum_{\ell=1}^{\mathrm{v}} \mathbf{x}_2(\tau(k_2)+\ell)\right) \end{bmatrix}. \tag{2.103}$$

Recall that $\Delta = [\Delta_1^\top \ \ldots \ \Delta_m^\top]^\top$ is the perturbation term, and $\odot$ is the componentwise product. Thus, from (2.65) and (2.103), we obtain

$$\begin{aligned} \overline{\mathbf{x}}(\tau(k+1)) &= \begin{bmatrix} (1-\epsilon_{k_1})\,\overline{\mathbf{x}}_1(\tau(k_1)) + \epsilon_{k_1}\left(P\left(\overline{\mathbf{x}}_1(\tau(k_1))\right) + \Delta_1\right) \\ (1-\epsilon_{k_2})\,\overline{\mathbf{x}}_2(\tau(k_2)) + \epsilon_{k_2}\left(P\left(\overline{\mathbf{x}}_2(\tau(k_2))\right) + \Delta_2\right) \end{bmatrix} \\ &= (1-\epsilon_k) \odot \overline{\mathbf{x}}(\tau(k)) + \epsilon_k \odot P\left(\overline{\mathbf{x}}(\tau(k))\right) + \epsilon_k \odot \Delta \\ &= R_{\epsilon_k}(\overline{\mathbf{x}}(\tau(k)) + \epsilon_k \odot \Delta. \end{aligned} \tag{2.104}$$

From (2.99) and (2.104), we conclude that $\overline{\mathbf{x}}(\tau(k+1))$ is close to $R_{\epsilon_k}(\overline{\mathbf{x}}(\tau(k))$ with high probability.

The following divergence result on the sequence of independent and identically distributed random variables will be useful to prove the main theorem.

Lemma 2.4.15. *(Wirth et al., 2019, Lemma C.1) Let $\{X_\ell\}_{\ell\in\mathbb{N}}$ be a sequence of real-valued random variables that are independent and identically distributed with well-defined expectation $\mathbb{E}(X_1) < 0$ and finite variance $\mathrm{Var}(\mathrm{X}_1)$. Moreover, let us assume that the positive real-valued sequence $\{\epsilon_\ell\}_{\ell\in\mathbb{N}}$*

is not summable but square summable, then for $L \in \mathbb{N}$, we have

$$\lim_{L\to\infty} \sum_{\ell=1}^{L} \epsilon_\ell X_\ell = -\infty \text{ and}$$
$$\lim_{k\to\infty} \sup_{L\geq 0} \sum_{\ell=k}^{k+L} \epsilon_\ell X_\ell = 0, \text{ almost surely.}$$

□

We call a sequence $\{\epsilon_\ell\}_{\ell\in\mathbb{N}}$ not summable but square summable if, for $L \in \mathbb{N}$, the following holds:

$$\epsilon_\ell \geq 0, \quad \sum_{\ell=1}^{L} \epsilon_\ell^2 < \infty, \quad \sum_{\ell=1}^{L} \epsilon_\ell = \infty.$$

We now present the proof of the main theorem in the following subsection.

2.4.3 Proof of Theorem 2.4.1 (convergence of average allocation to the KKT point)

The proof of the main theorem is presented in several steps. Step 0 describes the assumptions and chosen constants. Step 1 shows that if the state trajectories $\overline{\mathbf{x}}(\tau(k))$ start in Σ^m, they enter $P_{co}(2\overline{\delta})$ almost surely, where $\overline{\delta}$ (as in Lemma 2.4.3) is chosen such that $0 < 3\overline{\delta} < \delta^*$, δ^* is defined in Equation (2.77). Moreover, for the fixed-point $\mathbf{x}^* \in \Sigma^m$ (as in Lemma 2.4.2) and radius $\zeta > 0$, Step 2 shows that if the state trajectories $\overline{\mathbf{x}}(\tau(k))$ start in $P_{co}(3\overline{\delta})$, they enter the ball $B_H(\mathbf{x}^*, \zeta)$ almost surely. Step 3 presents the stability analysis and shows that the state trajectories converge almost surely to the fixed point.

Step 0 (Choosing constants) Let us fix $\zeta > 0$. We choose δ^- such that (2.63) holds, and ϖ satisfies Lemma 2.4.9. Additionally, let C_ζ be the constant as in (2.73), guaranteed by Corollary 2.4.7. Furthermore, let $\delta^* = \min\left\{\frac{C_\zeta \exp(\zeta)}{2\varpi}, \delta^-\right\}$ as in (2.77), and $\overline{\delta}$ is chosen such that $0 < 3\overline{\delta} < \delta^*$. Let δ^+ be chosen such that (2.67) holds and $C_{\overline{\delta}}$ be as in Lemma 2.4.3. Also, let C_0 be the constant as in (2.81), guaranteed by Lemma 2.4.12. We choose $\theta \in (0, 1)$ such that

$$\theta\left(1 + C_{\overline{\delta}}\right) - (1-\theta) < 0, \quad \text{and} \quad \theta C_0 - (1-\theta)\frac{C_\zeta}{2} < 0. \tag{2.105}$$

Step 1 This step shows that the state trajectories $\overline{\mathbf{x}}(\tau(k))$ starting in Σ^m will enter $P_{\text{co}}(2\overline{\delta})$ in finite number of steps. Let t_{k_0} be the time instant at which every resource has at least one capacity event; moreover, let σ_1 be the first hitting time, defined as

$$\sigma_1 \triangleq \min\left\{k \geq k_0 \mid \overline{\mathbf{x}}(\tau(k)) \in P_{\text{co}}(2\overline{\delta})\right\}. \tag{2.106}$$

For $\overline{\delta} > 0$, if $d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta})) > \overline{\delta}$, from Lemma 2.4.3 (iii), we have

$$\mathbb{P}_{\mathbf{x}_0}\left(d_1(\overline{\mathbf{x}}(\tau(k+1)), P_{\text{co}}(\overline{\delta})) \leq (1+\epsilon_k C_{\overline{\delta}}) d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta}))\right) \leq \theta. \tag{2.107}$$

And, from Lemma 2.4.3 (i), we have

$$\mathbb{P}_{\mathbf{x}_0}\left(d_1(\overline{\mathbf{x}}(\tau(k+1)), P_{\text{co}}(\overline{\delta})) \leq (1-\epsilon_k) d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta}))\right) \geq 1-\theta. \tag{2.108}$$

For $k \in \mathbb{N}$ and $\ell = 1, 2, \ldots, k$, let a_ℓ be random variables, defined as

$$a_\ell \triangleq \begin{cases} (1+\epsilon_\ell C_{\overline{\delta}}) & \text{with probability } \theta, \\ (1-\epsilon_\ell) & \text{with probability } 1-\theta. \end{cases}$$

Notice that ϵ_ℓ and $\epsilon_\ell C_{\overline{\delta}}$ are independent of the sample path; hence, the random variables a_ℓ are independent. Thus, for $\tau(k) < \sigma_1$ (that is, $\overline{\mathbf{x}}(\tau(k)) \notin P_{\text{co}}(2\overline{\delta})$) and $d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta})) > \overline{\delta}$, from (2.107) and (2.108), we obtain

$$d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta})) \leq \prod_{\ell=1}^{k} a_\ell d_1(\overline{\mathbf{x}}(W), P_{\text{co}}(\overline{\delta})). \tag{2.109}$$

After taking the logarithm of (2.109), we obtain

$$\begin{aligned} \log\left(d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta}))\right) &\leq \log\left(\prod_{\ell=1}^{k} a_\ell\right) + \log\left(d_1(\overline{\mathbf{x}}(W), P_{\text{co}}(\overline{\delta}))\right) \\ &\leq \sum_{\ell=1}^{k} \log(a_\ell) + \log\left(d_1(\overline{\mathbf{x}}(W), P_{\text{co}}(\overline{\delta}))\right). \end{aligned} \tag{2.110}$$

For $\epsilon_\ell \in (0,1)$, we obtain $\log(1-\epsilon_\ell) < -\epsilon_\ell$ and $\log(1+\epsilon_\ell C_{\overline{\delta}}) < (1+C_{\overline{\delta}})\epsilon_\ell$. Recall that we chose $\theta \in (0,1)$ such that (2.105) holds; thus, we obtain the following expectation

$$\begin{aligned}\mathbb{E}\left[\log(a_\ell)\right] &= \theta \log(1+\epsilon_\ell C_{\overline{\delta}}) + (1-\theta)\log(1-\epsilon_\ell) \\ &< \left(\theta(1+C_{\overline{\delta}}) - (1-\theta)\right)\epsilon_\ell < 0.\end{aligned}$$

From Lemma 2.4.15, we have $\lim_{k\to\infty} \sum_{\ell=1}^{k} \log(a_\ell) = -\infty$.

Thus, from (2.110), we obtain $\lim_{k\to\infty} \log\left(d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta}))\right) = -\infty$, almost surely, as long as $\overline{\mathbf{x}}(\tau(k)) \notin P_{\text{co}}(2\overline{\delta})$. So, $\lim_{k\to\infty} d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta})) = 0$, almost surely, as long as $\overline{\mathbf{x}}(\tau(k)) \notin P_{\text{co}}(2\overline{\delta})$; which contradicts that $d_1(\overline{\mathbf{x}}(\tau(k)), P_{\text{co}}(\overline{\delta})) > \overline{\delta} > 0$. Hence, for a finite k, we have $\overline{\mathbf{x}}(\tau(k)) \in P_{\text{co}}(2\overline{\delta})$, almost surely.

Step 2 For $\zeta > 0$, and for the fixed point $\mathbf{x}^* \in \Sigma^m$, this step shows that the state trajectories $\overline{\mathbf{x}}(\tau(k))$ starting in $P_{\text{co}}(3\overline{\delta})$ will reach the ball $B_H(\mathbf{x}^*, \zeta)$ in a finite number of steps. From Lemma 2.4.12, if $\overline{\mathbf{x}}(\tau(k)) \in P_{\text{co}}(3\overline{\delta})$, then we have

$$\mathbb{P}_{\mathbf{x}_0}\left(D_H\left(\overline{\mathbf{x}}(\tau(k+1)), \mathbf{x}^*\right) \leq D_H\left(\overline{\mathbf{x}}(\tau(k)), \mathbf{x}^*\right) + \epsilon_k C_0\right) \leq \theta. \tag{2.111}$$

For $\epsilon_\ell \in (0,1)$, we obtain $\log\left(1-\epsilon_\ell \frac{C_\zeta}{2}\right) < -\epsilon_\ell \frac{C_\zeta}{2}$. Thus, for $W_0 \in \mathbb{N}$ and all $W \geq W_0$, and for $D_H\left(\overline{\mathbf{x}}(\tau(k)), \mathbf{x}^*\right) \geq \zeta$, from Corollary 2.4.11 and Equation (2.99), we have

$$\mathbb{P}_{\mathbf{x}_0}\left(D_H\left(\overline{\mathbf{x}}(\tau(k+1)), \mathbf{x}^*\right) < D_H\left(\overline{\mathbf{x}}(\tau(k)), \mathbf{x}^*\right) - \epsilon_k \frac{C_\zeta}{2}\right) \geq 1-\theta. \tag{2.112}$$

For $k \in \mathbb{N}$ and $\ell = 1, 2, \ldots, k$, let b_ℓ be random variables, defined as

$$\mathrm{b}_\ell \triangleq \begin{cases} \epsilon_\ell C_0 & \text{with probability } \theta, \\ -\epsilon_\ell \dfrac{C_\zeta}{2} & \text{with probability } 1-\theta. \end{cases}$$

The random variables b_ℓ are independent by a similar argument as the random variables a_ℓ. Thus, if $\overline{\mathbf{x}}(\tau(k)) \in P_{\text{co}}(3\overline{\delta})$ and $D_H\left(\overline{\mathbf{x}}(\tau(k)), \mathbf{x}^*\right) \geq \zeta$, then from Equations (2.111) and (2.112), we obtain

$$D_H\left(\overline{\mathbf{x}}\left(\sigma_1(\overline{\mathbf{x}}(W)) + k\mathrm{v}\right), \mathbf{x}^*\right) \leq D_H\left(\overline{\mathbf{x}}\left(\sigma_1(\overline{\mathbf{x}}(W))\right), \mathbf{x}^*\right) + \sum_{\ell=1}^{k} \mathrm{b}_\ell. \tag{2.113}$$

We chose $\theta \in (0,1)$ such that (2.105) holds; thus, we obtain

$$\mathbb{E}\left[\mathrm{b}_\ell\right] < \left(\theta C_0 - (1-\theta)\frac{C_\zeta}{2}\right)\epsilon_\ell < 0.$$

As $\mathbb{E}\left[\mathrm{b}_\ell\right] < 0$; from Lemma 2.4.15, we have $\lim_{k\to\infty} \sum_{\ell=1}^{k} \mathrm{b}_\ell = -\infty$, almost surely. Thus, from (2.113), we obtain $\lim_{k\to\infty} D_H\left(\overline{\mathbf{x}}\left(\sigma_1(\overline{\mathbf{x}}(W)) + k\mathrm{v}\right), \mathbf{x}^*\right) = -\infty$, which is a contradiction to the assumption that $D_H\left(\overline{\mathbf{x}}(\tau(k)), \mathbf{x}^*\right) \geq \zeta > 0$. Thus, if the state trajectories $\overline{\mathbf{x}}(\tau(k))$ start in $P_{\text{co}}(3\overline{\delta})$, then the distance between $\overline{\mathbf{x}}(\tau(k))$ and the fixed point $\mathbf{x}^*$ on the metric D_H will be non-negative and smaller than ζ. Hence, almost surely, the state trajectories enter the ball $B_H(\mathbf{x}^*, \zeta)$ in a finite number of steps.

Step 3 (Stability analysis) We present the following stability analysis to show almost sure convergence of $\overline{\mathbf{x}}(\tau(k))$ to the fixed point $\mathbf{x}^*$. Moreover, for large $k \in \mathbb{N}$, we show that $\overline{\mathbf{x}}(\tau(k)) \in B_H(\mathbf{x}^*, \zeta)$, almost surely. To do so, we fix $\frac{\zeta}{3} > 0$ and choose the constants described in Step 0 for this fixed value. After repeating Steps 1 and 2 for the fixed value $\frac{\zeta}{3}$, we deduce that $\overline{\mathbf{x}}(\tau(k)) \in B_H(\mathbf{x}^*, \frac{\zeta}{3})$, almost surely. Furthermore, we consider the radius $\frac{2\zeta}{3}$; from (2.113), we derive that if $\overline{\mathbf{x}}(\tau(k)) \in B_H(\mathbf{x}^*, \frac{2\zeta}{3})$, then the state trajectories will exit the ball $B_H(\mathbf{x}^*, \zeta)$ only if $\sum_{\ell=k}^{k+L} \mathrm{b}_\ell > \frac{\zeta}{3} > 0$. On the contrary, for $k, L \in \mathbb{N}$, from Lemma 2.4.15, we know that $\sum_{\ell=k}^{k+L} \mathrm{b}_\ell \leq 0$ for large k. Thus, almost surely, the state trajectories do not exit the ball $B_H(\mathbf{x}^*, \zeta)$ infinitely often. Hence, as ζ is arbitrary, our claim that $\overline{\mathbf{x}}(k)$ converges to the fixed point $\mathbf{x}^*$ almost surely, holds.

2.5 Numerical results

In this section, we use numerical results to illustrate the evolution of the average allocations over time. Although the convergence of average allocations to an optimal limit is guaranteed under

general assumptions, we observe that the number of iterations required for the convergence depends intricately on the number of agents, their characteristic cost functions, additive-increase parameters and the values of Γ_1 and Γ_2. The optimal limit is obtained using a solver, whereas the average allocations are obtained by instantaneous allocations averaged over all past capacity events of the resources.

2.5.1 Analysis 1

In numerical Analysis 1, we consider 45 agents that compete to access two shared resources with capacities $\mathcal{C}_1 = 5$ and $\mathcal{C}_2 = 6$. Each agent i's cost f_i is a function of its average allocations of both resources. The cost functions are given in Figure 2.1: The agents come in three classes. Agents 1 to 15 choose costs from the first case of Equation (2.114), agents 16 to 30 choose costs from the second case, and agents 31 to 45 choose costs from the third case. Therein, we generate uniformly distributed (integer) random variables $a_i \in [10, 30]$, $b_i \in [15, 35]$, $c_i \in [1, 3]$, $d_i \in [1, 5]$, $e_i \in [1, 4]$, $g_i \in [15, 25]$, and $h_i \in [10, 20]$, for each agent at the start of the experiment to obtain different costs for agents at a time step.

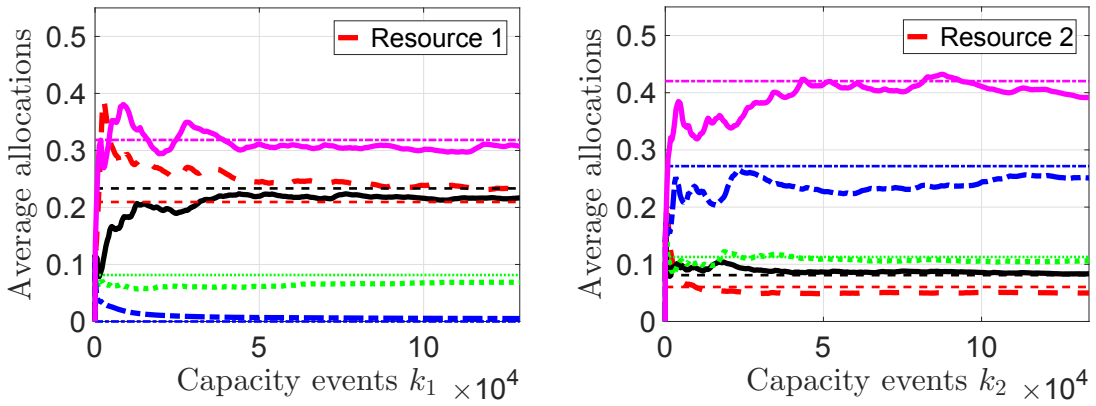
$$f_i(x_i, y_i) = \begin{cases} \frac{a_i}{2}x_i^2 + \frac{b_i}{4}x_i^4 + \frac{b_i}{2}y_i^2 + \frac{a_i}{4}y_i^4 + \frac{h_i}{4}(c_i x_i + d_i y_i)^4 & i = 1, \ldots, 15, \\ \frac{b_i}{2}x_i^2 + \frac{b_i}{4}y_i^2 + \frac{g_i}{4}(d_i x_i + e_i y_i)^4 & i = 16, \ldots, 30, \\ \frac{b_i}{2}x_i^4 + \frac{b_i}{3}y_i^4 + \frac{g_i}{4}(c_i x_i + e_i y_i)^4 & i = 31, \ldots, 45. \end{cases} \tag{2.114}$$

Figure 2.1: The cost functions used in numerical Analysis 1 to plot Figures 2.3,2.4, 2.5, 2.6.

$$f_i(x_i, y_i) = \begin{cases} \frac{a_i}{2}x_i^2 + \frac{a_i}{4}x_i^4 + \frac{a_i}{2}y_i^2 + \frac{a_i}{4}y_i^4 + \frac{a_i}{4}(c_i x_i + c_i y_i)^4 & i = 1, 2, \\ \frac{a_i}{2}x_i^2 + \frac{a_i}{4}y_i^2 + \frac{a_i}{4}(c_i x_i + c_i y_i)^4 & i = 3, 4, \\ \frac{a_i}{2}x_i^4 + \frac{a_i}{3}y_i^4 + \frac{a_i}{2}(c_i x_i + c_i y_i)^2 & i = 5, 6. \end{cases} \tag{2.115}$$

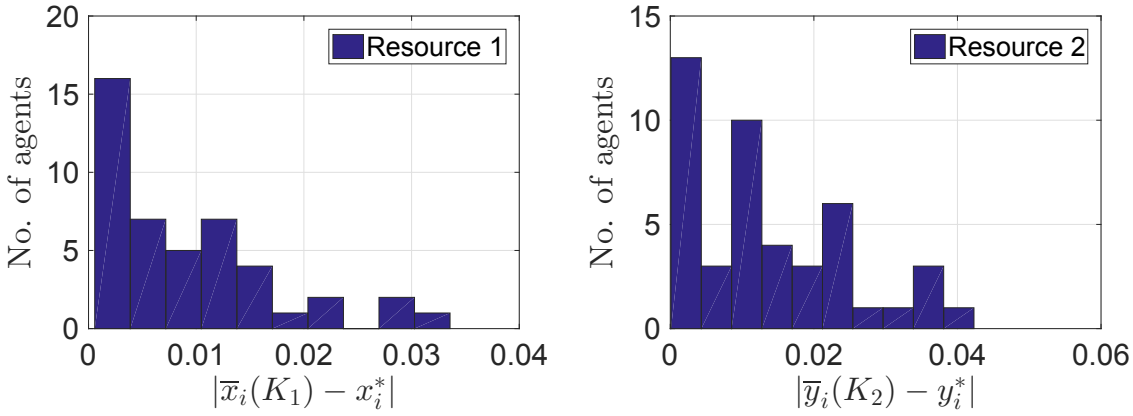
Figure 2.2: The cost functions used in Analysis 2 to plot Figure 2.8.

Figures 2.3, 2.4, 2.5, and 2.6 summarize the numerical Analysis 1. Figure 2.3 shows that the average allocations of agents for both resources converge over time to the solver's optimal values, x_i^*, y_i^*. Figure 2.4 shows the distribution of the absolute difference between the average allocations and the solver's optimal values at the last capacity events of resources 1 and 2, which is denoted by K_1 and K_2, respectively.



(a) (b)

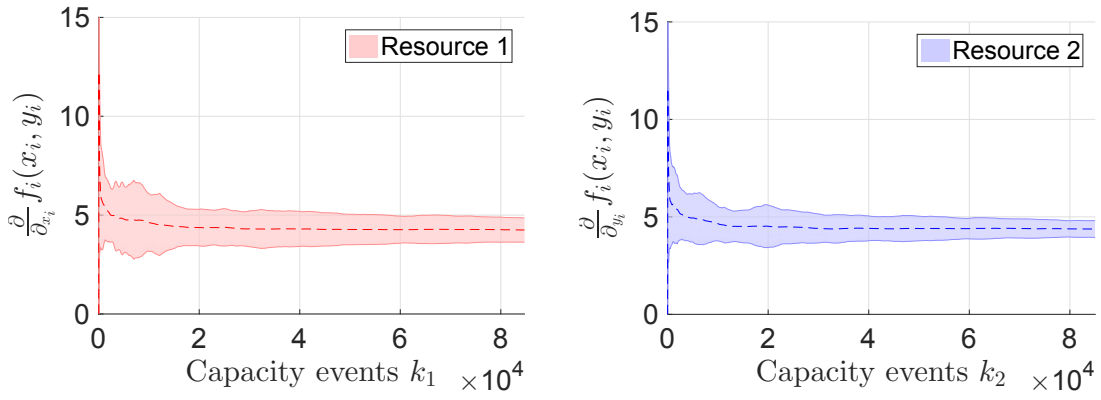
Figure 2.3: Evolution of the average allocations: (a) of resource 1, (b) of resource 2, for five randomly chosen agents, for Analysis 1. The cost functions used in Analysis 1 is presented in Equation (2.114).



(a) (b)

Figure 2.4: Histogram of the absolute difference between the average allocations and the solver's optimal values at the last capacity event of (a) resource 1 and (b) resource 2, for Analysis 1.

Figure 2.5 presents the partial derivatives of the cost functions of agents with the shaded error bars suggesting the width of one standard deviation either way of the mean. As derived in Section 2.2.2, to achieve optimality for the optimization Problem (2.3), the partial derivatives of all agents' cost functions for a particular resource should be in consensus, that is, for agents i, u, we should have $\frac{\partial}{\partial x_i} f_i(x_i, y_i) = \frac{\partial}{\partial x_u} f_u(x_u, y_u)\Big|_{x_i=x_i^*, y_i=y_i^*, x_u=x_u^*, y_u=y_u^*}$; analogously, $\frac{\partial}{\partial y_i} f_i(x_i, y_i) = \frac{\partial}{\partial y_u} f_u(x_u, y_u)\Big|_{x_i=x_i^*, y_i=y_i^*, x_u=x_u^*, y_u=y_u^*}$ (cf. (2.14)). Figure 2.5 shows that the error of partial derivatives across all agents decreases over time; in other words, the partial derivatives concentrate over time around the same value, so they are in consensus, eventually.



(a) (b)

Figure 2.5: Evolution of partial derivatives of the cost functions: (a) for resource 1, (b) for resource 2. Error bars are over the mean of partial derivatives of all agents and the error of one standard deviation, for Analysis 1.

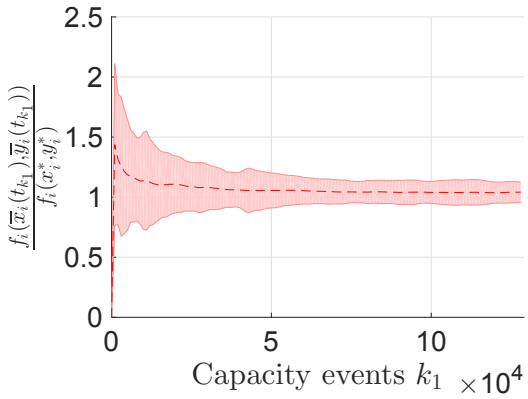
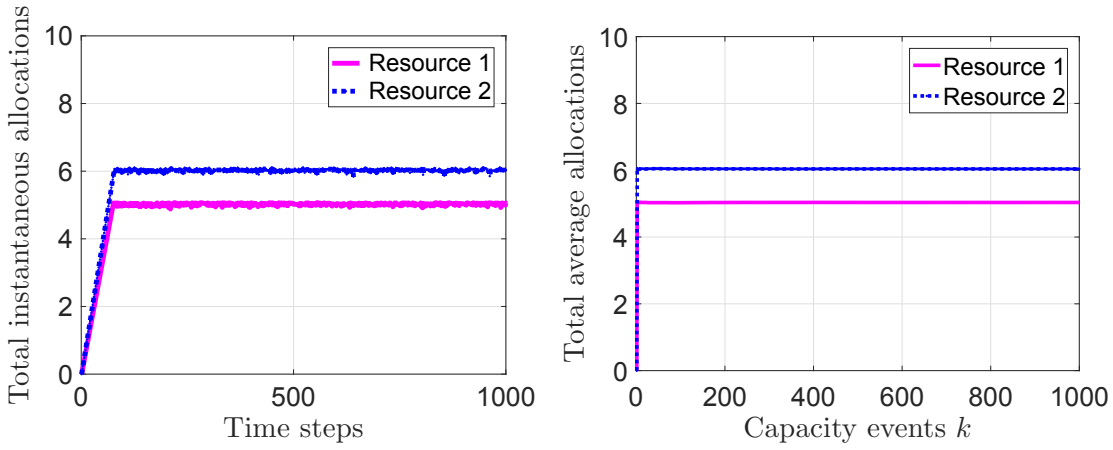


Figure 2.6: (a) Evolution of the ratio of cost over average allocations and the solver's optimal cost; error bar is over the mean of the ratio of cost of all agents and the error of one standard deviation, for Analysis 1.

Figure 2.6 illustrates that the ratio of costs over average allocations and the solver's optimal costs across all agents concentrate close to 1 over time, corroborating our claim of optimality of Problem (2.3). Note that when we say the ratio of total costs, we mean $\frac{\sum_{i=1}^n f_i(\overline{x}_i(t_k),\overline{y}_i(t_k))}{\sum_{i=1}^n f_i(x_i^*,y_i^*)}$, where t_k is the time-step at which the k'th capacity event occurs. Furthermore, $\overline{x}_i(t_k)$ is agent i's average allocation over the capacity events of resource 1 up to time-step t_k; analogously, $\overline{y}_i(t_k)$ is defined.

Figure 2.7(a) shows the evolution of aggregate instantaneous allocations $\sum_{i=1}^n x_i(\nu)$ and $\sum_{i=1}^n y_i(\nu)$ at time steps $\nu \in \mathbb{N}$; we observe that it is concentrated around the respective resource capacity.

Furthermore, Figure 2.7(b) shows the evolution of aggregate average allocations $\sum_{i=1}^{n} \overline{x}_i(t_k)$ and $\sum_{i=1}^{n} \overline{y}_i(t_k)$ at capacity events; we observe that it is concentrated very close to the respective resource capacity.



(a) (b)

Figure 2.7: (a) Evolution of aggregate instantaneous allocations of resources, (b) evolution of aggregate average allocations of resources, capacities are $\mathcal{C}_1 = 5$ and $\mathcal{C}_2 = 6$, for Analysis 1.

2.5.2 Analysis 2

In *numerical Analysis 2*, we consider 6 agents that compete to access two shared resources with capacities $\mathcal{C}_1 = 5$ and $\mathcal{C}_2 = 6$. To analyze the rate of convergence, we present the number of iterations required for average allocations to reach close to the optimal limit. We show this through the evolution of the ratio of total cost over average allocations and the solver's optimal cost.

Similar to Analysis 1, we generate random coefficients of the cost functions. We consider the cost functions presented in (2.115) of Figure 2.2. The coefficients a_i, c_i follow (folded) normal distribution $\mathcal{N}(\mu, \sigma^2)$—wherein the absolute values of the random variables are considered. Here, μ denotes the mean, and σ^2 denotes the variance of the random variables.

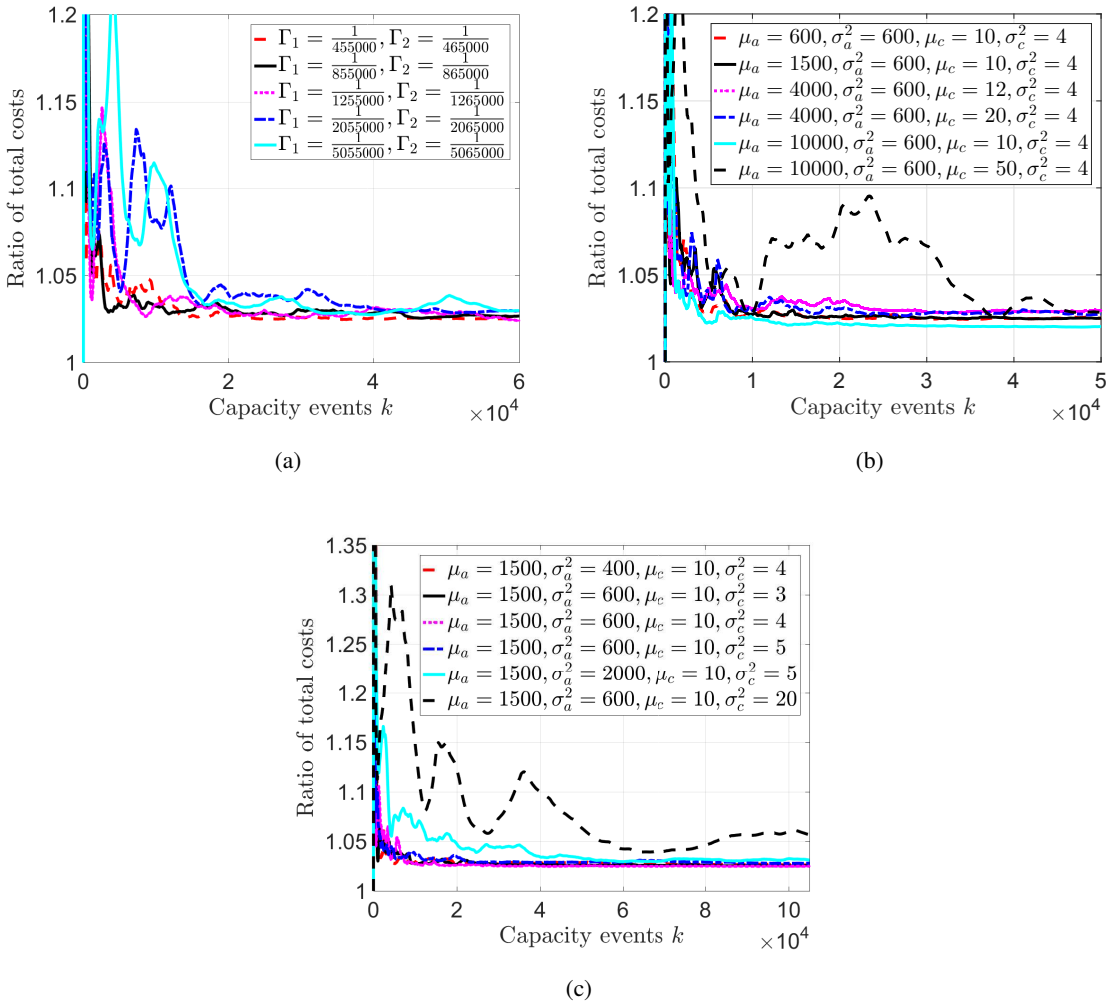


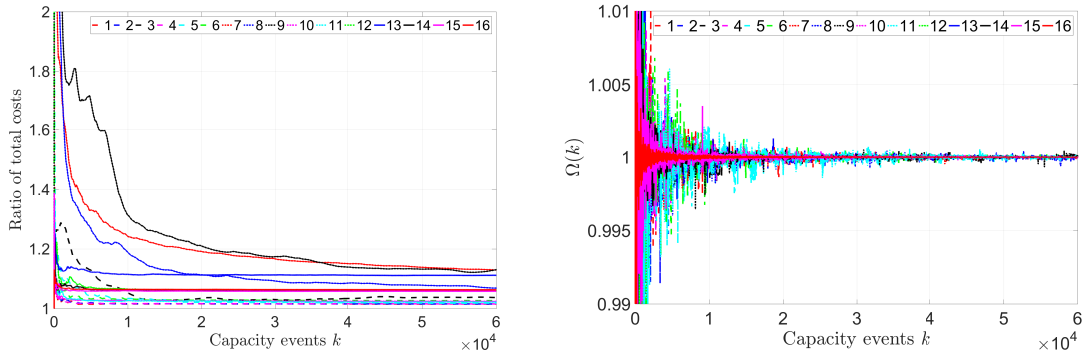
Figure 2.8: Evolution of the ratio of total cost over average allocations and the solver's total optimal cost: (a) with different values of Γ_1, Γ_2, (b) with different values of mean, and (c) with different values of variance, of the random variables, for numerical Analysis 2. The cost functions are listed in Equation (2.115). μ_a denotes the mean of the random variables (vector) $\mathbf{a} = (a_1, \ldots, a_6)$; μ_c denotes the mean of the random variables $\mathbf{c} = (c_1, \ldots, c_6)$, with folded normal distribution. In the folded normal distribution, the absolute values of the random variables are considered. Moreover, σ_a^2 denotes the variance of the random variables $\mathbf{a}$, and σ_c denotes the variance of the random variables $\mathbf{c}$. Subfigure (a) is based on $\mu_a = 1500$, $\sigma_a^2 = 600$, $\mu_c = 10$, and $\sigma_c^2 = 4$. The cost functions used in Analysis 2 is presented in Equation (2.115).

We observe that as the values of Γ_1 and Γ_2 decrease, the number of iterations required for the convergence increases, shown in Figure 2.8(a). We also observe that as the mean of the parameter random variables (coefficients of the characteristic cost functions) increases, the number of

iterations required to converge to the optimal values increases, shown in Figure 2.8(b). Furthermore, we observe that as the variance of the coefficients of the cost functions increases, the number of iterations required to converge to optimal values increases, shown in Figure 2.8(c). Finally, as the additive-increase parameters increase, the number of iterations required for the convergence increases. Note that the increase of mean in combination with the increase of variance plays a significant role in increasing the number of iterations required for the convergence, shown in Figure 2.9.

2.5.3 Analysis 3

Although the same conclusion holds for the rate of convergence described in Analysis 2, to capture different cost functions (see Table 2.1), we conduct several experiments in *Analysis 3*. Again we consider 6 agents that compete to access two shared resources with capacities $\mathcal{C}_1 = 5$ and $\mathcal{C}_2 = 6$. The coefficients of the cost functions are uniformly distributed random variables. We present the evolution of the ratio of total costs in Figure 2.9(a). The parameters and the corresponding legends of Figure 2.9 are listed in Tables 2.2, 2.3, and 2.4. Moreover, let $\Omega(k)$ denote the ratio $\frac{\sum_{i=1}^{n} f_i(\overline{x}_i(t_{k+1}),\overline{y}_i(t_{k+1})) - \sum_{i=1}^{n} f_i(x_i^*,y_i^*)}{\sum_{i=1}^{n} f_i(\overline{x}_i(t_k),\overline{y}_i(t_k)) - \sum_{i=1}^{n} f_i(x_i^*,y_i^*)}$ and Ω^* denote the rate of convergence. Thus, for a large k, we obtain $\Omega(k) \approx \Omega^*$. We present the evolution of $\Omega(k)$ in Figure 2.9(b), we observe that Ω^* is close to 1.



(a) (b)

Figure 2.9: (a) Evolution of the ratio of total cost over average allocations and the solver's total optimal cost, (b) evolution of $\Omega(k)$, for Analysis 3.

Coefficients	$f_i(x_i, y_i)$
$g_i = h_i = 0$	$$\begin{cases} \frac{a_i}{2}x_i^2 + \frac{b_i}{4}x_i^4 + \frac{b_i}{2}y_i^2 + \frac{a_i}{4}y_i^4, \\ \frac{b_i}{2}x_i^2 + \frac{b_i}{4}y_i^2, \\ \frac{b_i}{2}x_i^4 + \frac{b_i}{3}y_i^4. \end{cases} \quad (2.116)$$ Separable functions.
$c_i = d_i = e_i = 1$	$$\begin{cases} \frac{a_i}{2}x_i^2 + \frac{b_i}{4}x_i^4 + \frac{b_i}{2}y_i^2 + \frac{a_i}{4}y_i^4 + \frac{h_i}{4}(x_i + y_i)^4, \\ \frac{b_i}{2}x_i^2 + \frac{b_i}{4}y_i^2 + \frac{g_i}{4}(x_i + y_i)^4, \\ \frac{b_i}{2}x_i^4 + \frac{b_i}{3}y_i^4 + \frac{g_i}{4}(x_i + y_i)^4. \end{cases} \quad (2.117)$$ Non-separable functions.

Table 2.1: Cost functions used in Analysis 3, these are special cases of cost functions in (2.114).

Function type	Parameters	α_1, α_2	β_1, β_2	Γ_1	Γ_2	Legend
(2.116)	$a_i, b_i \in [0, 1]$.	0.01, 0.0125	0.7, 0.6	$\frac{1}{5}$	$\frac{1}{5}$	1
	$a_i \in [10, 30]$, $b_i \in [15, 35]$.			$\frac{1}{500}$	$\frac{1}{500}$	2
				$\frac{1}{25500}$	$\frac{1}{26000}$	3
	$a_i \in [40, 100]$, $b_i \in [5, 150]$.			$\frac{1}{2000}$	$\frac{1}{2000}$	4

Table 2.2: Parameters for Analysis 3, and the legends used in Figure 2.9.

Function type	Parameters	α_1, α_2	β_1, β_2	Γ_1	Γ_2	Legend
(2.114)	$a_i, b_i \in [0,1]$, $c_i, d_i, e_i \in [0,1]$, $g_i, h_i \in [0,1]$.	$0.01, 0.0125$	0.7, 0.6	$\frac{1}{25}$	$\frac{1}{25}$	5
	$a_i \in [10,30]$, $b_i \in [15,35]$, $c_i, d_i, e_i \in [1,2]$, $g_i \in [10,20]$, $h_i \in [15,25]$.			$\frac{1}{8500}$	$\frac{1}{8800}$	6
	$a_i \in [10,30]$, $b_i \in [15,35]$, $c_i, d_i, e_i \in [1,3]$, $g_i \in [10,20]$, $h_i \in [15,25]$.			$\frac{1}{28500}$	$\frac{1}{28800}$	7
	$a_i \in [10,30]$, $b_i \in [15,35]$, $c_i \in [1,5]$, $d_i \in [1,4]$, $e_i \in [1,3]$, $g_i \in [10,20]$, $h_i \in [15,25]$.			$\frac{1}{89500}$	$\frac{1}{90600}$	8
	$a_i \in [10,30]$, $b_i \in [15,35]$, $c_i \in [1,6]$, $d_i \in [1,5]$, $e_i \in [1,3]$, $g_i \in [10,20]$, $h_i \in [15,25]$.			$\frac{1}{220500}$	$\frac{1}{225600}$	9

Table 2.3: Parameters for Analysis 3, and the legends used in Figure 2.9.

Function type	Parameters	α_1, α_2	β_1, β_2	Γ_1	Γ_2	Legend
(2.117)	$a_i, b_i \in [0,1]$, $g_i, h_i \in [0,1]$.	0.01, 0.0125	0.7, 0.6	$\frac{1}{30}$	$\frac{1}{30}$	10
	$a_i \in [40,100]$, $b_i \in [5,150]$, $g_i \in [10,20]$, $h_i \in [15,25]$.			$\frac{1}{7000}$	$\frac{1}{7000}$	11
		0.03, 0.035				12
		0.05, 0.06				13
		0.03, 0.035	0.4, 0.5			14
			0.2, 0.3	$\frac{1}{2000}$	$\frac{1}{2000}$	15
			0.8, 0.9.			16

Table 2.4: Parameters for Analysis 3, and the legends used in Figure 2.9.

Finally, we present a comparative analysis of single resource allocation algorithm and the proposed multi-resource allocation algorithm in Appendix 2.8.

2.6 Conclusion

We proposed a stochastic distributed iterative multi-resource allocation algorithm. It is based on the additive-increase multiplicative-decrease (AIMD) algorithm. Wirth and his co-authors (Wirth et al., 2019) modified the basic AIMD algorithm to solve a class of distributed optimization problems for a single resource with no inter-agent communication but little with a central agent. We extended those results for multiple heterogeneous divisible resources. We modeled the system as a non-homogeneous Markov chain with place-dependent probabilities and presented the proof of convergence of average allocations to optimal allocations. Moreover, we presented the numerical results to corroborate the efficacy of the proposed algorithm. It would be of considerable interest to

apply the algorithm to real-world applications.

2.7 Appendix: Analysis of the total instantaneous demands

This section presents an analysis of the sum of instantaneous demands of all agents at a time instant. We show that the sum does not exceed the capacity of the respective resource. We also verify that the total demands at capacity events are equal.

Recall that we consider all agents have the same additive increase factor that is $\alpha_{ji} = \alpha_j = \alpha_{ju}$, for $i, u = 1, 2, \ldots, n$. Hence, for ease of analysis, we can rewrite (2.35) as

$$x_{ji}(t_{k_j+1}) = \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) + (t_{k_j+1} - t_{k_j})\alpha_{ji}. \tag{2.118}$$

We now want to calculate the sum of instantaneous demands of all agents at time instant t_ν, $\nu \in \mathbb{N}$, after the occurence of the k_j'th capacity event but before the $k_j + 1$'th capacity event. Moreover, let $\Delta t = t_\nu - t_{k_j} \leq t_{k_{j+1}} - t_{k_j}$ denote the time-lapsed (duration) after the k_j'th capacity event. We obtain the instantaneous demand of agent i at time instant t_ν as

$$\begin{aligned} x_{ji}(t_\nu) &= \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) + (\Delta t)\alpha_{ji} \\ &\leq \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) + (t_{k_{j+1}} - t_{k_j})\alpha_{ji}. \end{aligned} \tag{2.119}$$

By taking the sum of (2.119) of all agents in the network, we obtain

$$\begin{aligned} \sum_{i=1}^{n} x_{ji}(t_\nu) &\leq \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) + \frac{\mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j})}{n\alpha_j} \sum_{i=1}^{n} \alpha_{ji} \\ &= \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) + \frac{\mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j})}{n\alpha_j} n\alpha_j \\ &= \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) + \mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j)x_{ji}(t_{k_j}) \\ &= \mathcal{C}_j. \end{aligned}$$

Hence, we obtain

$$\sum_{i=1}^{n} x_{ji}(t_\nu) \leq \mathcal{C}_j.$$

Moreover, by following similar steps as previous, it is straightforward to show that $\sum_{i=1}^{n} x_{ji}(t_{k_j+1}) = \mathcal{C}_j$—the sum of instantaneous demands at a capacity event is equal to the resource capacity. Thus, by taking the sum of (2.35) of all agents in the network, we obtain

$$\begin{aligned}
\sum_{i=1}^{n} x_{ji}(t_{k_j+1}) &= \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j}) + \frac{\mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j})}{n\alpha_j} \sum_{i=1}^{n} \alpha_{ji} \\
&= \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j}) + \frac{\mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j})}{n\alpha_j} n\alpha_j \\
&= \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j}) + \mathcal{C}_j - \sum_{i=1}^{n} \boldsymbol{\beta}_{ji}(k_j) x_{ji}(t_{k_j}) \\
&= \mathcal{C}_j.
\end{aligned}$$

Finally, we proceed as follows to verify that the total demands at capacity events are equal, that is, $\sum_{i=1}^{n} x_{ji}(t_{k_j+1}) = \sum_{i=1}^{n} x_{ji}(t_{k_j})$, for $k_j \in \mathbb{N}$. Recall that $x_{ji}(k_j)$ denotes $x_{ji}(t_{k_j})$ and $\mathbf{x}_j(k_j)$ denotes $\begin{bmatrix} x_{j1}(k_j) & x_{j2}(k_j) & \dots & x_{jn}(k_j) \end{bmatrix}^\top$. We formulate the evolution of the instantaneous demands of resource j at capacity events as follows (see (2.36) for the AIMD matrix for resource j):

$$\begin{aligned}
&\mathbf{x}_j(k_j+1) \\
&= \left(\begin{bmatrix} \boldsymbol{\beta}_{j1}(k_j) & 0 & \dots & 0 \\ 0 & \boldsymbol{\beta}_{j2}(k_j) & \dots & 0 \\ \vdots & & & \vdots \\ 0 & & \dots & \boldsymbol{\beta}_{jn}(k_j) \end{bmatrix} + \frac{1}{n} \mathbf{e} \begin{bmatrix} 1 - \boldsymbol{\beta}_{j1}(k_j) & \dots & 1 - \boldsymbol{\beta}_{jn}(k_j) \end{bmatrix} \right) \mathbf{x}_j(k_j).
\end{aligned} \tag{2.120}$$

After slight mathematical manipulation, we obtain

$$\mathbf{x}_j(k_j+1) = \begin{bmatrix} \boldsymbol{\beta}_{j1}(k_j)x_{j1}(k_j) \\ \boldsymbol{\beta}_{j2}(k_j)x_{j2}(k_j) \\ \vdots \\ \boldsymbol{\beta}_{jn}(k_j)x_{jn}(k_j) \end{bmatrix} + \frac{1}{n} \begin{bmatrix} (1-\boldsymbol{\beta}_{j1}(k_j))x_{j1}(k_j) + \ldots + (1-\boldsymbol{\beta}_{jn}(k_j))x_{jn}(k_j) \\ \vdots \\ (1-\boldsymbol{\beta}_{j1}(k_j))x_{j1}(k_j) + \ldots + (1-\boldsymbol{\beta}_{jn}(k_j))x_{jn}(k_j) \end{bmatrix}. \tag{2.121}$$

By taking the sum of instantaneous demands of all agents, from (2.121), we obtain

$$\begin{aligned} \sum_{i=1}^{n} x_{ji}(k_j+1) &= \boldsymbol{\beta}_{j1}(k_j)x_{j1}(k_j) + \boldsymbol{\beta}_{j2}(k_j)x_{j2}(k_j) + \ldots + \boldsymbol{\beta}_{jn}(k_j)x_{jn}(k_j) \\ &\quad + ((1-\boldsymbol{\beta}_{j1}(k_j))x_{j1}(k_j) + \ldots + (1-\boldsymbol{\beta}_{jn}(k_j))x_{jn}(k_j)) \\ &= \boldsymbol{\beta}_{j1}(k_j)x_{j1}(k_j) + \boldsymbol{\beta}_{j2}(k_j)x_{j2}(k_j) + \ldots + \boldsymbol{\beta}_{jn}(k_j)x_{jn}(k_j) \\ &\quad + \sum_{i=1}^{n} x_{ji}(k_j) - (\boldsymbol{\beta}_{j1}(k_j)x_{j1}(k_j) + \ldots + \boldsymbol{\beta}_{jn}(k_j)x_{jn}(k_j)) \\ &= \sum_{i=1}^{n} x_{ji}(k_j). \end{aligned}$$

Thus, the sum of agents' instantaneous demands at capacity events is the same. Hence, we have $\sum_{i=1}^{n} x_{ji}(k_j+1) = \sum_{i=1}^{n} x_{ji}(k_j) = \mathcal{C}_j$, for $j = 1, 2, \ldots, m$.

2.8 Appendix: Comparison of numerical results of single resource and multiple resource cases

This section presents several experiments to compare the results obtained by the single resource allocation algorithm and the proposed multi-resource allocation algorithm. Recall that, in the single resource case, the cost function of an agent depends on the average allocation of a single resource. In contrast, in the multi-resource case, the cost functions are coupled through average allocations of multiple resources. Subsection 2.8.1 presents results with separable cost functions. It shows that the sum of the total costs obtained by the single resource algorithms and the total cost by

the multi-resource algorithm converges to the same value. Subsection 2.8.2 presents results with non-separable cost functions. Our multi-resource allocation algorithms provide close to optimal values (by a solver). In contrast, the single resource allocation algorithms are not efficient for such scenarios and provide suboptimal solutions.

2.8.1 Separable cost function

For the single resource case, the cost functions for resource 1 are listed in (2.122) and the cost functions for resource 2 are listed in (2.123). The cost functions for the multi-resource case are listed in (2.124), wherein the cost functions are coupled through average allocations of two resources. For an accurate comparison of results, we chose separable cost functions. We consider 6 agents and two resources in the network. We chose the resource capacities $\mathcal{C}_1 = 5$ for resource 1 and $\mathcal{C}_2 = 6$ for resource 2, respectively. Different costs are generated with uniformly distributed (integer) random coefficients $a_i \in [10, 30]$, $b_i \in [15, 35]$, $c_i \in [20, 30]$ at a time step. Additionally, coefficients of the cost functions, a_i, b_i, c_i, are generated at the start of the algorithm. The same values are used in each of the three analyses—single resource analysis with resource 1, single resource analysis with resource 2, and multi-resource (coupled) analysis with resources 1 and 2.

$$f_i(x_i) = \begin{cases} \frac{a_i}{2}x_i^2 + \frac{b_i}{4}x_i^4 & i = 1, 2, \\ \frac{b_i}{2}x_i^2 & i = 3, 4, \\ \frac{b_i}{2}x_i^4 & i = 5, 6. \end{cases} \tag{2.122}$$

Figure 2.10: The cost functions for allocating resource 1.

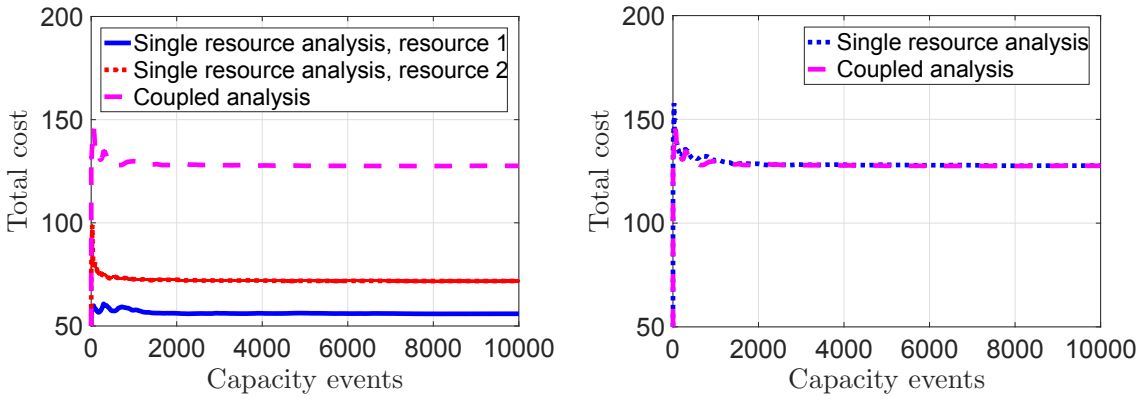
$$f_i(y_i) = \begin{cases} \frac{c_i}{2} y_i^2 + \frac{b_i}{4} y_i^4 & i = 1, 2, \\ \frac{c_i}{2} y_i^2 & i = 3, 4, \\ \frac{a_i}{2} y_i^4 & i = 5, 6. \end{cases} \tag{2.123}$$

Figure 2.11: The cost functions for allocating resource 2.

$$f_i(x_i, y_i) = \begin{cases} \frac{a_i}{2} x_i^2 + \frac{b_i}{4} x_i^4 + \frac{c_i}{2} y_i^2 + \frac{b_i}{4} y_i^4 & i = 1, 2, \\ \frac{b_i}{2} x_i^2 + \frac{c_i}{2} y_i^2 & i = 3, 4, \\ \frac{b_i}{2} x_i^4 + \frac{a_i}{2} y_i^4 & i = 5, 6. \end{cases} \tag{2.124}$$

Figure 2.12: The cost functions for allocating resources 1 and 2.

Figure 2.13(a) illustrates the evolution of the total cost over average allocations obtained by single resource allocation algorithm for resource 1, single resource allocation algorithm for resource 2, and multi-resource allocation algorithm for resources 1 and 2. We observe that in all three cases, the total costs over average allocations converge over time. Figure 2.13(b) shows the evolution of the sum of the total costs by single resource algorithms for resource 1 and resource 2 that is $\sum_{i=1}^{n} f_i(\overline{x}_i(t_k)) + \sum_{i=1}^{n} f_i(\overline{y}_i(t_k))$, and the evolution of the total cost by the multi-resource allocation algorithm $\sum_{i=1}^{n} f_i(\overline{x}_i(t_k), \overline{y}_i(t_k))$. We observe that the sum of the total costs obtained by the single resource algorithms, and the total cost by the multi-resource algorithm converge to the same value.



(a) (b)

Figure 2.13: (a) Evolution of the total cost over average allocations by single resource algorithm for resource 1, single resource algorithm for resource 2, and multi-resource algorithm for resources 1 and 2, (b) evolution of the sum of the total costs by single resource algorithms for resource 1 and resource 2, and evolution of the total cost by the multi-resource algorithm for resources 1 and 2.

We now aim to obtain close to the single resource allocation results for resource 1 from the multi-resource allocation algorithm. We do so by assigning the capacity of resource 2 to 0 so that resource 2 is not active; only resource 1 is allocated. We compare the results in Figure 2.14 that illustrates the evolution of the total cost over average allocations obtained by single resource algorithm for resource 1, and multi-resource allocation algorithm with $\mathcal{C}_2 = 0$. We observe that the total costs over average allocations reach very close to each other over time.

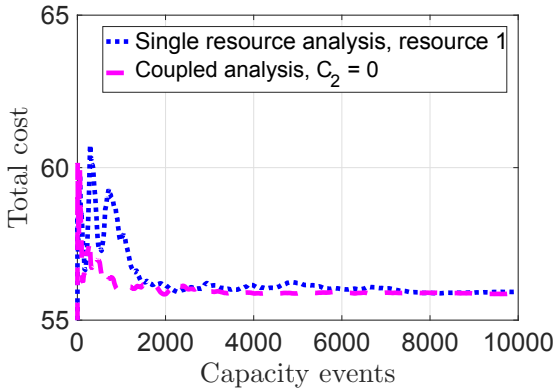


Figure 2.14: Evolution of the total cost over average allocations by single resource algorithm for resource 1 and multi-resource allocation algorithm with $\mathcal{C}_2 = 0$.

Figure 2.15 shows the evolution of the sum of instantaneous allocations and the sum of average

allocations by the single resource algorithm for resource 1 and multi-resource allocation algorithm with $\mathcal{C}_2 = 0$. We observe that after some time of the start of the algorithm (after the occurrence of the first capacity event), the sum of instantaneous allocations concentrates around the respective resource capacities, as shown in Figure 2.15 (a) (single resource case), and Figure 2.15(c) (multi-resource case). Furthermore, the sum of average allocations concentrates very close to the respective resource capacities, as shown in Figure 2.15 (b) (single resource case), and Figure 2.15 (d) (multi-resource case).

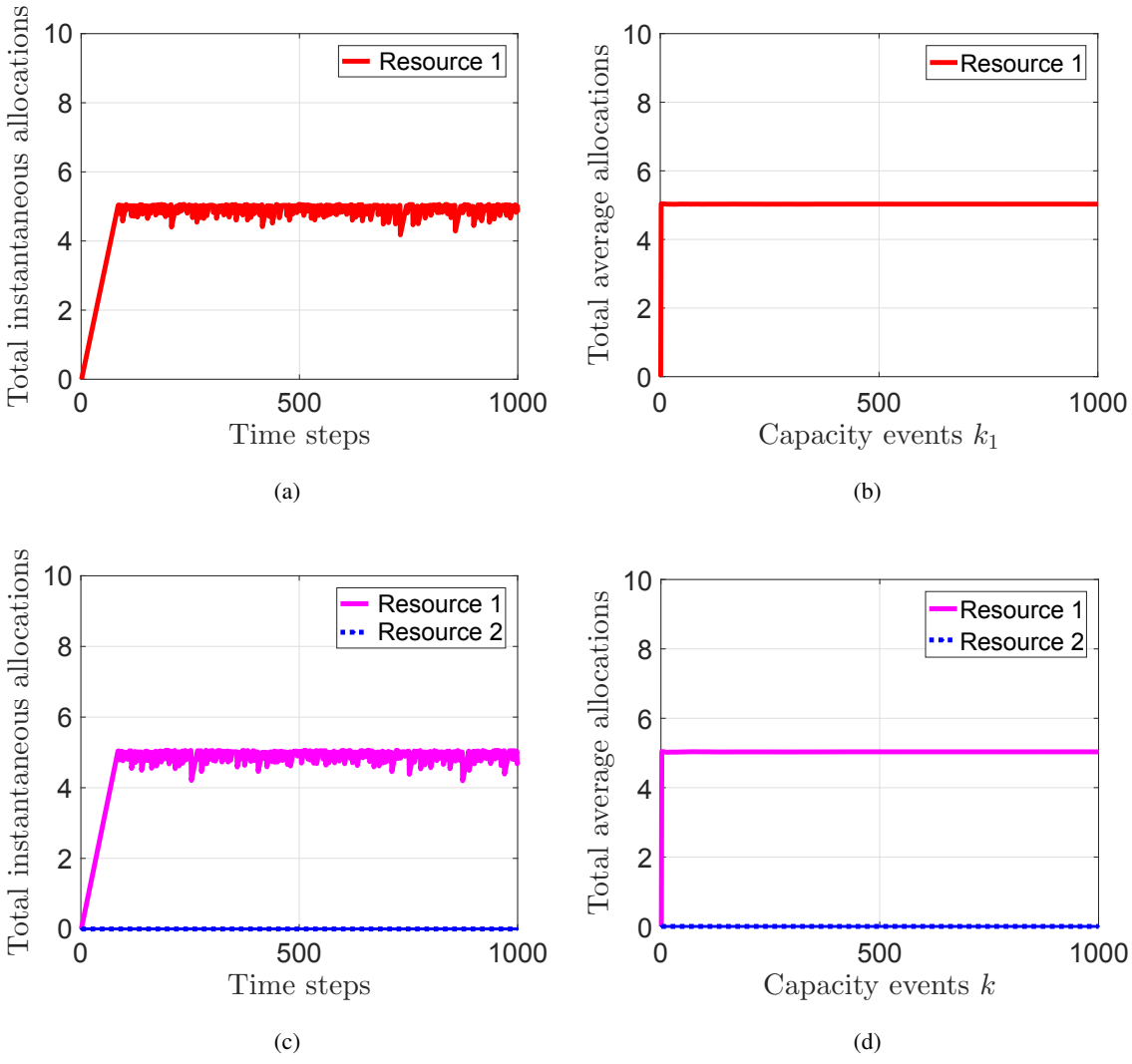


Figure 2.15: Single resource allocation for resource 1 with capacity $\mathcal{C}_1 = 5$: (a) Evolution of aggregate instantaneous allocations, (b) evolution of aggregate average allocations. Multi-resource allocation with capacities $\mathcal{C}_1 = 5$ and $\mathcal{C}_2 = 0$: (c) Evolution of aggregate instantaneous allocations of resources, (d) evolution of aggregate average allocations of resources.

Similar to the previous analysis on resource 1, we now compare the results obtained by the single resource algorithm for resource 2 and the multi-resource allocation algorithm by assigning the capacity of resource 1 to 0. The same conclusion holds in this case also. The total costs over average allocations reach very close to each other over time, as shown in Figure 2.16.

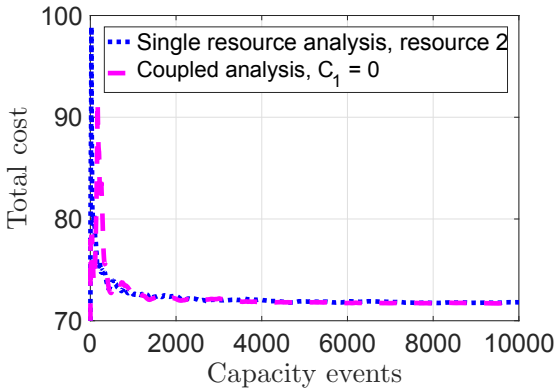


Figure 2.16: Evolution of the total cost over average allocations by single resource algorithm for resource 2 and multi-resource allocation algorithm with $\mathcal{C}_1 = 0$.

Additionally, the sum of instantaneous allocations concentrates around the respective resource capacities, as shown in Figure 2.17 (a) (single resource case), and Figure 2.17 (c) (multi-resource case). Also, the sum of average allocations concentrates very close to the respective capacities, as shown in Figure 2.17 (b) (single resource case), and Figure 2.17 (d) (multi-resource case).

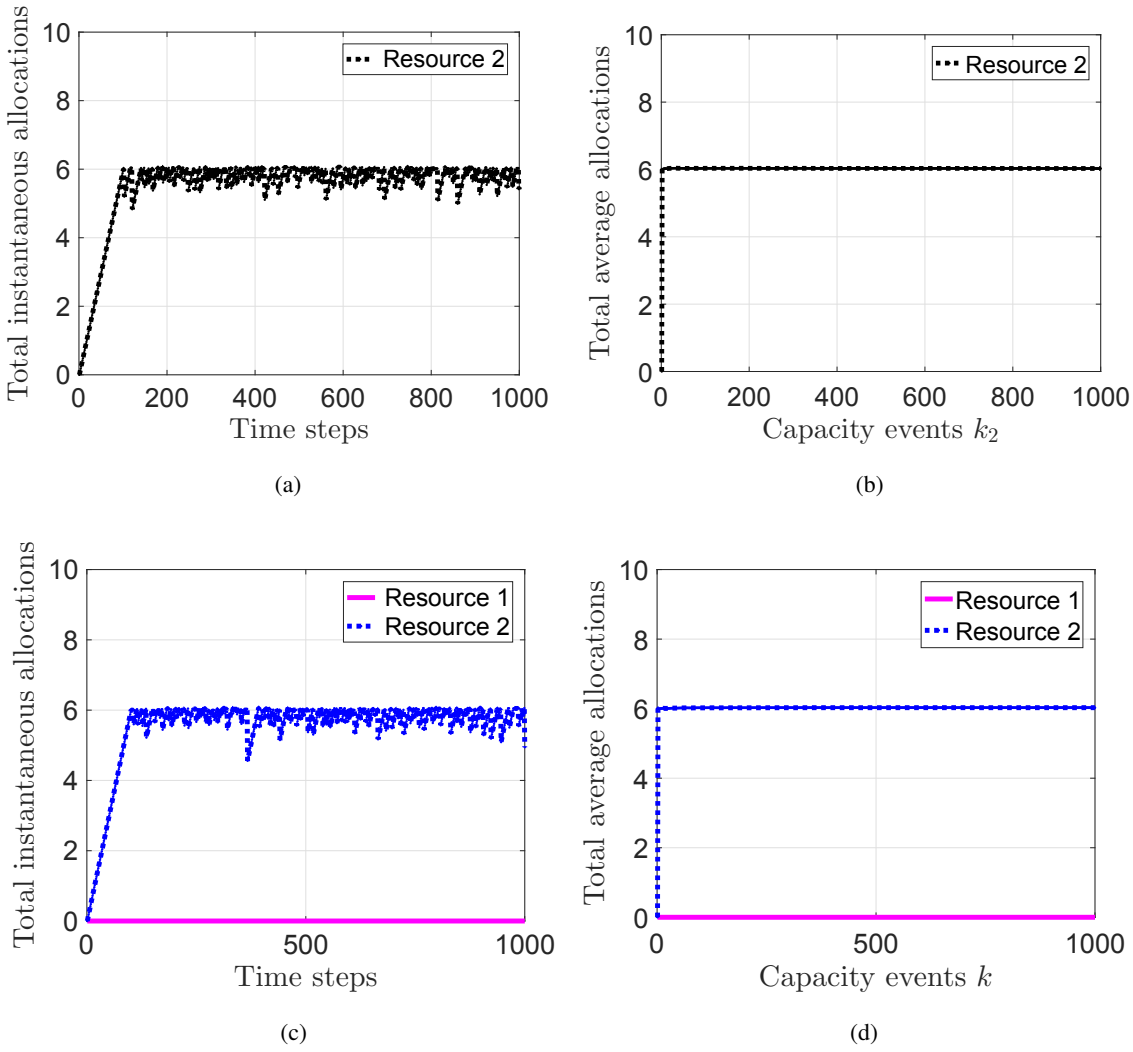


Figure 2.17: Single resource allocation for resource 2 with capacity $\mathcal{C}_2 = 6$: (a) Evolution of aggregate instantaneous allocations, (b) evolution of aggregate average allocations. Multi-resource allocation with capacities $\mathcal{C}_1 = 0$ and $\mathcal{C}_2 = 6$: (c) Evolution of aggregate instantaneous allocations, (d) evolution of aggregate average allocations of resources.

2.8.2 Non-separable cost function

In this subsection, we consider scenarios where the cost functions are non-separable, in the sense that the allocations of one resource depend on the allocations of other resources. Moreover, the cost functions can not be split into independent cost functions, as in the case of separable functions. Our algorithms are suited for such scenarios and provide close to optimal values by a solver. In contrast, the single resource allocation algorithms are not efficient for such scenarios and provide suboptimal

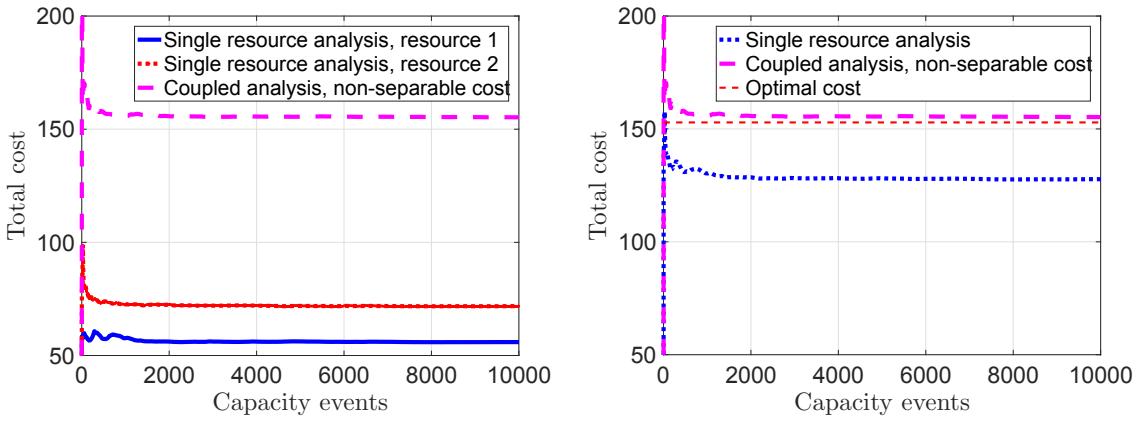
solutions.

To show this, we used the same experimental settings as described in Section 2.8.1 and chose the same values of coefficients of the cost functions. In addition, we use the cost functions listed in (2.125) for multi-resource case, and we use the cost functions listed in (2.122) and (2.123) for single resource cases, for resource 1 and resource 2, respectively.

$$f_i(x_i, y_i) = \begin{cases} \frac{a_i}{2}x_i^2 + \frac{c_i}{2}y_i^2 + \frac{b_i}{4}(x_i + y_i)^4 & i = 1, 2, \\ \frac{b_i}{2}x_i^2 + \frac{c_i}{2}y_i^2 & i = 3, 4, \\ \frac{b_i}{2}x_i^4 + \frac{a_i}{2}y_i^4 & i = 5, 6. \end{cases} \tag{2.125}$$

Figure 2.18: The cost functions for allocating resources 1 and 2, non-separable cost functions for coupled analysis.

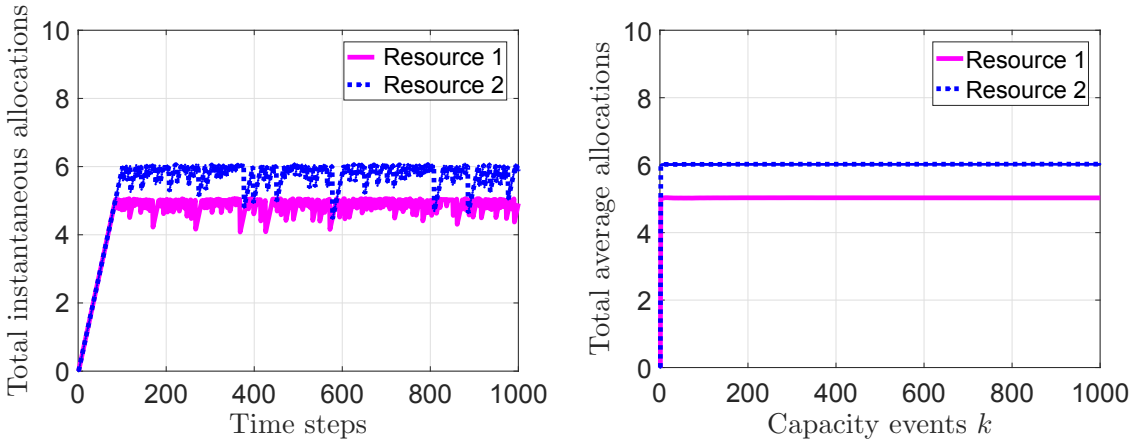
Figure 2.19(a) illustrates the evolution of the total cost over average allocations obtained by the single resource allocation algorithm for resource 1, single resource allocation algorithm for resource 2, and multi-resource allocation algorithm for resources 1 and 2 with non-separable cost function (as listed in (2.125)). We observe that the total costs over average allocations converge over time. Figure 2.19(b) shows that the sum of the total costs obtained by the single resource algorithms and the total cost by the multi-resource algorithm with the non-separable cost function converges to different values. Moreover, the total cost by the multi-resource algorithm is close to the total optimal cost by the solver; whereas, the sum of the total costs obtained by the single resource algorithms is much lower than the total cost by the multi-resource algorithm and is suboptimal, as we expect this because of the nature of the cost functions used.



(a) (b)

Figure 2.19: (a) Evolution of the total cost over average allocations by single resource algorithm for resource 1, single resource algorithm for resource 2, and multi-resource algorithm for resources 1 and 2, (b) evolution of the sum of the total costs by single resource algorithms for resource 1 and resource 2, and evolution of the total cost by the multi-resource algorithm for resources 1 and 2. The dotted line shows the optimal cost by the solver. The multi-resource algorithm and the solver use the non-separable cost functions listed in Equation (2.125). Furthermore, the single resource algorithms use the cost functions listed in Equations (2.122) and (2.123) for resource 1 and resource 2, respectively.

Figure 2.20 shows the evolution of the sum of instantaneous allocations and the sum of average allocations by the multi-resource allocation algorithm. Similar to previous analyses, we observe that after some time of the start of the algorithm, the sum of instantaneous allocations concentrates around the respective resource capacities, as shown in Figure 2.20(a). Furthermore, the sum of average allocations concentrates very close to the respective capacities, as shown in Figure 2.20(b). Interested readers can refer to Figures 2.15(a) and 2.17(a) for the evolution of the sum of instantaneous allocations and Figures 2.15(b) and 2.17(b) for the evolution of the sum of average allocations by the single resource algorithms for resource 1 and resource 2, respectively.



(a) (b)

Figure 2.20: (a) Evolution of aggregate instantaneous allocations, (b) evolution of aggregate average allocations of resources, capacities $\mathcal{C}_1 = 5$ and $\mathcal{C}_2 = 6$ for multi-resource allocation with the non-separable cost functions listed in Equation (2.125).

Chapter 3

AIMD based Derandomized Distributed Algorithm for Divisible Multi-resource Allocation

In this chapter, we develop a derandomized AIMD algorithm to solve a class of distributed optimization problems for multiple shared resource allocation. The algorithm is a derandomized version of the stochastic additive-increase and multiplicative-decrease (AIMD) algorithm of Chapter 2. The developed solution uses a one-bit feedback signal for each resource between the system and the agents and does not require inter-agent communication. Additionally, each agent has private cost functions, which are strictly convex, twice continuously differentiable, and increasing in each variable. Empirically we show that the long-term average allocations of multiple shared resources converge to optimal allocations, and the system achieves minimum social cost. Furthermore, we show that the proposed derandomized AIMD algorithm converges faster than the stochastic AIMD algorithm (Alam et al., 2018a) and both the approaches provide approximately the same solutions.

3.1 Introduction

The number of Internet of things (IoT) devices, such as smartphones, smart-watches, fitness trackers, connected cars, cameras, etc., is increasing very rapidly (Columbus, 2017). Such devices are constrained by computational resources and battery life (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015; Atzori, Iera, & Morabito, 2010); by providing additional shared resources for task offloading, these devices can be used to build many emerging smart applications. Some of the representative smart applications are Apple Siri, Google Assistant, IBM Watson, Amazon Alexa, etc.; they use Cloud technologies for task offloading. Because Clouds are usually hosted at faraway locations worldwide from the IoT devices, offloading the tasks on the Clouds incur delay, which is not suitable for many latency-sensitive mobile applications; for example, cognitive assistance (Ha et al., 2014). Interested readers can find some exciting cognitive assistance applications in (Z. Chen et al., 2017). Such applications collect the data through sensors of wearable devices, offload them on mini Cloud data-centers called *Cloudlets* (Satyanarayanan, Bahl, Caceres, & Davies, 2009) for processing and receiving the processed information in real-time, which assists a cognitively impaired person. Cloudlets are the enabling technology for computing resource-intensive and latency-sensitive mobile applications (Satyanarayanan, 2017). It stays in close proximity to the IoT devices, usually at one hop (Masip-Bruin, Marin-Tordera, Jukan, & Ren, 2018). A recent survey of such technologies can be found in (Shi, Cao, Zhang, Li, & Xu, 2016). The IoT devices receive the resources on-demand from a Cloudlet to perform interactive tasks. However, the latency tolerant tasks can be offloaded on the Cloud. A diagram of such a system is presented in Figure 3.1. A similar three-tier architecture is presented in (Song et al., 2017), where a human-centric real-time positioning system using wearable devices is proposed. The resource provisioning in a Cloudlet is dynamic, and the IoT devices receive resources as a virtual machine (VM). Interested readers can find details of on-demand VM provisioning in (Echeverria, Root, Bradshaw, & Lewis, 2014). Since each device works for a different purpose, it requires a different amount of shared resources. We assume that the VMs are customized, which are created based on the resource requirement of IoT devices.

Due to the increase in the number of IoT devices, optimal allocation of shared resources to achieve

social optimum with little communication overhead is a challenging problem. It becomes more challenging to achieve social optimum when the IoT devices do not communicate their information to other IoT devices. As a step in this direction, Alam et al. (Alam et al., 2018a) propose a distributed stochastic additive increase and multiplicative decrease (AIMD) algorithm to allocate multiple shared resources to IoT devices. The algorithm is a distributed stochastic algorithm and involves very little communication overhead. Based on this work, we propose a distributed and deterministic AIMD algorithm, which is a derandomized version of it. The proposed algorithm incurs very little communication overhead, and it converges much faster than the stochastic AIMD algorithm of (Alam et al., 2018a). Interested readers can find recent works on resource allocation in (Angelakis, Avgouleas, Pappas, Fitzgerald, & Yuan, 2016; Q. Lu, Yao, Qi, He, & Guan, 2016). In the proposed algorithm, we assume that each IoT device has its cost function, and it does not share the cost function or resource allocation history with other participating IoT devices.

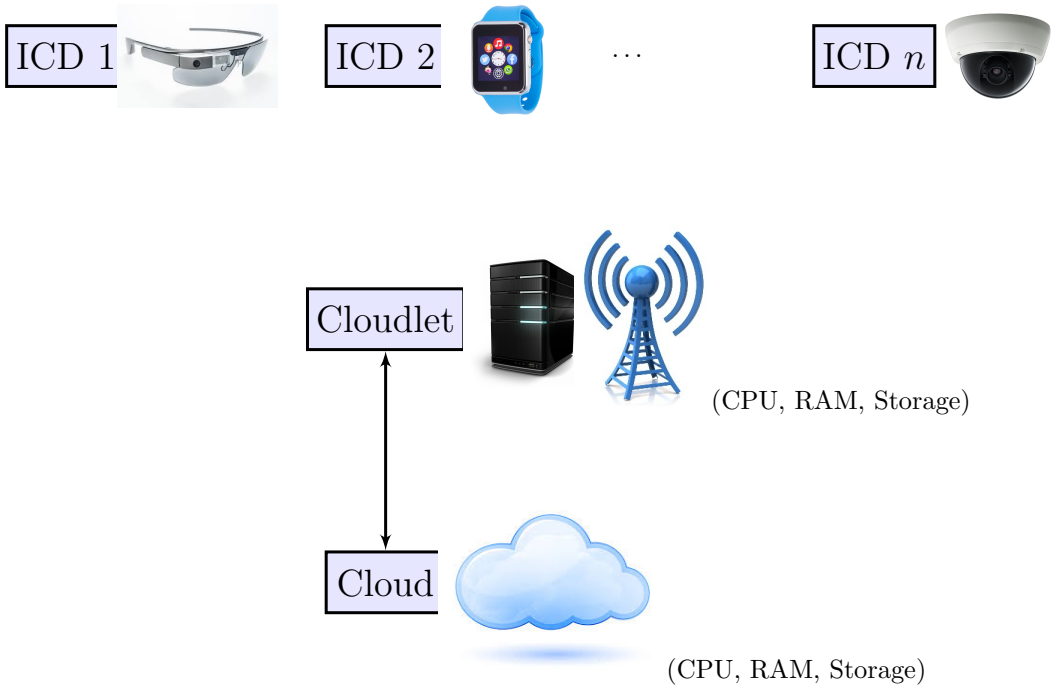


Figure 3.1: A three-tier architecture of IoT devices — Cloudlets — Cloud: IoT devices offload their tasks on Cloudlets. They receive computing resources such as CPU, memory, storage, etc., from Cloudlets with little latency. Larger and latency tolerant tasks can be offloaded on Cloud. Here ICD denotes an IoT device.

As a brief background, in stochastic AIMD algorithm (described in Chapter 2), the devices keep increasing their demand of each resource linearly by a constant called *additive increase factor* until they receive a *capacity event* notification from the control unit. The control unit sends a capacity event notification for each resource to all IoT devices to notify that the total demand of the resource has reached its capacity. After receiving this notification, the devices respond in a probabilistic way to decrease their resource demands abruptly by a constant called *multiplicative decrease factor*. The devices again start increasing their demands linearly until they receive the next capacity event notification; this process repeats over time and repeats for all the resources in the system. We modify the stochastic AIMD algorithm (Alam et al., 2018a) and propose a deterministic algorithm to allocate multiple shared resources.

Let us assume that there is a Cloudlet that hosts several computing resources, and IoT devices offload their tasks on it. We also assume that the Cloudlet has a control unit that keeps track of the aggregate demands of resources. The system aims to optimally allocate resources hosted on Cloudlet and achieve social optimum cost. The proposed deterministic AIMD algorithm works as follows: at the start of the algorithm, the control unit of the Cloudlet broadcasts a set of parameters to all the IoT devices in the system. Each IoT device has its cost function, and it does not share the costs with other IoT devices. After receiving the parameters from the control unit, IoT devices start the AI phase, and they keep increasing the resource demand linearly by a positive constant called *additive increase factor* until they receive the *capacity event* notification from the control unit of the Cloudlet. After receiving capacity event notifications from the control unit, they abruptly decrease their demand in a deterministic way, based on, multiplicative decrease factor, average resource allocation, and derivative of the cost functions. The devices can start increasing their demands again until they receive the next capacity event notification from the control unit. This process repeats over time and repeats for every resource in the system.

We observe empirically that the proposed deterministic AIMD algorithm converges faster than the stochastic AIMD algorithm. Furthermore, we observe that both approaches provide approximately the same results over time. The long-term average allocations are approximately equal to the optimal allocation values obtained by solving the same optimization problem in a centralized setting.

The following are three main contributions to this chapter. First, we propose a distributed, private, and deterministic resource allocation algorithm to allocate multiple shared resources to IoT devices. The algorithm is a derandomized version of the stochastic AIMD algorithm (Alam et al., 2018a). Second, we describe simulation settings in detail and verify our results' efficacy; we compare the results with the stochastic AIMD algorithm. We also compare the results of the deterministic AIMD with the results obtained by solving the same optimization problem in a centralized setting. Third, we present a use case of a tourist attraction center, where we assume heterogeneous IoT devices, such as a set of wearable devices, surveillance cameras, etc. These IoT devices receive on-demand shared resources from the Cloudlets for task offloading. Such facilities can be useful in realizing the potential of IoT devices in emerging applications such as assisting tourists with a physical disability, uploading pictures of scenes on social media, getting detailed real-time information about a painting in a museum, real-time foreign language translation, etc.

3.2 Problem formulation

Suppose that a Cloudlet hosts a pool of shared resources, e.g., CPUs, GPUs, RAM, storage, network bandwidth, etc. For the sake of generality, we assume that there are m shared resources $R^1, R^2, \ldots, R^m$ with capacities $\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^m$, respectively. Let there are n heterogeneous IoT devices, $ICD\ 1$, $ICD\ 2$, ..., $ICD\ n$, such as smart wearable devices, cameras, etc. The Cloudlet has a control unit, which coordinates with the Cloudlet and the participating IoT devices. We index IoT devices with $i = 1, 2, \ldots, n$ and use $j = 1, 2, \ldots, m$ to index the resources hosted on the Cloudlet. Let $\mathbb{N}$ denote the set of natural numbers. We use k to index time steps, where $k \in \mathbb{N}$. Furthermore, we assume that each IoT device has a cost function $f_i : \mathbb{R}_+^m \to \mathbb{R}_+$, which associates a cost to a certain allotment of resources. An IoT device demands the amount of shared computing resources from the Cloudlet according to its need and the application it performs. The system aims that each IoT device receives the optimal allocation and the system achieves a minimum social cost. We now list the assumptions for the cost functions of each IoT device.

Assumption 3.2.1. *We assume that the cost function f_i is twice continuously differentiable, convex, and increasing in each variable.*

For all i and j, we denote by $x_i^j \in \mathbb{R}_+$ the amount of resource j allotted to IoT device i. We aim to solve the following optimization problem in a distributed way with no inter-agent communication:

Problem 3.2.2.

$$\begin{aligned} \min_{x_1^1,\ldots,x_n^m} \quad & \sum_{i=1}^{n} f_i(x_i^1, x_i^2, \ldots, x_i^m), \\ \textit{subject to} \quad & \sum_{i=1}^{n} x_i^j = \mathcal{C}^j, \quad j = 1, 2, \ldots, m, \\ & x_i^j \geq 0, \quad i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m. \end{aligned}$$

We denote the solution to the minimization problem by $\mathbf{x}^* \in (\mathbb{R}_+^n)^m$, where $\mathbf{x}^* = (x_1^{*1}, \ldots, x_n^{*m})$. Because of the compactness of the constraint sets and strict convexity of the cost functions, there exists a unique optimal solution of Problem 3.2.2. Let $x_i^j(k)$ and $\overline{x}_i^j(k)$ denote the amount of instantaneous allocations and the amount of average allocations of resource j of IoT device i at the time step k, respectively. For all i, j and k, the average allocation is calculated as follows:

$$\overline{x}_i^j(k) = \frac{1}{k+1} \sum_{\ell=0}^{k} x_i^j(\ell). \tag{3.1}$$

We assume here that an IoT device can obtain any amount of resource in $[0, \mathcal{C}^j]$. Our aim here is to propose a distributed, private and deterministic iterative scheme to allocate multiple shared resources to IoT devices, such that the long-term average allocations of resources converge to the optimal allocations. Thus, we aim to achieve:

$$\overline{\mathbf{x}}(k) \to \mathbf{x}^*, \text{ when } k \to \infty. \tag{3.2}$$

Let $\frac{\partial}{\partial_x} f_i(x, y)$ denote the partial derivative of $f_i(.)$ with respect to x. Based on the analysis on optimality (refer Chapter 1.4.5), the derivatives of cost functions of all IoT devices for a particular resource should make a consensus to achieve optimality for Problem 3.2.2.

3.3 Algorithm

The proposed algorithm is a distributed, private, iterative, and deterministic AIMD algorithm. It is used to allocate multiple shared divisible resources to IoT devices. The algorithm is deterministic in the sense that it does not involve randomness in resource allocation phases. The block diagram of the system is presented in Figure 3.2. We assume that there is a Cloudlet that hosts computing resources, and the control unit is a sub-system of the Cloudlet. Let $\alpha^j \in (0, \mathcal{C}^j]$ be the *additive increase factor*, $\beta^j \in [0, 1)$ be the *multiplicative decrease factor* and $\Gamma_j \in \mathbb{R}_+$ be the normalization factor of resource j, for $j = 1, 2, \ldots, m$. Let $S^j(k) \in \{0, 1\}$ denote the *capacity event* at time step k for resource j, for all j and k. At the start of the system, the control unit initializes the parameters Γ^j, α^j, and β^j with desired values. It also initializes the capacity event $S^j(0)$ with 0. After initialization, the control unit broadcasts Γ^j, α^j, β^j, and $S^j(0)$ to all the participating IoT devices, for all resources. Each IoT device runs its algorithm to demand shared resources.

The algorithm of an IoT device works as follows, after joining the system, an IoT device starts increasing its demand of shared resources linearly until it receives a capacity event notification from the control unit. The control unit broadcasts this notification when the total demand of a resource reaches its capacity. After receiving this capacity event, the IoT device decreases its demands abruptly and in a deterministic way. It does so by using multiplicative decrease factor β^j and its *scaling factor* $\lambda_i^j(k) \in (0, 1]$. The control unit updates and broadcasts the capacity event $S^j(k + 1) = 1$ when the total demand $\sum_{i=1}^n x_i^j(k)$ of resource j reaches the capacity $\mathcal{C}^j$ at time step k, i.e.,

$$S^j(k+1) = \begin{cases} 0 & \text{if } \sum_{i=1}^n x_i^j(k) < \mathcal{C}^j \\ 1 & \text{if } \sum_{i=1}^n x_i^j(k) \geq \mathcal{C}^j. \end{cases} \tag{3.3}$$

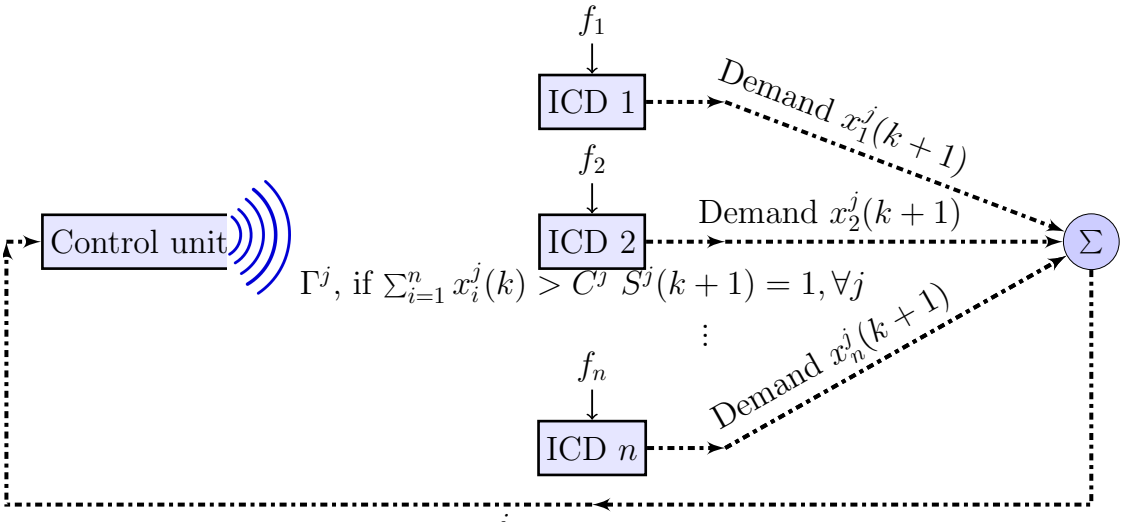


Figure 3.2: Block diagram of the multi-resource allocation deterministic AIMD model.

We now formally describe the deterministic AIMD algorithm; it has the following two phases: *additive increase (AI)* and *multiplicative decrease (MD)*.

(i) *Additive increase (AI)*: In this phase, an IoT device keeps increasing its resource demand linearly by additive increase factor, α^j, until it receives a capacity event signal $S^j(k) = 1$ from the control unit. The control unit broadcasts the capacity event to notify the IoT devices that the aggregate demand has reached the capacity of the resource. AI phase is formulated as:

$$x_i^j(k+1) = x_i^j(k) + \alpha^j.$$

(ii) *Multiplicative decrease (MD)*: In this phase, after receiving the capacity event signal from the control unit, the IoT devices reduce their demands synchronously in a deterministic way. To do so, they use the multiplicative decrease factor, $\beta^j \in [0, 1)$, and scaling factor, $\lambda_i^j(k) \in (0, 1]$. The value of the scaling factor, $\lambda_i^j(k)$, is calculated according to (3.4). The multiplicative decrease phase is formulated as:

$$x_i^j(k+1) = \Big(\lambda_i^j(k)\beta^j + \Big(1 - \lambda_i^j(k)\Big)\Big)\, x_i^j(k).$$

After abruptly reducing the resource demands, the IoT devices again start to increase their demands linearly until they receive the next capacity event; this process repeats over time.

Now, let Γ^j be the normalization constant received by an IoT device i for resource j from the control unit. It is used to keep scaling factor $\lambda_i^j(k) \in (0, 1]$. The scaling factor $\lambda_i^j(k)$ of IoT device i depends on its average resource allocation, the derivative of its cost function for resource j, and normalization constant Γ^j. For all i, j, and k, we calculate the scaling factor as follows:

$$\lambda_i^j(k) = \Gamma^j \frac{\frac{\partial}{\partial_x}\Big|_{x=\overline{x}_i^j(k)} f_i\left(\overline{x}_i^1(k), \ldots, \overline{x}_i^m(k)\right)}{\overline{x}_i^j(k)}. \tag{3.4}$$

Now, let $\mathcal{F}$ be the set of convex, twice continuously differentiable and increasing cost functions. The control unit calculates Γ^j as follows:

$$\Gamma^j = \inf_{\mathbf{x}\in(\mathbb{R}_+^n)^m, f_i\in\mathcal{F}} \left(\frac{x_i^j}{\frac{\partial}{\partial_x}\Big|_{x=x_i^j} f_i(x_i^1, x_i^2, \ldots, x_i^m)} \right), \text{ for } j = 1, 2, \ldots, m. \tag{3.5}$$

Notice that the method of calculating scaling factor $\lambda_i^j(k)$ and normalization factor Γ^j is the same as described in (Alam et al., 2018a). However, for clarity, we use the term scaling factor for

$\lambda_i^j(k)$ instead of probability.

Algorithm 5: Algorithm of control unit

Input: $\mathcal{C}^j$, for $j = 1, 2, \ldots, m$.

Output: $S^j(k)$, for $j = 1, 2, \ldots, m$, $k \in \mathbb{N}$.

Initialization: $S^j(0) \leftarrow 0$, for $j = 1, 2, \ldots, m$, and broadcast Γ^j, for $j = 1, 2, \ldots, m$;

while $k \in \mathbb{N}$ **do**

  while $j = 1, 2, \ldots, m$ **do**

    if $\sum_{i=1}^{n} x_i^j(k) \geq \mathcal{C}^j$ **then**

      $S^j(k+1) \leftarrow 1$;

      broadcast $S^j(k+1)$;

    else

      $S^j(k+1) \leftarrow 0$;

    end

  end

end

The algorithm of the control unit is presented in Algorithm 5, and the distributed deterministic

algorithm for each IoT device is described in Algorithm 6.

Algorithm 6: Deterministic AIMD algorithm of IoT device i (D-AIMD i)

Input: $S^j(k)$, $k \in \mathbb{N}$ and Γ^j, for $j = 1, 2, \ldots, m$.

Output: $x_i^j(k+1)$, for $j = 1, 2, \ldots, m$, $k \in \mathbb{N}$.

Initialization: Agent i sets its states $x_i^j(0)$[1] and $\overline{x}_i^j(0) \leftarrow x_i^j(0)$, for $j = 1, 2, \ldots, m$;

while *IoT device i participates at $k \in \mathbb{N}$* **do**

  while $j = 1, 2, \ldots, m$ **do**

    if $S^j(k) = 1$ **then**

$$\lambda_i^j(k) \leftarrow \Gamma^j \frac{\frac{\partial}{\partial x}\Big|_{x=\overline{x}_i^j(k)} f_i\big(\overline{x}_i^1(k), \overline{x}_i^2(k), \ldots, \overline{x}_i^m(k)\big)}{\overline{x}_i^j(k)};$$

$$x_i^j(k+1) \leftarrow \Big(\lambda_i^j(k)\beta^j + \big(1 - \lambda_i^j(k)\big)\Big) x_i^j(k);$$

    else

$$x_i^j(k+1) \leftarrow x_i^j(k) + \alpha^j;$$

    end

$$\overline{x}_i^j(k+1) \leftarrow \tfrac{k+1}{k+2}\overline{x}_i^j(k) + \tfrac{1}{k+2} x_i^j(k+1);$$

  end

end

By following the approach with appropriate values of $\lambda_i^j(k)$, α^j, β^j, and Γ^j, the long-term average resource allocations converge to optimal allocations that is $\overline{\mathbf{x}}(k) \to \mathbf{x}^*$, when $k \to \infty$ and hence the system achieves a minimum social cost. Notice that, in the additive increase (AI) phase, the stochastic and the deterministic AIMD algorithms follow the same allocation rule. However, in the multiplicative decrease (MD) phase, they follow different allocation rules. In the stochastic AIMD algorithm, the MD phase is probabilistic—an IoT device responds to the capacity event asynchronously, either by reducing its demand multiplicatively by β^j or by not responding to the capacity event. However, in the deterministic AIMD, the MD phase is deterministic—all the participating IoT devices back-off synchronously using parameters β^j and $\lambda_i^j(k)$.

[2]We can initialize $x_i^j(0)$ with any value in $[0, \mathcal{C}^j]$ such that $\sum_{i=1}^n x_i^j(0) \leq \mathcal{C}^j$, for $j = 1, 2, \ldots, m$.

Observation 3.3.1. *Based on the empirical results, we observe that the average allocations converge to optimal allocations. We also observe that the deterministic AIMD algorithm converges much faster than the stochastic AIMD algorithm.*

Remark 3.3.2 (Communication overhead)**.** *Because of the faster convergence of the deterministic AIMD algorithm, the number of capacity events (in bits) required to reach consensus of derivatives is less than that of the stochastic AIMD algorithm. As described in (Alam et al., 2018a) for m resources in the system, the communication overhead is $\sum_{\ell=0}^{k}\sum_{j=1}^{m} S^j(\ell)$ bits until the k'th time step, where $S^j(\ell) \in \{0,1\}$. In the worst case scenario, this will be mk bits until time step k.*

As we stated in Chapter 1, we restate that because our models consider a central server that keeps track of aggregate demands and sends feedback signals in the network, the system may fail if the central server stops working. However, we can fix this by adding a backup server that keeps all the information as the central server and receives feedback signals from the central server. When the central server stops sending the feedback signals, the backup server takes charge and works as the central server until the central server comes live. Such settings are explored extensively by the machine learning community as federated learning. Wherein several agents collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020). For clarity, we restate it in each chapter in the thesis.

3.4 Experiments

This section presents the simulation settings and results of the deterministic AIMD algorithm and compares it with the results of the stochastic AIMD algorithm, which is simulated using the same parameters. We observe that the average allocations of a resource converge at approximately the same value for each IoT device in both approaches. We also observe that deterministic AIMD converges faster than the stochastic AIMD.

Now, suppose that there are 60 IoT devices such as Google glasses, smart-watches, and cameras being used in and around a place of tourist attraction. Let us assume that the management of the place has installed a Cloudlet there and aims to allocate resources optimally to all participating IoT devices and to achieve a social minimum cost. We assume further that the Cloudlet hosts three shared computing resources, RAM, CPU cycles, and disk storage. Let the capacities of these resources be $\mathcal{C}^1 = 32$ GB, $\mathcal{C}^2 = 20$ GHz and $\mathcal{C}^3 = 250$ GB, respectively. For the convenience of scalability of parameters, we suppose that 10 GB of storage is represented by GB^{D}, then we write $\mathcal{C}^3 = 25\ \mathrm{GB}^{\mathrm{D}}$. The Cloudlet has a control unit that notifies the IoT devices when the demands reach the capacity of a resource. At the start of the system, the control unit broadcasts the parameters $\alpha^1 = 0.025$ GB, $\alpha^2 = 0.02$ GHz, $\alpha^3 = 0.0225\ \mathrm{GB}^{\mathrm{D}}$, $\beta^1 = 0.7$, $\beta^2 = 0.85$ and $\beta^3 = 0.75$ to all the participating IoT devices. It also broadcasts the normalization factors $\Gamma^1 = \Gamma^2 = \Gamma^3 = 1/90$.

Let a_i, b_i, and c_i represent the cost of RAM, CPU cycles and disk storage, respectively and d_i represents any other costs incurred. Let $X = \{1, 2, \ldots, 25\}$, $Y = \{1, 2, \ldots, 20\}$, $Z = \{1, 2, \ldots, 15\}$ and $U = \{1, 2, \ldots, 10\}$ be uniformly distributed random variables, they are used to obtain different cost functions. We use $a_i \in X, b_i \in Y, c_i \in Z$ and $d_i \in U$ as the numerical values of these uniformly distributed random variables in the simulation. Let for all i, the cost functions of IoT device i is calculated as follows,

$$f_i(x_i^1, x_i^2, x_i^3) = \begin{cases} (i)a_i\big((x_i^1)^2 + \frac{1}{2}(x_i^1)^4\big) + b_i\big(2(x_i^2)^4 + \frac{1}{2}(x_i^2)^6\big) \\ +c_i\big((x_i^3)^2 + \frac{1}{4}(x_i^3)^4\big) + \frac{1}{8}d_i(x_i^3)^8 \\ (ii)a_i(x_i^1)^2 + b_i\big((x_i^2)^2 + \frac{1}{2}(x_i^2)^4\big) + \frac{3}{2}c_i(x_i^3)^4 \\ (iii)\frac{1}{3}a_i(x_i^1)^6 + b_i(x_i^2)^2 + c_i(x_i^3)^2 + d_i\big(\frac{1}{6}(x_i^2)^6 + \frac{1}{8}(x_i^3)^4\big). \end{cases} \tag{3.6}$$

The following are some of the results obtained from the simulation. As described in Section 3.2, for the optimization Problem 3.2.2, at the optimal values the derivatives of cost functions with respect to a particular resource make consensus.

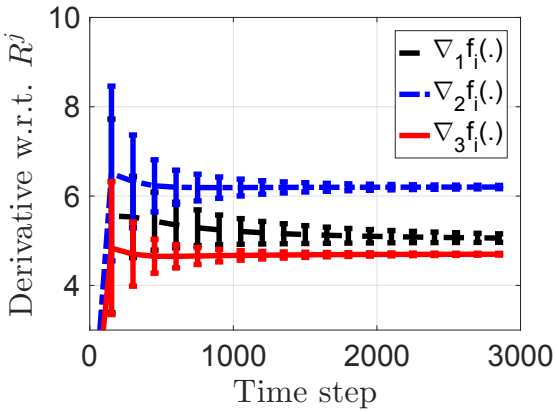
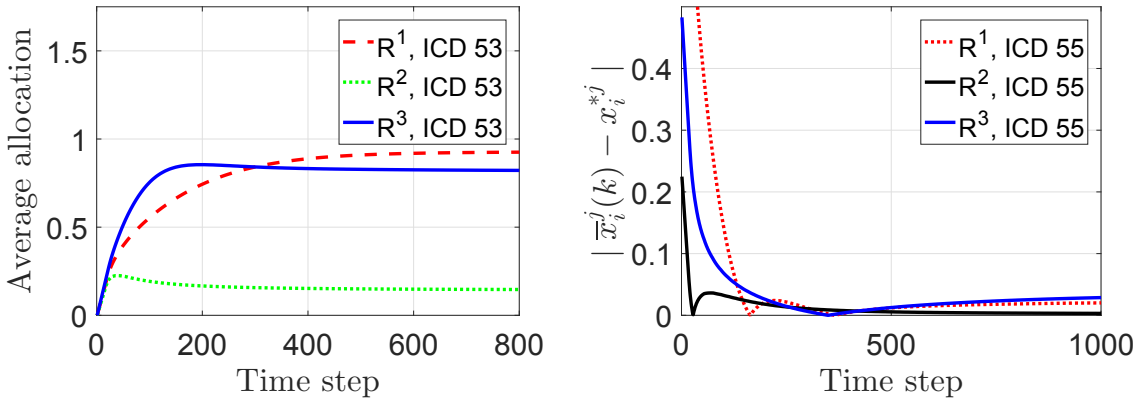


Figure 3.3: Results of deterministic AIMD: Evolution of the profile of derivatives of cost functions $\frac{\partial}{\partial_x} f_i(.)$ of all IoT devices of a single simulation.



(a) (b)

Figure 3.4: Results of deterministic AIMD: (a) evolution of average allocation $\overline{x}_i^j(k)$ of resources, (b) evolution of absolute difference of average allocation and optimal allocation.

We observe from Figure 3.3 that the derivatives of cost functions of IoT devices with respect to a particular resource gather around the same value over time and hence make a consensus. Figure 3.4(a) shows that the average allocations of resources converge over time. Thus, the long-term average allocations of resources are optimal allocations. We verify this claim by simulating the same optimization problem with the same parameters in a centralized setting. We see that the absolute difference of the average allocation and the optimal allocation calculated in a centralized setting are close to zero (see Figure 3.4(b)). Furthermore, Figure 3.5 illustrates that the ratio of the

sum of cost functions for average allocations and the sum of cost functions for calculated optimal allocations are approximately 1.

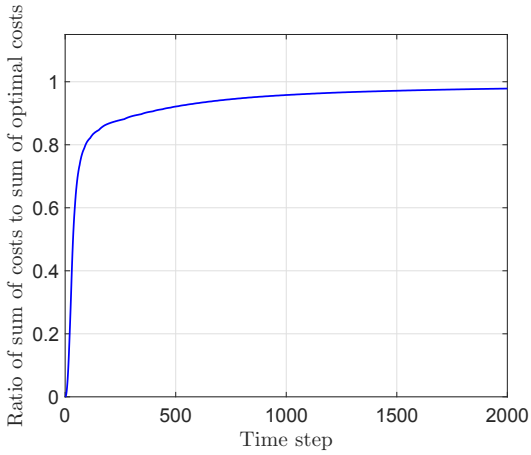
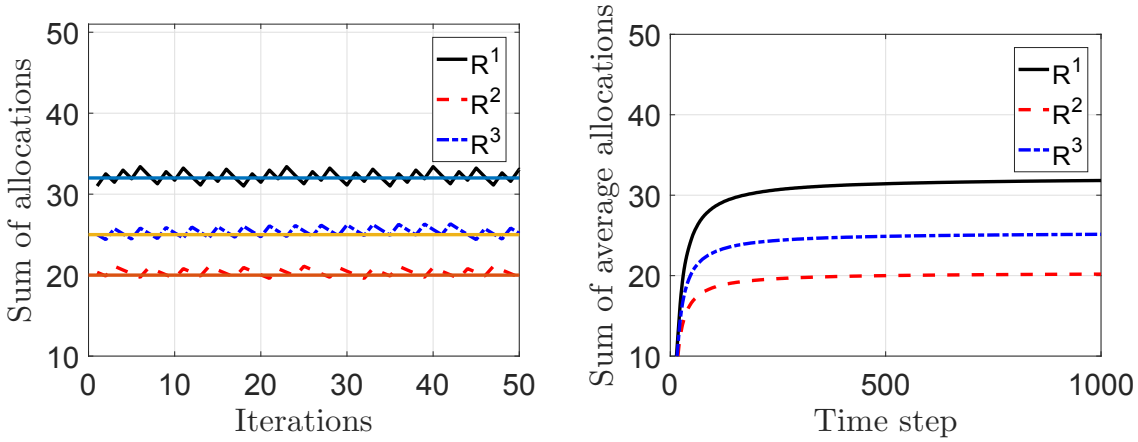


Figure 3.5: Results of deterministic AIMD: Ratio of the sum of cost functions to the sum of optimal cost functions.



(a) (b)

Figure 3.6: Results of deterministic AIMD: (a) total allocation of resources for last 50 time steps, (b) evolution of sum of average allocation of resources, the capacities are $\mathcal{C}^1 = 32$ GB, $\mathcal{C}^2 = 20$ GHz, and $\mathcal{C}^3 = 25$ GB$^{\mathrm{D}}$.

Figure 3.6(a) shows the overshoots of the allocations; we observe that the sum of instantaneous allocations is concentrated near the respective capacities of the resources. To reduce the overshoots, we introduce a constant $\gamma^j \in (0, 1)$ and update the capacity event $S^j(k+1) = 1$, when $\sum_{i=1}^{n} x_i^j(k) > \gamma^j \mathcal{C}^j$ as described in (Alam et al., 2018a). From, Figure 3.6(b), we observe that

the sum of average allocations is approximately equal to the respective capacities of the resources; hence we say that the sum of average allocations satisfies the constraints of the optimization problem.

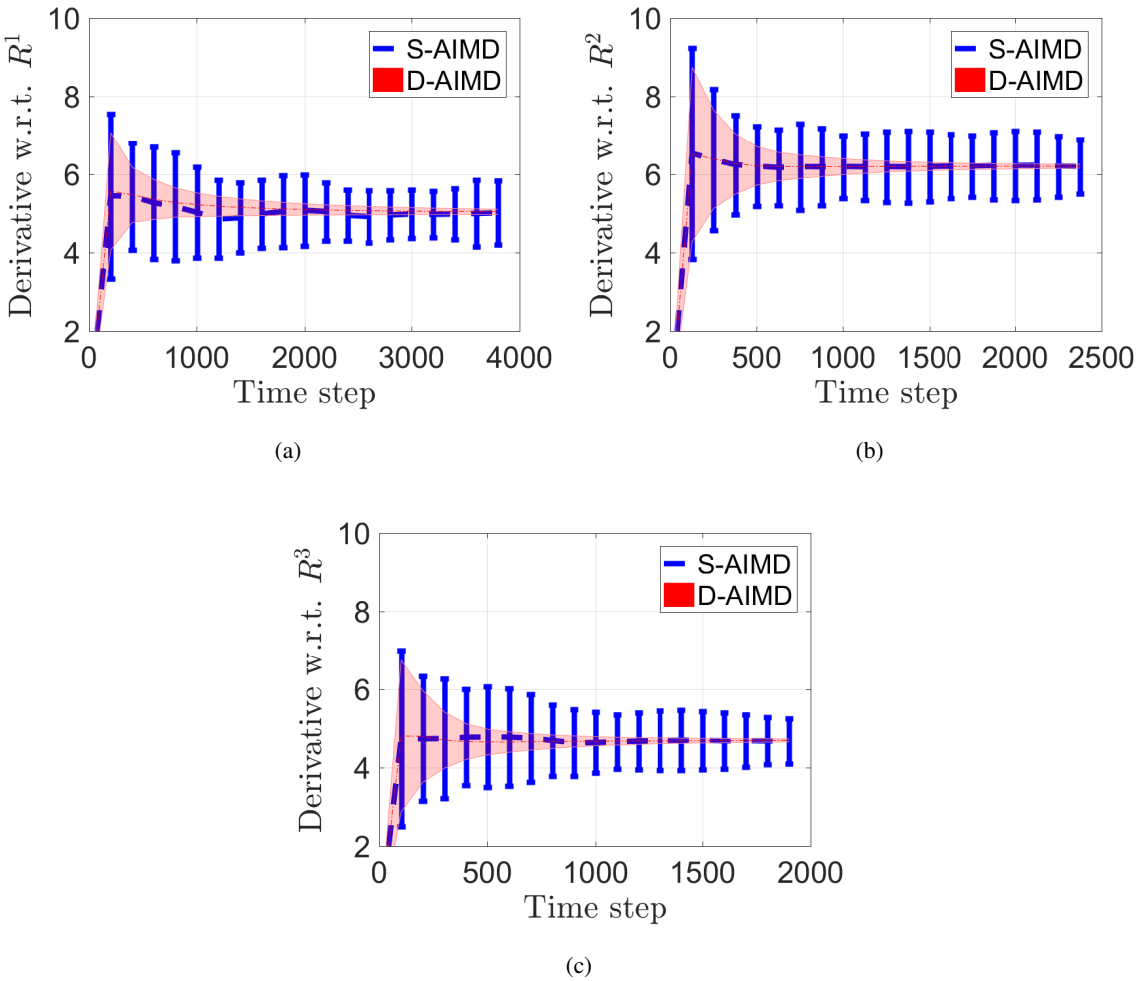


Figure 3.7: Evolution of the profile of derivatives of cost functions $\frac{\partial}{\partial_x} f_i(.)$ of all IoT devices of a single simulation of D-AIMD and S-AIMD algorithms with respect to — (a) resource 1, (b) resource 2, and (c) resource 3. Legends: S-AIMD represents the stochastic AIMD and D-AIMD represents the deterministic AIMD.

We compare the results of the deterministic AIMD approach and the stochastic approach. To do so, we simulate the stochastic AIMD approach (Alam et al., 2018a) with the same parameters and cost functions. Figure 3.7 illustrates the derivatives of cost functions with respect to resources 1, 2, and 3, respectively, of a single simulation of all IoT devices of the deterministic AIMD (D-AIMD)

and the stochastic AIMD (S-AIMD). We present the derivatives of the D-AIMD as shaded error bars. We see that the derivatives of all IoT devices with respect to a particular resource of the deterministic AIMD algorithm concentrate more and more around the same value much faster than the derivatives obtained from the stochastic AIMD algorithm. Hence, the derivatives of the deterministic AIMD converge much faster than that of the stochastic AIMD; therefore, they reach a consensus faster than the stochastic AIMD. Furthermore, we observe in Figure 3.8 that the average allocations of resources 1, 2, and 3, respectively, in the deterministic AIMD algorithm converge faster than the average allocations in the stochastic AIMD algorithm. However, the average allocations of both approaches reach approximately the same value over time.

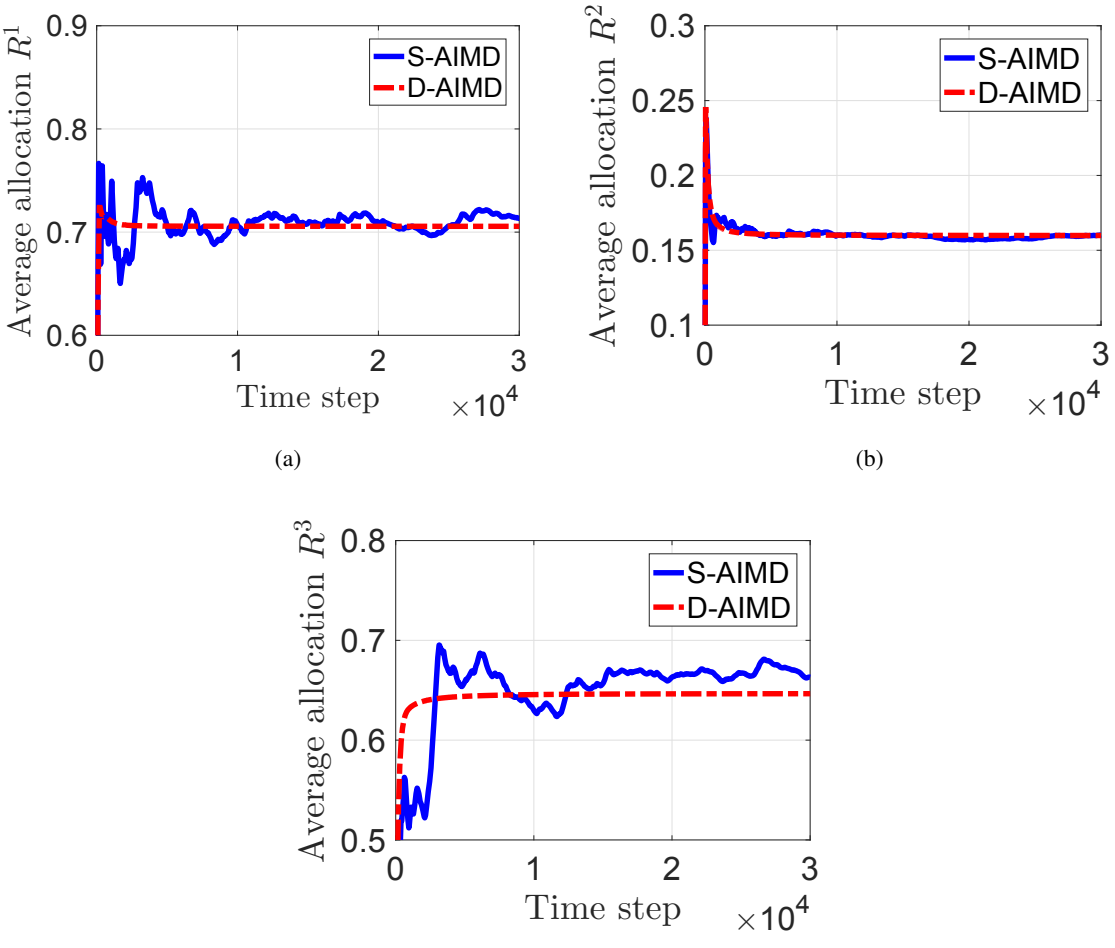
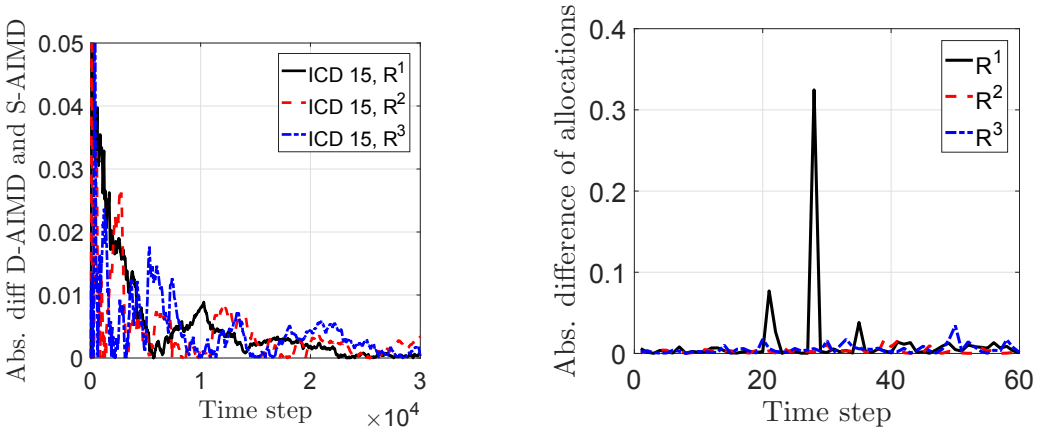


Figure 3.8: Evolution of average allocation of resources of IoT device 42 of D-AIMD and S-AIMD algorithms — (a) resource 1, (b) resource 2, and (c) resource 3.



(a) (b)

Figure 3.9: (a) Evolution of absolute difference of average allocations obtained from D-AIMD and S-AIMD, (b) absolute difference of average allocations obtained from D-AIMD and S-AIMD at time step 30000.

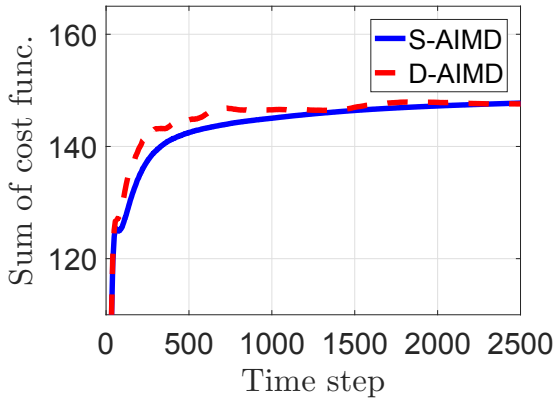


Figure 3.10: Evolution of sum of cost functions over the average allocation of S-AIMD and D-AIMD.

Figure 3.9(a) illustrates the absolute difference of average allocations obtained from D-AIMD and S-AIMD, where the absolute difference is close to zero. We observe that most of the absolute differences of average allocations reach an order of 10^{-3} over time (Figure 3.9(b)); hence both the approaches provide approximately the same long-term average allocations. Figure 3.10 illustrates the convergence of the sum of cost functions $\sum_{i=1}^{n} f_i(x_i^1(k), x_i^2(k), x_i^3(k))$ obtained from both the approaches, we observe that the sum of cost functions obtained from deterministic AIMD converges faster than the sum of cost functions obtained from stochastic AIMD, but they converge to the same

value over time nevertheless. Therefore, both approaches provide the same social optimum value.

Table 3.1: The simulation is run on Intel Core i5-6500, CPU 3.2 GHz, 8 GB RAM. The execution time of each simulation and the corresponding number of capacity events broadcast by the control unit are presented here.

No. of iterations	No. of capacity events (bits) for resources $1, 2$, and 3		Execution time (seconds)	
	S-AIMD	D-AIMD	S-AIMD	D-AIMD
500	184, 187, 137	266, 350, 318	0.666405	0.344306
1000	381, 388, 283	498, 719, 631	1.437232	0.681150
3000	1185, 1199, 835	1569, 2151, 1908	6.655815	3.984121
5000	1918, 1974, 1413	2439, 3563, 3288	16.052666	10.643811
10000	3889, 3950, 2800	5446, 6880, 6151	65.529710	50.616229
30000	11515, 11851, 8357	-	989.557591	-

Additionally, the execution time and the number of capacity events broadcast by the control unit for different simulations are presented in Table 3.1 of both the D-AIMD and the S-AIMD algorithms. We would like to clarify that these results are based on simulation performed on a single computer, and we do not consider the communication delay between IoT devices and the control unit. We observe that the execution time of the deterministic AIMD algorithm is less than that of the stochastic AIMD algorithm, whereas the number of capacity events is more than stochastic AIMD for a particular simulation. Notice that because the D-AIMD converges faster than the S-AIMD; therefore, in the D-AIMD, the control unit broadcasts fewer capacity events to reach consensus. Furthermore, we would like to clarify that in Table 3.1 we have not recorded the number of capacity events and execution time for D-AIMD at 30000 time steps because, in the simulation, we observe that the derivatives converge in less than 5000 time steps.

3.5 Conclusion

We propose a distributed, private, and deterministic algorithm to solve a set of distributed optimization problems for multiple divisible resource allocations with little communication overhead.

The proposed algorithm is a derandomized version of a stochastic AIMD algorithm. The solution has several features, such as the IoT devices need not communicate with each other to reach social-optimum. Second, the solution is communication-efficient, in the sense that the control unit broadcasts a one-bit feedback signal for each resource when the demand of a resource exceeds the capacity of the resource. Third, we showed empirically that the deterministic AIMD algorithm converges faster than the stochastic AIMD algorithm. We further showed that both approaches provide approximately the same long-term average allocation and achieve the same social-optimum cost. It is interesting to deploy the proposed algorithm in real applications using IoT devices and a Cloudlet and analyze its performance. It is also interesting to prove the convergence of average allocations theoretically. Additionally, the algorithm can be extended in several application areas, like smart grid, intelligent transportation systems, supply chain management, to name a few.

Chapter 4

Stochastic Distributed Algorithm for Unit-demand Resource Allocation

In this chapter, we consider a control problem involving several agents coupled through multiple shared resources. These resources are unit-demand (indivisible), allocated either one unit or not allocated; each agent's consumption is modeled as a Bernoulli random variable. Controlling the number of such agents in a probabilistic manner, subject to capacity constraints, is ubiquitous in smart cities. For instance, such agents can be humans in a feedback loop—who respond to a price signal or automated decision-support systems that strive toward system-level goals. In this chapter, we consider both a single resource allocation and multi-resource allocations for the same population of agents. For example, when a network of devices allocates resources to deliver several services, these services are coupled through capacity constraints on the resources. We provide fundamental guarantees of convergence for the single resource allocation case and extend it to multiple resources. Furthermore, we propose a distributed stochastic algorithm for multiple unit-demand resource allocation. We also present an example illustrating the performance of the algorithm.

4.1 Introduction

Classical control has much to offer in a smart city context. However, while this is, without doubt, true, many problems arising in the context of smart cities reveal subtle constraints that are relatively

unexplored by the control community. At a high level, both classical control and smart-city control deal with regulation problems. Nevertheless, in many (perhaps most) smart-city applications, control involves orchestrating the aggregate effect of a number of agents who respond to a signal (sometimes called a *price*) in a probabilistic way. A fundamental difference between classical control and smart-city control is the need to study the effect of control signals on the statistical properties of the populations that we wish to influence while at the same time ensuring that the control is in some sense optimal. This fundamental difference concerns the need for ergodic feedback systems, and even though this problem is rarely studied in control, it is the issue that is perhaps the most pressing in real-life applications. Since the need for predictability at the level of individual agents underpins an operator's ability to write economic contracts.

Our starting point for this work is the previous works of (Fioravanti, Marecek, Shorten, Souza, & Wirth, 2017; Griggs et al., 2016), and the observation that many problems in smart cities can be cast in a framework, where a large number of agents, such as people, cars, or machines, often with unknown objectives—compete for a limited resource. It is challenging to allocate a resource in a manner that utilizes it optimally and gives a guaranteed level of service to each of the agents competing for that resource. For example, allocating parking spaces (Arnott & Rowse, 1999; Khalid et al., 2021; Lin, Rivano, & Mouel, 2017; Teodorovic & Lucic, 2006), regulating cars competing for shared road space (Jones, 2014), or allocating shared bikes (DeMaio, 2009; Raviv & Kolka, 2013), are examples in which resource utilization should be maximized while at the same time delivering a certain quality of service (QoS) to individual agents is a major constraint. As noted in (Alam, Shorten, Wirth, & Yu, 2020; Fioravanti et al., 2017), at a high level, these are primarily optimal control problems but with the added objective of controlling the macroscopic properties of the agents' population. Thus, the design of feedback systems for deployment in smart cities must combine notions of regulation, optimization, and the existence of the unique invariant measure (Fioravanti, Marecek, Shorten, Souza, & Wirth, 2019).

Specifically, in this chapter, we consider the problem of controlling a number of agents coupled through multiple shared resources, where each agent demands the resources in a probabilistic manner. This work builds strongly on the previous work (Griggs et al., 2016) in which the optimal control and ergodic control of the population of agents are considered for a single resource. As we

have mentioned, controlling a network of agents which demand resources in a *probabilistic manner* is ubiquitous in smart cities. In many smart-city applications, the probabilistic intent of agents can be natural (where humans are in a feedback loop and respond, for example, to a price signal) or designed (implemented in a decision support system) so that the network achieves system-level goals. Often, such feedback loops are coupled together as agents contribute or participate in multiple services. For example, when a network of devices allocates resources to deliver several services, these services are coupled through the consumption of multiple shared resources; usually, we call such resources as *unit-demand or indivisible resources* which are either allocated one unit of the resource or not allocated. A concrete manifestation of such a system is the IBM Research's project *parked cars as a service delivery platform* (Cogill et al., 2014). Here, networks of parked cars collaborate to offer services to city managers. Examples of services include *wifi coverage*, *finding missing objects*, and *gas leak detection and localization*. Here, vehicle owners allocate parts of their resources stochastically to contribute to different services, each of which is managed by a feedback loop. The allocation between services is usually coupled via a nonlinear function representing the trade-off between resource allocation (energy, sensors) and the reward for participating in delivering a particular service. It is our firm belief that such systems are ubiquitous in smart cities and represent a new class of problems in feedback systems. In a similar direction, Katsikouli et al. (Katsikouli et al., 2020) developed algorithms to manage the particulate matter in smart cities arising from tyre abrasion using a feedback control strategy.

Our main contribution in this chapter is to establish stochastic schemes for a practically important class of problems for several agents coupled through multiple unit-demand shared resources. Each agent demands the shared resources in a probabilistic manner based on its private cost function and constraints. The constraints are based on multiple shared resources. This scheme is a generalization of the single unit-demand resource allocation algorithm proposed in (Griggs et al., 2016) and follows more relaxed constraints than (Alam et al., 2020). Additionally, the results on convergence are derived for networks with a single resource and extended to multiple resources. To check the efficacy of our algorithm, we present a use case of regulating the number of electric vehicles that share a limited number of level 1 and level 2 charging points. It illustrates that the agents receive the optimal charging points in long-term averages to maximize social welfare.

4.2 Preliminaries

Suppose that n agents are coupled through m unit-demand (indivisible) shared resources, and each agent has a cost function that depends on the allocation of these shared resources. Let the capacity of resources be $C_1, C_2, \ldots, C_m$, respectively. We denote $\mathcal{N} \triangleq \{1, 2, \ldots, n\}$, $\mathcal{M} \triangleq \{1, 2, \ldots, m\}$, and use $i \in \mathcal{N}$ as an index for agents and $j \in \mathcal{M}$ to index the resources. Let $\xi_{ji}(k)$ denote independent Bernoulli random variable which represents the instantaneous allocation of resource j of agent i at time step k. Furthermore, let $y_{ji}(k) \in [0, 1]$ denote the average allocation of resource j of agent i at time step k. We define $y_{ji}(k)$ as follows,

$$y_{ji}(k) \triangleq \frac{1}{k+1} \sum_{\ell=0}^{k} \xi_{ji}(\ell), \tag{4.1}$$

for $i = 1, 2, \ldots, n$, and $j = 1, 2, \ldots, m$. Let $[\mathbf{y}]_i \in [0, 1]^m$ denote $(y_{1i}, y_{2i}, \ldots, y_{mi})$ and $\mathbf{y}_j \in [0, 1]^n$ denote $(y_{j1}, y_{j2}, \ldots, y_{jn})$. Additionally, let $\mathbf{y} \in ([0, 1]^n)^m$ denote $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$. Note that we use bold letters to denote vectors. Moreover, we assume that agent i has a cost function $g_i : [0, 1]^m \to \mathbb{R}_+$ which associates a cost to a certain allotment of the shared resources to the agent. We assume that g_i is twice continuously differentiable, strictly convex, and increasing in all variables, for all i. We also assume that the agents do not share their cost functions or allocation information with other agents, and there is no inter-agent communication. Moreover, we assume that there is a central agent or control unit that tracks the aggregate demands of resources and broadcasts price signals in the network. Then instead of defining the resource allocation problem in terms of the instantaneous allocation $\xi_{ji}(k) \in \{0, 1\}$, for all i, j and k, we define the objective and constraints in terms of averages as follows,

$$\begin{aligned} \min_{y_{11}, \ldots, y_{mn}} \quad & \sum_{i=1}^{n} g_i(y_{1i}, \ldots, y_{mi}), \\ \text{subject to} \quad & \sum_{i=1}^{n} y_{ji} = C_j, \quad j = 1, \ldots, m, \\ & y_{ji} \geq 0, \quad i = 1, \ldots, n, \text{ and, } j = 1, \ldots, m. \end{aligned} \tag{4.2}$$

Let $\mathbf{y}^* = (y_{11}^*, \ldots, y_{mn}^*) \in ([0, 1]^n)^m$ denote the solution to (4.2). Let $\mathbb{N}$ denote the set of

natural numbers, and let $k \in \mathbb{N}$ denote the time steps. Next, our objective is to propose a distributed iterative algorithm that determines instantaneous allocation $\{\xi_{ji}(k)\}$ and ensures that the long-term average allocation, as defined in (4.1) converge to optimal allocation (treated in Section 4.4) that is,

$$\lim_{k\to\infty} y_{ji}(k) = y^*_{ji}, \text{ for } i = 1, 2, \ldots, n, \text{ and } j = 1, 2, \ldots, m,$$

thereby achieving the minimum social cost in the sense of long-term averages. By compactness of the constraint set, optimal solutions exist. The assumption that the cost function g_i is strictly convex leads to strict convexity of $\sum_{i=1}^{n} g_i$, which follows that the optimal solution is unique.

4.2.1 Optimality conditions

Let $\mathcal{L} : ([0,1]^n)^m \times \mathbb{R}^m \times (\mathbb{R}^n)^m \to \mathbb{R}$, and let $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_m) \in \mathbb{R}^m$ and $\boldsymbol{\lambda} = (\lambda_{11}, \lambda_{12}, \ldots, \lambda_{mn}) \in (\mathbb{R}^n)^m$ are Lagrange multipliers of the constraints in Problem (4.2). We now define Lagrangian of the optimization problem (4.2) as follows:

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \triangleq \sum_{i=1}^{n} g_i(y_{1i}, \ldots, y_{mi}) - \sum_{j=1}^{m} \mu_j \left(\sum_{i=1}^{n} y_{ji} - C_j \right) + \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_{ji} y_{ji}.$$

Now, let $\frac{\partial}{\partial y_{ji}} g_i(y_{1i}, \ldots, y_{mi})$ denote (partial) derivative of the cost function g_i with respect to y_{ji}, for all i and j. For strictly positive $y^*_{1i}, \ldots, y^*_{mi}$, $i = 1, 2, \ldots, n$, following a similar analysis as in Chapter 2, we find that the derivatives of the cost functions of all agents competing for a particular resource reach consensus at optimal allocations. That is, the following holds, for $i, u \in \mathcal{N}$, and $j \in \mathcal{M}$:

$$\frac{\partial}{\partial y_{ji}} g_i([\mathbf{y}]_i)\Big|_{[\mathbf{y}]_i = [\mathbf{y}]^*_i} = \frac{\partial}{\partial y_{ju}} g_u([\mathbf{y}]_u)\Big|_{[\mathbf{y}]_u = [\mathbf{y}]^*_u}. \tag{4.3}$$

Recall that $[\mathbf{y}]_i$ denotes $(y_{1i}, y_{2i}, \ldots, y_{mi})$. Moreover, the Karush-Kuhn-Tucker (KKT) conditions are satisfied by the consensus of derivatives (see (4.3)) of the cost functions that are necessary and sufficient conditions of optimality of the optimization Problem (4.2); a similar analysis is done in (Alam et al., 2018a, 2020; Wirth et al., 2019), readers can find further details of the KKT conditions at Chapter 5.5.3 (Boyd & Vandenberghe, 2004). In this chapter, we use this principle to show that

the proposed algorithm reaches optimal values asymptotically. The consensus of derivatives of cost functions are also used in (Wirth et al., 2019) (single resource), (Alam et al., 2018a, 2020) (multi-resource—stochastic), (Alam, Shorten, Wirth, & Yu, 2018b) (multi-resource—derandomized) to show the convergence of allocations to optimal values. For ease of readability of the chapter, we presented the optimality results here. It also appears in Chapter 1.4.5.

4.3 Allocating a single unit-demand resource

In this section, we consider the single unit-demand resource case by Griggs and co-authors (Griggs et al., 2016) and briefly describe the distributed, iterative, and stochastic allocation algorithm. We also provide results on convergence and optimality properties with a few assumptions.

With a single resource, we simplify notation by dropping the index j. Each agent i has a strictly convex cost function $g_i : [0,1] \to \mathbb{R}_+$. The binary random variable $\xi_i(k) \in \{0,1\}$ denotes the allocation of the unit resource for agent i at time step k. Let $y_i(k)$ be the average allocation up to time step k of agent i that is,

$$y_i(k) \triangleq \frac{1}{k+1} \sum_{\ell=0}^{k} \xi_i(\ell), \quad \text{for } i = 1, 2, \ldots, n. \tag{4.4}$$

Let the vectors $\boldsymbol{\xi}(k) = (\xi_1(k), \ldots, \xi_n(k)) \in \{0,1\}^n$ and $\mathbf{y}(k) = (y_1(k), \ldots, y_n(k)) \in [0,1]^n$, for all k.

The idea is to choose the probability for random variable ξ_i so as to ensure convergence to the socially optimum value and to adjust overall consumption to the desired level (capacity constraint) C by applying a signal Ω to the probability, we call this *price signal*. When an agent joins the network at time step $k \in \mathbb{N}$, it receives the price signal $\Omega(k)$. At each time step k, the central agency updates $\Omega(k)$ using a gain parameter τ, past utilization of the resource, and the resource capacity. After the update, it broadcasts the new value to all agents in the network:

$$\Omega(k+1) \triangleq \Omega(k) - \tau\Big(\sum_{i=1}^{n} \xi_i(k) - C\Big), \tag{4.5}$$

$$\text{where } \tau \in \left(0, \left(\max_{\mathbf{y}\in[0,1]^n} \sum_{i=1}^{n} \frac{y_i}{g_i'(y_i)}\right)^{-1}\right). \tag{4.6}$$

Upon receiving this signal, agent i responds in a probabilistic fashion based on its available information. The probability distribution function $\sigma_i(\cdot)$ uses the average allocation of the resource to agent i and the derivative g_i' of the cost function g_i, is given by:

$$\sigma_i(\Omega(k), y_i(k)) \triangleq \Omega(k)\frac{y_i(k)}{g_i'(y_i(k))}, \text{ for } i \in \mathcal{N}. \tag{4.7}$$

Agent i updates its resource demand at each time step either by demanding one unit of the resource or not demanding it, as follows,

$$\xi_i(k+1) = \begin{cases} 1 & \text{with probability } \sigma_i(\Omega(k), y_i(k)); \\ 0 & \text{with probability } 1 - \sigma_i(\Omega(k), y_i(k)). \end{cases} \tag{4.8}$$

We point out that for the above formulation, we require assumptions on the cost function g_i and the admissible value of Ω because the scheme requires that (4.7) does, in fact, define a probability. For ease of notation, we define $v_i(z) \triangleq z/g_i'(z)$, where $z \in [0,1]$, and $v(\mathbf{y})$ to be the vector with components $v_i(y_i)$, for $i = 1, 2, \ldots, n$, where $\mathbf{y} \in [0,1]^n$.

Definition 4.3.1 (Admissibility)**.** *Let $n \in \mathbb{N}$, and let $g_i : [0,1] \to \mathbb{R}_+$ be continuously differentiable and strictly convex, for $i = 1, \ldots, n$. We call the set $\{g_i, i = 1, \ldots, n\}$ and $\Omega > 0$ admissible, if*

(i) v_i is well defined on $[0,1]$, for $i = 1, \ldots, n$,

(ii) there are constants $0 < a < b < 1$, such that $\sigma_i(\Omega, z) = \Omega v_i(z) \in [a,b]$, for $i = 1, \ldots, n$, and $z \in [0,1]$.

The definition of admissibility imposes several restrictions on the possible cost function g_i, similar to those imposed in (Wirth et al., 2019). See this reference for a detailed discussion and possible relaxation. For the case that Ω is a constant, that is, Ω does not depend on the time step $k \in \mathbb{N}$, (4.5) is not active. Therefore, the convergence of the scheme follows using tools from classical stochastic approximation (Borkar, 2008).

Definition 4.3.2 (Invariant set (Borkar, 2008)). *Let $\mathbf{x} \in \mathbb{R}^n$ and $w : \mathbb{R}^n \to \mathbb{R}^n$. Furthermore, let $A \subset \mathbb{R}^n$ be a closed set. The set A is called an invariant set of the ordinary differential equation $\dot{\mathbf{x}}(t) = w(\mathbf{x}(t))$, $\mathbf{x}(0) = \mathbf{x}_0$, if $\mathbf{x}_0 \in A$ implies that the corresponding solution trajectory $\mathbf{x}(t)$ is also in A.*

Definition 4.3.3 (Internally chain transitive set (Borkar, 2008)). *Let $A \subset \mathbb{R}^n$ be a closed set. It is called an internally chain transitive if for any $\mathbf{a}, \mathbf{b} \in A$ and any $\epsilon > 0$, $T > 0$, there exist $\nu \geq 1$ and points $\mathbf{x}_0 = \mathbf{a}, \ldots, \mathbf{x}_\nu = \mathbf{b} \in A$ such that the trajectories of the ordinary differential equation $\dot{\mathbf{x}}(t) = w(\mathbf{x}(t))$, $\mathbf{x}(0) = \mathbf{x}_0$, initiated at $\mathbf{x}_\ell$ meets with the ϵ-neighborhood of $\mathbf{x}_{\ell+1}$ for $0 \leq \ell \leq \nu$ after time $t \geq T$.*

Theorem 4.3.4. *(Borkar, 2008, Theorem 11.2.5) Let $\mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^n$. If $w : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous. Then the ordinary differential equation $\dot{\mathbf{x}}(t) = w(\mathbf{x}(t))$, $\mathbf{x}(0) = \mathbf{x}_0$ is well posed, that is, it has a unique solution $\mathbf{x}(t)$ for the initial value $\mathbf{x}_0$.*

Theorem 4.3.5. *(Borkar, 2008, Theorem 2.2) Let $\mathbf{x}, \mathbf{M} \in \mathbb{R}^n$ and $w : \mathbb{R}^n \to \mathbb{R}^n$. For an initial value $\mathbf{x}(0) = \mathbf{x}_0$ and $k \in \mathbb{N}$, let $\mathbf{x}$ be formulated as follows*

$$\mathbf{x}(k+1) = \mathbf{x}(k) + a(k)\left[w(\mathbf{x}(k)) + \mathbf{M}(k+1)\right]. \tag{4.9}$$

If Assumptions 4.3.6 (i) to (iv) are satisfied then $\{\mathbf{x}(k)\}$ converges to a compact connected internally chain-transitive invariant set of the ordinary differential equation $\dot{\mathbf{x}}(t) = w(\mathbf{x}(t))$, almost surely, for $t \geq 0$.

Assumption 4.3.6. *(i) The map w is Lipschitz continuous.*

(ii) Step-size $a(k) > 0$, for $k \in \mathbb{N}$, and

$$\sum_{\ell=0}^{\infty} a(\ell) = \infty, \text{ and } \sum_{\ell=0}^{\infty} \left(a(\ell)\right)^2 < \infty.$$

(iii) $\{\mathbf{M}(k)\}$ is a martingale difference sequence with respect to the σ-algebra $\mathcal{F}_k$ generated by

the events up to time step k, that is

$$\mathbb{E}\left(\mathbf{M}(k+1) \mid \mathcal{F}_k\right) = 0,$$

almost surely, for $k \in \mathbb{N}$. Moreover, for l^2-norm $\|\cdot\|^2$, martingale difference sequence $\{\mathbf{M}(k)\}$ is square-integrable that is,

$$\mathbb{E}\left(\|\mathbf{M}(k+1)\|^2 \mid \mathcal{F}_k\right) \leq \eta(1 + \|\mathbf{x}(k)\|^2),$$

almost surely, for $k \in \mathbb{N}$, and $\eta > 0$.

(iv) Sequence $\{\mathbf{x}(k)\}$ is almost surely bounded.

Theorem 4.3.7. *For $i = 1, 2, \ldots, n$ and $k \in \mathbb{N}$, let $y_i(k)$ be defined as in (4.4) and $\mathbf{y} = (y_1, \ldots, y_n)$. Let $\mathbf{y}(0) = \mathbf{y}_0$ be initial conditions and $\Omega > 0$ be constant. Assume that the cost function $g_i : [0, 1] \to \mathbb{R}_+$ is strictly convex, twice continuously differentiable and strictly increasing in each variable, for $i = 1, \ldots, n$. Also, assume that $\{g_i, i = 1, \ldots, n\}$ and Ω are admissible. Then $\{\mathbf{y}(k)\}$ converges to a unique value $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$ characterized by the condition,*

$$\Omega = g_i'(\hat{y}_i), \quad \text{for } i = 1, \ldots, n. \tag{4.10}$$

Proof. By definition, we have,

$$\mathbf{y}(k+1) = \frac{k}{k+1}\mathbf{y}(k) + \frac{1}{k+1}\boldsymbol{\xi}(k+1). \tag{4.11}$$

Let $\boldsymbol{\sigma}(\Omega, \mathbf{y}(k))$ denote the vectors with entries $\sigma_i(\Omega, y_i(k))$, for $i = 1, 2, \ldots, n$, and $k = 0, 1, 2, \ldots$ Thus, (4.11) may be reformulated as

$$\begin{aligned} &\mathbf{y}(k+1) \\ &= \mathbf{y}(k) + \frac{1}{k+1}\left[(\boldsymbol{\sigma}(\Omega, \mathbf{y}(k)) - \mathbf{y}(k)) + (\boldsymbol{\xi}(k+1) - \boldsymbol{\sigma}(\Omega, \mathbf{y}(k)))\right]. \end{aligned} \tag{4.12}$$

Furthermore, let $(\boldsymbol{\xi}(k+1) - \boldsymbol{\sigma}(\Omega, \mathbf{y}(k)))$ be denoted by $\mathbf{M}(k+1)$, and the step-size $\frac{1}{k+1}$ be denoted by $a(k)$, for $k \in \mathbb{N}$. Also, let $(\boldsymbol{\sigma}(\Omega, \mathbf{y}(k)) - \mathbf{y}(k))$ be denoted by $w(\mathbf{y}(k))$. After replacing these values in (4.12), we obtain

$$\mathbf{y}(k+1) = \mathbf{y}(k) + a(k)\left[w(\mathbf{y}(k)) + \mathbf{M}(k+1)\right]. \tag{4.13}$$

We now verify that Assumptions 4.3.6 (i) to (iv) are satisfied for formulation (4.13). Recall that $w(\mathbf{y}(k)) = (\boldsymbol{\sigma}(\Omega, \mathbf{y}(k)) - \mathbf{y}(k))$; thus, the map $w : \mathbf{y} \mapsto \boldsymbol{\sigma}(\Omega, \mathbf{y}) - \mathbf{y} = \Omega v(\mathbf{y}) - \mathbf{y}$ is Lipschitz, which satisfies Assumption 4.3.6 (i). Also, the step-size $a(k) = \frac{1}{k+1}$ is positive, for $k = 0, 1, 2, \ldots$, and we derive that

$$\sum_{\ell=0}^{\infty} a(\ell) = \infty, \text{ and } \sum_{\ell=0}^{\infty} \big(a(\ell)\big)^2 < \infty,$$

which satisfy Assumption 4.3.6 (ii). Additionally, we note that the expectation:

$$\mathbb{E}\left(\boldsymbol{\xi}(k+1) - \boldsymbol{\sigma}(\Omega, \mathbf{y}(k)) \mid \mathcal{F}_k\right) = 0, \tag{4.14}$$

where $\mathcal{F}_k$ is the σ-algebra generated by the events up to time step k. This follows immediately from the definition of the probability $\sigma_i(\cdot)$. By (4.14), $\{\mathbf{M}(k)\}$ is a martingale difference sequence with respect to σ-algebra; also, the sequence $\{\boldsymbol{\xi}(k+1) - \boldsymbol{\sigma}(\Omega, \mathbf{y}(k))\}$ is of course bounded. With little manipulation, we show that the martingale difference sequence $\{\mathbf{M}(k)\}$ is square-integrable—which satisfy Assumption 4.3.6 (iii). Moreover, the iterate $\mathbf{y}(k) \in [0, 1]^n$ is bounded almost surely, which satisfies Assumption 4.3.6 (iv). Thus, it follows from Theorem 4.3.5 that almost surely $\{\mathbf{y}(k)\}$ converges to a compact connected internally chain-transitive invariant set of the ordinary differential equation,

$$\begin{aligned} \dot{\mathbf{y}}(t) &= w(\mathbf{y}(t)) \\ &= \Omega v(\mathbf{y}(t)) - \mathbf{y}(t), \end{aligned} \tag{4.15}$$

with initial conditions $\mathbf{y}(0) = \mathbf{y}_0$. Moreover, as $w(\cdot)$ is Lipschitz continuous, thus from Theorem

4.3.4 we obtain that the ordinary differential equation (4.15) has a unique solution.

We now analyze the critical or equilibrium point(s) of the ordinary differential equation (o.d.e.) (4.15). Let the equilibrium points of the o.d.e. (4.15) be $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n) \in [0,1]^n$. We know that the equilibrium points satisfy that $w(\mathbf{y}(t)) = 0$. Thus, by definition, we obtain

$$\Omega v(\hat{\mathbf{y}}) - \hat{\mathbf{y}} = 0.$$

Furthermore, we obtain

$$\Omega v_i(\hat{y}_i) - \hat{y}_i = 0, \quad \text{for } i = 1, 2, \ldots, n.$$

We now show that these equilibrium points are asymptotically stable in $[0,1]^n$. To do so, let us choose a point smaller than the equilibrium point $\hat{y}_i$, let the point be $y_i = 0$. As $0 < \Omega v_i(y_i) < 1$, thus we have

$$\Omega v_i(y_i) - y_i > 0, \quad \text{at } y_i = 0, \text{for } i = 1, 2, \ldots, n.$$

Thus, the slope of the o.d.e. is positive and hence, the solution trajectory $y_i(t)$ is going upward towards the equilibrium point $\hat{y}_i$. Let us now choose a point greater than the equilibrium point $\hat{y}_i$, let the point be $y_i = 1$; as $0 < \Omega v_i(y_i) < 1$, thus we have

$$\Omega v_i(y_i) - y_i < 0, \quad \text{at } y_i = 1, \text{for } i = 1, 2, \ldots, n.$$

Thus, the slope is negative, and hence, the solution trajectory $y_i(t)$ is going downward towards the equilibrium point. In both cases, the trajectory is going towards the equilibrium point. We conclude that the equilibrium point $\hat{y}_i$ is asymptotically stable, for $i = 1, 2, \ldots, n$. Recall that for an unstable solution, the solution trajectories tend away from the equilibrium points.

As we have $\Omega \hat{y}_i / g_i'(\hat{y}_i) - \hat{y}_i = 0$ at equilibrium points. Thus, we obtain

$$\Omega = g_i'(\hat{y}_i), \quad \text{for } i = 1, 2, \ldots, n.$$

As g_i is strictly convex, g_i' is strictly increasing; hence, the equilibrium point $\hat{y}_i$ is unique, for $i = 1, 2, \ldots, n$. This determines the unique equilibrium point of $\{\mathbf{y}(k)\}$. The proof is complete. □

4.4 Allocating multiple unit-demand resources

We turn our attention in this section to the case of multiple unit-demand (indivisible) resources shared by the same population of agents as in the single resource case. We present a new algorithm that generalizes the single unit-demand resource algorithm of Section 4.3 to multiple unit-demand shared resources. The agents are coupled through multiple shared resources.

Each agent in the network runs the distributed unit-demand multi-resource allocation algorithm. Let $\tau_j \in (0, 1)$ be the gain parameter, and let $\Omega_j(k)$ denote a normalization factor updated and broadcast by the central agent, we call it the *price signal*. Additionally, let C_j represent the capacity of resource j, for all j. The central agent updates $\Omega_j(k)$ according to (4.16) at each time step k and broadcasts it to all agents in the network, for all j. When an agent joins the network at time step k, it receives the parameter $\Omega_j(k)$ for resource j, for all j. Each agent's algorithm updates its resource demand at a time step—either by demanding one unit of the resource or not demanding it.

The price signal $\Omega_j(k)$ depends on its value at the previous time step, τ_j, capacity C_j, and the total utilization of resource j at the previous time step, for all j and k. After receiving this signal, agent i's algorithm responds in a probabilistic manner. It calculates its probability $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$ using its average allocation $[\mathbf{y}]_i(k)$ of resource j and the derivative of its cost function, for all j and k, as described in (4.17). Agent i finds out the outcome of Bernoulli trial for resource j at time step k, outcome 1 occurs with probability $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$ and outcome 0 with probability $1 - \sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$; based on the value 0 or 1, the algorithm decides whether to demand one unit of the resource j or not. If the value is 1, then the algorithm demands one unit of the resource; otherwise, it does not demand the resource; analogously, it is done for all the resources in the network. This process repeats over time. Following this, the average allocations converge to optimal allocations. The proposed *unit-demand multi-resource allocation* algorithm for the central agent is

presented in Algorithm 7, and the algorithm for each agent is presented in Algorithm 8.

Algorithm 7: Algorithm of the central agent.

Input: $C_1, \ldots, C_m, \tau_1, \ldots, \tau_m, \xi_{11}(k), \ldots, \xi_{mn}(k)$, for $k \in \mathbb{N}$ and $i \in \mathcal{N}$.

Output: $\Omega_1(k+1), \Omega_2(k+1), \ldots, \Omega_m(k+1)$, for $k \in \mathbb{N}$.

Initialization: $\Omega_j(0) \leftarrow 0.350$, for $j \in \mathcal{M}$,

foreach $k \in \mathbb{N}$ **do**

  foreach $j \in \mathcal{M}$ **do**

    calculate $\Omega_j(k+1)$ according to (4.16) and broadcast it in the network;

  end

end

Algorithm 8: Multi resource allocation algorithm of agent i.

Input: $\Omega_1(k), \Omega_2(k), \ldots, \Omega_m(k)$, for $k \in \mathbb{N}$.

Output: $\xi_{1i}(k+1), \xi_{2i}(k+1), \ldots, \xi_{mi}(k+1)$, for $k \in \mathbb{N}$.

Initialization: $\xi_{ji}(0) \leftarrow 1$ and $y_{ji}(0) \leftarrow \xi_{ji}(0)$, for $i \in \mathcal{N}$, $j \in \mathcal{M}$.

foreach $k \in \mathbb{N}$ **do**

  foreach $j \in \mathcal{M}$ **do**

    $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k)) \leftarrow \Omega_j(k) \frac{y_{ji}(k)}{\frac{\partial}{\partial y}\Big|_{y=y_{ji}(k)} g_i([\mathbf{y}]_i(k))}$;

    generate Bernoulli independent random variable $b_{ji}(k)$ with the parameter $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$;

    if $b_{ji}(k) = 1$ **then**

      $\xi_{ji}(k+1) \leftarrow 1$;

    else

      $\xi_{ji}(k+1) \leftarrow 0$;

    end

    $y_{ji}(k+1) \leftarrow \frac{k+1}{k+2} y_{ji}(k) + \frac{1}{k+2} \xi_{ji}(k+1)$;

  end

end

After introducing the algorithms, we now describe how to calculate different parameters. We

choose a small *gain parameter* τ_j in $(0,1)$. We define $\Omega_j(k+1)$ which is based on the utilization of resource $j \in \mathcal{M}$ at time step k, as

$$\Omega_j(k+1) \triangleq \Omega_j(k) - \tau_j \left(\sum_{i=1}^{n} \xi_{ji}(k) - C_j \right). \tag{4.16}$$

After receiving the price signal $\Omega_j(k)$ from the central agent at time step k, agent i responds with probability $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$ in the following manner to demand resource j at next time step:

$$\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k)) \triangleq \Omega_j(k) \frac{y_{ji}(k)}{\frac{\partial}{\partial y}\Big|_{y=y_{ji}(k)} g_i([\mathbf{y}]_i(k))}. \tag{4.17}$$

Notice that $\Omega_j(k)$ is used to bound the probability $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k)) \in (0,1)$, for all i, j and k. Following the algorithm, the long-term average allocations converge to optimal allocations. Let $\boldsymbol{\xi}_j(k) \in \{0,1\}^n$ and $\mathbf{y}_j(k) \in [0,1]^n$ denote the vectors with entries $\xi_{ji}(k), y_{ji}(k)$, respectively, and $\boldsymbol{\sigma}_j(\Omega_j(k), \mathbf{y}(k))$ denotes the vector with entries $\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$, for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and $k = 0, 1, 2, \ldots$ We now present the convergence results for a fixed constant Ω_j.

4.4.1 Proof of convergence of average allocations for multiple resources

This section describes convergence result for multi-resource case with a constant Ω_j, for $j = 1, 2, \ldots, m$. Moreover, similar to the single resource case, we can restate the definition of admissibility as in Definition 4.3.1 and the theorem of convergence of average allocation $\mathbf{y}_j(k)$ for a constant Ω_j, for $j = 1, 2, \ldots, m$, as in Theorem 4.3.7. We state the generalized theorem of convergence of average allocations of multi-resource as follows.

Theorem 4.4.1 (Convergence of average allocations)**.** *Let $n, m \in \mathbb{N}$. Assume that the cost function $g_i : [0,1]^m \to \mathbb{R}_+$ is strictly convex, continuously differentiable and strictly increasing in each variable, for $i = 1, \ldots, n$. Furthermore, let $\Omega_j > 0$, and assume that $\{g_i, i = 1, \ldots, n\}$ and Ω_j are admissible, for $j = 1, \ldots, m$. Then for unique value $\hat{\mathbf{y}}_j = (\hat{y}_{j1}, \hat{y}_{j2}, \ldots, \hat{y}_{jn})$; almost surely we have $\lim_{k\to\infty} \mathbf{y}_j(k) = \hat{\mathbf{y}}_j$, where $\hat{\mathbf{y}}_j$ is characterized by the condition,*

$$\Omega_j = \frac{\partial}{\partial y_{ji}} g_i([\mathbf{y}]_i)\Big|_{[\mathbf{y}]_i = [\hat{\mathbf{y}}]_i}, \tag{4.18}$$

for $i = 1, \ldots, n$, *and* $j = 1, \ldots, m$.

Proof. For $j = 1, 2, \ldots, m$, we reformulate the average allocation $\mathbf{y}_j(k)$ as:

$$\mathbf{y}_j(k+1) = \frac{k}{k+1}\mathbf{y}_j(k) + \frac{1}{k+1}\boldsymbol{\xi}_j(k+1). \tag{4.19}$$

Which may be further reformulated as:

$$\begin{aligned} &\mathbf{y}_j(k+1) \\ &= \mathbf{y}_j(k) + \frac{1}{k+1}\left[(\boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k)) - \mathbf{y}_j(k)) + (\boldsymbol{\xi}_j(k+1) - \boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k)))\right]. \end{aligned} \tag{4.20}$$

Let $(\boldsymbol{\xi}_j(k+1) - \boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k)))$ be denoted by $\mathbf{M}_j(k+1)$, and the step-size $\frac{1}{k+1}$ be denoted by $a(k)$, for $k \in \mathbb{N}$. Also, let $(\boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k)) - \mathbf{y}_j(k))$ be denoted by $\omega_j(\mathbf{y}_j(k))$. After replacing these values in (4.20), we obtain

$$\mathbf{y}_j(k+1) = \mathbf{y}_j(k) + a(k)\left[\omega_j(\mathbf{y}_j(k)) + \mathbf{M}_j(k+1)\right]. \tag{4.21}$$

Here, for a fixed j, $\{\mathbf{M}_j(k)\}$ is a martingale difference sequence with respect to σ-algebra; moreover, the sequence $\{\boldsymbol{\xi}_j(k+1) - \boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k))\}$ is bounded. For a fixed j, (4.21) is similar to (4.13); the proof follows the single resource case. □

Remark 4.4.2 (Privacy of an agent)**.** *The central agent only knows about the aggregate utilization* $\sum_{i=1}^{n} \xi_{ji}(k)$ *of resource* $j \in \mathcal{M}$ *at time step* k *that ensures the privacy of the probability distribution function and the cost function of an agent.*

Notice that there is no inter-agent communication required to achieve social optimum cost, but the central agent broadcasts the price signals $\Omega_j(k)$ at each time step. Hence, the network has very little communication overhead. Suppose that $\Omega_j(k)$ takes the floating-point values represented by μ bits. If there are m unit-demand (indivisible) resources in the network, then the communication overhead in the network will be μm bits per time unit.

As we stated in Chapter 1, we restate that because our models consider a central server that

keeps track of aggregate demands and sends feedback signals in the network, the system may fail if the central server stops working. However, we can fix this by adding a backup server that keeps all the information as the central server and receives feedback signals from the central server. When the central server stops sending the feedback signals, the backup server takes charge and works as the central server until the central server comes live. Such settings are explored extensively by the machine learning community as federated learning. Wherein several agents collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020). For clarity, we restate it in each chapter in the thesis.

4.5 Application to electric vehicle charging

In this section, we present a hypothetical scenario to regulate the number of electric vehicles that share a limited number of level 1 and level 2 charging points, using Algorithms 7 and 8. We illustrate through numerical results that utilization of charging points (level 1 or level 2) is concentrated around its capacity. Moreover, agents receive the optimal charging points in long-term averages; we verify this using the consensus of derivatives of cost functions of agents, which satisfies all the KKT conditions for the optimization Problem 4.2, as described in Section 4.2.

As a background, the transportation sector in the US contributed around 27% of greenhouse gas (GHG) emissions in 2015, in which light-duty vehicles like cars have a 60% contribution. Furthermore, the share of carbon dioxide (CO_2) is 96.7% of all GHG gases from the transportation sector (EPA, 2017). To put it in context, currently, we have more than 1 billion vehicles (electric (EV) as well as internal combustion engine (ICE)) on the road worldwide (Sousanis, 2015); the number is increasing very rapidly, which will result in increased CO_2 emissions in future. Though electric-only vehicles produce zero emissions, the electricity generating units produce GHG emissions at the

sources, depending on the power generation technique used; for example, thermal-electric, hydro-electric, wind power, nuclear power, etc. The annual CO_2 emissions by an electric vehicle (EV) is $2,079.7$ kg (share of CO_2 emissions in producing electricity for charging an EV) and an ICE is $5,186.8$ kg (Energy, 2018).

Consider a situation where a city sets aside several free (no monetary cost) *electric vehicle supply equipment* (EVSE) that supports level 1 and level 2 chargers at a public EV charging station to serve the residents or to promote usage of electric vehicles or for load balancing on the power grids, etc. The voltage and current rating of chargers vary; interested readers can find details at (Q. Wang, Liu, Du, & Kong, 2016; Yilmaz & Krein, 2013). Briefly, the level 1 charger works at 110–120 Volt (V) AC, 15–20 Ampere (A), and the level 2 charger works at 240 V AC, 20–40 A. It takes around 8–12 hours to fully charge the battery of an EV with a level 1 charger; additionally, it takes around 4–6 hours to fully charge the battery with a level 2 charger. The charging time depends on the battery capacity, onboard charger capacity, and a few other factors (Schey, 2014). Suppose now that the city has installed C_1 EVSEs, which support level 1 chargers, and C_2 EVSEs, which support level 2 chargers. We consider n electric cars in the network; their cost functions are coupled through average allocations of level 1 and level 2 charging points. The city must decide whether to allocate a level 1 charging point or a level 2 charging point to an electric car to regulate the number of cars utilizing charging points for social welfare. Clearly, in such a situation, charging points should be allocated in a distributed manner that preserves the privacy of individual car users but also maximizes the benefit to the city. We use the proposed distributed stochastic algorithm, which ensures the privacy of electric car users and allocates charging points optimally to maximize social welfare; for example, to minimize total electricity cost or CO_2 emissions.

On average, 0.443 kg of CO_2 is produced to generate and distribute 1 kWh of electric energy with mixed energy sources (assoc. BEV, 2009). Based on the voltage and current ratings of level 1 and level 2 chargers, and using fundamental formulations, we calculate the amount of CO_2 emissions in generation and distribution of electricity, illustrated in Table 4.1. We use this data to formulate the cost function g_i of electric car i.

Charger type	power (kW)	CO_2 emission in four hours
Level 1	1.65–2.40	2.92–4.25 kg
Level 2	4.80–9.60	8.51–17.01 kg

Table 4.1: Amount of CO_2 emissions in generation and distribution of electricity for level 1 and level 2 chargers in four-hours duration.

Suppose that there are n electric cars competing for the level 1 and level 2 charging points; each car user has private cost function g_i which depends on the average allocations of level 1 and level 2 charging points, $y_{1i}(k)$ and $y_{2i}(k)$, respectively, for $i = 1, 2, \ldots, n$. We assume that the city agency (central agent) broadcasts the price signals $\Omega_1(k)$ and $\Omega_2(k)$ in the network after every four hours; for the simplicity of calculations and due to the charging rate of level 2 chargers, we chose a duration of 4 hours. Moreover, suppose that the cost functions are classified into four classes based on—the type of vehicle, its battery capacity, on-board charger capacity, and a few other factors. We consider that a vehicle belongs to one of the classes. Let a and b be constants, based on the values in Table 4.1, we let $a = 2.9$, $b = 8.51$. Additionally, let c_{1i}, c_{2i} be uniformly distributed random variables, where $c_{1i} \in [1, 1.5]$, $c_{2i} \in [1, 2]$, for $i = 1, 2, \ldots, n$. The cost function g_i is listed in (4.22), where first and second terms represent CO_2 emissions at a basic assumed rate of charging of the battery, whereas third and subsequent terms represent the amount of CO_2 emissions due to different charging losses or factors. We observe that no allocation of charging points produces zero emissions. The cost functions are as follows:

$$g_i(y_{1i}, y_{2i}) = \begin{cases} (i) \quad ay_{1i} + by_{2i} + ac_{1i}(y_{1i})^2 + bc_{2i}(y_{2i})^4, \\ (ii) \quad ay_{1i} + by_{2i} + ac_{1i}(y_{1i})^4/2 + bc_{2i}(y_{2i})^2, \\ (iii) \quad ay_{1i} + by_{2i} + ac_{1i}(y_{1i})^4/3 + ac_{1i}(y_{1i})^6 + bc_{2i}(y_{2i})^4, \\ (iv) \quad ay_{1i} + by_{2i} + ac_{1i}(y_{1i})^2 + bc_{2i}(y_{2i})^6/6. \end{cases} \tag{4.22}$$

For the simulation setup, we consider $n = 1200$ electric cars competing for level 1 and level 2 chargers. The cost functions of electric cars are coupled through average allocations of level 1 and level 2 chargers. Moreover, we classify the electric cars as follows—cars 1 to 300 belong to class 1, cars 301 to 600 belong to class 2, cars 601 to 900 belong to class 3, and cars 901 to 1200 belong to

class 4. Each class has a set of cost functions; the cost functions of class 1 are presented in (4.22)(i) and analogously for other classes. We consider the capacity of level 1 chargers $C_1 = 400$, and the capacity of level 2 chargers $C_2 = 500$. The parameters of the algorithms are initialized with the following values: $\Omega_1(0) = 0.328$, $\Omega_2(0) = 0.35$, $\tau_1 = 0.001275$, and $\tau_2 = 0.001175$. We use the proposed Algorithm 7 and Algorithm 8 to allocate charging points to the electric cars in the network. A car user sends a request for the charging point to the city agency in a probabilistic manner based on its private cost function g_i and its previous average allocation of level 1 and level 2 charging points. The car users do not share their cost functions or history of their allocations with other car users or with the city agency. Note a limitation of this application, following the proposed algorithm, in some cases, a car user can receive access to both level 1 and level 2 charging points for a single car, which may not be desired in real-life scenarios.

We present simulation results of the automatic allocation of charging points here. We observe that the electric car users receive optimal allocations of both types of charging points and minimize the overall CO_2 emissions. Figure 4.1 shows the evolution of average allocations of charging points.

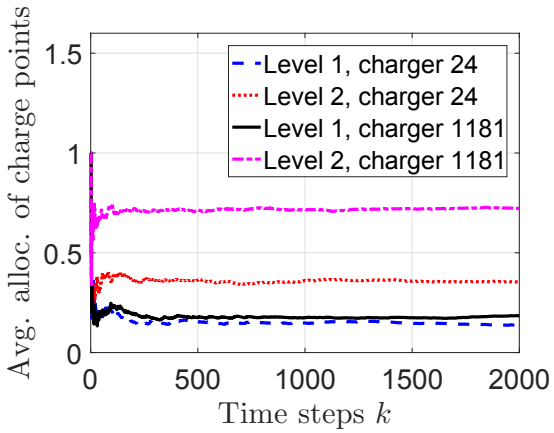
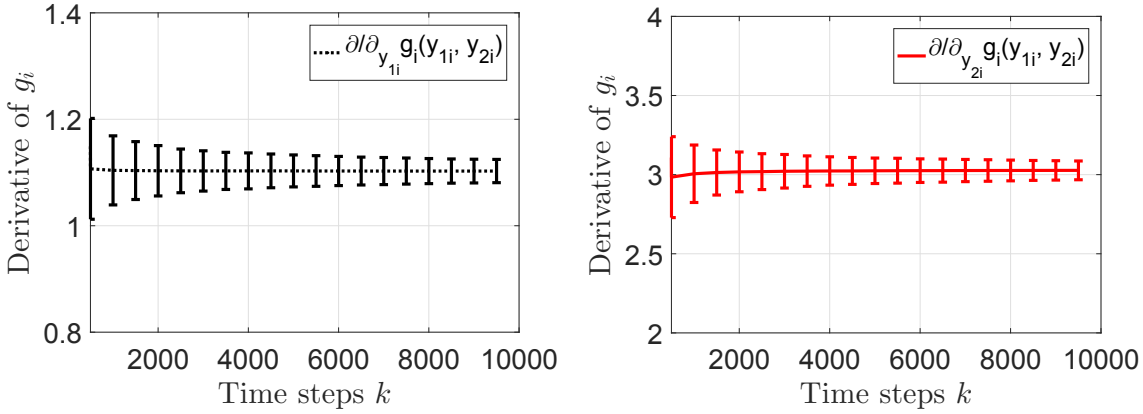


Figure 4.1: Evolution of average allocations of charging points.

As described earlier in (4.3), to show the optimality of the solution, the partial derivatives of the cost functions of all the cars with respect to a particular type of charger should make a consensus. The profile of derivatives of cost functions of the cars with respect to level 1 and level 2 chargers for a single simulation is illustrated in Figure 4.2(a) and 4.2(b), respectively. We observe that they converge with time and hence make a consensus, which satisfies the KKT conditions for optimality.

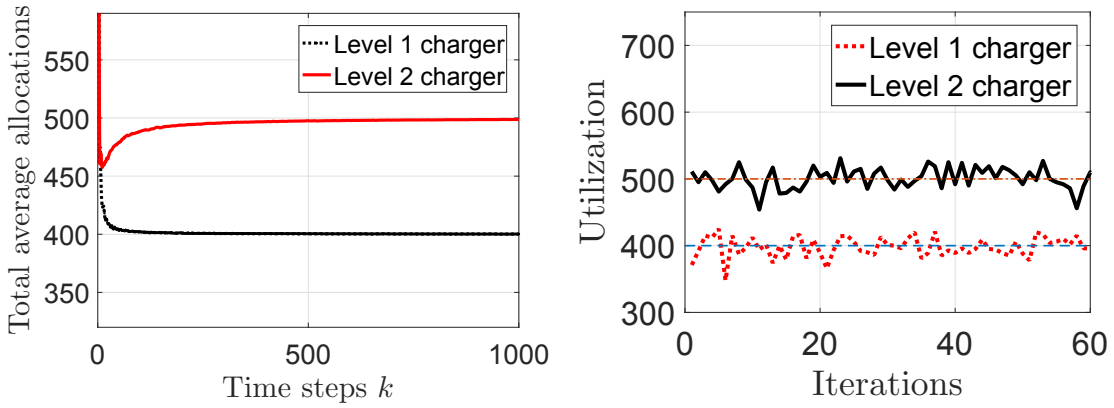
Note that we use third and subsequent terms of (4.22) to calculate the derivative $\frac{\partial}{\partial y_{ji}} g_i(\cdot)$ which shifts its value by constants a or b without affecting the KKT points, but it provides faster convergence in the simulation. The empirical results thus obtained show the convergence of the long-term average allocations of charging points to their respective optimal values using the consensus of derivatives of the cost functions; hence, the network achieves social optimum value.



(a) (b)

Figure 4.2: (a) evolution of the profile of derivatives of cost functions g_i of all the electric cars in the network with respect to level 1 chargers, (b) evolution of the profile of derivatives of cost functions g_i of all the electric cars in the network with respect to level 2 chargers.

Figure 4.3(a) illustrates the sum of the average allocations $\sum_{i=1}^{n} y_{ji}(k)$ over time, for $i = 1, 2, \ldots, n$ and $j = 1, 2$. We observe that the sum of the average allocations of charging points converge to respective capacity over time that is, for large k, $\sum_{i=1}^{n} y_{ji}(k) \approx C_j$, for all j. We further illustrate the utilization of charging points for the last 60 time steps in Figure 4.3(b). It is observed that most of the time, the total allocation of charging points is concentrated around its capacity. Figure 4.4 shows the evolution of price signals $\Omega_1(k)$ and $\Omega_2(k)$ defined in (4.16).



(a) (b)

Figure 4.3: (a) Evolution of the sum of average allocations of charging points, (b) utilization of charging points over the last 60 time steps, capacities of level 1 and level 2 chargers are $C_1 = 400$ and $C_2 = 500$, respectively.

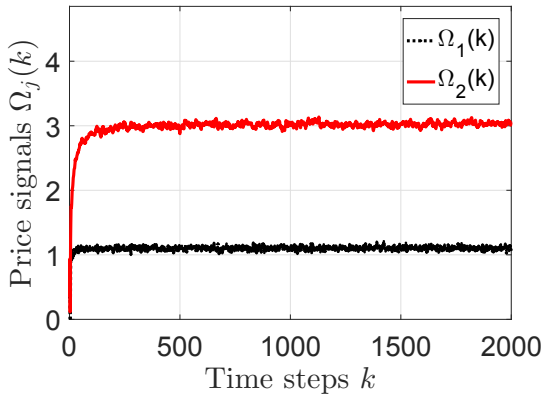


Figure 4.4: Evolution of price signals $\Omega_1(k)$ and $\Omega_2(k)$, defined in (4.16).

4.6 Conclusion

We developed a new distributed stochastic algorithm to solve a class of multiple indivisible (unit-demand) resource allocation problems. The solution does not require communication between agents but little communication with a central agent. The chapter extends an existing unit-demand single resource allocation algorithm to multiple unit-demand shared resources. In the

single-resource case, based on a constant price signal, we showed the convergence of long-term average allocations to unique equilibrium points using techniques from classical stochastic approximation. We further extended the results to multiple shared resources. Additionally, experiments show that the long-term average allocations converge rapidly to optimum values in the multi-resource case.

Open problems are to prove convergence with a time-varying price signal for single-resource as well as multi-resource cases. Another open problem is to analyze the rate of convergence of the algorithms.

Chapter 5

Distributed Algorithms for Prosumer Markets

Internet-of-Things (IoT) enables the development of sharing economy applications. In many sharing economy scenarios, agents both produce as well as consume a resource; we call them *prosumers*. A community of prosumers agrees to sell excess resources to another community in a prosumer market. In this chapter, we develop a control-theoretic approach to regulate the number of prosumers in a prosumer community, where each prosumer has a cost function that is coupled through its time-averaged production and consumption of the resource. Furthermore, each prosumer runs its distributed algorithm and takes only binary decisions in a probabilistic way, whether to produce one unit of the resource or not and to consume one unit of the resource or not. In the developed approach, prosumers do not explicitly exchange information with each other due to privacy reasons but little exchange of information is required for feedback signals broadcast by a central agency. In the developed approach, prosumers achieve the optimal values asymptotically. Furthermore, our algorithm is suitable to implement in an IoT context with minimal demands on infrastructure. We describe two use cases; community-based car-sharing and collaborative energy storage for prosumer markets. We also present numerical results to check the efficacy of the algorithms.

5.1 Introduction

Recently, consumers across a range of sectors have started to embrace shared ownership of resources and services with guaranteed access, as opposed to more traditional business models that focus on sole-ownership only. The reasons for this trend are multi-faceted, and range from societal issues, such as the need to reduce wastage, and more general environmental concerns (Narasimhan et al., 2018), (Hamari, Sjoklint, & Ukkonen, 2016), (J. Lan, Ma, Zhu, Mangalagiu, & Thornton, 2017), (T. D. Chen & Kockelman, 2016), to pure monetary opportunities arising from increased connectivity (and the ability that this gives to advertise the availability of unused resources and services) (Crisostomi et al., 2017). Well-known examples of successful companies building *sharing economy* products include Airbnb (hospitality), Lyft (ride sharing) (Fraiberger & Sundararajan, 2015), Bird and Lime (scooter sharing), Mobike (bike sharing) (J. Lan et al., 2017), and Google (Google reviews—information sharing).

Roughly speaking, several types of sharing application classes are discerned (as described in (Crisostomi et al., 2017)).

A. **Opportunistic sharing:** Services based on opportunistic sharing of resources exploit large-scale availability of either unused resources or obsolete business models or both. Examples of products in this area include the parking application JustPark (`www.justpark.com`) and the peer-to-peer car-sharing services Getaround (`www.getaround.com`). The key enablers for such products are mechanisms for informing agents of available resources, their delivery, and payments.

B. **Federated negotiation and sharing:** In this case, agents group together to negotiate better contracts for utilities (electricity, gas, water, health, etc.) or to provide mutually beneficial services such as collaborative energy storage. The key enablers for such products are mechanisms for grouping communities and enforcing contractual obligations for federations of like-minded consumers.

C. **Bespoke sharing:** In this case, products are designed with the specific objective of being shared rather than for sole ownership. A basic example of such systems is devices and services that allow several users to share a single electric charge-point. Other examples include time-shared apartments or cars that are owned by several people rather than a single person (Crisostomi et al., 2017).

D. **Hybrid sharing:** Finally, opportunities also exist for sharing economy to support the regular economy. We readily find examples of such systems in the hospitality industry, where ad-hoc sharing economy infrastructure (spare rooms in local houses) can be used as a buffer to accommodate excess demand in the regular economy (hotels).

The common characteristic in all of the above application classes is the ability for community-wide communication and actuation to enable services to be bought and sold and so that contracts can be enforced.

The development of sharing economy applications (Huckle, Bhattacharya, White, & Beloff, 2016), (Kortuem & Bourgeois, 2016) is facilitated by Internet-of-Things (IoT). For example, IoT helps to perform secure payments, track the location and condition of an object, and list a few. Interested readers can find several IoT-based applications in (Atzori et al., 2010), (Al-Fuqaha et al., 2015) and the papers cited therein. Therefore, while the value of the sharing economy is not in question (Goudin, 2016; Hamari et al., 2016) and while many of the essential infrastructural elements needed for the deployment of such systems are being developed rapidly, there is an additional requirement for structured platforms to enable distributed *community-wide* buying and distributed *community-wide* selling. Currently, such platforms are at a very early stage of development, with significant opportunities for improvement.

Our objective in this chapter is to address this deficit partially and to develop tools to support the design of community-based prosumer markets. We define *prosumers* as agents that both produce and consume a resource (Ritzer & Jurgenson, 2010). Specifically, we are interested in developing

light algorithms that can easily be deployed on modest IoT platforms and that can be used to support distributed community-wide buying and selling of resources. Here, by *light*, we mean algorithms that place low demands on the infrastructure, both in terms of computational power and actuation and connectivity requirements of individual prosumers. A fundamental requirement is also that such algorithms are scale-free in the sense that they can operate across a range of community sizes; from small communities of a few prosumers to larger communities made up of very many prosumers. These constraints are directly related to the challenges associated with uncertainties that arise in the context of sharing economy problems. Typically, at any time instant, one does not know how many prosumers are participating in the sharing scheme; whether prosumers can or are willing to communicate with each other (perhaps due to privacy considerations), and whether enough computational power is available to the whole network to allocate resources in real-time optimally. A further complication is that we would like any scheme that we develop to be backward compatible with old IoT platforms that support only essential interaction between prosumers and infrastructure. Thus, there is considerable interest in developing *light* algorithms that place only modest demands on infrastructure, yet can be used to implement complex policies in the face of the uncertainties mentioned above.

Given this context, we are particularly interested in situations where communities come together to purchase and sell related commodities simultaneously. Such systems arise, for example, in energy systems where agents both produce and consume energy (Ritzer & Jurgenson, 2010), (Moret & Pinson, 2018), (Agnew & Dargusch, 2015).

5.2 Prosumer markets and communities

Prosumers are the agents that both produce and consume resources (Ritzer & Jurgenson, 2010), (Moret & Pinson, 2018), (Parag & Sovacool, 2016). We are interested in prosumer markets that facilitate a community of prosumers for distributed production and consumption. Such markets are emerging rapidly in the energy sector (Inderberg, Tews, & Turner, 2018; Schill, Zerrahn, & Kunz, 2017), but also in other areas such as shared mobility (Grijalva, Costley, & Ainsworth, 2011). Parag

and Sovacool (Parag & Sovacool, 2016) classify prosumer markets according to three network architectures [1].

(i) *Prosumer-to-prosumer model:* In this peer-to-peer model, prosumers interact (buy or sell resource) directly with each other as depicted in Figure 5.1. This model is widespread. For example, consider the case of the car-sharing platform Turo. Here, car owners list their cars on the Turo sharing platform, and the riders book the cars of their choice through this platform for a certain period with a fee. A similar peer-to-peer model is proposed in (C. Zhang et al., 2018) for energy trading in microgrids.

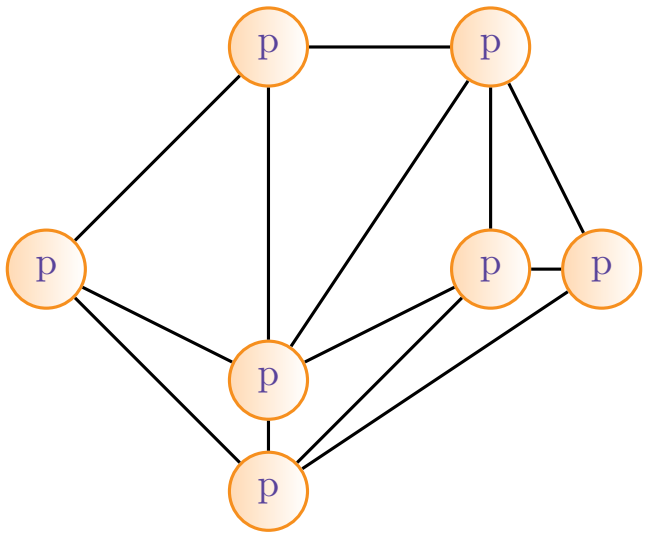
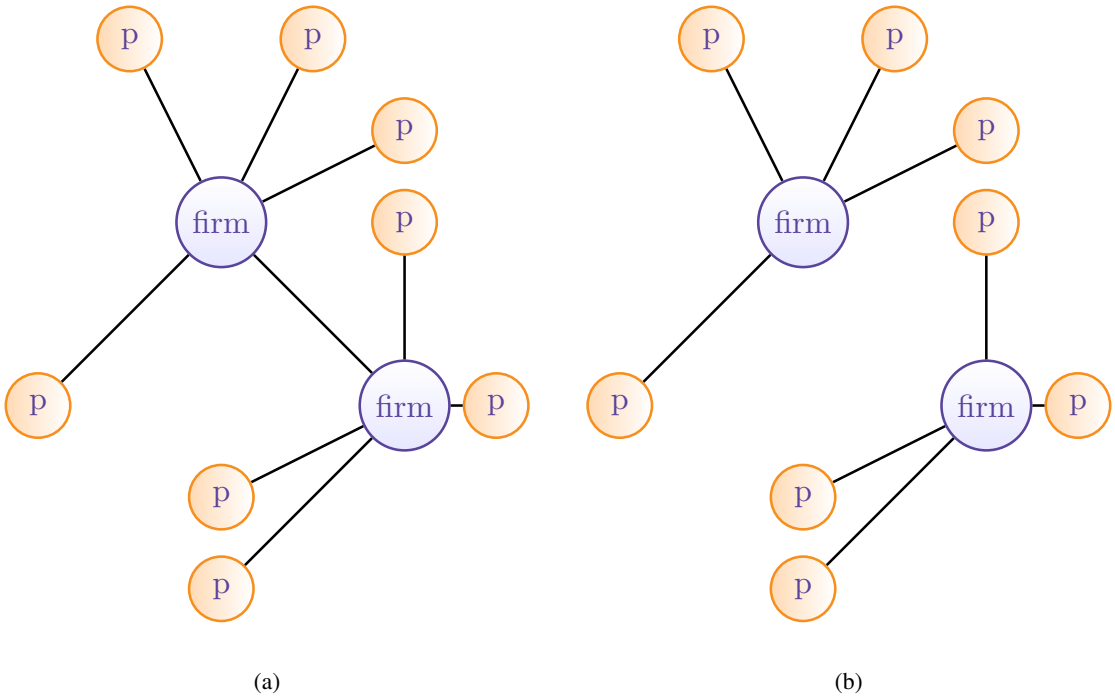


Figure 5.1: Prosumer-to-prosumer model. Here *p* represents a prosumer.

(ii) *Prosumer-to-firm model:* In this model, prosumers interact with a local firm directly. There are two types of prosumer to firm models, *prosumer-to-interconnected-firm* model, and *prosumer-to-isolated-firm* model. In the prosumer-to-interconnected-firm model, prosumers are connected to a local firm, which may be connected to the main firm as presented in Figure 5.2(a). For example, suppose that prosumers produce energy from renewable sources and are connected to a microgrid. A prosumer satisfies its energy needs from the microgrid and the energy

[1] In the network architectures, *p* represents a prosumer.

it produces. If the prosumer produces more energy than it needs; then, it can return excess energy to the microgrid; this microgrid may be connected to the main grid. Whereas, in the prosumer-to-isolated-firm model, prosumers are connected to the local firm, which works in isolation as depicted in Figure 5.2(b). An example of the prosumer-to-isolated-firm model is Island microgrid (de Souza Ribeiro, Saavedra, de Lima, & de Matos, 2011) in which prosumers and microgrid work together to fulfill the energy need of prosumers on the Island.



(a) (b)

Figure 5.2: Prosumer-to-firm models—(a) prosumer-to-interconnected-firm model, (b) prosumer-to-isolated-firm model.

(iii) *Community-based prosumer model:* In this model, prosumers are located in the same geographic location who have similar resource needs and resource production pattern; more generally, they share common goals and interests. These prosumers are grouped to interact with each other and efficiently manage the resource needs of the community, as depicted in Figure 5.3. In this case, communities may also exchange resources with each other. A recent example of community-based trip sharing is found at (Hasan et al., 2018) in which

the algorithm clusters commuters in communities to optimize car usage. We clarify that, for simplicity, we consider a single prosumer community that interacts with another community (*external community*) in the rest of the chapter unless otherwise stated.

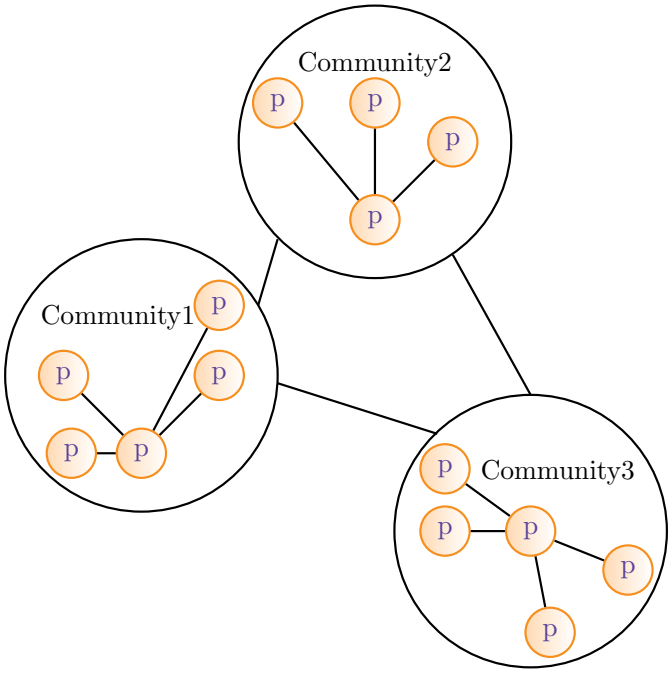


Figure 5.3: Community-based prosumer model.

Figures 5.1, 5.2, and 5.3 are adapted from (Parag & Sovacool, 2016).

While work on analytics to help design prosumer markets is still in its infancy, somewhat surprisingly, few papers have begun to deal with some of the complex market design issues associated with such systems (Moret & Pinson, 2018), (Gkatzikis et al., 2014), (Iosifidis & Tassiulas, 2017), (Georgiadis et al., 2020). Roughly speaking, these papers deal with two main issues; (i) the existence of market equilibria and (ii) methods to allocate resources amongst (competing) prosumers. It is in this latter context that this present work is placed. Generally speaking, resource allocation algorithms for prosumer markets have until now been formulated in an optimization context and can be categorized as is traditionally done for classical optimization models. Namely, resources are either allocated centrally (Einav, Farronato, & Levin, 2016) as in the case of Uber, whereby drivers are assigned to the passengers; or in a distributed fashion as in the case of Airbnb, whereby

guests and hosts choose each other; or using hybrid of the above two, as in the case of Didi Chuxing (Narasimhan et al., 2018). As we have said, typically, these allocation problems are formulated in an optimization context, paying particular attention to the certain constraints that arise in the sharing economy. These include privacy of the individual, fair allocation of resources, the satisfaction of service level agreements, and ever-increasing regulatory constraints (for example, in the case of Airbnb).

Our contribution in this chapter is to address these problems using a different approach. Namely, we shall consider these problems (in a control-theoretic context) as regulation problems with optimality constraints. In particular, we are interested in applications where prosumer communities both buy and sell resources from or to one, or more external entities. Thus, we take the view that such communities have a contract to both buy and sell a pre-specified level of a resource at a given time instant, and the objective of the allocation algorithm is to ensure that these levels of demand are met. Given this basic setting, we ask the question as to whether this can be done without any explicit exchange of information between individual prosumers in situations where prosumers take only binary decisions (buy or sell), and whether, given these constraints, optimal use of the resource can be realized. We shall see in the next section that it is indeed possible to formulate the problem and develop distributed algorithms that achieve all of these properties and which can be implemented in an IoT context with minimal demands on infrastructure.

5.2.1 Contribution

We are primarily interested in applications where prosumer communities buy and sell resources via contracts to one or more external entities and must ensure that they meet certain demands in real-time. Thus, the objective of any allocation algorithm is to ensure that these levels of demand are met. Typically, a problem of this nature would be solved in a standard optimization framework. Unfortunately, this approach is not available to us due to the uncertainty that prevails for this system class. For example, the number of prosumers participating at a given time instant may vary, as may the contracted level of prosumption. Also, for reasons of privacy, prosumers may only communicate with each other or with the infrastructure in a limited fashion, making the communication graph unknown *a-priori*, from the point of view of algorithm development. Thus, our approach is

to formulate these allocation problems in a control-theoretic setting, where the effect of these uncertainties and other disturbances can be dealt with using feedback. Our principal contribution is, therefore, to develop distributed control algorithms that can asymptotically achieve optimality.

5.3 Problem statement

Let us assume that a prosumer market consists of a prosumer community and an *external community*. The prosumer community has $N \in \mathbb{N}$ prosumers producing and consuming a resource, which agrees to sell its excess resource to the external community. We also assume that the prosumer market has a control unit, which measures the aggregate consumption and production of the resource. It communicates with the prosumer community as well as the external community; we call it a *sharing platform*. Additionally, notice that N is not known to the individual prosumers participating in the scheme. For simplicity, we assume that there is a single resource produced and consumed by all prosumers, though our formulation can easily be extended to multiple resources and multiple prosumer communities. In our model, the process of production and consumption takes place at discrete time instants $t_0 < t_1 < t_2 < \ldots$, where $t_0 = 0$. At each time instant, the overall production and consumption of the previous time instant are evaluated and adjusted. Additionally, we assume that communities and external agencies are contracted to, *on aggregate*, consume and produce a certain amount of the resource at time instant t_k, for $k = 0, 1, 2, \ldots$

We assume that each prosumer has limited actuation and at any time instant t_k, for $k = 0, 1, 2, \ldots$, either it consumes one unit of the resource or it does not consume it. Similarly, each prosumer either produces one unit of the resource at a time instant, or it does not produce it. Thus, for each prosumer i, we denote by $x_i(k)$ the amount of the resource consumed, and by $y_i(k)$ the amount of the resource produced at time instant t_k.[2]

We also assume that there are constants $C_x \geq 0$ and $C_y \geq 0$, specifying the aggregate consumption and production bounds, respectively. Notice that the constants C_x and C_y are known to the sharing

[2]Depending on the application, the processes of production and consumption may be more appropriately modeled on a continuous time-scale. In this case, we interpret time instants t_k at those times in which the prosumption over the interval $(t_{k-1}, t_k]$ is accounted for.

platform but not to individual prosumers in the market. Thus, at each time instant t_k, we require:

$$\sum_{i=1}^{N} x_i(k) = C_x, \quad \text{and,} \tag{5.1}$$

$$\sum_{i=1}^{N} y_i(k) = C_y. \tag{5.2}$$

Our primary objective is to ensure that these prosumption bounds are met at each time instant t_k, for $k = 0, 1, 2, \ldots$ However, we are particularly interested in situations where production and consumption are coupled together. For example, in communities that are formed to produce energy, the time-averaged production and consumption of energy might be coupled through battery storage requirements. Furthermore, in the community-based car-sharing prosumer market, the average number of delivered and received "cars" might be coupled through the desired value of utilization of cars; and similarly, production of a resource might depend on its consumption. Thus, in these situations, we would like to ensure that consumption and production bounds (5.1) and (5.2) are met asymptotically. To formulate this as a long-term requirement, we introduce the time-averaged consumption:

$$\overline{x}_i(k) \triangleq \frac{1}{k+1} \sum_{\ell=0}^{k} x_i(\ell), \quad \text{for } i = 1, \ldots, N, \tag{5.3}$$

with the time-averaged production $\overline{y}_i(k)$ defined analogously. Now, let $T_i \in \mathbb{R}_+$ be the desired value of utilization of the resource, for $i = 1, \ldots, N$. Thus, depending on the application (refer Section 5.5, cf. (5.26)), we might require:

$$\lim_{k\to\infty} \overline{x}_i(k) + \overline{y}_i(k) = T_i, \quad \text{for } i = 1, \ldots, N, \tag{5.4}$$

with some additional constraints on $\overline{x}_i(k)$ and $\overline{y}_i(k)$. In several applications, such as smart energy trading wherein prosumers may aim to sell a fixed fraction of its produced energy from renewable sources to others, it might require, for $\alpha_i \in [0, 1]$:

$$\lim_{k\to\infty} \overline{x}_i(k) = \alpha_i T_i, \quad \text{and,} \lim_{k\to\infty} \overline{y}_i(k) = (1-\alpha_i) T_i, \quad \text{for } i = 1, \ldots, N. \tag{5.5}$$

Fortunately, in problems that we consider, there is flexibility in satisfying these constraints and it is enough to satisfy that for sufficiently large k, we have:

$$\overline{x}_i(k) + \overline{y}_i(k) \quad \approx \quad T_i, \quad \text{and,} \tag{5.6}$$

$$\overline{x}_i(k) \approx \alpha_i T_i, \quad \text{and,} \quad \overline{y}_i(k) \approx (1-\alpha_i) T_i, \tag{5.7}$$

for $i = 1, \ldots, N$. To formulate this mathematically, we associate a cost $g_i : [0,1]^2 \to \mathbb{R}_+, (x_i, y_i) \mapsto g_i(x_i, y_i)$ to the deviation of the actual long-term prosumptions.

Assumption 5.3.1 (Cost function)**.** *The cost function $g_i(\cdot)$ is strictly convex, strictly increasing in each variable, and is continuously differentiable, for all i.*

Given this basic setting, we are interested in solving the following optimization problem:

Problem 5.3.2 (Optimization)**.**

$$\min_{x_1,\ldots,x_N,y_1,\ldots,y_N} \quad \sum_{i=1}^{N} g_i(x_i, y_i), \tag{5.8}$$

$$\textit{subject to} \quad \sum_{i=1}^{N} x_i = C_x, \tag{5.9}$$

$$\sum_{i=1}^{N} y_i = C_y, \tag{5.10}$$

$$x_i \geq 0, \tag{5.11}$$

$$y_i \geq 0, \quad \textit{for } i = 1, 2, \ldots, N. \tag{5.12}$$

We assume that for this optimization problem an optimal point $(\mathbf{x}^*, \mathbf{y}^*) \in \mathbb{R}_+^{2N}$ exists, where $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_N^*)$ and $\mathbf{y}^* = (y_1^*, y_2^*, \ldots, y_N^*)$. The optimal point is unique by Assumption 5.3.1 of strict convexity of the cost function $g_i(\cdot)$. As our problem is timed, we aim to design a distributed algorithm determining, for each time instant t_k, values of $x_i(k)$ and $y_i(k)$, such that for

the long-term averages, we have:

$$\lim_{k\to\infty} \overline{x}_i(k) = x_i^*, \quad \lim_{k\to\infty} \overline{y}_i(k) = y_i^*, \quad \text{for } i = 1, \ldots, N. \tag{5.13}$$

Also, it is desirable that the constraints of the optimization problem are satisfied by $\overline{x}_i(k)$ and $\overline{y}_i(k)$, for every k.

Typically, a problem of this nature can be solved in a standard optimization framework. Unfortunately, this approach is not available to us for many reasons:

(i) The number of prosumers N in the prosumer community, and the constraints on C_x and C_y may vary with time, making an off-line computation of an optimal solution difficult.

(ii) To preserve privacy, we also assume that prosumers do not necessarily communicate with each other, and they only communicate with the infrastructure in a limited fashion. Thus, even the communication graph is unknown *a-priori* for this particular problem class. In particular, individual cost function g_i is often private and is not shared by prosumers.

(iii) We are interested in algorithms that self organize and converge to an optimal solution even in the presence of disturbances in the state information.

Our approach, therefore, is to treat the above problem as a feedback (stochastic) control problem, which changes the formulation in a minor way; namely, we allow:

$$\sum_{i=1}^{N} x_i(k) \approx C_x, \text{ and } \sum_{i=1}^{N} y_i(k) \approx C_y, \quad \text{for all } k. \tag{5.14}$$

In other words, we allow the instantaneous prosumption to undershoot or overshoot the reference values by a small amount. Then, given this background, and from Assumption 5.3.1 for the cost function g_i, for all prosumers i, we shall demonstrate that an elementary feedback control algorithm

can be devised to solve an approximate version of Problem 5.3.2. As we shall see, this algorithm requires only a few bits of message transfer as intermittent feedback from a control unit (sharing platform) to prosumers in the prosumer market, but no inter-prosumer communication is required.

5.4 Algorithms for community-based prosumer markets

The algorithms developed in this section are motivated by the following elementary argument. We consider only the case in which at time instant t_k, either 0 or 1 unit of the resource is consumed or produced, for $k = 0, 1, 2, \ldots$ For the sake of argument, we only consider the case of pure consumption in this preamble.

Let $z(k)$ denote the number of times an agent (consumer) consumes a unit resource until time instant t_k, where $k = 0, 1, 2, \ldots$ Let $\overline{z}(k) \triangleq \frac{1}{k+1} z(k)$, denote the time-averaged consumption of the resource until time instant t_k. We assume that the consumption at time instant t_k is decided by a stochastic procedure, where the probability of consumption of one unit of the resource is a function $p(\overline{z}(k))$. Additionally, we assume that this probability conditioned on $\overline{z}(k)$ is independent of the previous history of the process. Then:

$$z(k+1) = z(k) + w(k), \tag{5.15}$$

where $w(k)$ is a random variable taking the value 0 or 1. Thus, the following holds true:

$$\begin{aligned} \overline{z}(k+1) &= \frac{k+1}{k+2}\overline{z}(k) + \frac{1}{k+2} w(k) && (5.16) \\ &= \overline{z}(k) + \frac{1}{k+2}\big(w(k) - \overline{z}(k)\big). && (5.17) \end{aligned}$$

Recall that $p(\overline{z}(k))$ denotes the probability that $w(k) = 1$ at time instant t_k, for $k = 0, 1, 2, \ldots$ Then, we rewrite (5.17) as:

$$\overline{z}(k+1) = \overline{z}(k) + \frac{1}{k+2}\big(p(\overline{z}(k)) - \overline{z}(k)\big) + \frac{1}{k+2}\big(w(k) - p(\overline{z}(k))\big).$$

Note that systems of this form are discussed extensively in (Borkar, 2008). In particular, the term $\{w(k) - p(\overline{z}(k))\}$ is a martingale difference sequence and is treated as noise. It is shown in (Borkar, 2008) that under mild assumptions, $\overline{z}(k)$ converges almost surely. The basic idea of the remainder of this section is to construct stochastic feedback algorithms that mimic this argument. In particular, our basic idea in the sequel is to choose the probability distribution $p(\overline{z}(k))$, so that the stochastic system both solves a regulation problem and also optimization problem of the form of Problem 5.3.2, simultaneously.

5.4.1 Optimality conditions

In this subsection, we briefly discuss the optimality conditions for Problem 5.3.2, using Lagrangian multipliers. These optimality conditions lead to the state dependent probabilities that we alluded to in the preamble.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_N)$. Also, let μ^1, μ^2 and $\lambda^1 = (\lambda_1^1, \ldots, \lambda_N^1)$, $\lambda^2 = (\lambda_1^2, \ldots, \lambda_N^2)$ be the Lagrange multipliers corresponding to the equality constraints (5.9), (5.10) and the inequality constraints (5.11), (5.12), respectively. The Lagrangian of Problem 5.3.2 is defined as $\mathcal{L} : \mathbb{R}^{2N} \times \mathbb{R}^2 \times \mathbb{R}^{2N} \to \mathbb{R}$, where

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mu^1, \mu^2, \lambda^1, \lambda^2) = \sum_{i=1}^{N} g_i(x_i, y_i) - \mu^1(\sum_{i=1}^{N} x_i - C_x) - \mu^2(\sum_{i=1}^{N} y_i - C_y) + \sum_{i=1}^{N} \lambda_i^1 x_i + \sum_{i=1}^{N} \lambda_i^2 y_i.$$

We assume that the optimal value of Problem 5.3.2 is obtained for positive values of consumption and production. We, therefore, let $x_i^*, y_i^* \in (0, 1]$ denote the optimal point of Problem 5.3.2. By this assumption, the inequality constraints are not active, and it follows that the corresponding optimal Lagrange multipliers are $\lambda^{*1} = 0 = \lambda^{*2}$.

Additionally, let μ^{*1}, μ^{*2} be the optimal Lagrange multipliers for the equality constraints. The first order optimality condition is that the gradient vanishes, and inspection shows that the gradient condition decouples. Recall that $\frac{\partial}{\partial_{x_i}} g_i(\cdot)$ denotes the partial derivative of $g_i(\cdot)$ with respect to x_i and $\frac{\partial}{\partial_{y_i}} g_i(\cdot)$ denotes the partial derivative of $g_i(\cdot)$ with respect y_i. Then, we arrive at the following

conditions:

$$\frac{\partial}{\partial_{x_i}} g_i(x_i, y_i)\Big|_{x_i=x_i^*, y_i=y_i^*} = \mu^{*1}, \quad \text{for } i = 1, 2, \ldots, N,$$

and,

$$\frac{\partial}{\partial_{y_i}} g_i(x_i, y_i)\Big|_{x_i=x_i^*, y_i=y_i^*} = \mu^{*2}, \quad \text{for } i = 1, 2, \ldots, N.$$

In other words, we have:

$$\frac{\partial}{\partial_{x_i}} g_i(x_i, y_i)\Big|_{x_i=x_i^*, y_i=y_i^*} = \frac{\partial}{\partial_{x_j}} g_j(x_j, y_j)\Big|_{x_j=x_j^*, y_j=y_j^*}, \text{ for } i, j \in \{1, 2, \ldots, N\}. \tag{5.18}$$

Analogously, we obtain:

$$\frac{\partial}{\partial_{y_i}} g_i(x_i, y_i)\Big|_{x_i=x_i^*, y_i=y_i^*} = \frac{\partial}{\partial_{y_j}} g_j(x_j, y_j)\Big|_{x_j=x_j^*, y_j=y_j^*}, \text{ for } i, j \in \{1, 2, \ldots, N\}. \tag{5.19}$$

We find that the optimal values satisfy all the Karush-Kuhn-Tucker (KKT) conditions, which are necessary and sufficient conditions for optimality of differentiable convex functions (Chap. 5.5.3 (Boyd & Vandenberghe, 2004)). Hence, the derivatives of the cost functions of all prosumers with respect to consumption as well as production must reach consensus at the optimal point.
This type of consensus condition has been used in (Griggs et al., 2016; Wirth et al., 2019) (single resource case) and (Alam et al., 2018a) (multi-resource case) to derive place-dependent probabilities that ensure convergence to the consensus condition and thus, to the optimal point.

5.4.2 Algorithm for consumption

Here, we briefly describe the distributed algorithm proposed in (Griggs et al., 2016) for allocating a single resource to consumers. In this model, no production is taking place, hence the corresponding variables are omitted. Suppose that there are N consumers in a community. We assume that consumer i of the community has a cost function $g_i : [0, 1] \to \mathbb{R}_+$, which is strictly convex, strictly increasing in each variable, and continuously differentiable, for $i = 1, 2, \ldots, N$.

The random variable $x_i(k) \in \{0, 1\}$ denotes the consumption of the unit resource for consumer i at time instant t_k, for all i and k. As before, let $\overline{x}_i(k)$ be the time-averaged consumption of consumer i until time instant t_k, for all i and k.

The idea is to choose probabilities so as to ensure convergence to the social optimum and to adjust overall consumption by the community to the reference value (capacity) C_x, by applying a *feedback signal* $\Omega(k)$ to the probabilities. At each time instant t_k, the control unit updates $\Omega(k)$ using a gain parameter $\tau > 0$, the past aggregate consumption of the resource, and the capacity C_x as described in (5.20) and then, broadcasts the new value to all consumers in the community:

$$\Omega(k+1) \triangleq \Omega(k) - \tau\Big(\sum_{i=1}^{N} x_i(k) - C_x\Big). \tag{5.20}$$

After receiving this signal, consumer i responds in a probabilistic way. The probability distribution $\sigma_i(\cdot)$ is calculated using the time-averaged consumption $\overline{x}_i(k)$ and the derivative $g_i'(\cdot)$ of the cost function $g_i(\cdot)$, as follows:

$$\sigma_i(\Omega(k), \overline{x}_i(k)) \triangleq \Omega(k)\frac{\overline{x}_i(k)}{g_i'(\overline{x}_i(k))}. \tag{5.21}$$

Notice that the cost function g_i is chosen as increasing function in each variable so that the probability $\sigma_i(\cdot)$ is in the valid range, for all i. Now, consumer i updates its resource consumption at each time instant t_k, either by consuming one unit of the resource or not consuming it, as follows:

$$x_i(k+1) = \begin{cases} 1 & \text{with probability } \sigma_i(\Omega(k), \overline{x}_i(k)); \\ 0 & \text{with probability } 1 - \sigma_i(\Omega(k), \overline{x}_i(k)). \end{cases}$$

Empirical results show that the time-averaged consumption $\overline{x}_i(k)$ converges to the optimal value x_i^* asymptotically.

Remark 5.4.1 (Integral control action)**.** *Equation* (5.20) *defines what is called an integral control action in tracking control. The overall consumption by the community should "track" the available capacity, i.e., approximate it. In other words, the objective of the integrator is to ensure that the tracking error* $e(k) = \sum_{i=1}^{N} x_i(k) - C_x \approx 0$ *asymptotically.*

Remark 5.4.2 (Consensus of derivatives)**.** *The policy for $\sigma_i(\Omega(k), \overline{x}_i(k))$ in (5.21) is to ensure that asymptotically, $g_i'(\overline{x}_i(k)) = g_j'(\overline{x}_j(k))$, for all consumers i and j. As is discussed in (Griggs et al., 2016), the convergence of the above algorithm and strict convexity of all cost functions (and of course assuming the existence of a feasible solution in the constraint set) is enough to imply that Problem 5.3.2 is solved asymptotically for the consumption case.*

5.4.3 Algorithm for coupled prosumption

The case of *coupled prosumption* of a single resource is characterized by the constraint (5.4), that is, consumption and production are coupled through the desired value of utilization of the resource. The algorithm follows the case of exclusive consumption closely taking into account the cost function as discussed in the problem formulation (Section 5.3). Before proceeding, note that the following discussion extends to an arbitrary number of resources, but is presented for a single resource, both to aid exposition and to be consistent with the application class that is the principal consideration in this chapter. Interested readers can look at (Alam et al., 2018a) for preliminary results on multi-resource allocation. With this background in mind, following the discussion for consumption, let us assume that there is a community of N prosumers in a prosumer market. The prosumer community sells the excess resource to an external community. Furthermore, let τ_x, τ_y denote the gain parameters for consumption and production, $\Omega_x(k)$, $\Omega_y(k)$ denote the *feedback signals* for both processes, and $C_x,\ C_y$ represent the respective contract (capacity) constraints. We assume the existence of a centralized control unit (sharing platform) in the prosumer market that can measure the aggregate response of the prosumer community and broadcast a feedback signal in the prosumer market at each time instant t_k, for both consumption and production type. Specifically, the sharing platform updates the feedback signal $\Omega_x(k)$, as follows:

$$\Omega_x(k+1) \triangleq \Omega_x(k) - \tau_x\Big(\sum_{i=1}^{N} x_i(k) - C_x\Big), \quad \text{for all } k, \tag{5.22}$$

with $\Omega_y(k)$ updated analogously. After receiving a feedback signal prosumer i's algorithm responds in a probabilistic manner. The probability that prosumer i responds to the feedback signal is given

by:

$$\sigma_{i,x}(k) \triangleq \Omega_x(k) \frac{\frac{\partial}{\partial_x}\Big|_{x=\overline{x}_i(k)} g_i\big(\overline{x}_i(k), \overline{y}_i(k)\big)}{\overline{x}_i(k)}, \quad \text{for all } i \text{ and } k. \tag{5.23}$$

Here, $\sigma_{i,x}(k)$ denotes the probability of prosumer i, responding to a demand for consumption of the resource, at time instant t_k; similarly, $\sigma_{i,y}(k)$ denotes the probability of prosumer i, responding for production of the resource at time instant t_k, for all k. As in the consumption case, the cost function g_i is chosen as increasing function in each variable, to keep the probabilities $\sigma_{i,x}(k)$ and $\sigma_{i,y}(k)$ in the valid range. However, the definition of $\sigma_{i,x}(k)$ and $\sigma_{i,y}(k)$ is slightly different from the consumption case, described previously. Now, prosumer i updates its consumption and production of the resource at each time instant in the following way:

$$x_i(k+1) = \begin{cases} 1 & \text{with probability } \sigma_{i,x}(k); \\ 0 & \text{with probability } 1 - \sigma_{i,x}(k), \end{cases} \quad \text{and,}$$

$$y_i(k+1) = \begin{cases} 1 & \text{with probability } \sigma_{i,y}(k); \\ 0 & \text{with probability } 1 - \sigma_{i,y}(k), \end{cases}$$

for all i and k. Empirically, we observe that the time-averaged consumption $\overline{x}_i(k)$ converges to the optimal consumption value x_i^*, and similarly, the time-averaged production $\overline{y}_i(k)$ converges to the optimal production value y_i^* asymptotically, for all prosumers in the community. We describe the algorithm of the sharing platform in Algorithm 9 and the algorithm of prosumers of the community

in Algorithm 10.

Algorithm 9: Algorithm of control unit (sharing platform).

Input: $C_x, C_y, \tau_x, \tau_y, x_i(k), y_i(k)$, for $k = 0, 1, 2, \ldots$ and $i = 1, 2, \ldots, N$.

Output: $\Omega_x(k+1), \Omega_y(k+1)$, for $k = 0, 1, 2, \ldots$

Initialization: $\Omega_x(0) \leftarrow 0.06$ and $\Omega_y(0) \leftarrow 0.06$[1],

foreach $k = 0, 1, 2, \ldots$ **do**

calculate $\Omega_x(k+1)$ and $\Omega_y(k+1)$ as follows and broadcast them in the prosumer community;

$$\Omega_x(k+1) \leftarrow \Omega_x(k) - \tau_x\Big(\sum_{i=1}^{N} x_i(k) - C_x\Big), \quad \text{and,}$$

$$\Omega_y(k+1) \leftarrow \Omega_y(k) - \tau_y\Big(\sum_{i=1}^{N} y_i(k) - C_y\Big).$$

end

We make the following remarks.

Remark 5.4.3 (Communication overhead and privacy)**.** *There is no explicit communication between prosumers. Thus, the algorithm is low cost in terms of communication and is private.*

Remark 5.4.4 (Probability bounds)**.** *The gain parameters τ_x, τ_y are small constants chosen to ensure $\sigma_{i,x}(k)$ and $\sigma_{i,y}(k)$ are the probabilities; namely, are in [0,1], for all i and k.*

Remark 5.4.5 (Consensus of partial derivatives)**.** *The well-posedness of our algorithm follows from the assumption of strict convexity of $g_i(\cdot)$, and that the constraint sets are closed and bounded; namely, that there exists unique optimal solution to Problem 5.3.2. To show that the long-term average values converge to the optimal values, we use the consensus of (partial) derivatives of the cost functions of prosumers as described previously. That is:*

$$\lim_{k\to\infty} \frac{\partial}{\partial_x}\Big|_{x=\overline{x}_i(k)} g_i(\overline{x}_i(k), \overline{y}_i(k)) = \lim_{k\to\infty} \frac{\partial}{\partial_x}\Big|_{x=\overline{x}_i(k)} g_j(\overline{x}_j(k), \overline{y}_j(k)),$$

[1]We initialize $\Omega_x(0)$ and $\Omega_y(0)$ with positive real numbers.

and similarly,

$$\lim_{k\to\infty} \frac{\partial}{\partial_y}\Big|_{y=\overline{y}_i(k)} g_i(\overline{x}_i(k),\overline{y}_i(k)) = \lim_{k\to\infty} \frac{\partial}{\partial_y}\Big|_{y=\overline{y}_i(k)} g_j(\overline{x}_j(k),\overline{y}_j(k)),$$

for all $i,j \in \{1,2,\ldots,N\}$.

Algorithm 10: Algorithm of prosumer i.

Input: $\Omega_x(k), \Omega_y(k)$, for $k = 0,1,2,\ldots$

Output: $\overline{x}_i(k+1), \overline{y}_i(k+1)$, for $k = 0,1,2,\ldots$

Initialization: $x_i(0), y_i(0) \leftarrow 1$ and $\overline{x}_i(0) \leftarrow x_i(0)$ and $\overline{y}_i(0) \leftarrow y_i(0)$.

while *prosumer* i *is active at* $k = 0,1,2,\ldots$ **do**

$$\sigma_{i,x}(k) \leftarrow \Omega_x(k) \frac{\frac{\partial}{\partial_x}\Big|_{x=\overline{x}_i(k)} g_i\big(\overline{x}_i(k),\overline{y}_i(k)\big)}{\overline{x}_i(k)};$$
$$\sigma_{i,y}(k) \leftarrow \Omega_y(k) \frac{\frac{\partial}{\partial_y}\Big|_{y=\overline{y}_i(k)} g_i\big(\overline{x}_i(k),\overline{y}_i(k)\big)}{\overline{y}_i(k)};$$

calculate outcome of the random variables;

$$x_i(k+1) \leftarrow \begin{cases} 1 & \text{w. p. } \sigma_{i,x}(k) \\ 0 & \text{w. p. } 1-\sigma_{i,x}(k); \end{cases}$$
$$y_i(k+1) \leftarrow \begin{cases} 1 & \text{w. p. } \sigma_{i,y}(k) \\ 0 & \text{w. p. } 1-\sigma_{i,y}(k); \end{cases}$$

update $\overline{x}_i(k+1)$ and $\overline{y}_i(k+1)$ as follows;

$$\overline{x}_i(k+1) \leftarrow \frac{k+1}{k+2}\overline{x}_i(k) + \frac{1}{k+2}x_i(k+1);$$
$$\overline{y}_i(k+1) \leftarrow \frac{k+1}{k+2}\overline{y}_i(k) + \frac{1}{k+2}y_i(k+1);$$

end

As we stated in Chapter 1, we restate that because our models consider a central server that keeps track of aggregate demands and sends feedback signals in the network, the system may fail if the central server stops working. However, we can fix this by adding a backup server that keeps all the information as the central server and receives feedback signals from the central server. When the central server stops sending the feedback signals, the backup server takes charge and works as the central server until the central server comes live. Such settings are explored extensively by the machine learning community as federated learning. Wherein several agents collaborate to train a global model without sharing their local on-device data. Each agent updates the global model with their local dataset and parameters and shares the updates with the central server. The central server aggregates the updates by agents and updates the global model (McMahan et al., 2017), (Konecny et al., 2016), (Bonawitz et al., 2019), (Kairouz et al., 2021). The optimization techniques are called *federated optimization* (Reddi et al., 2021), (T. Li et al., 2020), (Konecny et al., 2015), (J. Wang et al., 2020). For clarity, we restate it in each chapter in the thesis.

5.5 Use cases

We now describe two use cases for the community-based prosumer market. The first is a transportation example and concerns a car-sharing prosumer market. The second example is from the energy sector and considers a prosumer market, where produced and consumed energy is coupled through storage constraints.

5.5.1 Community-based car-sharing

In this use case, we consider a community of N households (prosumers) with several cars, not all of which are required each day. We assume that cars are pooled and shared amongst community members to multiplex and monetize the excess capacity but that the average aggregate daily community demand for cars is known. We let T_i denote the average number of cars desired to be used by each household i, for all i. Suppose that C_x cars are required within the community each day, and an excess of C_y cars are made available to an external community each day. For simplicity, we assume that C_x and C_y are fixed. Notice that a household is a prosumer in the sense that it requires cars for

its transportation needs and supplies excess car-days to an external community. For each prosumer i, let $x_i(k) \in \{0,1\}$ denote that prosumer i requires a car at day k or not, and let $y_i(k) \in \{0,1\}$ denote that prosumer i supplies a car to an external community at day k or not. Thus, we assume that aggregated over the entire community, the demand for *shared cars* on a given day k is:

$$\sum_{i=1}^{N} x_i(k) = C_x, \tag{5.24}$$

leaving the *excess* capacity so that C_y cars can be supplied to an external community:

$$\sum_{i=1}^{N} y_i(k) = C_y. \tag{5.25}$$

Thus, $\sum_{\ell=0}^{k} x_i(\ell)$ is the number of days a car was required by prosumer i over k days, and $\sum_{\ell=0}^{k} y_i(\ell)$ is the number of days that the same prosumer made a car available to an external community. Thus, $\sum_{\ell=0}^{k} \big(x_i(\ell) + y_i(\ell)\big)$ is the total number of days that a car was used as a result of prosumer i. Over some days, prosumer i will require that the time-averaged of this number be equal to the desired value of utilization of cars, T_i. For example, if a prosumer has two cars, thus, over seven days period (a week) it has 14 car-days. Now, suppose that the prosumer only needs access to 10 car-days over a week. Then, this prosumer might choose $\sum_{\ell=0}^{6} x_i(\ell)$ and $\sum_{\ell=0}^{6} y_i(\ell)$, such that $\sum_{\ell=0}^{6} \big(x_i(\ell) + y_i(\ell)\big) = 12$. In this case, the prosumer sells access to two excess car-days over the week (two car-days rather than the maximum of four to provide some margin in case that a car is required for personal use more than expected). Let $\overline{x}_i(k)$ denote the average number of days prosumer i requires a car over k days, and let $\overline{y}_i(k)$ denote the average number of days prosumer i makes a car available to an external community over k days, for all i and k. Thus, over a long period, this constraint can be scaled by the number of days to yield:

$$\overline{x}_i(k) + \overline{y}_i(k) \approx T_i, \tag{5.26}$$

with the cost of not achieving this goal captured by a penalty function $g_i(\overline{x}_i + \overline{y}_i - T_i)$. For sufficiently large k, we might also require that $\overline{x}_i(k) \approx \alpha_i T_i$ and $\overline{y}_i(k) \approx (1-\alpha_i)T_i$ with $\alpha_i \in [0,1]$, for all i. This latter constraint can be formulated in terms of a cost via a penalty function. For

example, in residential areas in Ireland; two car households are common, meaning that households can in principle both consume and produce cars simultaneously, hence act as prosumers. However, this may not always be possible, and prosumers may be required to make alternative arrangements or pay penalties, should they not be able to meet contractual demands. To formulate the cost function, we associate costs $h_i : [0,1] \to \mathbb{R}_+, x_i \mapsto h_i(x_i)$ to the deviation from $\alpha_i T_i$ and $l_i : [0,1] \to \mathbb{R}_+, y_i \mapsto l_i(y_i)$ to the deviation from $(1-\alpha_i)T_i$, for all i. Then, the aim of the sharing platform is to minimize:

$$\sum_{i=1}^{N} \big(g_i(\overline{x}_i + \overline{y}_i - T_i) + h_i(\overline{x}_i - \alpha_i T_i) + l_i(\overline{y}_i - (1-\alpha_i)T_i)\big), \tag{5.27}$$

subject to the additional constraints listed in Problem 5.3.2.

5.5.2 Collaborative energy storage

In this use case, we again assume that N households in a community participate in a prosumer market. Each household is connected to the grid and has installed solar panels. Every household has batteries to store the energy either from the solar panel or the grid. The households act as prosumers in the sense that they can consume stored energy as well as sell excess energy for monetary benefits. Let $x_i(k) \in \{0,1\}$ denote that household i consumes stored energy at day k or not, and let $y_i(k) \in \{0,1\}$ denote that household i sells stored energy to an external community at day k or not, for all i. Let $\overline{x}_i(k)$ denote the time-averaged amount of stored energy consumed by household i over k days, and let $\overline{y}_i(k)$ denote the time-averaged amount of stored energy sold by household i to an external community over k days, for all i. Now, we assume that $\sum_{i=1}^{N} x_i(k) = C_x$, be the aggregated consumption of stored energy over the entire community on a given day k; whereas, the excess energy C_y is sold to an external community, with $\sum_{i=1}^{N} y_i(k) = C_y$. Furthermore, we assume that each household may have a constraint on the amount of energy stored in order to realize the above strategy. As in the previous use case, this *soft* constraint is captured as follows. Let T_i be the desired amount of energy stored by household i. Then, on average, over a sufficiently large given period k,

household i may expect to store the following desired amount of energy temporarily:

$$\overline{x}_i(k) + \overline{y}_i(k) \approx T_i, \tag{5.28}$$

with the deviation from this goal captured by a penalty function $g_i(\overline{x}_i + \overline{y}_i - T_i)$. We might also require that, roughly speaking, a certain amount of storage is reserved for consumption and a certain amount to sell (production). Hence, again, the objective of the sharing platform is to minimize:

$$\sum_{i=1}^{N} \big(g_i(\overline{x}_i + \overline{y}_i - T_i) + h_i(\overline{x}_i - \alpha_i T_i) + l_i(\overline{y}_i - (1-\alpha_i)T_i)\big),$$

subject to the constraints listed in Problem 5.3.2. Notice that the definition of $h_i(\cdot)$ and $l_i(\cdot)$ is same as the previous use case.

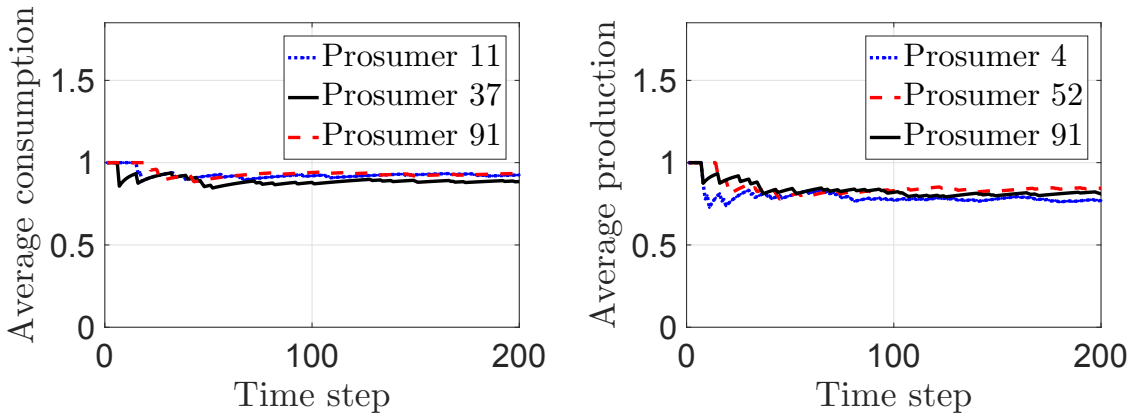
5.6 Numerical results

In this section, we present numerical results for 99 prosumers participating in a prosumer market. We assume that these $N = 99$ in the prosumer market are grouped into two prosumer communities. Prosumers 1 to 50 are grouped in Community 1, and Prosumers 51 to 99 are grouped in Community 2. Additionally, each community has a cost function type, and prosumers of a particular community use the cost function type of that community but with randomized parameter values. Recall that prosumers can produce the resource as well as consume it. Let time instants $t_0 < t_1 < t_2 < \ldots$, represent days, and let the capacity constraints be $C_x = 90$ and $C_y = 80$. Furthermore, let the cost factors $a_i \in \{1, 2, \ldots, 10\}$ and $b_i \in \{1, 2, \ldots, 15\}$ be drawn from uniformly distributed random variables. Additionally, let $\delta = 11.75$ and $\gamma = 11.65$. Notice that the values of δ and γ are chosen in such a way that the cost function $g_i(\cdot)$ is increasing in each variable. Recall that this is done to keep the probabilities $\sigma_{i,x}$ and $\sigma_{i,y}$ in $[0, 1]$. The cost functions are presented as

follows:

$$g_i(\overline{x}_i(k),\overline{y}_i(k)) = \begin{cases} \delta\big(\overline{x}_i(k)+\overline{y}_i(k)\big) + \frac{1}{4}a_i\big(\overline{x}_i(k)+\overline{y}_i(k)-T_1\big)^2 + \\ \frac{1}{2}\big(\overline{x}_i(k)-\frac{1}{2}T_1\big)^2 + \frac{1}{2}\big(\overline{y}_i(k)-\frac{1}{2}T_1\big)^2 \qquad \text{Community 1,} \\ \gamma\big(\overline{x}_i(k)+\overline{y}_i(k)\big) + \frac{1}{8}a_i\big(\overline{x}_i(k)+\overline{y}_i(k)-T_2\big)^2 + \\ \frac{5}{4}b_i\big(\overline{x}_i(k)+\overline{y}_i(k)-T_2\big)^4 + \frac{1}{2}\big(\overline{x}_i(k)-\frac{1}{2}T_2\big)^2 + \\ \frac{1}{2}\big(\overline{y}_i(k)-\frac{1}{2}T_2\big)^2 \qquad \text{Community 2.} \end{cases}$$

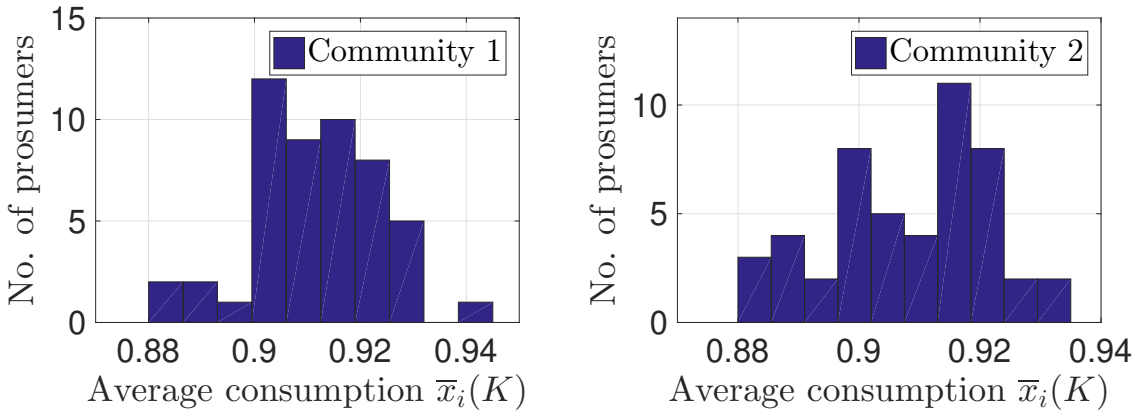
Now, we present the numerical results and show the convergence of the long-term average of prosumption by each prosumer in Figure 5.4. Figure 5.4(a) shows the time-averaged consumption and Figure 5.4(b) shows the time-averaged production over 200 days. In the context of car-sharing, these are the average of cars used by a prosumer and the average of cars shared with another person by the same prosumer over 200 days, respectively.



(a) (b)

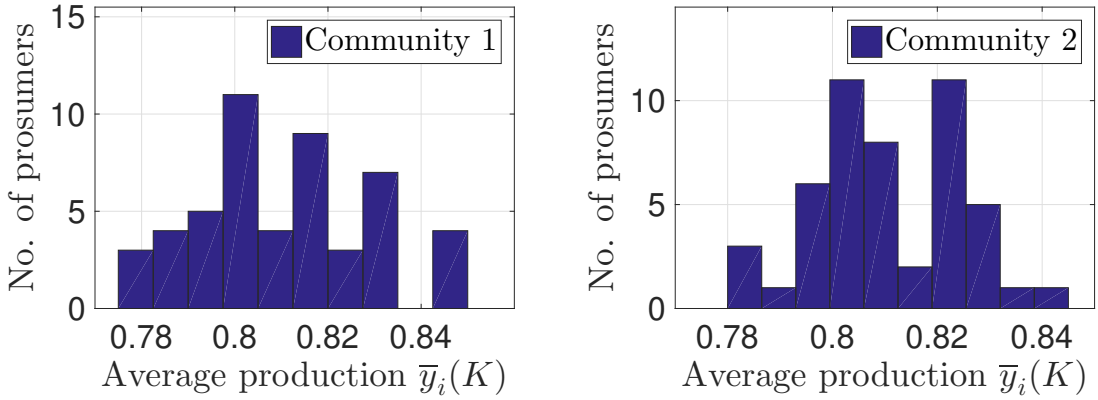
Figure 5.4: (a) Evolution of time-averaged consumption of the resource, and (b) evolution of time-averaged production of the resource.

Figures 5.5 and 5.6 illustrate the average prosumptions (consumption and production), respectively, on the 200^{th} day by every prosumer of a particular community.



(a) (b)

Figure 5.5: At time index $K = 200$—(a) average consumption $\overline{x}_i(K)$ by prosumers of Community 1, (b) average consumption $\overline{x}_i(K)$ by prosumers of Community 2.



(a) (b)

Figure 5.6: At time index $K = 200$—(a) average production $\overline{y}_i(K)$ by prosumers of Community 1, and (b) average production $\overline{y}_i(K)$ by prosumers of Community 2.

Furthermore, the absolute difference between the desired value of utilization T_i of cars and the actual utilization $\overline{x}_i(k) + \overline{y}_i(k)$ by prosumer i for a certain period is shown in Figures 5.7 and 5.8. Figure 5.7 illustrates the evolution of the absolute difference between T_i and $\overline{x}_i(k) + \overline{y}_i(k)$ for individual prosumers. Here, we observe that gradually the difference comes closer to zero. Additionally, in Figure 5.8(a) and 5.8(b), we observe that the absolute difference between the quantities is close to zero for most of the prosumers.

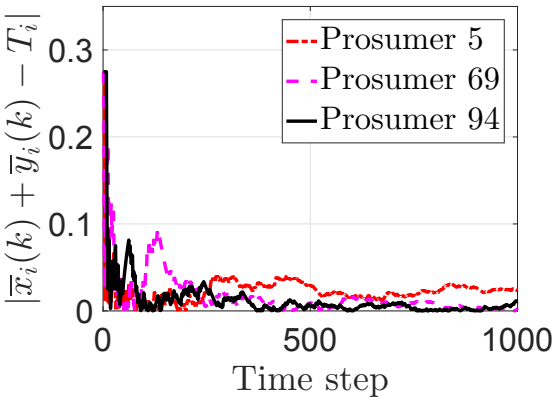
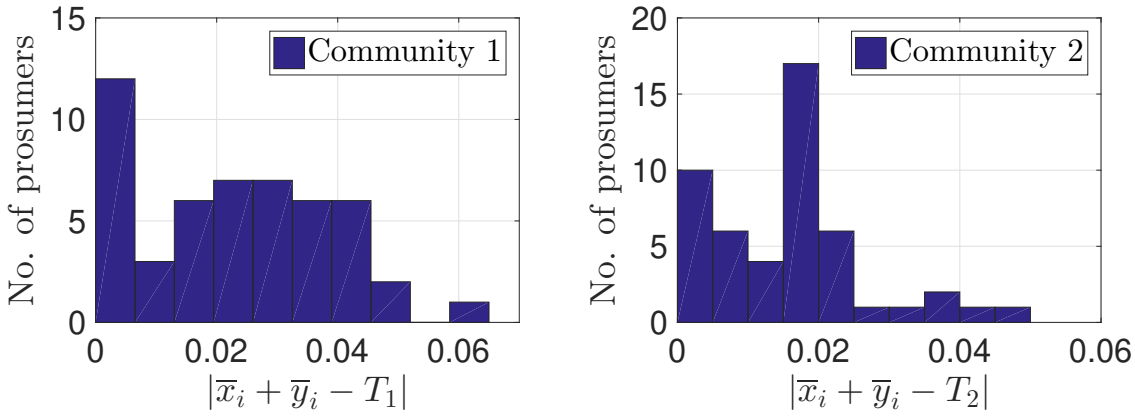


Figure 5.7: Evolution of absolute difference between desired value of utilization T_i and actual utilization of the resource $\overline{x}_i(k) + \overline{y}_i(k)$ of individual prosumers.

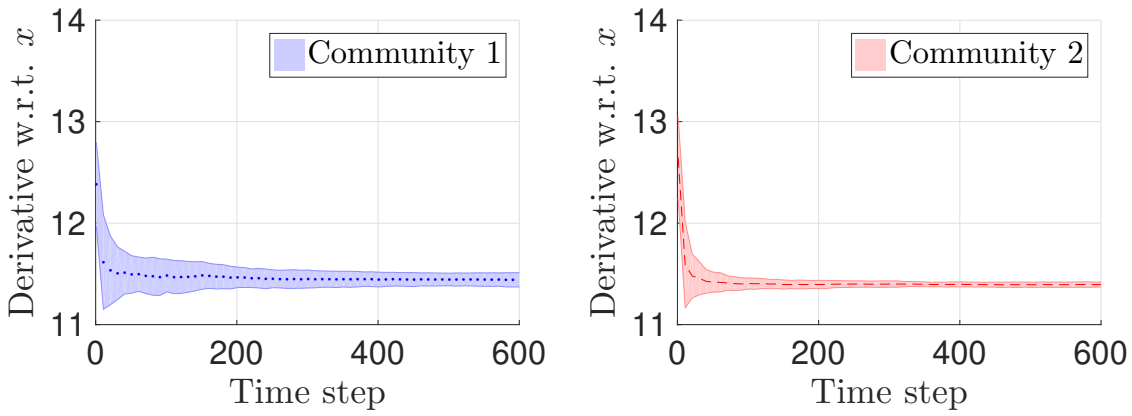


(a) (b)

Figure 5.8: (a) Absolute difference between desired value of utilization and actual utilization $|\overline{x}_i(K) + \overline{y}_i(K) - T_1|$ of prosumers of Community 1, here $T_1 = 1.74$, and (b) absolute difference between desired value of utilization and actual utilization $|\overline{x}_i(K) + \overline{y}_i(K) - T_2|$ of prosumers of Community 2, here $T_2 = 1.725$ and time index $K = 200$.

Now, we analyze the derivatives of the cost functions and see whether they gather close to each other to make consensus over time or not. Recall that the derivative of the cost function with respect to consumption is $\frac{\partial}{\partial_{x_i}} g_i(\cdot)$ and with respect to production is $\frac{\partial}{\partial_{y_i}} g_i(\cdot)$, that are shown in Figure 5.9 for a single simulation. We plot the shaded error-bars as depicted in Figure 5.9(a) and 5.9(b). It is observed that in both the cases the derivatives gather close to each other over time. Thus, the derivatives make consensus asymptotically with respect to the respective prosumer

community, which is a necessary and sufficient condition for optimality as described in Subsection 5.4.1. Notice that because both the derivatives (with respect to consumption and production) are the same, therefore, we illustrate here just one of them. We clarify here that because of the chosen initial values, the probability $\sigma_{i,x}(k)$ may overshoot at the start of the algorithm; to keep it in the valid range, we use $\min\left\{1, \Omega_x(k)\frac{\frac{\partial}{\partial x}\Big|_{x=\overline{x}_i(k)} g_i(\overline{x}_i(k),\overline{y}_i(k))}{\overline{x}_i(k)}\right\}$. Similar step is used to keep $\sigma_{i,y}(k)$ in the valid probability range.



(a) (b)

Figure 5.9: (a) Evolution of derivatives of $g_i(\cdot)$ w. r. t. x for prosumers of Community 1, and (b) evolution of derivatives of $g_i(\cdot)$ w. r. t. x for prosumers of Community 2.

Now, we analyze the aggregate consumption $\sum_{i=1}^{N} x_i(k)$ and production $\sum_{i=1}^{N} y_i(k)$ by the prosumer communities. The aggregate consumption $\sum_{i=1}^{N} x_i(k)$ is presented in Figure 5.10(a) for last 40 time instants (days), and similarly, the aggregate production by the communities $\sum_{i=1}^{N} y_i(k)$ is shown in Figure 5.10(b). Notice that the aggregate prosumption is close to the respective capacity constraints C_x and C_y; overshoots and undershoots are due to the assumption of soft constraints, as described previously. Additionally, Figure 5.10(c) shows the time-averaged consumption $\sum_{i=1}^{N} \overline{x}_i(k)$ by all prosumers in the prosumer market until 1000 time instants, and similarly, the time-averaged production $\sum_{i=1}^{N} \overline{y}_i(k)$ by all prosumers in the prosumer market for the same period, these averages are approximately equal to the respective capacities, satisfying the capacity constraints of Problem 5.3.2.

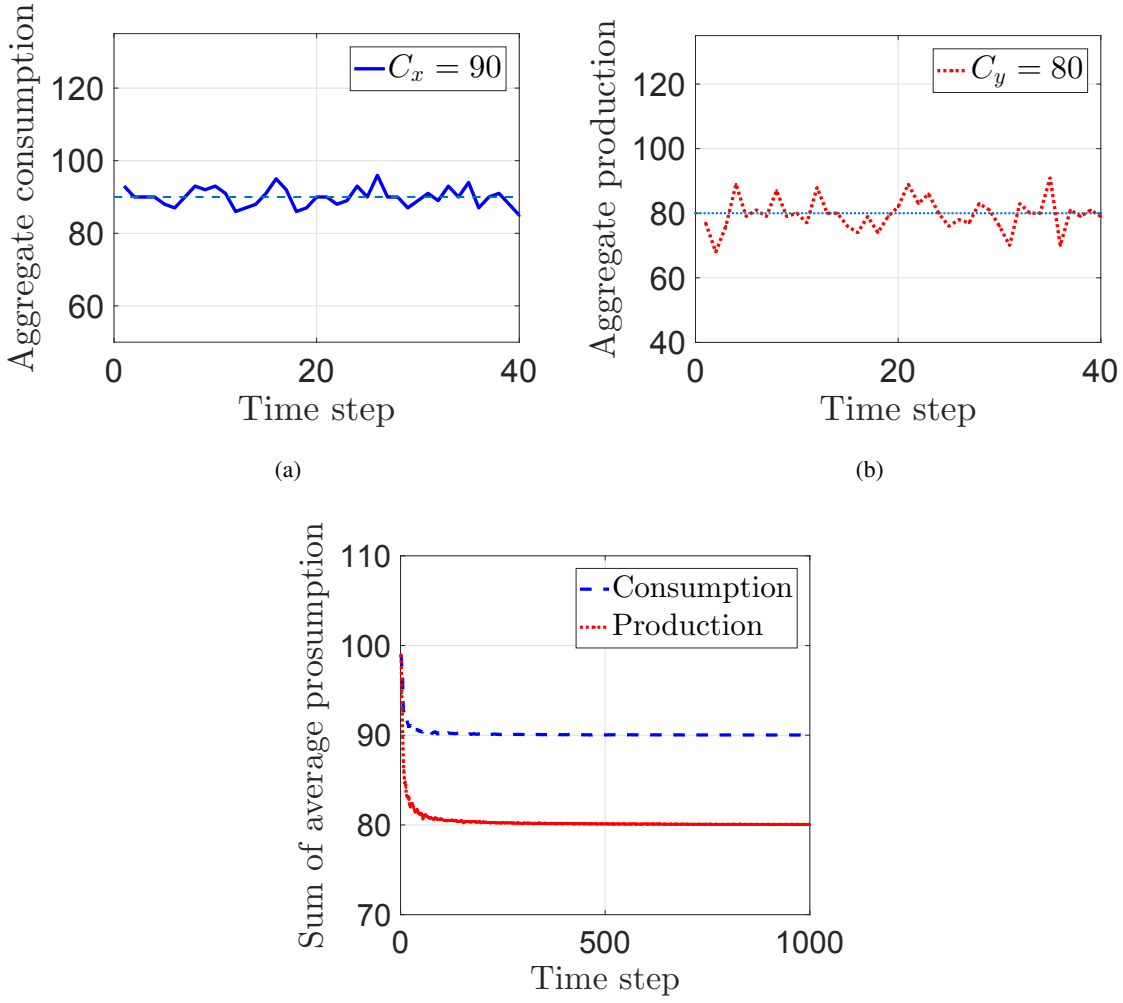
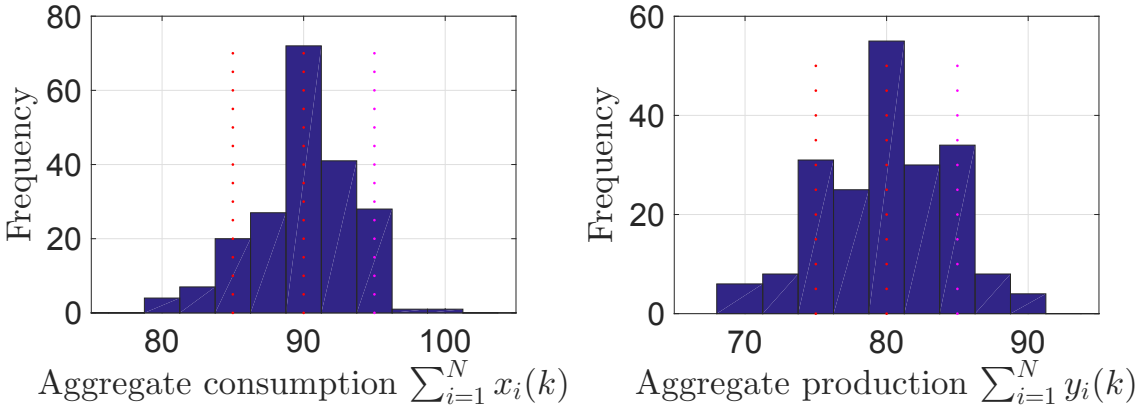


Figure 5.10: Aggregate prosumption for last 40 time instants—(a) aggregate consumption $\sum_{i=1}^{N} x_i(k)$, (b) aggregate production $\sum_{i=1}^{N} y_i(k)$, and (c) evolution of sum of time-averaged prosumption.

In addition to the above results, we observe in Figure 5.11(a) and 5.11(b) that most of the time the aggregate consumption $\sum_{i=1}^{N} x_i(k)$ and the aggregate production $\sum_{i=1}^{N} y_i(k)$ are close to their respective capacities C_x and C_y. Furthermore, the evolution of feedback signals Ω_x and Ω_y are shown in Figure 5.12.



(a) (b)

Figure 5.11: Frequency of prosumption—(a) frequency of aggregate consumption, (b) frequency of aggregate production for last 200 time instants.

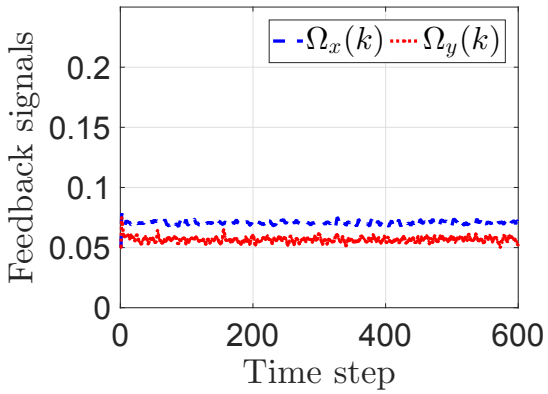


Figure 5.12: Evolution of feedback signals $\Omega_x(k)$ and $\Omega_y(k)$.

5.7 Conclusion

We developed distributed control algorithms to solve regulation problems with optimality constraints for the community-based prosumer market. The algorithm is based on ideas from stochastic approximation but formulated in a control-theoretic setting. The algorithm reaches optimality asymptotically, while simultaneously regulating instantaneous capacity constraints. To do so, the algorithm does not require communication between prosumers but little communication with the

sharing platform. Additionally, the algorithm is light and is suitable to implement in an Internet-of-Things (IoT) context with minimal demands on infrastructure. Two applications are described, and numerical results are presented to demonstrate the efficacy of the algorithms. Future work will explore the theoretical aspects of the algorithm (convergence properties), new applications and use cases, and the development of policies to reach more complicated equilibria.

Chapter 6

Conclusion and Future Directions

6.1 Conclusion

In this thesis, we studied the distributed multi-resource allocation problems in which several agents in a network collaborate to access multiple shared resources. The agents can be Internet-of-things devices such as fitness trackers, autonomous cars, mobile devices, surveillance cameras, wearable devices, etcetera. The agents may not wish to exchange allocation history or gradient of their cost functions with other agents; however, they wish to achieve social-optimum cost over the network. Such resource allocation problems are ubiquitous in smart cities, smart grids, sharing economy, cloud computing, edge-computing, etcetera. Wherein agents in a network need multiple-divisible and multiple-indivisible shared resources to complete their tasks and wish to achieve social-optimum values. Such distributed resource allocation problems are challenging to solve, particularly when the agents are constrained through communications, computational capabilities or do not wish to communicate with other agents in the network due to privacy reasons. Furthermore, when the cost functions of agents are non-separable and are coupled through the allocation of multiple resources, in such cases, the single resource allocation algorithms are not efficient and provide suboptimal solutions.

We developed several distributed iterative algorithms to solve such resource allocation problems for multiple-divisible and multiple-indivisible resources. The resource allocation problems are modeled as distributed optimization problems in which several agents in a network collaborate to

access the shared resources and minimize the sum of the total cost. In the algorithms, we consider a central server that keeps track of aggregate resource demands and sends feedback signals to agents in the network. There is no inter-agent communication required in the algorithms but little with the central server, which significantly reduces the communication overhead of the models. Additionally, the agents do not share their allocation history and are unaware of the total capacity and the number of agents in the network. Furthermore, an agent demands the resources based on its local computation using its cost function, resource allocations, and other parameters. In recent literature, such settings have attracted significant attention from the research community as federated optimization in which the agents do not need to share their local data or information, and a global model is trained; notably, from the machine learning community, as in federated learning. Federated learning is a distributed machine learning model in which several clients (agents) collaborate to train a global model without sharing their local on-device data. Because our models consider a central server that keeps track of aggregate demands and sends feedback signals in the network, the system may fail if the central server stops working. However, we can fix this by adding a backup server that keeps all the information as the central server and receives feedback signals. When the central server stops sending the feedback signals, the backup server takes charge and works as the central server until the central server comes live.

The first algorithm we developed is a stochastic distributed algorithm for multiple divisible resources, described in Chapter 2. It is a generalization of the additive-increase multiplicative-decrease (AIMD) algorithm for a single resource allocation. We assume that each agent has a private cost function coupled with multiple divisible resources; these cost functions are strictly convex, twice continuously differentiable, and increasing in each variable. Also, the agents do not share their allocation history and are unaware of the total capacity and the number of agents in the network. In our model, we consider a central server that keeps track of aggregate resource demands. The central server broadcasts a one-bit feedback signal (called a *capacity event*) in the network when the aggregate demand reaches its capacity. Thus, the algorithm incurs little communication overhead. We proposed AIMD matrices for multiple resources and modeled the system as a non-homogeneous Markov chain with place-dependent probabilities. We showed that the accumulative

average (long-term average) allocations of resources converge to the optimal values. Additionally, we presented numerical results to check the efficacy of our algorithm.

The second algorithm we developed is a distributed derandomized AIMD algorithm for multiple divisible shared resource allocation, described in Chapter 3. The algorithm is a deterministic version of the stochastic additive-increase and multiplicative-decrease (AIMD) algorithm. We consider a central server in the system that keeps track of aggregate demand by agents in the network and broadcasts a one-bit feedback signal when the aggregate demand reaches the capacity of the resource. Furthermore, we assume that each agent has private cost functions coupled through its allocation of multiple resources and are strictly convex, twice continuously differentiable, and increasing in each variable. Also, the developed solution does not require inter-agent communication. Thus, the algorithm incurs little communication overhead. Moreover, we showed empirically that the long-term average allocations of multiple shared resources converge to optimal allocations, and the system achieves minimum social cost. Additionally, we showed empirically that the developed derandomized AIMD algorithm converges faster than the stochastic AIMD algorithm, and both approaches provide approximately the same solutions.

The third algorithm we developed is a distributed stochastic algorithm for multiple indivisible (unit-demand) resource allocations, described in Chapter 4. Indivisible resources can be allocated one unit or zero units. We assume that each agent has private cost functions coupled through its allocations of multiple indivisible resources and are strictly convex, twice continuously differentiable, and increasing in each variable. Additionally, we consider a central server in the system that keeps track of aggregate demand by agents in the network and broadcasts an error signal (referred to as *price signal*) at each time step. The agents calculate their resource demands probabilistically based on cost functions, average allocations, and price signals. Each agent's resource demand is modeled as a Bernoulli random variable, and no inter-agent communication is required. We provided results on convergence for the single and multiple resource allocation cases using ideas from stochastic approximation techniques. Furthermore, we presented an example of allocating chargers to electric vehicles, illustrating the algorithm's performance.

Finally, in Chapter 5, we developed a distributed stochastic algorithm for applications in sharing economy settings where prosumer communities buy and sell resources through contracts to one or

more external entities and ensure that they meet particular demands in real-time. The algorithm solves regulation problems with optimality constraints for the community-based prosumer market. Recall that prosumers are agents that both produce and consume a resource. Furthermore, in a prosumer market, a community of prosumers sells excess resources to another community. In the algorithm, each prosumer has a cost function coupled through the time-averaged production and consumption of the resource. Each prosumer runs its distributed algorithm and takes binary decisions whether to produce a resource or not and to consume the resource in a probabilistic way. The prosumers do not communicate with each other; however, a little with the infrastructure (sharing platform), so the communication graph is unknown a-priori. The sharing platform keeps track of the aggregate consumption and production of the resource by all prosumers and sends signals to prosumers in the community to calculate their demands for the next step. Following the algorithm, the prosumers achieve the social-optimal cost in long-term average prosumption. Finally, we describe two use cases, community-based car-sharing and collaborative energy storage for prosumer markets. We also present numerical results to check the efficacy of the algorithms.

6.2 Future directions

In future work, we can find the theoretical bounds for the rate of convergence of the divisible and indivisible resource allocation algorithms. Additionally, the models can be used in several application areas such as cloud computing, federated learning, edge computing, smart grids, energy trading, sharing economy, or wireless sensor networks—wherein sensors have the minimal processing power and battery life.

It is also interesting to deploy the algorithms in real applications using Internet-of-Things devices and edge computing and analyze the system's performance. For the de-randomized AIMD algorithm of Chapter 3, it is an open problem to prove the convergence of average allocations. It is also interesting to study the stability analysis of the AIMD based algorithms.

Additionally, it is interesting to investigate the fairness properties such as envy-freeness, max-imin, proportionality for the divisible and indivisible resource allocation problems where agents do not communicate with each other and keep their preferences private. It is also interesting to explore

the cases where the cost function of an agent is coupled with the allocation of mixed resources—both divisible and indivisible. Furthermore, for the indivisible case in Chapter 4, it is an open problem to prove the convergence results wherein $\Omega(k+1)$ is updated with a constant step size τ, and average allocation is formulated with a decreasing step size using ideas from stochastic approximation.

Furthermore, it will be interesting to propose privacy algorithms based on our developed algorithms to provide certain privacy guarantees to agents in the network. An analyst can learn statistics on the population of agents; however, it should not be able to infer an agent's resource demand or its cost function, as in differential privacy models.

Finally, it will be interesting to apply our algorithms in federated learning. For example, the indivisible allocation algorithms of Chapter 4 can be used for client selection in federated learning.

Appendix A

An Overview of Notations

A.1 Basic notations

Symbol	Description
$\mathbb{N}$	The set of natural numbers.
$\mathbb{R}$	The set of real numbers.
$\mathbb{R}_+$	The set of non-negative real numbers.
n	The number of agents in a network.
$\mathbb{R}_+^n$	The set of vectors in $\mathbb{R}^n$ with non-negative values.
m	The number of resources in the network.
i	Index to denote an agent, $i = 1, 2, \ldots, n$.
j	Index to denote a resource, $j = 1, 2, \ldots, m$.
$k \in \mathbb{N}$	Discrete time steps.
$x_i(k)$	The instantaneous resource demand of agent i at time step k.
$\overline{x}_i(k)$	The average allocation of agent i until time step k; defined as $\overline{x}_i(k) \triangleq \frac{1}{k+1}\sum_{\ell=0}^{k} x_i(\ell)$.
$\mathcal{C}$	The capacity of a resource.
$f_i : \mathbb{R}_+^m \to \mathbb{R}_+$	The cost function of agent i coupled through allocation of m resources.

A.2 Notations used in Chapter 2

Symbol	Description
$\nu \in \mathbb{N}$	The time steps.
t_ν	The discrete time instants, $\nu \in \mathbb{N}$.
$k \in \mathbb{N}$	Index to denote the capacity events.
$x_i(t_\nu) = x_i(\nu)$	The resource demand of agent i at time instant t_ν, $\nu \in \mathbb{N}$.
$\overline{x}_i(t_k) = \overline{x}_i(k)$	The average resource allocation until the k'th capacity event, defined in (2.1).
$\mathcal{C}$	The capacity constraint of a single resource.
$f_i : \mathbb{R}_+^m \to \mathbb{R}_+$	The cost function of agent i.
$\mathbf{x} \in [0, \mathcal{C}]^n$	The vector $(x_1, \ldots, x_n)$.
$\mathbf{x}^* \in [0, \mathcal{C}]^n$	The unique optimal point $(x_1^*, \ldots, x_n^*)$ of optimization problem (2.2).
$\mathcal{C}_1, \mathcal{C}_2$	The capacities of resources 1 and 2, respectively.
$x_{ji}(k_j)$	The demand of resource j by agent i at time instant t_{k_j}.
$\overline{x}_{ji}(k_j)$	The average allocation of resource j to agent i until the k_j'th capacity event, defined in (2.1).
$\mathbf{x}_1 \in [0, \mathcal{C}_1]^n$	The vector $(x_{11}, \ldots, x_{1n})$.
$\mathbf{x}_2 \in [0, \mathcal{C}_2]^n$	The vector $(x_{21}, \ldots, x_{2n})$.
$\mathbf{x}^* \in (\mathbb{R}_+^n)^m$	A unique optimal point of optimization problem (2.3), $\mathbf{x}^* = (x_{11}^*, \ldots, x_{mn}^*)$.
α	Additive-increase factor for a single resource.
β	Multiplicative-decrease factor for a single resource.
α_j	Additive-increase factor for resource j.
β_j	Multiplicative-decrease factor for resource j.

Symbol	Description
$\lambda_{ji}(k_j)$	Response probability of agent i at the k_j'th capacity event for resource j. Defined in (2.17) and (2.18), for $j = 1, 2$.
Γ_j	The normalization factor.
$H(\mathbf{x}, \mu)$	Lagrangian of optimization problem (2.2), defined in (2.6).
$\mu \in \mathbb{R}$	Lagrange multipliers of optimization problem (2.2), refer (2.6).
$H(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r})$	Lagrangian of optimization problem (2.3), defined in (2.13).
$\boldsymbol{\mu} \in \mathbb{R}^2$	The Lagrange multipliers (μ_1, μ_2) of (2.3), refer (2.13).
$\mathbf{s} \in \mathbb{R}^n$	The Lagrange multipliers $(s_1, s_2, \ldots, s_n)$ of (2.3), refer (2.13).
$\mathbf{r} \in \mathbb{R}^n$	The Lagrange multipliers $(r_1, r_2, \ldots, r_n)$ of (2.3), refer (2.13).
$\mathbf{x}^\top$	The transpose of vector $\mathbf{x} \in \mathbb{R}^n$.
Σ	The standard simplex in $\mathbb{R}^n$, defined by $\Sigma \triangleq \{\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n_+ \mid \sum_{i=1}^n x_i = 1\}$.
Σ^m	m product space of the simplex Σ.
$\mathbf{x} \in \Sigma^m$	The vector $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$.
$\overline{\mathbf{x}} \in \Sigma^m$	The vector $(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_m)$.
$[\mathbf{x}]_i$	The allocation vector $(x_{1i}, x_{2i}, \ldots, x_{mi})$ of agents i, for $i = 1, 2, \ldots, n$, and m resources.
$\mathbf{e} \in \mathbb{R}^n$	The vector of ones, $\mathbf{e} = [1 \quad 1 \quad \ldots \quad 1]^\top \in \mathbb{R}^n$.
$\mathbf{e}_\ell \in \mathbb{R}^n$	The ℓ'th standard basis vector; e.g., $\mathbf{e}_1 = [1 \quad 0 \quad \ldots \quad 0]^\top \in \mathbb{R}^n$.

Symbol	Description
$\odot$	The Hadamard product (or componentwise product), defined in (2.19).
$\operatorname{diag}(X, Y)$	The diagonal matrix $\begin{bmatrix} X & 0; & 0 & Y \end{bmatrix}$, for $X, Y \in \mathbb{R}^{n\times n}$.
$\mathbf{x} \gg \mathbf{y}$, for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$	$x_i > y_i$, for $i = 1, \ldots, n$.
$\operatorname{ri} \Sigma$	Relative interior of the simplex Σ, defined by $\{\mathbf{x} \in \Sigma : \mathbf{x} \gg 0\}$.
$\operatorname{conv} X$	The convex hull of a set $X \subset \mathbb{R}^n$, defined as the smallest convex set containing X.
$\|\mathbf{x}\|_1$, $\mathbf{x} \in \mathbb{R}^n$	The 1-norm on $\mathbb{R}^n$, defined by $\|\mathbf{x}\|_1 \triangleq \sum_{i=1}^n \|x_i\|$.
d_H	Hilbert metric on $\operatorname{ri} \Sigma$, defined in (2.22).
$\exp(d_H(\mathbf{x}, \mathbf{y}))$	Exponential form of Hilbert metric, defined in (2.23).
$d_1(v, X)$	The distance of a point v to the set X in $\mathbb{R}^n$, with respect to 1-norm, defined in (2.20).
$\boldsymbol{\beta}_i(k) \in \{\beta, 1\}$	The stochastic multiplicative decrease factors of agent i, if agent i responds to the k'th capacity event then $\boldsymbol{\beta}_i(k) = \beta$, if it does not respond then $\boldsymbol{\beta}_i(k) = 1$, for $i = 1, 2, \ldots, n$.
$\boldsymbol{\beta}(k)$	The vector of stochastic multiplicative decrease factors, $\boldsymbol{\beta}(k) \triangleq \begin{bmatrix} \boldsymbol{\beta}_1(k) & \ldots & \boldsymbol{\beta}_n(k) \end{bmatrix}^\top$.
I	The $n \times n$ identity matrix.
I^a	An $a \times a$ identity matrix, $a \in \mathbb{N}$.
$t_{k+1} - t_k$	The time difference between the k'th and the $k+1$'th capacity events, defined in (2.25).
$\mathbf{A}(k)$	The stochastic AIMD matrix for a single resource, defined in (2.28).

Symbol	Description
A_q, $1 \leq q \leq 2^n$	An AIMD matrix with n agents in a network sharing a single resource.
$\mathcal{A} = \{A_1, \ldots, A_{2^n}\}$	The set of all AIMD matrices with n agents in a network sharing a single resource.
$\mathbb{E}[\mathbf{A}(k)]$	The expectation of $\mathbf{A}(k)$.
$\mathbf{w}_p$	The right Perron eigen vector of $\mathbb{E}[\mathbf{A}(k)]$, defined in (2.31).
k_j	A counter for the number of capacity events associated with resource j.
K	The ordered set of all capacity events for all the resources.
K_j	The subset of K associated with resource j.
ϕ	The index map, defined in (2.32).
$\mathcal{A}_j$	The set of the AIMD matrices associated with resource j.
$A_{j,q}$	The q'th matrix in the set $\mathcal{A}_j$.
$\mathbf{A}(k)$	The stochastic AIMD matrix for two resources, $k \in K$, defined in (2.37).
$\mathbf{w}_{p_j}$	The right Perron eigen vector associated with resource j, defined in (2.40).
$\overline{\mathbf{S}}_j(k_j)$	The matrix defined in (2.41).
$\overline{\mathbf{S}}(k)$	The diagonal matrix $\operatorname{diag}(\overline{\mathbf{S}}_1(k_1), \overline{\mathbf{S}}_2(k_2))$, defined in (2.42).
$\|X\|_1$, for $X \in \mathbb{R}^{n \times n}$	The max column sum norm, defined in (2.44)
$\mathbb{P}_{\boldsymbol{\lambda}}$	The probability measure with fixed probability $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{y})$, for $\mathbf{y} \in \Sigma^m$.
$Q : \Sigma^m \to \mathbb{R}$	A strictly convex function, see Assumption 2.3.2.
$R : \Sigma^m \to ([0,1]^n)^m$	The map is the gradient of the strictly convex function Q, see Assumption 2.3.2.
$\boldsymbol{\xi}(k)$	The state vector of the resources, defined in (2.49).

Symbol	Description
$\mathbf{V}(k) \in \mathbb{R}^{4n\times 4n}$	The diagonal matrices of the stochastic AIMD matrices, defined in (2.52).
$V_k \in \mathbb{R}^{4n\times 4n}$	The diagonal matrices of the AIMD matrices, defined in (2.53).
$\mathcal{G}(k)$	The set of all V_k matrices.
$\mathbf{x}^* \in \mathrm{ri}\Sigma^m$	The KKT point of Problem (2.3), refer Lemma 2.4.2.
$W \in \mathbb{N}$	The fixed time-window, see (2.55).
v	The length of the averaging period, see (2.55).
$\overline{\mathbf{x}}(W+\mathrm{v})$	Defined in (2.55), also see (2.60).
$\overline{\mathbf{x}}(W)$	The average allocation until W capacity events.
ϵ_{v}	Denotes $\frac{\mathrm{v}}{W+\mathrm{v}+1}$, refer (2.56).
$P_j : \Sigma^m \to \Sigma$	The expectation of the invariant measure with fixed probabilities $\boldsymbol{\lambda}_j(\mathbf{y})$, defined in (2.57).
$P(\mathbf{y})$	The vector $(P_1(\mathbf{y}), \ldots, P_m(\mathbf{y}))$, defined in (2.58).
$\Delta_j \in \mathbb{R}^n$	The perturbation term associated with $\mathbf{x}_j$.
$\Delta \in (\mathbb{R}^n)^m$	The vector of perturbation terms $(\Delta_1, \ldots, \Delta_m)$, defined in (2.59).
$\delta^- > 0$	The constant used in (2.63).
$\overline{B}_1(0,\delta)$	The closed 1-norm ball of radius δ and center 0.
$R_\epsilon : \Sigma^m \to \Sigma^m$	Defined as $R_\epsilon(\mathbf{x}) \triangleq (1-\epsilon)\mathbf{x} + \epsilon P(\mathbf{x}),$ for $0 \le \epsilon \le 1$, see (2.65).
$P_{\mathrm{co}}(\delta)$	Defined as $P_{\mathrm{co}}(\delta) \triangleq \mathrm{conv}\, P(\Sigma^m) + \overline{B}_1(0,\delta)$, refer (2.66).
δ^+	The constant defined as $\delta^+ \triangleq \max_{\mathbf{y}\in\Sigma^m}\{d_1(\mathbf{y}, P(\Sigma^m))\}$, refer (2.67).
$\{\epsilon_k\}_{k\in\mathbb{N}} \subset (0,1)$	A sequence in $(0,1)$.
$C_{\overline{\delta}}$	A constant, refer Lemma 2.4.3.

Symbol	Description
$B_H(\mathbf{x}, \delta)$	The closed ball with respect to Hilbert metric, centered at $\mathbf{x}$ with radius δ.
$\zeta_j > 0$	A constant, refer Lemma 2.4.4.
$C_{\zeta_j} > 0$	The constant defined in (2.71).
$C_\zeta > 0$	The constant defined in (2.73).
$\zeta > 0$	A constant, see Corollary 2.4.6.
$\varpi > 0$	The constant defined in (2.76).
δ^*	The constant defined as $\delta^* \triangleq \min\left\{\frac{C_\zeta \exp(\zeta)}{2\varpi}, \delta^-\right\}$, see (2.77).
$P_e(\mathbf{y})$	The diagonal matrix $\operatorname{diag}\left(P_1(\mathbf{y})\mathbf{e}^\top, P_2(\mathbf{y})\mathbf{e}^\top\right)$, defined in (2.82).
$\overline{\mathbf{S}}(m\mathrm{v})$	The diagonal matrix $\operatorname{diag}\left(\overline{\mathbf{S}}_1(\mathrm{v}), \overline{\mathbf{S}}_2(\mathrm{v}), \ldots, \overline{\mathbf{S}}_m(\mathrm{v})\right)$, cf. (2.42).
$\mathbf{M}(m\hat{\mathrm{v}})$	The random variable denotes $\left\|\overline{\mathbf{S}}(m\hat{\mathrm{v}}) - P_e(\mathbf{y})\right\|_1$, $\hat{\mathrm{v}} \in \mathbb{N}$, defined in (2.89).
σ^2	The variance of $\mathbf{M}(m\hat{\mathrm{v}})$.
$\tau(k)$	Defined as $\tau(k) \triangleq W + k\mathrm{v}$, refer (2.100).
ϵ_{k_j}	Defined as $\epsilon_{k_j} \triangleq \frac{\mathrm{v}}{W+(k_j+1)\mathrm{v}+1}$, refer (2.101).
ϵ_k	Defined as $\epsilon_k \triangleq \begin{bmatrix} \epsilon_{k_1} & \epsilon_{k_2} \end{bmatrix}^\top$, refer (2.102).
$\overline{\mathbf{x}}(\tau(k+1))$	Defined in (2.103).
σ_1	Defined as $\sigma_1 \triangleq \min\left\{k \geq k_0 \mid \overline{\mathbf{x}}(\tau(k)) \in P_{\mathrm{co}}(2\overline{\delta})\right\}$, $k_0 \in \mathbb{N}$, see (2.106).

A.3 Notations used in Chapter 3

Symbol	Description
k	Discrete time steps, $k \in \mathbb{N}$.
n	The number of IoT devices (agents).
i	The index for IoT devices, $i = 1, 2, \ldots, n$.
j	The index for resources, $j = 1, 2, \ldots, m$.
R^j	Resource j, for $j = 1, 2, \ldots, m$.
$\mathcal{C}^j$	The capacity of resource j, for $j = 1, 2, \ldots, m$.
$ICD\ i$	The IoT device i, for $i = 1, 2, \ldots, n$.
$f_i : \mathbb{R}_+^m \to \mathbb{R}_+$	The cost function of IoT device i, $i = 1, 2, \ldots, n$.
$x_i^j(k)$	The instantaneous demand of agent i for resource j at time step $k \in \mathbb{N}$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.
$\mathbf{x}^* \in (\mathbb{R}_+^n)^m$	The optimal point $(x_1^{*1}, \ldots, x_n^{*m})$ of the optimization Problem 3.2.2.
$\overline{x}_i^j(k)$	The average allocation of agent i for resource j at time step k, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, defined in (3.1).
$\overline{\mathbf{x}}(k)$	The vector $(\overline{x}_1^1(k), \ldots, \overline{x}_n^m(k))$.
$S^j(k) \in \{0, 1\}$	The capacity event signal for resource j, $S^j(k) = 1$ if capacity event occurs and $S^j(k) = 0$ if it does not occur, refer (3.3).
$\alpha^j \in (0, \mathcal{C}^j]$	Additive increase factor for resource j.
$\beta^j \in [0, 1)$	Multiplicative decrease factor for resource j.
$0 < \lambda_i^j(k) \leq 1$	The scaling factor of agent i for resource j at time step k, defined in (3.4).
Γ^j	The normalization factor, defined in (3.5).

A.4 Notations used in Chapter 4

Symbol	Description
k	Discrete time steps, $k \in \mathbb{N}$.
n	The number of agents in a network.
m	The number of unit-demand (indivisible) shared resources.
C	The capacity of a single resource.
$\mathcal{N}$	The set $\{1, 2, \ldots, n\}$.
$\mathcal{M}$	The set $\{1, 2, \ldots, m\}$.
$\xi_i(k) \in \{0, 1\}$	The demand by agent i for a single resource at time step k—an independent Bernoulli random variable.
$y_i(k) \in [0, 1]$	The average allocation of agent i of a single resource until time step k, defined as $y_i(k) \triangleq \frac{1}{k+1} \sum_{\ell=0}^{k} \xi_i(\ell)$.
$g_i : [0, 1] \to \mathbb{R}_+$	The cost function of agent i for a single resource.
$\boldsymbol{\xi}(k)$	The vector $(\xi_1(k), \ldots, \xi_n(k)) \in \{0, 1\}^n$, for $k \in \mathbb{N}$.
$\mathbf{y}(k)$	The vector $(y_1(k), \ldots, y_n(k)) \in [0, 1]^n$, for $k \in \mathbb{N}$.
$\Omega(k+1)$	The normalization factor or price signal of a single resource, $\Omega(k+1) \triangleq \Omega(k) - \tau\Big(\sum_{i=1}^{n} \xi_i(k) - C\Big)$, see (4.5).
$\sigma_i(\Omega(k), y_i(k))$	The probability distribution function, $\sigma_i(\Omega(k), y_i(k)) \triangleq \Omega(k) \frac{y_i(k)}{g_i'(y_i(k))}$, for $i \in \mathcal{N}$, see (4.7).
$\boldsymbol{\sigma}(\Omega, \mathbf{y}(k))$	The vector $(\sigma_1(\Omega, y_1(k)), \ldots, \sigma_n(\Omega, y_n(k)))$, for $k \in \mathbb{N}$.
$\mathbf{M}(k+1)$	Denotes $(\boldsymbol{\xi}(k+1) - \boldsymbol{\sigma}(\Omega, \mathbf{y}(k)))$; the martingale difference.
$a(k)$	A decreasing step-size, $a(k) = \frac{1}{k+1}$, for $k \in \mathbb{N}$.
$w(\mathbf{y}(k))$	Denotes $(\boldsymbol{\sigma}(\Omega, \mathbf{y}(k)) - \mathbf{y}(k))$, refer (4.12) and (4.13).
C_j	The capacity of resource j.

Symbol	Description
$\xi_{ji}(k) \in \{0,1\}$	The demand of agent i for resource j at time step k; an independent Bernoulli random variable.
$y_{ji}(k) \in [0,1]$	The average allocation of resource j to agent i until time step k, $y_{ji}(k) \triangleq \frac{1}{k+1}\sum_{\ell=0}^{k}\xi_{ji}(\ell)$.
$[\mathbf{y}]_i \in [0,1]^m$	The vector $(y_{1i}, y_{2i}, \ldots, y_{mi})$, for $i = 1, 2, \ldots, n$.
$\mathbf{y}_j \in [0,1]^n$	The vector $(y_{j1}, y_{j2}, \ldots, y_{jn})$, for $j = 1, 2, \ldots, m$.
$\mathbf{y} \in ([0,1]^n)^m$	The vector $(y_{11}, \ldots, y_{mn})$.
$g_i : [0,1]^m \to \mathbb{R}_+$	The cost function of agent i coupled through allocations of m resources.
$\tau_j \in (0,1)$	The gain parameter.
$\Omega_j(k+1)$	The normalization factor or price signal of resource j, defined in (4.16).
$\sigma_{ji}(\Omega_j(k), [\mathbf{y}]_i(k))$	The probability distribution function of agent i for resource j, defined in (4.17).
$\boldsymbol{\xi}_j(k) \in \{0,1\}^n$	The vector $(\xi_{j1}(k), \ldots, \xi_{jn}(k))$.
$\boldsymbol{\sigma}_j(\Omega_j(k), \mathbf{y}(k))$	The vector $(\sigma_{j1}(\Omega_j(k), [\mathbf{y}]_1(k)), \ldots, \sigma_{jn}(\Omega_j(k), [\mathbf{y}]_n(k)))$.
$\omega_j(\mathbf{y}_j(k))$	Denotes $(\boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k)) - \mathbf{y}_j(k))$.
$\{\mathbf{M}_j(k+1)\}$	Denotes $\{\boldsymbol{\xi}_j(k+1) - \boldsymbol{\sigma}_j(\Omega_j, \mathbf{y}(k))\}$, the martingale difference sequence of resource j.

A.5 Notations used in Chapter 5

Symbol	Description
$k \in \mathbb{N}$	Discrete time steps.
N	The number of agents.
C_x	Aggregate consumption bound or the total consumption capacity of the resource.
C_y	Aggregate production bound or the total production capacity of the resource.
$x_i(k)$	The amount of the resource consumed by agent i at time step $k \in \mathbb{N}$.
$y_i(k)$	The amount of the resource produced by agent i at time step $k \in \mathbb{N}$.
$\overline{x}_i(k)$	The time-averaged consumption of the resource, defined in (5.3).
$\overline{y}_i(k)$	The time-averaged production of the resource.
$T_i \in \mathbb{R}_+$	The desired value of utilization of the resource, for $i = 1, \ldots, N$.
$g_i : [0,1]^2 \to \mathbb{R}_+$	The cost function of agent i.
$(\mathbf{x}^*, \mathbf{y}^*) \in \mathbb{R}_+^{2N}$	The optimal point of optimization Problem (5.3.2).
$\Omega(k+1)$	The feedback signal for the consumption case, defined as $\Omega(k+1) \triangleq \Omega(k) - \tau\Big(\sum_{i=1}^{N} x_i(k) - C_x\Big)$, see (5.20).
$\sigma_i(\Omega(k), \overline{x}_i(k))$	The probability distribution for the consumption case, defined as $\sigma_i(\Omega(k), \overline{x}_i(k)) \triangleq \Omega(k)\frac{\overline{x}_i(k)}{g_i'(\overline{x}_i(k))}$, refer (5.21).
$\Omega_x(k)$	The feedback signal for consumption (for coupled prosumption case), defined in (5.22).
$\Omega_y(k)$	The feedback signal for production (for coupled prosumption case).
$\sigma_{i,x}(k)$	The probability distribution for consumption (for coupled prosumption case), defined in (5.23).
$\sigma_{i,y}(k)$	The probability distribution for production (for coupled prosumption case).

References

Agnew, S., & Dargusch, P. (2015). Effect of residential solar and storage on centralized electricity supply systems. *Nature Climate Change*, *5*(315).

Alam, S. E., Shorten, R., Wirth, F., & Yu, J. Y. (2018a, Sep.). Communication-efficient distributed multi-resource allocation. In *IEEE international smart cities conference (ISC2)* (pp. 1–8).

Alam, S. E., Shorten, R., Wirth, F., & Yu, J. Y. (2018b, Oct). Derandomized distributed multi-resource allocation with little communication overhead. In *Allerton conference on communication, control, and computing* (pp. 84–91).

Alam, S. E., Shorten, R., Wirth, F., & Yu, J. Y. (2020). Distributed algorithms for Internet-of-Things enabled prosumer markets: A control theoretic perspective. In E. C. et al. (Ed.), *Analytics for the sharing economy: Mathematics, engineering and business perspectives.* Springer.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, *17*(4), 2347–2376.

Angelakis, V., Avgouleas, I., Pappas, N., Fitzgerald, E., & Yuan, D. (2016, Oct). Allocation of heterogeneous resources of an IoT device to flexible services. *IEEE Internet of Things Journal*, *3*(5), 691–700.

Arnott, R., & Rowse, J. (1999). Modeling parking. *Journal of Urban Economics*, *45*(1), 97 – 124.

assoc. BEV, E. (2009, April). *Energy consumption, CO_2 emissions and other considerations related to battery electric vehicles.*

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of things: A survey. *Computer Networks*, *54*(15), 2787 – 2805.

Avrachenkov, K. E., Borkar, V. S., & Pattathil, S. (2017). Controlling G-AIMD by index policy. In *IEEE annual conference on decision and control, CDC* (pp. 120–125).

Aysal, T. C., Yildiz, M. E., Sarwate, A. D., & Scaglione, A. (2009). Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, *57*(7), 2748–2761.

Aziz, H., Caragiannis, I., Igarashi, A., & Walsh, T. (2022). Fair allocation of indivisible goods and chores. *Auton. Agents Multi Agent Syst.*, *36*(1), 3.

Aziz, H., & Rey, S. (2020). Almost group envy-free allocation of indivisible goods and chores. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20* (pp. 39–45).

Baklanov, A., Garimidi, P., Gkatzelis, V., & Schoepflin, D. (2021). Achieving proportionality up to the maximin item with indivisible goods. In *The conference on artificial intelligence, AAAI* (pp. 5143–5150).

Bei, X., Li, Z., Liu, J., Liu, S., & Lu, X. (2021). Fair division of mixed divisible and indivisible goods. *Artificial Intelligence*, *293*.

Benabbou, N., Chakraborty, M., Elkind, E., & Zick, Y. (2019). Fairness towards groups of agents in the allocation of indivisible items. In *Proceedings of the international joint conference on artificial intelligence, IJCAI* (pp. 95–101).

Benadè, G., Procaccia, A. D., & Qiao, M. (2019). Low-distortion social welfare functions. In *The AAAI conference on artificial intelligence, AAAI* (pp. 1788–1795).

Benjaafar, S., Kong, G., Li, X., & Courcoubetis, C. (2018). Peer-to-peer product sharing: Implications for ownership, usage, and social welfare in the sharing economy. *Management Science*, *65*(2).

Berahas, A. S., Bollapragada, R., Keskar, N. S., & Wei, E. (2019). Balancing communication and computation in distributed optimization. *IEEE Transactions on Automatic Control*, *64*(8), 3141–3155.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... Roselander, J. (2019). Towards federated learning at scale: System design. In *Proceedings of machine learning and systems* (Vol. 1, pp. 374–388).

Borkar, V. S. (2008). *Stochastic approximation*. Cambridge University Press.

Boutilier, C., Caragiannis, I., Haber, S., Lu, T., Procaccia, A. D., & Sheffet, O. (2012). Optimal social choice functions: A utilitarian view. In *Proceedings of the ACM conference on electronic commerce* (pp. 197–214).

Boutilier, C., Caragiannis, I., Haber, S., Lu, T., Procaccia, A. D., & Sheffet, O. (2015). Optimal social choice functions: A utilitarian view. *Artificial Intelligence*, *227*, 190–213.

Boyd, S., Ghosh, A., Prabhakar, B., & Shah, D. (2006). Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, *14*(SI), 2508–2530.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–122.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, *3*, 1–122.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York, NY, USA: Cambridge University Press.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (2016). *Handbook of computational social choice* (1st ed.). Cambridge University Press.

Cai, L., Shen, X., Pan, J., & Mark, J. W. (2005). Performance analysis of TCP-friendly AIMD algorithms for multimedia applications. *IEEE Transaction on Multimedia*, *7*(2), 339–355.

Carli, R., & Dotoli, M. (2020). Distributed alternating direction method of multipliers for linearly constrained optimization over a network. *IEEE Control Systems Letters*, *4*(1), 247–252.

Chakraborty, M., Igarashi, A., Suksompong, W., & Zick, Y. (2021, aug). Weighted envy-freeness in indivisible item allocation. *ACM Trans. Econ. Comput.*, *9*(3).

Chang, T. H., Nedic, A., & Scaglione, A. (2014). Distributed constrained optimization by consensus-based primal-dual perturbation method. *IEEE Transactions on Automatic Control*, *59*(6), 1524–1538.

Chen, T. D., & Kockelman, K. M. (2016). Carsharing's life-cycle impacts on energy use and greenhouse gas emissions. *Transportation Research Part D: Transport and Environment*, *47*, 276 – 284.

Chen, Z., Hu, W., Wang, J., Zhao, S., Amos, B., Wu, G., ... Satyanarayanan, M. (2017). An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance. In *Proceedings of the ACM/IEEE symposium on edge computing* (pp. 14:1–14:14).

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., ... Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*, *30*(1), 3–31.

Chiu, D., & Jain, R. (1989). Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, *17*(1), 1–14.

Cogill, R., Gallay, O., Griggs, W., Lee, C., Nabi, Z., Ordonez, R., ... Zhuk, S. (2014, Nov). Parked cars as a service delivery platform. In *International conference on connected vehicles and expo (ICCVE)* (pp. 138–143).

Columbus, L. (2017, Dec). 2017 roundup of Internet of things forecasts. *Forbes*.

Conitzer, V., Freeman, R., Shah, N., & Vaughan, J. W. (2019). Group fairness for the allocation of indivisible goods. In *The thirty-third AAAI conference on artificial intelligence, AAAI* (pp. 1853–1860).

Corless, M., King, C., Shorten, R., & Wirth, F. (2016). *AIMD dynamics and distributed resource allocation* (No. 29). Philadelphia, PA: SIAM.

Courcoubetis, C., & Weber, R. (2012). Economic issues in shared infrastructures. *IEEE/ACM Transactions on Networking*, *20*(2), 594–608.

Crisostomi, E., Liu, M., Raugi, M., & Shorten, R. (2014). Plug-and-play distributed algorithms for optimized power generation in a microgrid. *IEEE Transactions on Smart Grid*, *5*(4), 2145–2154.

Crisostomi, E., Shorten, R., Studli, S., & Wirth, F. (2017). *Electric and plug-in hybrid vehicle networks: Optimization and control*. CRC press (Taylor and Francis Group).

DeMaio, P. (2009). Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation*, *12*(4), 41–56.

de Souza Ribeiro, L. A., Saavedra, O. R., de Lima, S. L., & de Matos, J. G. (2011). Isolated micro-grids with renewable hybrid generation: The case of Lencois island. *IEEE Transactions on Sustainable Energy*, *2*(1), 1–11.

Dogan, E. (2021). Population monotonicity in fair division of multiple indivisible goods. *Int. J. Game Theory*, *50*(2), 361–376.

Duchi, J. C., Agarwal, A., & Wainwright, M. J. (2012, March). Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, *57*(3), 592–606.

Echeverria, S., Root, J., Bradshaw, B., & Lewis, G. (2014, Nov). On-demand VM provisioning for cloudlet-based cyber-foraging in resource-constrained environments. In *6th international conference on mobile computing, applications and services* (pp. 116–124).

Einav, L., Farronato, C., & Levin, J. (2016). Peer-to-peer markets. *Annual Review of Economics*, *8*, 615–635.

Elgabli, A., Park, J., Bedi, A. S., Issaid, C. B., Bennis, M., & Aggarwal, V. (2021). Q-GADMM: Quantized group ADMM for communication efficient decentralized machine learning. *IEEE Transactions on Communications*, *69*(1), 164–181.

Elkind, E., Faliszewski, P., & Slinko, A. (2009). On distance rationalizability of some voting rules. In *Proceedings of conference on theoretical aspects of rationality and knowledge* (pp. 108–117).

Endriss, U. (2014). Social choice theory as a foundation for multiagent systems. In J. P. Müller, M. Weyrich, & A. L. C. Bazzan (Eds.), *Multiagent system technologies* (pp. 1–6).

Energy, U. D. (2018, Feb.). *Emissions from hybrid and plug-in electric vehicles.*

EPA, U. (2017, July). *Fast facts: U.S. transportation sector GHG emissions 1990-2015* (No. EPA-420-F-17-013).

Fioravanti, A. R., Marecek, J., Shorten, R. N., Souza, M., & Wirth, F. R. (2017, Dec). On classical control and smart cities. In *IEEE annual conference on decision and control (CDC)* (pp. 1413–1420).

Fioravanti, A. R., Marecek, J., Shorten, R. N., Souza, M., & Wirth, F. R. (2019). On the ergodic control of ensembles. *Automatica*, *108*.

Fossati, F., Moretti, S., Perny, P., & Secci, S. (2020). Multi-resource allocation for network slicing. *IEEE/ACM Transactions on Networking*, *28*(3), 1311–1324.

Fraiberger, S., & Sundararajan, A. (2015). Peer-to-peer rental markets in the sharing economy.

SSRN Electronic Journal.

Freeman, R., Sikdar, S., Vaish, R., & Xia, L. (2019). Equitable allocations of indivisible goods. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19* (pp. 280–286).

Freeman, R., Zahedi, S. M., & Conitzer, V. (2017). Fair and efficient social choice in dynamic settings. In *Proceedings of the international joint conference on artificial intelligence, IJCAI* (pp. 4580–4587).

Georgiadis, L., Iosifidis, G., & Tassiulas, L. (2020). On the efficiency of sharing economy networks. *IEEE Transactions on Network Science and Engineering*, *7*(3), 1094–1110.

Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., & Stoica, I. (2011). Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the USENIX conference on networked systems design and implementation* (pp. 323–336).

Gkatzikis, L., Iosifidis, G., Koutsopoulos, I., & Tassiulas, L. (2014). Collaborative placement and sharing of storage resources in the smart grid. In *International conference on smart grid communications* (pp. 103–108).

Goudin, P. (2016). *The cost of non-Europe in the sharing economy: Economic, social and legal challenges and opportunities* (Tech. Rep.). European Parliamentary Research Service.

Griggs, W. M., Yu, J. Y., Wirth, F. R., Hausler, F., & Shorten, R. (2016). On the design of campus parking systems with QoS guarantees. *IEEE Trans. Intelligent Transportation Systems*, *17*(5), 1428–1437.

Grijalva, S., Costley, M., & Ainsworth, N. (2011). Prosumer-based control architecture for the future electricity grid. In *IEEE international conference on control applications* (pp. 43–48).

Gündoğan, A., Gürsu, H. M., Pauli, V., & Kellerer, W. (2020). Distributed resource allocation with multi-agent deep reinforcement learning for 5G-V2V communication. In *Proceedings of the international symposium on theory, algorithmic foundations, and protocol design for mobile networks and mobile computing* (pp. 357–362).

Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P., & Satyanarayanan, M. (2014). Towards wearable cognitive assistance. In *Proceedings of the annual international conference on mobile systems, applications, and services* (pp. 68–81).

Halpern, D., & Shah, N. (2021). Fair and efficient resource allocation with partial information. In *Proceedings of the thirtieth international joint conference on artificial intelligence, IJCAI* (pp. 224–230).

Hamari, J., Sjoklint, M., & Ukkonen, A. (2016). The sharing economy: Why people participate in collaborative consumption. *Journal of the Association for Information Science and Technology*, *67*(9), 2047–2059.

Han, D., Liu, K., Sandberg, H., Chai, S., & Xia, Y. (2021). Privacy-preserving dual averaging with arbitrary initial conditions for distributed optimization. *IEEE Transactions on Automatic Control*.

Han, S., Topcu, U., & Pappas, G. J. (2017). Differentially private distributed constrained optimization. *IEEE Transactions on Automatic Control*, *62*(1), 50–64.

Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., & Williams, P. (2010, July). Foundations for smarter cities. *IBM Journal of Research and Development*, *54*(4), 1–16.

Hartfiel, D. J. (2002). *Nonhomogeneous matrix products*. Singapore: World Scientific.

Hasan, M. H., Hentenryck, P. V., Budak, C., Chen, J., & Chaudhry, C. (2018). Community-based trip sharing for urban commuting. In *AAAI conference on artificial intelligence* (pp. 6589–6597).

Hernandez-Munoz, J. M., Vercher, J. B., Munoz, L., Galache, J. A., Presser, M., Gomez, L. A. H., & Pettersson, J. (2011). The future Internet. In *Smart cities at the forefront of the future Internet* (pp. 447–462).

Huckle, S., Bhattacharya, R., White, M., & Beloff, N. (2016). Internet of things, blockchain and shared economy applications. *Procedia Computer Science*, *98*, 461–466.

Huo, X., & Liu, M. (2022). Privacy-preserving distributed multi-agent cooperative optimization—paradigm design and privacy analysis. *IEEE Control Systems Letters*, *6*, 824–829.

Inderberg, T. J., Tews, K., & Turner, B. (2018). Is there a prosumer pathway? exploring household solar energy development in Germany, Norway, and the United Kingdom. *Energy Research and Social Science*, *42*, 258 – 269.

Iosifidis, G., & Tassiulas, L. (2017). Dynamic policies for cooperative networked systems. In

Workshop on the economics of networks, systems and computation (pp. 1–6).

Jacobson, V. (1988, Aug.). Congestion avoidance and control. *SIGCOMM Comput. Commun. Rev.*, *18*(4), 314–329.

Joe-Wong, C., Sen, S., Lan, T., & Chiang, M. (2013). Multiresource allocation: Fairness–efficiency tradeoffs in a unifying framework. *IEEE/ACM Transactions on Networking*, *21*(6), 1785–1798.

Jones, I. (2014). *Road space allocation: The intersection of transport planning, governance and infrastructure* (PhD.).

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, *14*(1–2), 1–210.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T. (2020). SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the international conference on machine learning* (Vol. 119, pp. 5132–5143).

Katsikouli, P., Ferraro, P., Richardson, H., Cheng, H., Anderson, S., Mallya, D., ... Shorten, R. (2020). Distributed ledger enabled control of tyre induced particulate matter in smart cities. *Frontiers in Sustainable Cities*, *2*, 48.

Kawase, Y., & Sumita, H. (2020). On the max-min fair stochastic allocation of indivisible goods. In *The conference on artificial intelligence, AAAI* (pp. 2070–2078).

Khalid, M., Wang, K., Aslam, N., Cao, Y., Ahmad, N., & Khan, M. K. (2021). From smart parking towards autonomous valet parking: A survey, challenges and future works. *Journal of Network and Computer Applications*, *175*.

Khamse-Ashari, J., Lambadaris, I., Kesidis, G., Urgaonkar, B., & Zhao, Y. (2019). A cost-aware fair allocation mechanism for multi-resource servers. *IEEE Networking Letters*, *1*(1), 34–37.

Kia, S. S., Cortes, J., & Martinez, S. (2015). Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica*, *55*, 254 – 264.

Koloskova, A., Stich, S., & Jaggi, M. (2019). Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the international conference on machine learning* (Vol. 97, pp. 3478–3487).

Konecny, J., McMahan, B., & Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. In *NIPS workshop on optimization for machine learning.*

Konecny, J., McMahan, H. B., Ramage, D., & Richtarik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, *arXiv:1610.02527 [cs.LG]*.

Kortuem, G., & Bourgeois, J. (2016). The Internet of things for the open sharing economy. In *Proceedings of the ACM international joint conference on pervasive and ubiquitous computing* (pp. 666–669).

Lan, G., Lee, S., & Zhou, Y. (2018). Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*.

Lan, J., Ma, Y., Zhu, D., Mangalagiu, D., & Thornton, T. F. (2017). Enabling value co-creation in the sharing economy: The case of mobike. *Sustainability*, *9*(9), 1–20.

Léauté, T., & Faltings, B. (2013, May). Protecting privacy through distributed computation in multi-agent decision making. *J. Artif. Int. Res.*, *47*(1), 649–695.

Lee, S., & Nedic, A. (2016). Asynchronous gossip-based random projection algorithms over networks. *IEEE Transactions on Automatic Control*, *61*(4), 953–968.

Lee, S., Nedic, A., & Raginsky, M. (2018). Coordinate dual averaging for decentralized online optimization with nonseparable global objectives. *IEEE Transactions on Control of Network Systems*, *5*(1), 34–44.

Leung, K.-C., Lai, C., & Ding, H. (2021). Leave no cash on the table: An optimal approach for controlling wireless TCP AIMD. *IEEE Transactions on Network Science and Engineering*, *8*(3), 2235–2248.

Li, B., Cen, S., Chen, Y., & Chi, Y. (2020). Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. In *Proceedings of the international conference on artificial intelligence and statistics* (Vol. 108, pp. 1662–1672).

Li, C., Chen, S., Li, J., & Wang, F. (2019). Distributed multi-step subgradient optimization for multi-agent system. *Systems & Control Letters*, *128*, 26–33.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In *Proceedings of machine learning and systems, MLSys.*

Lin, T., Rivano, H., & Mouel, F. L. (2017, Dec). A survey of smart parking solutions. *IEEE*

Transactions on Intelligent Transportation Systems, *18*(12), 3229–3253.

Liu, L., & Han, Z. (2015). Multi-block ADMM for big data optimization in smart grid. In *International conference on computing, networking and communications* (pp. 556–561).

Liu, S., Chen, P. Y., & Hero, A. O. (2018). Accelerated distributed dual averaging over evolving networks of growing connectivity. *IEEE Transactions on Signal Processing*, *66*(7), 1845–1859.

Liu, S., Qiu, Z., & Xie, L. (2017). Convergence rate analysis of distributed optimization with projected subgradient algorithm. *Automatica*, *83*, 162–169.

Lu, Q., Yao, J., Qi, Z., He, B., & Guan, H. (2016). Fairness-efficiency allocation of CPU-GPU heterogeneous resources. *IEEE Transactions on Services Computing*.

Lu, T., & Boutilier, C. (2011). Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the international joint conference on artificial intelligence, IJCAI* (pp. 280–286).

Makhdoumi, A., & Ozdaglar, A. (2014). Broadcast-based distributed alternating direction method of multipliers. In *Annual Allerton conference on communication, control, and computing* (pp. 270–277).

Masip-Bruin, X., Marin-Tordera, E., Jukan, A., & Ren, G. (2018). Managing resources continuity from the edge to the cloud: Architecture and performance. *Future Generation Computer Systems*, *79*, 777–785.

Matt, P. A., & Toni, F. (2006). Egalitarian allocations of indivisible resources: Theory and computation. In *Cooperative information agents X* (pp. 243–257).

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of international conference on artificial intelligence and statistics* (Vol. 54, pp. 1273–1282).

Mohanty, S. P., Choppali, U., & Kougianos, E. (2016, July). Everything you wanted to know about smart cities: The Internet of things is the backbone. *IEEE Consumer Electronics Magazine*, *5*(3), 60–70.

Moret, F., & Pinson, P. (2018). Energy collectives: A community and fairness based approach to future electricity markets. *IEEE Transactions on Power Systems*.

Murhekar, A., & Garg, J. (2021). On fair and efficient allocations of indivisible goods. In *The conference on artificial intelligence, AAAI* (pp. 5595–5602).

Narasimhan, C., Papatla, P., Jiang, B., Kopalle, P. K., Messinger, P. R., Moorthy, S., ... Zhu, T. (2018). Sharing economy: Review of current research and future directions. *Customer Needs and Solutions*, *5*(1), 93–106.

Nedic, A. (2011). Asynchronous broadcast-based convex optimization over a network. *IEEE Transactions on Automatic Control*, *56*(6), 1337–1351.

Nedic, A., & Ozdaglar, A. (2009, Jan.). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, *54*(1), 48–61.

Nedic, A., Ozdaglar, A., & Parrilo, P. A. (2010, Apr.). Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, *55*(4), 922–938.

Nedic, A., Pang, J., Scutari, G., & Sun, Y. (2018). Distributed optimization over networks. In *Multi-agent optimization: Lecture notes in mathematics* (Vol. 2224, pp. 1–84).

Nguyen, D. H. (2021). Optimal solution analysis and decentralized mechanisms for peer-to-peer energy markets. *IEEE Transactions on Power Systems*, *36*(2), 1470–1481.

Nguyen, D. T., Le, L. B., & Bhargava, V. K. (2019). A market-based framework for multi-resource allocation in fog computing. *IEEE/ACM Transactions on Networking*, *27*(3), 1151–1164.

Nguyen, T. T., Roos, M., & Rothe, J. (2013, Jul). A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. *Annals of Mathematics and Artificial Intelligence*, *68*(1), 65–90.

Oh, H., Procaccia, A. D., & Suksompong, W. (2021). Fairly allocating many goods with few queries. *SIAM J. Discret. Math.*, *35*(2), 788–813.

Ohseto, S. (2021). Strategy-proof and Pareto efficient allocation of indivisible goods: General impossibility domains. *Int. J. Game Theory*, *50*(2), 419–432.

Parag, Y., & Sovacool, B. K. (2016). Electricity market design for the prosumer era. *Nature Energy*, *1*.

Poullie, P., Bocek, T., & Stiller, B. (2018). A survey of the state-of-the-art in fair multi-resource allocations for data centers. *IEEE Transactions on Network and Service Management*, *15*(1), 169–183.

Pu, S., & Nedic, A. (2021). Distributed stochastic gradient tracking methods. *Mathematical Programming*, *187*(1), 409–457.

Pu, S., Shi, W., Xu, J., & Nedic, A. (2021). Push–pull gradient methods for distributed optimization in networks. *IEEE Transactions on Automatic Control*, *66*(1), 1–16.

Raviv, T., & Kolka, O. (2013). Optimal inventory management of a bike-sharing station. *IIE Transactions*, *45*(10), 1077–1093.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konecny, J., ... McMahan, H. B. (2021). Adaptive federated optimization. In *International conference on learning representations, ICLR*.

Ritzer, G., & Jurgenson, N. (2010). Production, consumption, prosumption: The of capitalism in the age of the digital 'prosumer'. *Journal of Consumer Culture*, *10*(1), 13–36.

Rivera, J., Goebel, C., & Jacobsen, H. (2017). Distributed convex optimization for electric vehicle aggregators. *IEEE Transactions on Smart Grid*, *8*(4), 1852–1863.

Robbins, H., & Monro, S. (1951, Sept.). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*(3), 400–407.

Rockafellar, R. T. (2015). *Convex analysis*. Princeton University Press.

Romao, L., Margellos, K., Notarstefano, G., & Papachristodoulou, A. (2021). Subgradient averaging for multi-agent optimisation with different constraint sets. *Automatica*, *131*, 109738.

Salehisadaghiani, F., & Pavel, L. (2016). Distributed Nash equilibrium seeking: A gossip-based algorithm. *Automatica*, *72*, 209–216.

Sampath, L. P. M. I., Paudel, A., Nguyen, H. D., Foo, E. Y. S., & Gooi, H. B. (2021). Peer-to-peer energy trading enabled optimal decentralized operation of smart distribution grids. *IEEE Transactions on Smart Grid*.

Sattler, F., Wiedemann, S., Muller, K. R., & Samek, W. (2020). Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(9), 3400–3413.

Satyanarayanan, M. (2017, Jan). The emergence of edge computing. *Computer*, *50*(1), 30–39.

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009, Oct). The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*(4), 14-23.

Schey, S. (2014). *Canadian electric vehicle infrastructure deployment guidelines.*

Schill, W., Zerrahn, A., & Kunz, F. (2017). Prosumage of solar electricity: Pros, cons, and the system perspective. *DIW Berlin*(1637).

Shah, S. N., Incremona, G. P., Bolzern, P., & Colaneri, P. (2019). Optimization based AIMD saturated algorithms for public charging of electric vehicles. *Eur. J. Control*, *47*, 74–83.

Shang, F., Xu, T., Liu, Y., Liu, H., Shen, L., & Gong, M. (2021). Differentially private ADMM algorithms for machine learning. *IEEE Transactions on Information Forensics and Security*, *16*, 4733–4745.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016, Oct). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646.

Shorten, R., King, C., Wirth, F., & Leith, D. (2007). Brief paper: Modelling TCP congestion control dynamics in drop-tail environments. *Automatica*, *43*(3), 441–449.

Silvestre, D., Hespanha, J. P., & Silvestre, C. (2019). Broadcast and gossip stochastic average consensus algorithms in directed topologies. *IEEE Transactions on Control of Network Systems*, *6*(2), 474–486.

Song, M., Ou, Z., Castellanos, E., Ylipiha, T., Kamarainen, T., Siekkinen, M., ... Hui, P. (2017, Dec). Exploring vision-based techniques for outdoor positioning systems: A feasibility study. *IEEE Transactions on Mobile Computing*, *16*(12), 3361–3375.

Sousa, T., Soares, T., Pinson, P., Moret, F., Baroche, T., & Sorin, E. (2019). Peer-to-peer and community-based markets: A comprehensive review. *Renewable and Sustainable Energy Reviews*, *104*, 367–378.

Sousanis, J. (2015, Aug). *World vehicle population tops 1 billion units.*

Studli, S., Crisostomi, E., Middleton, R., & Shorten, R. (2012). A flexible distributed framework for realising electric and plug-in hybrid vehicle charging policies. *International Journal of Control*, *85*(8), 1130–1145.

Tallapragada, P., & Cortes, J. (2020). Hierarchical-distributed optimized coordination of intersection traffic. *IEEE Transactions on Intelligent Transportation Systems*, *21*(5), 2100–2113.

Teodorovic, D., & Lucic, P. (2006). Intelligent parking systems. *European Journal of Operational Research*, *175*(3), 1666 – 1681.

Tsianos, K. I., Lawlor, S., & Rabbat, M. G. (2012). Push-sum distributed dual averaging for convex optimization. In *IEEE conference on decision and control (CDC)* (pp. 5453–5458).

Tsitsiklis, J., Bertsekas, D., & Athans, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, *31*(9), 803–812.

Tsitsiklis, J. N. (1984). *Problems in decentralized decision making and computation* (Unpublished doctoral dissertation). Massachusetts Institute of Technology, Cambridge, MA, USA.

Tushar, W., Saha, T. K., Yuen, C., Smith, D., & Poor, H. V. (2020). Peer-to-peer trading in electricity networks: An overview. *IEEE Transactions on Smart Grid*, *11*(4), 3185–3200.

Tushar, W., Yuen, C., Mohsenian-Rad, H., Saha, T., Poor, H. V., & Wood, K. L. (2018). Transforming energy networks via peer-to-peer energy trading: The potential of game-theoretic approaches. *IEEE Signal Processing Magazine*, *35*(4), 90–111.

Tushar, W., Yuen, C., Saha, T. K., Morstyn, T., Chapman, A. C., Alam, M. J. E., ... Poor, H. V. (2021). Peer-to-peer energy systems for connected communities: A review of recent advances and emerging challenges. *Applied Energy*, *282*.

Umer, K., Huang, Q., Khorasany, M., Afzal, M., & Amin, W. (2021). A novel communication efficient peer-to-peer energy trading scheme for enhanced privacy in microgrids. *Applied Energy*, *296*.

Uribe, C. A., Lee, S., Gasnikov, A., & Nedic, A. (2021). A dual approach for optimal algorithms in distributed optimization over networks. *Optimization Methods and Software*, *36*(1), 171-210.

Vaya, M. G., Andersson, G., & Boyd, S. (2014). Decentralized control of plug-in electric vehicles under driving uncertainty. In *IEEE PES innovative smart grid technologies, Europe* (pp. 1–6).

Wang, J., Liu, Q., Liang, H., Joshi, G., & Poor, H. V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Annual conference on neural information processing systems, NeurIPS.*

Wang, Q., Liu, X., Du, J., & Kong, F. (2016). Smart charging for electric vehicles: A survey from the algorithmic perspective. *IEEE Communications Surveys and Tutorials*, *18*(2), 1500–1517.

Wang, W., Liang, B., & Li, B. (2015). Multi-resource fair allocation in heterogeneous cloud

computing systems. *IEEE Transactions on Parallel and Distributed Systems*, *26*(10), 2822–2835.

Wei, E., & Ozdaglar, A. (2012, Dec). Distributed alternating direction method of multipliers. In *IEEE conference on decision and control (CDC)* (pp. 5445–5450).

Wirth, F., Stanojevic, R., Shorten, R., & Leith, D. (2006). Stochastic equilibria of AIMD communication networks. *SIAM Journal on Matrix Analysis and Applications*, *28*(3), 703–723.

Wirth, F., Stüdli, S., Yu, J. Y., Corless, M., & Shorten, R. (2019). Nonhomogeneous place-dependent Markov chains, unsynchronised AIMD, and optimisation. *Journal of the ACM*, *66*(4), 24:1–24:37.

Yang, T., Yi, X., Wu, J., Yuan, Y., Wu, D., Meng, Z., ... Johansson, K. H. (2019). A survey of distributed optimization. *Annual Reviews in Control*, *47*, 278–305.

Yilmaz, M., & Krein, P. T. (2013, May). Review of battery charger topologies, charging power levels, and infrastructure for plug-in electric and hybrid vehicles. *IEEE Transactions on Power Electronics*, *28*(5), 2151–2169.

Yin, X., Zhu, Y., & Hu, J. (2021). A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.*, *54*(6).

Yuan, H., Zaheer, M., & Reddi, S. (2021). Federated composite optimization. In *Proceedings of the 38th international conference on machine learning* (Vol. 139, pp. 12253–12266).

Yuan, K., Ling, Q., & Yin, W. (2016). On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, *26*(3), 1835–1854.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014, Feb). Internet of things for smart cities. *IEEE Internet of Things Journal*, *1*(1), 22–32.

Zhang, C., Wu, J., Zhou, Y., Cheng, M., & Long, C. (2018). Peer-to-peer energy trading in a microgrid. *Applied Energy*, *220*, 1 – 12.

Zhang, J., Uribe, A. C., Mokhtari, A., & Jadbabaie, A. (2019). Achieving acceleration in distributed optimization via direct discretization of the heavy-ball ODE. In *American control conference (ACC)* (pp. 3408–3413).

Zhang, R., & Kwok, J. T. (2014). Asynchronous distributed ADMM for consensus optimization. In

Proceedings of the international conference on international conference on machine learning.

Zhou, Y., Ye, Q., & Lv, J. (2022). Communication-efficient federated learning with compensated Overlap-FedAvg. *IEEE Transactions on Parallel and Distributed Systems*, *33*(1), 192–205.