Selfishness in Strategic Resource Allocation Problems

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Vipin Ravindran Vijayalakshmi, M.Sc.

aus Thiruvananthapuram, Indien

Berichter:	Univ.-Prof. Dr. rer. nat. Britta Peis
	Univ.-Prof. Dr. Gerhard Woeginger
	Assistant Professor Dr. Alexander Skopalik

Tag der mündlichen Prüfung: 06.09.2021

Abstract

The advent of modern technology in the communication and the transportation industry encouraged the proliferation of its consumer base. Due to the rising demand on many of these modern applications and scenarios, centralization was no longer deemed a viable approach for managing their operations. These days, many modern infrastructures are designed to allow for multiple strategic users who make decisions that suit their individual utility. Consequently, non-cooperative game theory has emerged as an essential tool in analyzing and predicting the outcome of decentralized systems. We study various algorithmic aspects arising due to strategic behavior among multiple non-cooperative users in several classes of resource allocation problems using a game theoretic approach. The classes of strategic games we study, succinctly represent many of the socio-economic scenarios arising due to decentralization.

At first, we consider congestion games which are often used to model various scenarios of resource allocation by non-cooperative users. In these games, the resources could possibly correspond to edges in a road or communication network, machines in distributed systems, etc., and have cost functions with a diseconomy of scale. The players in a congestion game then iteratively pick feasible subsets of resources that maximize their individual utility. The players continue to strategically deviate, if necessary, until the game converges to a widely studied solution concept known as the pure Nash equilibrium. The hardness of computing a pure Nash equilibrium in congestion games has been of significant interest in the scientific community over the past two decades. As computing an exact pure Nash equilibrium is known to be hard, we study a weaker notion of pure Nash equilibrium called an approximate pure Nash equilibrium and to that extend, analyze an efficient algorithm that computes approximate pure Nash equilibria in congestion games with an approximation guarantee that is by far the best known in the literature.

The impact of non-coordination among the users on the self emerging solutions of a congestion game are provably bad. One of the many approaches used to alleviate the effects of non-cooperativeness is the introduction of taxes. Economic incentives have provably shown to influence the users to induce solutions that are significantly closer to the optimal solution. We present a mechanism to compute load dependent taxes for congestion games that are robust against the changes in the number of users and resources in the game.

In spite of being the predominant class of games to model resource allocation problems involving different users, congestion games lack an element of time dependence, especially in certain application scenarios such as the road transportation network. It is quite unnatural to assume that all users of a road section experience the same congestion. A cost sharing mechanism must also take into account the order in which a resource processes the users assigned to it. We extend congestion games with resource dependent priority list to model the impact of fixed order scheduling on the strategic behavior of users. We then study the question of existence and inefficiency of pure Nash equilibria in the extended model. It is quite natural to then consider a scheduling game on parallel machines, in which jobs try to minimize their completion time by choosing a machine to be processed on. Each machine uses an individual priority list to decide on the order according to which the jobs on the machine are processed. Here, we study the existence of a pure Nash equilibrium and characterize classes of instances in which a pure Nash equilibrium is guaranteed to exist. For each of these classes, we settle algorithmic questions such as tractability and the inefficiency of pure Nash equilibria.

Finally, we study the impact of non-cooperative users in a large scale distributed communication network such as the peer-to-peer overlay network, where the existence of non-cooperative and malicious users is considered as a rule rather than an exception. Due to its distributed architecture, users or peers are allowed to join and leave without admission control. A necessary condition to maintain the stability of a distributed overlay network that allows for efficient storage and retrieval of information is to have certain degree of commitment and coordination among the peers in the network. Moreover, the lack of a centralized mechanism entails that the peers in the network coordinate among themselves. However, peers joining and leaving the network in a non-coordinated manner leads to instability and loss of information. We model this scenario using an adversary and present an algorithm that is able to ensure stability in the network in the presence of large fraction of non-cooperative peers.

Zusammenfassung

Die kontinuierlichen Entwicklungen innerhalb der Kommunikation- sowie Transportindustrie führten zur einer stetigen Zunahme der Nutzer in den letzten Jahrzehnten. Aufgrund der damit einhergehenden Steigerung des Bedarfs an modernen Applikationen und Szenarios sind zentralisierte Lösungen heute kein zufriedenstellendes Konzept mehr diese Anwendungen umzusetzen. Moderne Infrastruktur-Systeme sind daher so entwickelt, dass viele strategische Nutzer eigenständig Entscheidungen tätigen können. Diese Eigenständigkeit hat jedoch die Folge, dass die jeweiligen Entscheidungen nur ihren persönlichen Nutzen maximieren. Dieser Tatsache geschuldet entwickelte sich die Nicht-kooperative Spieltheorie, in der die Ausgänge dieser dezentralen Systeme analysiert und vorhergesagt werden.

In der vorliegenden Arbeit studieren wir verschiedene algorithmische Fragestellungen, die sich aus der Existenz von nicht-kooperativen Spielern in Auslastungsspielen ergeben. Hierzu nutzen wir insbesondere Techniken aus der Spieltheorie sowie der Komplexitätstheorie. Die untersuchte Klasse der strategischen Spiele beschreibt zuverlässig diverse sozioökonomische Szenarien, die im Zusammenhang mit Dezentralisierung entstehen.

Zuerst betrachten wir Congestion-Games, die häufig in Modellen genutzt werden in dennen eine gewisse Anzahl an Ressourcen auf nicht-kooperative Spieler aufgeteilt werden. Hier präsentieren wir einen Algorithmus, der approximative Nash-Gleichgewichte in Congestion-Games berechnet und dessen Approximationsgüte bekannte Resultate in der Literatur stark verbessert. Darüber hinaus betrachten wir eine Erweiterung des Spieles, in der die Spieler durch finanzielle Anreize zu besseren Entscheidungen für die Allgemeinheit bewogen werden sollen. Hier präsentieren wir Kostenfunktionen für die Ressourcen, die von der jeweiligen Belegung der Ressourcen abhängen. Diese Kosten erweisen sich als robust gegenüber Veränderungen in der Anzahl der Spieler als auch der Ressourcen des Spiels.

Darüber hinaus untersuchen wir eine weitere Abwandlung von Congestion-Games, indem wir eine zeitliche Komponente hinzufügen. Die einzelen Resourcen werden nicht mehr simultan von allen Spielern genutzt, sondern nacheinander entsprechend einer Prioritätsliste der Ressource. In diesem Modell studieren wir die Frage der Existenz und der Effizienz von Nash-Gleichgewichten. Zudem untersuchen wir algorithmische Fragestellungen in den sogenannten Scheduling-Games.

Zuletzt untersuchen wir den Einfluss von nicht-kooperativen Benutzern in distributiven Kommunikationsnetzwerken, wie zum Beispiel in Peer-to-Peer Netzwerken, in denen die Existenz von nicht vertrauenswürdigen Nutzern angenommen werden muss. Unter Berücksichtigung dieser Unsicherheit für das System präsentieren wir einen Algortihmus, der Stabilität im Netzwerk garantieren kann, selbst wenn eine große Anzahl von nicht-kooperativen Spielern existiert.

Acknowledgments

In the recent years, I had the opportunity to conduct my research in an excellent environment and work with a number of people. I am grateful to the research training group UnRAVeL that I was given this opportunity.

In particular, I want to express my gratitude to my supervisors Britta Peis and Gerhard Woeginger for their guidance and support. I am grateful to Alexander Skopalik for his guidance and for sharing inspiring ideas. I want to thank Martin Grohe and Gerhard Lakemeyer for kindly agreeing to be part of my examination committee. I would like to thank my co-authors Thorsten Götte, Alexander Skopalik, Marc Schröder, and Tami Tamir for sharing their ideas and all the edifying discussions. I want to thank Marc Schröder, Björn Tauer, Niklas Rieken, Tim Hartmann, and Thorsten Götte for proof-reading several parts of this thesis. Furthermore, I would like to thank all my friends and colleagues who accompanied my scientific career. Particularly, I want to express my gratitude to Björn Tauer, Niklas Rieken, Daniel Schmand, Laura Vargas Koch, Marc Schröder, Veerle Timmermans, Dina Hermanns, Dennis Fischer, Tim Hartmann, Janosch Fuchs, Helen Bolke-Hermanns, Birgit Willms, Andreas Wierz, Martin Groß, Walter Unger, Sophia Wrede, Oliver Antons, Sharat Ibrahimpur, Amal Pillai, Thorsten Götte, and Andreas Schultz.

Finally, I would like to thank my parents Ravindran and Vijayalakshmi and my sister Ravina for their encouragement and support.

Contents

Chapter 1

Introduction

Algorithmic game theory studies the strategic behavior of users in many of the socio-economic situations involving different users [NRTV07]. The advent of technology and growing consumer base saw a rising demand for the decentralization of various large scale infrastructures. The absence of a central authority implied that the strategic decisions of individual users played a vital role in determining the overall efficacy and performance of an infrastructure. For example, there has been an increasing demand on road transportation networks across the world over the years. The steady rise in motor vehicle production and sales due to increasing demand for transportation of people and goods, warranted for well planned transportation networks. However, the non-cooperative behavior among the users of the network resulted in sub-optimal traffic flows and avoidable delays. The time needed for the users to commute then inevitably depends on the total amount of traffic in the network [U.S64], i.e., the delay a user incurs also depends on the strategic choices of other users in the network. Therefore, it is quite natural to model the strategic nature of users in a transportation network as a strategic game. A strategic game consists of a finite set of players and for each player, a finite set of strategies. A predefined cost function then determines the cost of a player conditioned on the strategies of the other players in the game. The players' objective is to iteratively pick feasible strategies that minimize their personal cost until the game converges to a stable solution in which none of the players deviate. Analyzing various factors such as traffic congestion, average travel time, etc., in a road transportation network can be studied by investigating the efficiency of the self emerging solutions in the corresponding game. The most commonly used solution concept among many others to characterize these self emerging solutions are the equilibrium concepts, most famous of them all being the Nash equilibrium. The concept of a Nash equilibrium was introduce by John Forbes Nash, Jr. in his seminal dissertation [Nas50]. A Nash equilibrium describes a solution in which none of the players have a unilateral deviation by which they could improve their cost or utility. The last two decades have seen significant amount of literature investigating algorithmic aspects such as inefficiency, existence and tractability of Nash equilibrium solutions. In particular, the existence and computational hardness of *pure* Nash equilibria [NRTV07, p. 12] in various strategic games that model a diseconomy of scale [FPT04, ARV08]. In addition to their theoretical significance, the existence and computation of pure Nash equilibria is of utmost importance to the practitioners as they succinctly describe and predict the uncertainty in various strategic problems arising due to non-coordination.

The existence of pure Nash equilibria in strategic games is not always guaranteed, e.g., Matching Pennies [NRTV07, see, Example 1.7]. However, certain classes of games always exhibit a pure strategy Nash equilibrium [MS96]. The most prominent of all are the congestion games introduced by Rosenthal [Ros73a]. Congestion games constitute an important class of games that succinctly represents a game theoretic model for resource allocation among non-cooperative and strategic users [EK12]. In particular, a congestion game is a cost-minimization game defined by a set of resources with a diseconomy of scale and a set strategic players. For congestion games, Rosenthal [Ros73a] using a potential function argument proved that it belongs to a class of games in which a pure Nash equilibrium always exists, i.e., the game always consists of a self-emerging solution in which no user is able to improve by unilaterally deviating. Moreover, the computational complexity of pure Nash equilibria in congestion games has been well studied [FPT04, ARV08, CS11]. The computation of exact pure Nash equilibria in general is shown to be hard [FPT04]. However, certain restricted structures in the strategy space [FPT04, ARV08] or cost functions [CS11] have shown positive results. The literature has also received considerable attention towards studying a rather weaker notion of equilibrium know as an *approximate* pure Nash equilibrium. To our knowledge, the concept of α-approximate pure Nash equilibria was introduced by Roughgarden and Tardos [RT00] in the context of non-atomic selfish routing games [NRTV07]. An α-approximate pure Nash equilibrium is a solution in which none of the users can unilaterally deviate to improve by a factor of at least α. The computational hardness of approximate pure Nash equilibria in congestion games has been well studied [CS11, SV08, CFGS11]. In particular, for congestion games with non-decreasing cost functions that are bounded degree polynomials [CFGS11]. We study the computation of approximate pure Nash equilibria in congestion games with non-decreasing cost functions and to that extend, present a polynomial time algorithm that improves the results in [CFGS11].

The outcome of a strategic game need not be unique. The order in which the players iteratively deviate essentially decides the feasible outcome to which the game converges. Therefore, to understand the uncertainty in the outcome of a resource allocation problem with a diseconomy of scale involving strategic users, it is crucial to be able to measure their worst-case inefficiency. There exists significant amount of literature investigating the bounds on the inefficiency of equilibria [KP99] for various non-cooperative games [AAE05, Ros73b, RS11b, CQ12, GLMM10, CCG$^+$15, CMS12, BFFM11, KST19, BV17]. It is easy to see the negative impact of non-cooperativeness on the quality of the equilibrium. One of the many approaches used to improve the inefficiency is the introduction of economic incentives [CKK06, FKK10, FS07, Swa12, BV16, Vij17]. Cole et al. [CDR06] study the influence of taxes and subsidies on the strategic behavior of players to improve inefficiency of equilibria in non-atomic selfish routing games [NRTV07, p. 462]. They investigate the influence of economic incentives and its ability to induce desirable solutions in the game. Bilò and Vinci [BV16] present an algorithm to compute optimal load dependent taxes in congestion games. We study universal load dependent taxes which are robust against perturbation in the game instance such as number of players and resources. We provide improved bounds on the inefficiency of equilibria in congestion games under robust load dependent taxes.

Although congestion games remain to be one of the predominant and well studied model for analyzing traffic behavior in transportation networks, it is easy to see that it lacks an element of time dependency and assumes that all users on the road experience the same delay. However, observe that a vehicle can only be delayed by the vehicles ahead of it. It would only but be more realistic, if we extend the model to incorporate some element of time dependency such that, it precisely reflects the delay incurred by a user at each section of the road network, e.g., based on the order in which they arrive at that section. Variants to congestion games that incorporates certain restricted structures of time dependence have been well studied [AGM$^+$08, FOV08, GMMT15, PNS16, BV20]. We relax these restrictive assumptions and present a model that generalizes the time dependency in congestion games. We study the existence, computational complexity, and settle bounds on the inefficiency of equilibria in these games. It is also quite natural then to theoretically understand the impact of fixed order scheduling from a parallel machine scheduling perspective.

Scheduling problems are considered to be some of the most fundamental topics in the area of theoretical computer science. A typical scheduling problem can be described as assigning a set of independent jobs to machines such that, the assignment minimizes a predefined objective, e.g., the maximum load across machines [Gra66], the sum of delays incurred by jobs [Smi56], etc. We refer the readers to the book of Michael L. Pinedo [Pin08] for a comprehensive introduction to various models of scheduling problems. Scheduling problems have traditionally been studied from a centralized point of view. However, the impact of order based time dependency on scheduling problems has only received marginal attention in the literature [AGA99, ANCK08, All15, BFO$^+$19]. In order to model time dependence, we consider priority based scheduling on parallel machines. In this variant of scheduling we consider, the request of some jobs are preferred over the others, i.e., each machine imposes a preferential order over the jobs assigned to it. The preference order may be machine dependent and need not be restricted to general rules such as fair cost sharing, first-in first-out, shortest processing time first, etc. We believe that many real-world applications such as airline boarding and vehicle parking, can be succinctly modeled using the aforementioned variant. We study the hardness in computing an optimal assignment under local fixed order scheduling and settle some of the complexity results for certain interesting settings of the problem.

Although scheduling problems have traditionally been studied from a centralized perspective, as a consequence of decentralization many modern job scheduling applications are designed to allow for multiple strategic users, i.e., the outcome of these applications are determined by the strategic decisions taken by the users instead of a central authority. As a result, non-cooperative game theory has become an essential component in the study of job-scheduling applications, where jobs are controlled by non-cooperative users who independently and strategically choose resources that are beneficial to them [CV07, AART06, GLMM10, Vöc07]. Additionally, the impact of priority based scheduling in these games with certain restrictive assumptions have been widely studied with respect to the inefficiency of equilibria [CKN04, ILMS09, CCG$^+$15, DN09, AJM08]. However, we believe that there are no natural justifications to these assumptions and therefore, we relax them and consider a significantly more general setting in which machines have arbitrary individual priority over the jobs. That is, each machine schedules those jobs that have chosen it according to its priority list. We study algorithmic aspects such as existence, inefficiency and computability of pure Nash equilibria in these games for objectives such as minimum makespan of the schedule and sum of job completion time.

Strategic behavior need not necessarily be restricted to users choice of resources, routes, etc., in a decentralized infrastructure. One can also consider the level of altruism shown by users in maintaining coordination and stability in a distributed network of systems, as a strategic decision. Traditional Internet Protocol (IP) networking architecture was primarily based on the client-server networking principles. In this model, users of the communication network participate either as a client or as a server. The communication framework typically comprises of a single server catering to many clients. Most file sharing applications, e.g., the File Transfer Protocol (FTP) were traditionally based on this client-server model of networking. However, the last few decades have seen file sharing applications steadily shifting towards ad-hoc networking principles like the Peer-To-Peer (P2P) overlay networks [Sch01, ECP$^+$05]. The users in P2P networks are often referred to as peers and participate in the communication model as both clients and servers simultaneously. Peers in the network provide computational and storage infrastructure for efficient information storage and retrieval in the network. This gives rise to a distributed networking architecture without the need for a central mechanism to orchestrate file sharing in the IP network. The self-orchestrating nature of the P2P network implied that peers were allowed to join and leave (churn) the network without any form of admission control. The peers coordinated among themselves to ensure stability in the network. This also allowed for better fault tolerance and scalability in the network [ECP$^+$05].

Although P2P overlay networks enabled significant reduction in computational overhead as a consequence of decentralized orchestration, it imposed considerable challenges on the network maintenance. A necessary condition for maintaining a decentralized network that allowed for efficient storage and retrieval of information is to have trusted peers, i.e., the network warrants a certain degree of commitment and coordination from the peers in the network. In large scale distributed systems without admission control, the existence of non-cooperative and malicious peers is considered as a rule rather than an exception [SR06]. This poses as a major challenge in maintaining P2P network stability [ECP$^+$05]. However, the severity of the problem is partly governed by the overlay topology. P2P overlay networks have been mainly classified into three types: *Structured*, *Unstructured* and *Hybrid*. We refer the readers to Darlagiannis [Dar05] and EK Lua et al. [ECP$^+$05] for a comprehensive introduction to these classes of overlay networks. Here, we only briefly discuss some of the specific details of structured and unstructured overlays that alludes to their efficacy in information storage and retrieval in a large scale distributed network of systems.

The predominant and naturally emerging form of P2P overlay network in the Internet today are the unstructured overlay networks. As the name suggests, an unstructured P2P overlay does not impose a structure on the overlay's topology. Peers are allowed to choose random neighborhood and data exchange is usually based on flooding and random walk techniques on the induced graph [GMS04, APR$^+$15]. Many file sharing applications such as BitTorrent are based on unstructured P2P networking principles [ECP$^+$05]. The omission of structure enables them to be built relatively fast and are highly resilient to peer dynamics such as churn and maliciousness [JC10, ECP$^+$05]. However, the randomness in the topology make them less suitable for file sharing frameworks that require data to be stored and retrieved from specific locations in the network. In contrast, the structured overlay networks enforce a strict structure on the topology. This enables the file sharing applications built over these networks to control the location where the information is stored and therefore, ensure efficient retrieval [RS11a, FSY05, AS21]. These networks typically employ the Distributed Hash Table (DHT) mechanism to organize and retrieve information in the network [NW07, ECP$^+$05]. Although structured overlay networks are efficient in terms of information storage and retrieval, they are usually less resilient to faults arising due to maliciousness and unpredictable participation (churn) of nodes in the network. The last two decades have seen extensive amount of work in building highly churn resistant structured overlay networks [Sch05, FSY05, AS07, AS21, DGS16]. One of the many approaches in the literature to tackle unpredictable churn is to design mechanisms that induce structural robustness in the overlay against churn. One such mechanism is to continuously reorganize the overlay to restore structure [MS06]. We model the unpredictability in node dynamics using an omniscient adversary and analyze a structured P2P overlay network based on a graph topology that we refer to as *Linearized DeBruijn Swarm*. We then present a distributed algorithm that maintains the stability of the network in the presence of adversarial churn.

1.1 Strategic Games and Equilibrium Concepts

We now introduce the classes of games we study in the remainder of this thesis and the necessary game theoretic preliminaries required for their analysis. We remark that we restrict our attention to the relevant details of only a selected class of games and deliberately refrain from presenting a broader introduction to game theory. We refer the readers to [NRTV07] for a comprehensive introduction to Algorithmic Game Theory. We begin by understanding the fundamental notion of a strategic game.

Definition 1.1 (Strategic Game)**.** A *strategic game* denoted by the tuple

$$\left(\mathcal{N}, (S_u)_{u\in\mathcal{N}}, (C_u)_{u\in\mathcal{N}}\right)$$

consists of a finite set of players $\mathcal{N} = \{1, \ldots, N\}$ and for each player $u \in \mathcal{N}$, a finite set of pure strategies S_u and a cost function $C_u : S \to \mathbb{R}_{\geq 0}$ that maps a state $s \in S \coloneqq S_1 \times S_2 \times \cdots \times S_N$ to the cost of player $u \in \mathcal{N}$.

For a state $s \coloneqq (s_1, \ldots, s_N) \in S$, let $C_u(s'_u, s_{-u})$ denote the cost of a player $u \in \mathcal{N}$ on strategy s'_u, while all the other players adopt their strategy in s.

Definition 1.2 (Mixed Nash Equilibrium)**.** Let σ_u be a probability distribution over the strategy set S_u of a player $u \in \mathcal{N}$. Then, the set $(\sigma_1, \ldots, \sigma_N)$ is a *mixed Nash equilibrium* if no player is able to improve their expected cost under the product distribution $\sigma = \times(\sigma_u)_{u\in\mathcal{N}}$ by unilaterally deviating, i.e., for all $u \in \mathcal{N}$ and for all σ'_u,

$$\mathbf{E}_{s\sim\sigma}\left[C_u(s)\right] \leq \mathbf{E}_{s_{-u}\sim\sigma_{-u}}\left[C_u(s'_u, s_{-u})\right].$$

Every finite game has at least one mixed Nash equilibrium [Nas50].

A state s is called a pure strategy profile, if each player $u \in \mathcal{N}$, picks exactly one strategy from the set of available strategies in S_u. For a pure strategy profile s and a player $u \in \mathcal{N}$, we denote by s_{-u} the strategy profile without player u. A pure strategy profile s is considered as a pure Nash equilibrium (PNE), if none of the players can improve their cost by unilaterally deviating to an alternative strategy. We say the game is in a pure Nash equilibrium in the strategy profile s. All strategic games which are also potential games exhibit at least one pure strategy Nash equilibrium [MS96].

Definition 1.3 (Pure Nash Equilibrium)**.** A state $s \in S \coloneqq S_1 \times S_2 \times \cdots \times S_N$ is a *pure Nash equilibrium* if no player $u \in \mathcal{N}$ can improve their cost by unilaterally deviating from their current strategy in s to another strategy $s'_u \in S_u$, i.e., for all players $u \in \mathcal{N}$ and for all $s'_u \in S_u$ it holds that,

$$C_u(s) \leq C_u(s'_u, s_{-u}).$$

For the sake of convenience we often refer to the pure Nash equilibrium condition in the above definition as the Nash inequality.

A strategy profile s is not a pure Nash equilibrium if there exists at least one player $u \in \mathcal{N}$ and at least one strategy $s'_u \in S_u$ such that,

$$C_u(s) > C_u(s'_u, s_{-u}).$$

It is natural for players to always deviate to alternative strategies that minimizes their cost. The deviation that minimizes a player's cost among all the other allowed deviations is termed as the best response.

Definition 1.4 (Best Response)**.** A strategy $s_u \in S_u$ for a player $u \in \mathcal{N}$ is called a *best response* if it is the best strategy with respect to a fixed strategy of the other players, i.e., for all $s'_u \in S_u$,

$$C_u(s_u, s_{-u}) \leq C_u(s'_u, s_{-u}).$$

Computing an exact pure Nash equilibrium is shown to be hard in certain classes of games [FPT04, ARV08]. Therefore, it is natural to consider a weaker notion of pure Nash equilibrium known as an approximate pure Nash equilibrium [RT00, SV08, CS11, CFGS11]. A strategy profile s is considered to be in an α-approximate pure Nash equilibrium for some $\alpha > 1$, if none of the players are able to improve their cost by a factor of α.

Definition 1.5 (α-Approximate PNE)**.** A state $s \in S := S_1 \times S_2 \times \cdots \times S_N$ is an *α-approximate pure Nash equilibrium* for some $\alpha > 1$, if no player $u \in \mathcal{N}$ can improve their cost by a factor of at least α by unilaterally deviating from their current strategy to another strategy, i.e., for all players $u \in \mathcal{N}$ and for all $s'_u \in S_u$ it holds that,

$$C_u(s) \leq \alpha \cdot C_u(s'_u, s_{-u}).$$

Similar to an exact best response, an α-best response is a unilateral deviation that also improves the cost of a player by factor at least α for a fixed $\alpha > 1$.

Definition 1.6 (α-Best Response)**.** A strategy $s_u \in S_u$ for a player $u \in \mathcal{N}$ is called an *α-best response* if it is the best factor α improving strategy with respect to a fixed strategy of the other players, i.e., for all $s'_u \in S_u$,

$$C_u(s_u, s_{-u}) \leq \alpha \cdot C_u(s'_u, s_{-u}).$$

Additional Equilibrium Concepts

Solution concepts such as pure Nash equilibrium and mixed Nash equilibrium are hard to compute in general [FPT04, DGP06]. However, there exist higher equilibrium concepts such as the correlated equilibrium which are more permissive than pure and mixed Nash equilibria and is computationally tractable [NRTV07, p. 79].

Definition 1.7 (Correlated Equilibrium [Aum74])**.** A probability distribution σ over the set of states S is said to be a *correlated equilibrium* if for every player $u \in \mathcal{N}$, every two strategies $s_u, s'_u \in S_u$ and every recommendation $s = (s_1, \ldots, s_N) \sim \sigma$, the expected cost for following the recommendation s_u is not greater than choosing s'_u instead, i.e.,

$$\mathbf{E}_{s\sim\sigma}\left[C_u(s) \mid s_u\right] \leq \mathbf{E}_{s\sim\sigma}\left[C_u(s'_u, s_{-u}) \mid s_u\right].$$

Definition 1.8 (Coarse Correlated Equilibrium [MV78])**.** A probability distribution σ over the set of states S is said to be a *coarse correlated equilibrium (CCE)* if for all $s_u, s'_u \in S_u$, and for all $u \in \mathcal{N}$,

$$\mathbf{E}_{s\sim\sigma}[C_u(s)] \leq \mathbf{E}_{s\sim\sigma}[C_u(s'_u, s_{-u})].$$

We refer the readers to Roughgarden [Rou15] for further intuitions to these equilibrium concepts.

1.2 Congestion Games

Congestion games belong to a class of strategic games that always exhibit a pure Nash equilibrium [Ros73a] and are often used to model many of the socio-economic scenarios involving strategic users such as the road transportation network [EK12]. In a congestion game the players compete over a set of resources in the game and therefore, the strategy space of a player comprises of subsets of resources.

Definition 1.9 (Congestion Game)**.** A *congestion game* denoted by

$$G = \left(\mathcal{N}, E, (S_u)_{u \in N}, (f_e)_{e \in E}\right)$$

consists of a set of players $\mathcal{N} = \{1, 2, \ldots, N\}$, who compete over a set of resources $E = \{e_1, e_2, \ldots, e_m\}$. Each player $u \in \mathcal{N}$ has a set of strategies denoted by $S_u \subseteq 2^E$. Each resource $e \in E$ has a non-negative and non-decreasing cost function $f_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ associated with it.

The cost of a player $u \in \mathcal{N}$ in any given state s of the game is given by

$$C_u(s) = \sum_{e \in s_u} f_e(n_e(s)).$$

Then, the social cost of a congestion game in a state s is denoted by

$$C(s) = \sum_{u \in \mathcal{N}} C_u(s).$$

For a strategic game G, the state $s^* \in S$ that minimizes the social cost is called the social optimum and its corresponding cost is denoted by $OPT(G)$.

Congestion games exhibit a cost revealing potential function called the Rosenthal potential. For a state s in a congestion game the value of the potential in the state s is given as

$$\Phi(s) = \sum_{e \in E} \Phi_e(s) = \sum_{e \in E} \sum_{i=1}^{n_e(s)} f_e(i),$$

where $n_e(s)$ denotes the number of players on resource e in the state s. The Rosenthal potential is an exact potential, i.e., the change in cost of a player $u \in \mathcal{N}$ unilaterally deviating from their strategy in s to an alternative strategy s'_u is then exactly equal to the change in potential. That is,

$$\Phi(s) - \Phi(s'_u, s_{-u}) = C_u(s) - C_u(s'_u, s_{-u}).$$

Using this property, Rosenthal [Ros73a] showed that every congestion game has a pure Nash equilibrium.

Imposing certain restrictions on the strategy space of a congestion game gives us interesting variants of congestion games.

Definition 1.10. A congestion game is

- *symmetric* if for all players $u, v \in \mathcal{N}$,

$$S_u = S_v,$$

  and *asymmetric*, otherwise.

- *singleton* if for all players $u \in \mathcal{N}$ and for all $s_u \in S_u$,

$$|s_u| = 1.$$

  We remark that these games are often referred to as *selfish scheduling on identical machines.*

1.3 Inefficiency of Equilibria

To understand the outcome of a strategic game, it is crucial to be able to measure the quality of its self emerging solutions. The *price of anarchy* (PoA) [KP99] and the *price of stability* (PoS) [ADK+08] measures the inherent inefficiency of equilibria in strategic games. Both PoA and PoS apply to all Nash equilibrium concepts including pure Nash equilibria (NE).

Let $\mathcal{G}$ be a family of strategic games, and let G be a game in $\mathcal{G}$. Let $\mathcal{E}(G)$ be the set of pure Nash equilibria of the game G and assume that $\mathcal{E}(G) \neq \emptyset$.

Definition 1.11 (Price of Anarchy)**.** The *price of anarchy* of G is the ratio between the maximum cost of an NE and the social optimum of G, i.e.,

$$\text{PoA}(G) = \max_{s \in \mathcal{E}(G)} C(s)/OPT(G).$$

The price of anarchy of $\mathcal{G}$ is

$$\text{PoA}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{PoA}(G).$$

Definition 1.12 (Price of Stability)**.** The *price of stability* of G is the ratio between the minimum cost of an NE and the social optimum of G, i.e.,

$$\text{PoS}(G) = \min_{s \in \mathcal{E}(G)} C(s)/OPT(G).$$

The price of stability of $\mathcal{G}$ is

$$\text{PoS}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{PoS}(G).$$

Caragiannis et al. [CFGS12] introduce the notion of *stretch* in congestion games. The stretch of a congestion game G is defined as the worst-case ratio of the potential function at a pure Nash equilibrium and the global optimum of the potential value in G.

Definition 1.13 (Stretch of a Congestion Game)**.** The *stretch* of a congestion game G is the ratio between the maximum potential of a pure Nash equilibrium and the potential at the potential minimizer s^*, i.e.,

$$\text{stretch}(G) = \max_{s \in \mathcal{E}(G)} \Phi(s)/\Phi(s^*).$$

The stretch of a family of congestion games $\mathcal{G}$ is

$$\text{stretch}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{stretch}(G).$$

1.4 Strategic Games with Fixed Order Scheduling

Restricting the strategy space of a user in a strategic game to singletons gives rise to a scheduling game [Vöc07]. The impact of imposing scheduling policies on non-cooperative scheduling problems were first studied by Christodoulou et al. [CKN04]. They introduced the idea of local scheduling policies to improve the price of anarchy. Note that, for the sake of consistency with the scheduling literature, in the following definition we use σ instead of s to denote a strategy profile and use s_j to denote the speed of a machine j.

Definition 1.14 (Scheduling Games with Priority Lists)**.** A *scheduling game with machine-dependent priority lists* is given by a tuple

$$G = (\mathcal{N}, M, (p_i)_{i \in \mathcal{N}}, (s_j)_{j \in M}, (\pi_j)_{j \in M}),$$

where $\mathcal{N}$ is a finite set of $N \geq 1$ jobs, M is a finite set of $m \geq 1$ parallel related machines, $p_i \in \mathbb{R}_{\geq 0}$ is the processing time of job $i \in \mathcal{N}$, $s_j \in \mathbb{R}_{\geq 0}$ denotes the speed of the machine $j \in M$, and $\pi_j : \mathcal{N} \to \{1, \ldots, N\}$ is the priority list of machine $j \in M$.

A strategy profile $\sigma = (\sigma_i)_{i \in \mathcal{N}} \in M^{\mathcal{N}}$ assigns a machine $\sigma_i \in M$ to every job $i \in \mathcal{N}$. Given a strategy profile σ, the jobs are processed according to their order in the machines' priority lists. The set of jobs that delay $i \in \mathcal{N}$ in s is denoted by,

$$B_i(\sigma) = \{k \in \mathcal{N} \mid \sigma_k = \sigma_i \ \wedge \ \pi_{\sigma_i}(k) \leq \pi_{\sigma_i}(i)\}.$$

The cost of job $i \in \mathcal{N}$ is equal to its completion time in σ, given by

$$C_i(\sigma) = \frac{\sum_{k \in B_i(\sigma)} p_k}{s_{\sigma_i}}.$$

The social cost of the game for the assignment σ is denoted by $C(\sigma)$. The cost is defined with respect to some objective, e.g., the makespan, i.e.,

$$C_{\max}(\sigma) := \max_{i \in \mathcal{N}} C_i(\sigma),$$

or the sum of completion times, i.e., $\sum_{i \in \mathcal{N}} C_i(\sigma)$.

Scheduling games with priority list can also be extended to games with arbitrary strategy spaces. Ackermann et al. [AGM+08] were the first to study a congestion game with priorities. They consider a model in which users with higher priority on a resource displace users with lower priorities. Later, Farzad et al. [FOV08] study priority based selfish routing with atomic users and analyze the inefficiency of their equilibria. We investigate a model that is very similar to Farzad et al.

Definition 1.15 (Weighted Congestion Games with Priority Lists)**.** A *weighted congestion game with resource-dependent priority lists* is given by a tuple

$$G = (\mathcal{N}, E, (S_u)_{u \in N}, (w_u)_{u \in \mathcal{N}}, (c_e)_{e \in E}, (\pi_e)_{e \in E}),$$

where $\mathcal{N}$ is a finite set players, E is a finite set of resources, $S_u \subseteq 2^E$ is the set of feasible strategies for player $u \in \mathcal{N}$, $w_u \in \mathbb{R}_{\geq 0}$ is the weight of player $u \in \mathcal{N}$, $c_e \in \mathbb{R}_{\geq 0}$ is the cost coefficient of resource $e \in E$, and $\pi_e : \mathcal{N} \to \{1, \ldots, N\}$ is the priority list of resource $e \in E$ that defines its preference over the players using it.

Given a strategy profile $s \in S := S_1 \times S_2 \times \cdots \times S_N$, for every player $u \in \mathcal{N}$ and resource $e \in s_u$, let

$$B_{ue}(s) = \{k \in \mathcal{N} \mid e \in s_k \ \wedge \ \pi_e(k) \leq \pi_e(u)\}$$

denote the set of players that delay player $u \in \mathcal{N}$ in the state s on resource e. The cost of a player $u \in \mathcal{N}$ is given by

$$C_u(s) = w_u \cdot \sum_{e \in s_u} \sum_{k \in B_{ue}(s)} c_e \cdot w_k.$$

The social cost of the game in a state s is then given by

$$C(s) = \sum_{u \in \mathcal{N}} C_u(s).$$

1.5 Overlay Networks under Churn

Overlay networks are dynamic logical network topologies built over an underlay network such as the Internet [Sch01, ECP+05]. We consider a model in which time proceeds in synchronous rounds and observe a dynamic set of nodes $\mathcal{V} := \big(V_0, V_1, \dots\big)$ such that V_t is the set of nodes in round t. The model assumes that the node set $\mathcal{V}$ is determined by an adversary. This implies, in every round t the adversary can propose a set $O_t \subset V_{t-1}$ that leaves the network and a set $J_t \subset V_t$ that joins the network, i.e., $V_t := (V_{t-1} \setminus O_t) \cup J_t$. The continuous addition and deletion of nodes in the network is referred to as churn. Each node u in the network is identified by a unique and immutable identifier (e.g. IP address) denoted by $\mathrm{id}(u)$ of size $O(\log n)$, where n is the minimal number of nodes in any given round. The edges are referred to as the logical links between the nodes in the network. For all $u, v \in V_t$, there exists a directed edge (u, v) in the network if the node u knows $\mathrm{id}(v)$. A node $u \in V_t$ can send a message to a node $v \in V_t$ if and only if it knows the identifier of v, i.e., $\mathrm{id}(v)$. Let us denote by D_t^u the neighborhood of node u in round t, i.e.,

$$D_t^u := \{v \in V_t \mid u \text{ knows } \mathrm{id}(v) \text{ in round } t\},$$

then $(D_t^u)_{u \in V_t}$ completely defines the logical links and therefore, the structure of the overlay network in round t. Exchange of messages in each round results in a series of directed communication graphs $\mathcal{G} := \big(G_0, G_1, \dots\big)$ with $G_t = (V_t, E_t)$ and

$$E_t := \{(u, v) \mid v \text{ receives a message from } u \text{ in round } t\}.$$

A node can send messages to $O(\log n)$ different nodes in each round with message length at most $O(\mathrm{polylog}\, n)$ bits. The model assumes an (a, b)-late adversary, i.e., the adversary can see the edges E_t in round $t + a$ and the set $(D_t^u)_{u \in V_t}$ in round $t + b$.

We consider an overlay network based on a topology that has a well defined structure called a Linearized DeBruijn Swarm (see, Figure 7.1).

Definition 1.16. For $n \in \mathbb{N}_{\geq 0}$ and $\kappa \geq 1$, let V be a set of nodes with $|V| \geq n$ positioned on the $[0, 1)$-interval and $\lambda := 2\log(\kappa n)$. Then, the *Linearized DeBruijn Swarm* $D := (V, E_L \cup E_{DB})$ with parameter $c \in \mathbb{N}_{\geq 0}$ has the following properties:

- $(v, w) \in E_L \iff w \in V$ and $d(v, w) \leq \frac{2c\lambda}{n}$.
- $(v, w) \in E_{DB} \iff w \in V$ and $d\left(\frac{v+i}{2}, w\right) \leq \frac{3c\lambda}{2n}$ with $i \in \{0, 1\}$,

where $d : V \times V \mapsto \mathbb{R}_{\geq 0}$ is the distance function.

1.6 Notations

- For $n \in \mathbb{Z}_+$, set $[n] = \{1, \dots, n\}$.
- For any $m, n \in \mathbb{Z}$ with $n \geq m$, set $[n]_m = \{m, m+1, \dots, n\}$.
- For a resource $e \in E$ and a state $s \in S$, $n_e(s'_u, s_{-u})$ denotes the number of players on resource e, when player u plays strategy s'_u, while all the other players play their strategy in s.
- For $a, b \in [0, 1)$ and $b \leq a$, $[a \pm b]$ denotes the interval $[a - b, a + b]$.

1.7 Complexity

Complexity theory distinguishes between optimization and decision problems. While optimization (minimization or maximization) problems such as *The Traveling Salesman Problem* [Wor86] ask for the optimal solution to a given instance of the problem, decision problems such as *3-Dimensional Matching* [Kar72] are posed as a YES or NO question. A crucial factor to consider while designing algorithms for non-trivial problems is their runtime efficiency. It is commonly accepted that an algorithm with a runtime which is polynomially bounded in its input size is efficient. We assume that the readers are familiar with the standard complexity classes P and NP. It is a widely believed hypothesis that P $\neq$ NP, i.e., there exist problems in NP that do not exhibit polynomial time algorithms. The complexity class NP consists of all decision problems for which there exists a polynomial time algorithm that can verify, given an instance of the decision problem and a polynomial sized certificate, whether the instance is a YES instance. A decision problem $X \in$ NP is NP-complete if every problem $X' \in$ NP can be transformed to an instance of X in polynomial time. An optimization problem can be formulated as a decision problem by imposing a bound on the value to be optimized. This allows us to study the hardness of an optimization problem by applying the theory of NP-completeness on its decision variant. Therefore, in order to prove if an optimization problem is hard, it is sufficient to show that its decision variant is hard. A problem X is NP-hard if every problem $X' \in$ NP can be transformed to an instance of X in polynomial time. We refer the readers to Garey and Johnson [GJ79], Cook [Coo00], and Karp [Kar72] for a comprehensive introduction to complexity theory and the notion of NP-completeness.

A natural approach to deal with intractability of NP-hard optimization problems is to investigate the existence of polynomial time approximation algorithms, i.e., efficient algorithms to compute good solutions that approximate the optimal solution closely. We refer the readers to [ACG$^+$13] for further details on approximability of NP-hard problems. An approximation algorithm for a minimization problem X has a performance guarantee of α, if and only if the algorithm outputs a feasible solution for each instance $I \in X$ with cost at most α times the cost of the optimal solution for I, i.e., $\mathit{cost}(I) \leq \alpha \cdot \mathit{OPT}(I)$. The complexity class APX is the class of all optimization problems with a constant factor approximation algorithm, i.e., $\alpha = O(1)$. An elegant description of the class APX can also be found in Hoogeveen et al. [HSW01] and Kann [Kan92]. It is desirable to derive approximation algorithms for optimization problems with α arbitrarily close to 1, if possible. A widely used approach for this is the *polynomial time approximation scheme* (PTAS). A PTAS for an optimization problem is a family of polynomial time approximation algorithms with an approximation guarantee of $(1 + \varepsilon)$ for every fixed $\varepsilon > 0$. Similarly, a *quasi-polynomial time approximation scheme* (QPTAS) for an input of size n has runtime

$n^{\log^{O(1)} n}$ for each fixed $\varepsilon > 0$. A natural question that arises is whether such approximation algorithms always exist for NP-hard problems. Papadimitriou and Yannakakis [PY91] introduce the concept of L-reduction and prove that if there exists an L-reduction from an optimization problem X to another optimization problem Y with parameters $\alpha, \beta > 0$ and if problem X has some polynomial time approximation scheme for every fixed $\varepsilon > 0$, this implies that there also exists a polynomial time algorithm for problems in Y with approximation guarantee of $1 + \alpha\beta\varepsilon$ for every fixed $\varepsilon > 0$.

Definition 1.17 (Papadimitriou and Yannakakis [PY91])**.** Let Π_1 and Π_2 be two optimization problems. We say Π_1 *L-reduces* to Π_2 if there exist polynomial time computable functions f, g and constants $\alpha, \beta > 0$ such that for each instance $I \in \Pi_1$ the following holds,

1. $f(I) \in \Pi_2$ such that, $OPT(f(I)) \leq \alpha \cdot OPT(I)$.
2. Given any solution φ to $f(I)$, $g(\varphi)$ is a feasible solution to I such that

$$|cost(g(\varphi)) - OPT(I)| \leq \beta \cdot |cost(\varphi) - OPT(f(I))|.$$

The class APX is closed under approximation preserving reductions such as an L-reduction. Therefore, APX-complete problems are the hardest of problems in the class APX and APX-hard problems are those which are at least as hard as all problems in APX. APX-complete problems do no exhibit a polynomial time approximation scheme [ALM+98]. It is widely believed that showing that a problem is APX-hard also implies that the problem does not exhibit a QPTAS [HPQ17].

1.8 Outline and Bibliographical Notes

The thesis has been divided into two parts. In the first part we study different models of non-cooperative games. Particularly, we focus on their algorithmic aspects such as computability, existence and the inefficiency of their self emerging solutions. The second part focuses on decentralized overlay networks. Here, we present an algorithm that maintains a routable overlay network under high adversarial churn. We would like to remark that due to the diversity in the models we consider, we deliberately refrain from discussing details of the related work at this point and refer the readers to the individual chapters. Also, we intentionally repeat the description and preliminaries of the relevant model in the individual chapters to allow for a chapter to be read separately. Here, we give a brief overview about the results presented in the subsequent chapters of the thesis. We would like to remark that most of the results presented in this thesis are part of peer-reviewed manuscripts published as journal articles or as extended abstracts in conference proceedings. In most cases the proofs are also available as arXiv preprint.

Approximate pure Nash equilibria

We study the computability of pure Nash equilibria in congestion games. Congestion games belong to a class of local search problems in the complexity class PLS. We refer the readers to Johnson, Papadimitriou, and Yannakakis [JPY88] and [NRTV07, Chapter 19] for an overview on the complexity class PLS. It has been established that computing an exact [FPT04] or even an approximate [SV08] pure Nash equilibrium in congestion games is in general PLS-complete. This alludes our attention towards computing approximate pure Nash equilibrium in certain restricted classes of congestion games. Particularly, we consider congestion games with non-decreasing cost functions. In their seminal paper, Caragiannis et al. [CFGS11] present a polynomial-time algorithm that computes a $(2+\varepsilon)$-approximate pure Nash equilibrium for games with linear cost functions and further results for polynomial cost functions. In Chapter 2 we show that this factor can be improved to $(1.61+\varepsilon)$ and also show further improved results for polynomial cost functions, by a seemingly simple modification to their algorithm that allows for the cost functions used during the best response dynamics to be different from the overall objective function. Interestingly, our modification to the algorithm also extends to efficiently computing improved approximate pure Nash equilibria in games with arbitrary non-decreasing resource cost functions.

Universal Load Dependent Taxes

In Chapter 3 we extend the techniques presented in Chapter 2 and show that our analysis exhibits an interesting method to compute universal load dependent taxes that improves the inefficiency of equilibria in congestion games. Furthermore, using linear programming duality we prove lower bounds on the price of anarchy under universal taxation, e.g., 2.012 for linear congestion games and further results for polynomial cost functions. Moreover, our cost functions are locally computable and in contrast to [BV16] are independent of the actual instance of the game.

The results in Chapters 2 and 3 are joint work of the author with Alexander Skopalik and are part of the extended abstract of the following conference proceedings and arXiv preprint:

- [RVS20] **Improving Approximate Pure Nash Equilibria in Congestion Games**
  Vipin Ravindran Vijayalakshmi and Alexander Skopalik
  In the proceedings of the 16th Conference on Web and Internet Economics (WINE), December 2020, pages 280-294.

- [SR20] **Improving approximate pure Nash equilibria in congestion games**
  Vipin Ravindran Vijayalakshmi and Alexander Skopalik
  arXiv preprint 2007.15520 (2020).

Fixed Order Scheduling

In Chapter 4 we study a simple, yet challenging variant of a parallel machine scheduling problem. Here, we consider a variant of a scheduling problem in which each machine has its own priority over the sequence in which jobs assigned to it are processed. Priority based scheduling succinctly models many socio-economic situations and therefore, duly warrant analysis from a theoretical perspective. We study the computational hardness and inapproximability of the optimal solution in this model under local and global priority rules. To the best of our knowledge, the model has only been considered marginally in the literature. The unpublished results presented in Chapter 4 are joint work of the author with Marc Schröder (*School of Business and Economics, Maastricht University, Netherlands*) and Tami Tamir (*School of Computer Science, The Interdisciplinary Center, Israel*).

- [RST21b] **Scheduling with machine-dependent priority lists**
  Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir
  Unpublished.

Non-cooperative Fixed Order Scheduling

In Chapter 5 we analyze the model described in Chapter 4 in a non-cooperative setting. Here, we consider a scheduling game on parallel related machines, in which jobs try to minimize their completion time by choosing a machine to be processed on. Inline with the model, each machine has an individual priority list to decide on the order in which the jobs assigned on the machine are to be processed. We prove that it is NP-hard to decide if a pure Nash equilibrium exists and characterize four classes of instances in which a pure Nash equilibrium is guaranteed to exist. For each of these classes, we present an algorithm that computes a Nash equilibrium, we prove that best-response dynamics converge to a Nash equilibrium, and we bound the inefficiency of Nash equilibria with respect to objectives such as makespan of the schedule and the sum of completion times. Additionally, we prove that although a pure Nash equilibrium is guaranteed to exist in instances with identical machines, it is NP-hard to approximate the best Nash equilibrium with respect to both objectives.

Weighted Congestion Games with Scheduling Policy

Chapter 6 generalizes the model discussed in Chapter 5 to allow for arbitrary strategy sets and studies weighted congestion games with priority lists. We show that in general, even with unit weight players, a pure Nash equilibrium need not exist by making use of the famous Condorcet paradox [MdC85]. We use this construction to prove that even with unweighted players, it is NP-hard to decide whether a pure Nash equilibrium exists. We also prove a certain inapproximability result for the notion of approximate pure Nash equilibrium in these games. On a positive note, for matroid congestion games with unweighted players, we show that a pure Nash equilibrium always exists. Finally, we analyze the price of anarchy with respect to the sum of weighted costs and show that the upper bound of 4 proven by Cole et al. [CCG$^+$15] for unrelated machine scheduling with Smith's rule also extends to congestion games with resource-dependent priority lists. This ratio is smaller than the price of anarchy of the atomic game with priorities defined by Farzad et al. [FOV08]. Furthermore, we extend our analysis to games with polynomial cost functions of maximum degree d.

The results presented in Chapters 5 and 6 are joint work of the author with Marc Schröder and Tami Tamir. Parts of the results presented in Chapter 5 and 6 can also be found in the extended abstracts of the following conference proceedings and journal publication:

- [RST21a] **Scheduling games with machine-dependent priority lists**
  Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir
  In Theoretical Computer Science, Volume 855, 2021, pages 90-103.
- [STRV19] **Scheduling Games with Machine-Dependent Priority Lists**
  Marc Schröder, Tami Tamir, and Vipin Ravindran Vijayalakshmi
  In the proceedings of the 15th Conference on Web and Internet Economics (WINE), December 2019, pages 286-300.

Parts of the results in Chapter 6 can be found in the following arXiv preprint:

- [STR19] **Scheduling (Congestion) Games with Machine-Dependent Priority Lists**
  Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir
  arXiv 1909.10199 preprint (2019).

Churn Resistant Overlay Networks

In Chapter 7 we investigate the maintenance of a structured overlay network under high adversarial churn, where an adversary may churn a constant fraction of nodes in the network over the course of $O(\log n)$ rounds. In particular, the adversary has almost up-to-date information on the network topology as it can only observe a slightly outdated topology that is at least 2 rounds old with a provably minimal restriction that new nodes can only join the network via nodes that have taken part in the network for at least two rounds. We present an algorithm based on four sub-routines that constructs a new overlay—completely independent of all previous overlays—every 2 rounds with congestion $O(\log^3 n)$ messages each round. We extend a topology based on Linearized DeBruijn Graph [RS11a] and present a structured overlay network called Linearized DeBruijn Swarm (LDS), a highly churn resistant overlay. We believe that our techniques can be adapted to a variety of classical P2P topologies, where nodes are mapped into the $[0, 1)$-interval. The results presented in Chapter 7 are joint work of the author with Thorsten Götte and Christian Scheideler. Preliminary results in Chapter 7 were presented in the following conference proceedings:

- [GRS19] **Always be Two Steps Ahead of Your Enemy**
  Thorsten Götte, Vipin Ravindran Vijayalakshmi, and Christian Scheideler
  In the proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2019, pages 1073-1082.

Parts of the results in Chapter 7 can also be found in the following arXiv preprint:

- [GRS18] **Always be Two Steps Ahead of Your Enemy**
  Thorsten Götte, Vipin Ravindran Vijayalakshmi, and Christian Scheideler
  arXiv preprint 1810.07077 (2018).

Finally, in Chapter 8 we briefly summarize our results and discuss future research directions.

Part I

Decentralized Resource Allocation

Chapter 2

Approximate Pure Nash Equilibria

Congestion games are a prominent class of games to model resource allocation by non-cooperative users. Fabrikant et al. [FPT04] show that computing a pure Nash equilibrium in congestion games is in general PLS-complete. This naturally draws attention to studying a weaker notion of the equilibrium known as an approximate pure Nash equilibrium. Skopalik and Vöcking [SV08] show that computing an approximate pure Nash equilibrium in general is also PLS-complete. In this chapter we consider a restricted class of congestion games, i.e., class of games with arbitrary non-decreasing cost functions and present an algorithm that computes a constant factor approximate pure Nash equilibrium in polynomial number of player best response moves.

2.1 Introduction

We introduced congestion games in Chapter 1. Let us recall that congestion games constitute an important class of games that succinctly represent a game theoretic model for resource allocation among non-cooperative strategic users. A canonical example for this is the road transportation network, where the time needed to commute is a function on the total amount of traffic in the network (see, e.g., [U.S64]). A rational user picks the shortest available route to their destination. Therefore, it is quite natural to model the strategic nature of users in a transportation network as a strategic game. As described in Section 1.1, a strategic game denoted by the tuple $\big(\mathcal{N}, (S_u)_{u\in\mathcal{N}}, (C_u)_{u\in\mathcal{N}}\big)$ consists of a finite set of players $\mathcal{N}$ and for each player $u \in \mathcal{N}$, a finite set of strategies S_u and a cost function $C_u : S \to \mathbb{R}_{\geq 0}$ mapping a state $s = (s_u)_{u\in\mathcal{N}} \in \times_{u\in\mathcal{N}} S_u$ to the cost of player $u \in \mathcal{N}$. A congestion game is a strategic game that succinctly represents a decentralized resource allocation problem involving selfish users. In particular, a congestion game is a cost minimization game defined by a set of resources E, a set of players $\mathcal{N} := \{1, \ldots, N\}$ with strategies $S_1, \ldots, S_N \subseteq 2^E$, and for each resource $e \in E$, a cost function $f_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$. Congestion games were first introduced by Rosenthal [Ros73a] and using a potential function argument proved that it belongs to a class of games in which a pure Nash equilibrium always exists, i.e., the game always consists of a self-emerging solution in which no player is able to improve by unilaterally deviating.

In this chapter, we study the computational aspects of pure Nash equilibria in congestion games. We begin by familiarizing ourselves once again with the model definition and the necessary preliminaries. However, we remark that the readers may choose to skip the following subsection.

2.1.1 The Model

A *congestion game* denoted by the tuple

$$G = \left(\mathcal{N}, E, (S_u)_{u\in\mathcal{N}}, (f_e)_{e\in E}\right)$$

consists of a set of N players, $\mathcal{N} = \{1, 2, \dots, N\}$, who compete over a set of resources, $E = \{e_1, e_2, \dots, e_m\}$. Each player $u \in \mathcal{N}$ has a set of strategies denoted by $S_u \subseteq 2^E$. Each resource $e \in E$ has a non-negative and non-decreasing cost function $f_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ associated with it. Let us denote by $n_e(s)$ the number of players on a resource $e \in E$ in the state s, then the cost contributed by a resource $e \in E$ to each player using it is denoted by $f_e(n_e(s))$. Therefore, the cost of a player $u \in \mathcal{N}$ in a state $s = (s_1, \dots, s_N) \in S_1 \times \cdots \times S_N$ of the game is given by

$$C_u(s) = \sum_{e\in E: e\in s_u} f_e(n_e(s)).$$

For an arbitrary state $s \in S$, $C_u(s'_u, s_{-u})$ denotes the cost of player u, when only player u deviates from the state s. A best-response move denoted by $\mathcal{BR}_u(s)$ is a move that minimizes a player's cost while all the other players are fixed to their strategy in s. With a slight abuse of notation, let $\mathcal{BR}_u(0)$ denote the best response of a player u assuming that no other player participates in the game.

A state $s \in S$ is a pure Nash equilibrium (NE), if there exists no player who could deviate to another strategy and decrease their cost, i.e., for all $u \in \mathcal{N}$ and for all $s'_u \in S_u$,

$$C_u(s) \leq C_u(s'_u, s_{-u}).$$

A weaker notion of an NE is the α-approximate pure Nash equilibrium for $\alpha > 1$, which is a state s in which no player has an improvement that decreases their cost by a factor of at least α, i.e., for all $u \in \mathcal{N}$ and for all $s'_u \in S_u$,

$$\alpha \cdot C_u(s'_u, s_{-u}) \geq C_u(s).$$

For congestion games the exact potential function

$$\Phi(s) = \sum_{e\in E} \Phi_e(n_e(s)) = \sum_{e\in E} \sum_{i=1}^{n_e(s)} f_e(i),$$

guarantees the existence of an NE by proving that every sequence of unilateral improving deviations converges to an NE. We denote the social or global cost of a state s as

$$C(s) = \sum_{u\in\mathcal{N}} C_u(s)$$

and the state that minimizes the social cost is called the optimal, i.e., $s^* = \arg\min_{s\in S} C(s)$. The inefficiency of equilibria is measured using the price of anarchy (PoA) [KP99], which is the worst-case ratio between the social cost of an equilibrium and the social optimum.

A local optimum is a state s in which there is no player $u \in \mathcal{N}$ with an alternative strategy s'_u such that, $C_u(s'_u, s_{-u}) < C_u(s)$ and an α-approximate local optimum is a state s in which there is no player u who has an α-best response move (in short α-move) with a strategy s'_u such that $\alpha \cdot C_u(s'_u, s_{-u}) < C_u(s)$. Let us remark that there is an interesting connection between an NE and a local optimum. A NE is a local optimum of the potential function Φ and similarly, a local optimum is a pure Nash equilibrium of a game in which we change the resource cost functions from $f(x)$ to the marginal contribution to the social cost, e.g., to $f'(x) = xf(x) - (x-1)f(x-1)$. Analogous to the PoA, the *stretch* of a congestion game is the worst-case ratio between the value of the potential function at an equilibrium and the potential minimizer [CFGS11, CFGS12].

2.1.2 Our Contribution

Caragiannis et al. [CFGS11] present a polynomial-time algorithm that computes a $(2+\varepsilon)$-approximate pure Nash equilibrium for congestion games with linear cost functions and further constant factor results for polynomial cost functions. We improve the approximation guarantee achieved in the computation of approximate pure Nash equilibria with the algorithm in Caragiannis et al. [CFGS11], using a linear programming approach which generalizes the smoothness condition in Roughgarden [Rou15], to modify the cost functions that users experience in the algorithm. We show the approximation factor can be improved to $(1.61 + \varepsilon)$ for games with linear cost functions and further improved results for polynomial cost functions, with a seemingly simple modification to their algorithm by allowing for the cost functions used during the best response dynamics be different from the overall objective function. Interestingly, our modification to the algorithm also extends to efficiently computing improved approximate pure Nash equilibrium in games with arbitrary non-decreasing resource cost functions. In Section 2.3 we present an adaptation of the algorithm in Caragiannis et al. [CFGS11]. Although we only make a seemingly simple modification to their algorithm, we would like to remark that the analysis is significantly involved and does not follow immediately from [CFGS11], since the subgame induced by the algorithm with the modified costs is not a potential game anymore. Table 2.1 lists our results for resource cost functions that are bounded degree polynomials of maximum degree d. The values for ρ_d in Table 2.1 were obtained using the python program in Listing A.1. In [Vij17] the author computes cost functions using a linear programming based approach that improves the stretch in congestion games. We remark that although the bounds on stretch in [Vij17] are very similar to ours, the cost functions computed using their LP doesn't provably guarantee to satisfy the necessary conditions required for the proofs of our algorithm. Our algorithm strongly relies on the fact that the cost functions satisfy the strong smoothness condition described in Section 2.2.

d	Previous Approx. [CFGS11, FGS14]	**Our Approx.** $\rho_d + \varepsilon$
1	$2 + \varepsilon$	$1.61 + \varepsilon$
2	$6 + \varepsilon$	$3.35 + \varepsilon$
3	$20 + \varepsilon$	$8.60 + \varepsilon$
4	$111 + \varepsilon$	$27.46 + \varepsilon$
5	$571 + \varepsilon$	$98.14 + \varepsilon$

Table 2.1: Approximate pure Nash equilibria of congestion games with polynomial cost functions of degree at most d.

Our main result is presented as Theorem 2.1, where the factor ρ_d is listed in Table 2.1.

Theorem 2.1. *For every $\varepsilon > 0$, the algorithm computes a $(\rho_d + \varepsilon)$-approximate pure Nash equilibrium for every congestion game with non-decreasing cost functions that are polynomials of maximum degree d in a number of steps which is polynomial in the number of players, ρ_d and $1/\varepsilon$.*

2.1.3 Related Work

Congestion games fit into the framework of local search problems. Fabrikant et al. [FPT04] show that computing a pure Nash equilibrium in both symmetric and asymmetric congestion games is PLS-complete [JPY88]. They show that regardless of the order in which local search is performed, there are initial states from where it could take exponential number of the steps before the game converges to a pure Nash equilibrium. Also, they show PLS-completeness for network congestion games with asymmetric strategy spaces. As a positive result, Fabrikant et al. [FPT04] present a polynomial time algorithm to compute pure Nash equilibria in certain restricted strategy spaces e.g. symmetric network congestion games. Ackermann et al. [ARV08] show that network congestion games with linear cost functions are PLS-complete. However, if the set of strategies of each player consists of the bases of a matroid over the set of resources, then they show that the lengths of all best response sequences are polynomially bounded in the number of players and resources.

To our knowledge, the concept of α-approximate pure Nash equilibria was introduced by Roughgarden and Tardos [RT00] in the context of non-atomic selfish routing games. An α-approximate pure Nash equilibrium is a state in which none of the users can unilaterally deviate to improve by a factor of at least α. Orlin et al. [OPS04] show that every local search problem in PLS admits a fully polynomial time ε-approximation scheme. Although their approach can be applied to congestion games, this does not yield an approximate pure Nash equilibrium, but rather only an approximate local optimum of the potential function. In case of congestion games, Skopalik and Vöcking [SV08] show that in general for arbitrary cost functions, finding an α-approximate pure Nash equilibrium is PLS-complete, for any $\alpha > 1$. However, for polynomial cost function (with non-negative coefficients) of maximum degree d, Caragiannis et al. [CFGS11] present an approximation algorithm. They present a polynomial time algorithm that computes a $(2 + \varepsilon)$-approximate pure Nash equilibrium for games with linear cost functions and an approximation guarantee of $d^{O(d)}$ for polynomial cost functions of maximum degree d. Interestingly, they use the convergence of a carefully chosen subsets of players to a $(1 + \varepsilon)$-approximate pure Nash equilibrium (of that subset) as a subroutine to generate a state which is an approximation of the minimal potential function value (of that subset), e.g., $2 \cdot OPT$ for linear congestion games. This approximation factor of the minimal potential then essentially turns into the approximation factor of the approximate pure Nash equilibrium. Feldotto et al. [FGS14] using a *path-cycle decomposition* technique bound this approximation factor of the potential for arbitrary non-decreasing cost functions.

2.2 Definition and Preliminaries

We begin by familiarizing ourselves with the notion of (λ, μ)-smoothness in cost-minimization games. After a long series of papers in which various authors (e.g. [CK05a, AAE05, ADG$^+$11]) show upper bounds on the price of anarchy of congestion games, Roughgarden exhibited that most of them essentially used the same technique, which is formalized as (λ, μ)-smoothness [Rou15]. A cost-minimization game is called (λ, μ)-smooth, if for every pair of outcomes $s, s^* \in S$ it holds that,

$$\sum_{u \in \mathcal{N}} C_u(s_u^*, s_{-u}) \leq \lambda \cdot C(s^*) + \mu \cdot C(s). \tag{2.1}$$

The price of anarchy of a (λ, μ)-smooth game with $\lambda > 0$ and $\mu < 1$ is then at most $\frac{\lambda}{1-\mu}$.

2.2.1 Revisiting (λ, μ)-smoothness

Observe that the original smoothness definition in (2.1) can be extended to allow for an arbitrary objective function $h : S \mapsto \mathbb{R}_{\geq 0}$ instead of the social cost function

$$C(s) = \sum_{u \in \mathcal{N}} C_u(s).$$

Definition 2.2. A cost-minimization game is (λ, μ)-*smooth* with respect to an objective function $h : S \mapsto \mathbb{R}_{\geq 0}$, if for every pair of outcome s, $s^* \in S$,

$$\lambda \cdot h(s^*) \geq \sum_{u \in \mathcal{N}} C_u(s_u^*, s_{-u}) - \sum_{u \in \mathcal{N}} C_u(s) + (1 - \mu) h(s).$$

From the definition above, we restate the central smoothness theorem [Rou15].

Theorem 2.3. *Given a (λ, μ)-smooth cost-minimization game G with $\lambda > 0$, $\mu < 1$ and an objective function $h : S \mapsto \mathbb{R}_{\geq 0}$, then for every pure Nash equilibrium s and the global optimum s^*,*

$$h(s) \leq \frac{\lambda}{1 - \mu} h(s^*).$$

Proof. Let s be an arbitrary pure Nash equilibrium and s^* be the optimal solution. From the Nash inequality we know that $\forall u \in \mathcal{N}$,

$$C_u(s) \leq C_u(s_u^*, s_{-u}).$$

Then, summing over all the N players gives,

$$\sum_{u \in \mathcal{N}} C_u(s) - \sum_{u \in \mathcal{N}} C_u(s_u^*, s_{-u}) \leq 0.$$

By the definition of (λ, μ) smoothness in Definition 2.2 we have,

$$(1 - \mu) \cdot h(s) \leq \lambda \cdot h(s^*) + \sum_{u \in \mathcal{N}} C_u(s) - \sum_{u \in \mathcal{N}} C_u(s_u^*, s_{-u}).$$

Then, using the Nash inequality the theorem follows. □

The smoothness framework introduced by Roughgarden [Rou15] also extends to equilibrium concepts such as mixed Nash and (coarse) correlated equilibria. The same is true for our variant with respect to an arbitrary objective function h. For the sake of completeness we rework Roughgarden's proof here.

Theorem 2.4 (Extension Theorem)**.** *For every (λ, μ)-smooth cost-minimization game G with respect to an arbitrary objective function $h : S \mapsto \mathbb{R}_{\geq 0}$, every coarse correlated equilibrium σ and every outcome s^*,*

$$\mathbf{E}_{s \sim \sigma}[h(s)] \leq \frac{\lambda}{1 - \mu} \cdot h(s^*).$$

Proof. The proof is analogous to [Rou15]. From the (λ, μ)-smoothness condition in Definition 2.2, we have,

$$\mathbf{E}_{s \sim \sigma}[h(s)] \leq \frac{1}{1 - \mu} \mathbf{E}_{s \sim \sigma} \left[\lambda \cdot h(s^*) + \sum_{u \in \mathcal{N}} C_u(s) - \sum_{u \in \mathcal{N}} C_u(s_u^*, s_{-u}) \right]$$

$$= \frac{1}{1-\mu}\left[\lambda \cdot h(s^*) + \sum_{u \in \mathcal{N}} \mathbf{E}_{s\sim\sigma}[C_u(s)] - \sum_{u \in \mathcal{N}} \mathbf{E}_{s\sim\sigma}[C_u(s_u^*, s_{-u})]\right]$$

from the definition of CCE,

$$\leq \frac{\lambda}{1-\mu} \cdot h(s^*).$$

Hence, the theorem. □

We remark that a variant to our extension of Roughgarden's smoothness framework is independently introduced as *generalized smoothness* in [CPM19]. From Definition 2.2 we note the following observation.

Observation 2.5. *Every (λ, μ)-smooth cost-minimization game G with $\lambda > 0$ and $\mu < 1$ is also $\left(\frac{\lambda}{1-\mu}, 0\right)$-smooth with its cost functions scaled by a factor $\frac{1}{1-\mu}$.*

Given a strategic game $G = \left(\mathcal{N}, (S_u)_{u\in\mathcal{N}}, (C_u)_{u\in\mathcal{N}}\right)$, one can determine λ and μ that satisfies the smoothness condition in Definition 2.2, for all pairs of solution s and s^*. However, since the state space S grows exponentially in the number of players, this would be computationally inefficient. Therefore, we typically have to work with games in which the players' costs and the objective function h can be represented in a succinct way. Recall that in congestion games, the players cost and the global objective function are implicitly defined by the resource cost function. In the following, we allow for an arbitrary, additive objective function $h(s)$, i.e., of the form

$$h(s) = \sum_{e\in E} h_e(n_e(s))$$

where the function $h_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ is non-decreasing for all $e \in E$.

We study games in which we change the cost functions C_u experienced by the players. As a consequence of Observation 2.5 and scaling the cost functions appropriately, we can always ensure that we satisfy the smoothness inequality with $\mu = 0$, to conveniently restate the smoothness condition as follows.

Lemma 2.6. *A congestion game is $(\lambda, 0)$-smooth with respect to an objective function $h(s) = \sum_{e\in E} h_e(n_e(s))$, if for every non-decreasing cost function $f'_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ and for every $0 \leq n, m \leq N$,*

$$\lambda \cdot h_e(m) \geq m f'_e(n+1) - n f'_e(n) + h_e(n).$$

Proof. Let s and s^* be arbitrary solutions. Summing the inequality of the lemma with $m = n_e(s^*)$ and $n = n_e(s)$ for all $e \in E$ gives,

$$\lambda \sum_{e\in E} h_e(n_e(s^*)) \geq \sum_{e\in E} n_e(s^*) f'_e(n_e(s)+1) - \sum_{e\in E} n_e(s) f'_e(n_e(s)) + \sum_{e\in E} h_e(n_e(s))$$
$$\lambda \cdot h(s^*) \geq \sum_{u\in\mathcal{N}} C'_u(s_u^*, s_{-u}) - \sum_{u\in\mathcal{N}} C'_u(s) + h(s)$$

which is the $(\lambda, 0)$-smoothness condition of Definition 2.2. □

From now on, we use $f' = (f'_e)_{e\in E}$ whenever we refer to the modified cost functions and denote the modified player cost by $C'_u(s) = \sum_{e\in s_u} f'_e(n_e(s))$.

Strong smoothness

In Section 2.3 we present an algorithm to compute an approximate pure Nash equilibrium in congestion games with an improved approximation guarantee than that in Caragiannis et al. [CFGS11]. The analysis of the algorithm uses the potential function argument on a subset of players $F \subseteq \mathcal{N}$. In particular, it needs the property that the subgame induced by every subset of players from $\mathcal{N}$ is $(\lambda, 0)$-smooth. Unfortunately, Lemma 2.6 does not guarantee this property. Therefore, we define a stronger notion of $(\lambda, 0)$-smoothness that guarantees that the smoothness condition also holds for the subgame induced by every arbitrary subset of players in $\mathcal{N}$. Let us denote by $n_e^F(s)$ the number of players in F that use the resource e in the state s.

Definition 2.7. A strategic game is *strongly* $(\lambda, 0)$*-smooth* with respect to an objective function $h : S \mapsto \mathbb{R}_{\geq 0}$ and for some $\lambda > 0$, if for every subset $F \subseteq \mathcal{N}$ and for every $s, s^* \in S$,

$$\lambda \cdot h_F(s^*) \geq \sum_{u \in F} C'_u(s^*_u, s_{-u}) - \sum_{u \in F} C'_u(s) + h_F(s),$$

where $h_F(s) := \sum_{e \in E} h_e\left(n_e(s)\right) - h_e\left(n_e^{\mathcal{N} \setminus F}(s)\right)$.

Now consider an arbitrary subset of players $F \subseteq \mathcal{N}$ and a state $s \in S$. Let us define the potential of this subset as the potential in the subgame induced by these players in s, i.e.,

$$\Phi_F(s) := \sum_{e \in E} \Phi_{F,e}(s) = \sum_{e \in E} \sum_{i=1}^{n_e^F(s)} f_e\left(i + n_e^{\mathcal{N} \setminus F}(s)\right).$$

Denote by $G_s^F := (F, E, (S_u)_{u \in F}, (f_e^F)_{e \in E})$ the subgame induced by freezing the remaining players from $\mathcal{N} \setminus F$ with $f_e^F(x) := f_e(x + n_e^{\mathcal{N} \setminus F}(s))$, where $n_e^{\mathcal{N} \setminus F}(s)$ is the number of players outside of F on resource e in the state s. Then, the following lemma gives a stronger notion of the $(\lambda, 0)$-smoothness condition.

Lemma 2.8. *For every congestion game G with non-decreasing cost functions $f'_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$, which is $(\lambda, 0)$-smooth with respect to the potential function Φ_e for every subgame G_s^F induced by an arbitrary subset $F \subseteq \mathcal{N}$ and arbitrary states $s, s^* \in S$, i.e.,*

$$\lambda \cdot \Phi_{F,e}(s^*) - n_e^F(s^*) \cdot f'_e(n_e(s) + 1) + n_e^F(s) \cdot f'_e(n_e(s)) \geq \Phi_{F,e}(s),$$

is also strongly $(\lambda, 0)$-smooth.

Proof. The proof is analogous to the proof of Lemma 2.6. Consider an arbitrary subset of players $F \subseteq \mathcal{N}$, an arbitrary state s, and a subgame $G_s^F := (F, E, (S_u)_{u \in F}, (f_e^F)_{e \in E})$ induced by freezing the remaining players from $\mathcal{N} \setminus F$, that is, let $f_e^F(x) := f_e(x + n_e^{\mathcal{N} \setminus F}(s))$ where $n_e^{\mathcal{N} \setminus F}(s)$ is the number of players outside of F on resource e in the state s. Let s^* be an arbitrary solution. Summing the inequality of the lemma with $m = n_e^F(s^*)$ and $n = n_e^F(s)$ for all $e \in E$ gives,

$$\lambda \sum_{e \in E} \Phi_{F,e}\left(s^*\right) \geq \sum_{e \in E} n_e^F(s^*) \cdot f'_e(n_e(s) + 1) - \sum_{e \in E} n_e^F(s) \cdot f'_e(n_e(s)) + \sum_{e \in E} \Phi_{F,e}(s)$$

equivalent to,

$$\lambda \cdot \Phi_F(s^*) \geq \sum_{u \in F} C'_u(s^*_u, s_{-u}) - \sum_{u \in F} C'_u(s) + \Phi_F(s),$$

which is the strong $(\lambda, 0)$-smoothness condition of Definition 2.7. □

This subset property is of particular importance for the algorithm we present in Section 2.3 to compute an approximate pure Nash equilibrium, but may be of independent interest as well. We are not aware of other approximation algorithms that can guarantee this property. We would like to remark that all references to $(\lambda, 0)$-smoothness in Section 2.3 imply strong $(\lambda, 0)$-smoothness.

2.3 Approximate Equilibria in Congestion Games

In this section we aim at improving the approximation factor of approximate pure Nash equilibria in congestion games with arbitrary non-decreasing resource cost functions. We extend an algorithm based on Caragiannis et al. [CFGS11] to compute an approximate pure Nash equilibrium in congestion games with arbitrary non-decreasing resource cost functions. A key element of this algorithm is the so called stretch of a (sub-) game. Recall that the stretch is the worst-case ratio of the potential function at an equilibrium and the global minimum of the potential.

The algorithm generates a sequence of improving moves that converges to an approximate pure Nash equilibrium in polynomial number of best-response moves. The idea is to divide the players into blocks based on their costs and hence, their prospective ability to drop the potential of the game. In each phase of the algorithm, players of two consecutive blocks are scheduled to make improving moves starting with the blocks of players with high costs. One block only makes q-moves, which are improvements by a factor of at least q which is close to 1. The other block does p-moves, where p is slightly larger than the stretch of a q-approximate equilibrium and slightly smaller than the final approximation factor.

The key idea here is that blocks first converge to a q-approximate equilibrium and thereby generate a state with a stretch of approximately p. Later, when players of a block are allowed to do p-moves, there is not much potential left to move. In particular, there is no significant influence on players of blocks that moved earlier possible. This finally results in the approximation factor of roughly p. We modify the algorithm in [CFGS11] by changing the costs seen by the players during their q-moves to be a set of modified cost functions $(f'_e)_{e\in E}$, satisfying smoothness condition of Lemma 2.8 for some constant $\lambda > 0$, and this results in a $\lambda(1+\varepsilon)$-approximate pure Nash equilibrium. Note that, λ is the stretch with respect to the modified cost functions. We present the algorithm as Algorithm 1, but note that only the definition of $\theta(q)$ using λ, the definition of p in Line 1, and the use of the modified cost functions $(f'_e)_{e\in E}$ in Line 11 has been changed.

2.3.1 Analysis of the Algorithm

We are now ready to prove Theorem 2.1, by restating it as follows. The proof of the theorem follows the proof scheme of Caragiannis et al. [CFGS11], which we have to rigorously rework to accommodate for our modifications stated above.

Theorem 2.9. *For every constant $\varepsilon > 0$ and every set of cost functions $(f'_e)_{e\in E}$ which are strongly $(\lambda, 0)$-smooth with respect to Φ, Algorithm 1 computes a $\lambda(1+\varepsilon)$-approximate pure Nash equilibrium for every congestion game with non-decreasing cost functions, in number of steps which is polynomial in the number of players, $\Delta := \max_{e\in E} \frac{f_e(N)}{f_e(1)}$, λ, and $1/\varepsilon$.*

Proof. The algorithm partitions the players into blocks $B_1, B_2, \ldots, B_{\hat{z}}$ such that, a player

$$u \in B_i \Leftrightarrow \ell_u \in (b_{i+1}, b_i],$$

Algorithm 1 Computing a $\lambda(1+\varepsilon)$-approximate pure Nash equilibrium in congestion games.

```
Input: Congestion game $G = \left(\mathcal{N}, E, (S_u)_{u\in\mathcal{N}}, (f_e)_{e\in E}\right)$, $f' := (f'_e)_{e\in E}$ and $\varepsilon > 0$.
Output: A state of $G$ in $\lambda(1+\varepsilon)$-approximate pure Nash equilibrium.
 1: Set $q = \left(1+\frac{1}{N^c}\right)$, $p = \left(\frac{1}{\theta(q)} - \frac{1+q+2\lambda}{N^c}\right)^{-1}$, $c = 10\log\left(\frac{\lambda}{\varepsilon}\right)$, $\Delta = \max_{e\in E}\frac{f_e(N)}{f_e(1)}$ and
    $\theta(q) = \frac{\lambda}{1+\frac{1-q}{q}N\lambda}$, where
    $\lambda := \min\{\lambda' \in \mathbb{R}_{\geq 0} : \lambda'$ satisfies Lemma 2.8 with respect to cost functions $(f'_e)_{e\in E}\}$.
 2: foreach $u \in \mathcal{N}$ do
 3:     set $\ell_u = C_u\left(\mathcal{BR}_u(0)\right)$;
 4: end for
 5: Set $\ell_{min} = \min_{u\in\mathcal{N}}\ell_u$, $\ell_{max} = \max_{u\in\mathcal{N}}\ell_u$ and $\hat{z} = 1 + \lceil\log_{2\Delta N^{2c+2}}\left(\ell_{max}/\ell_{min}\right)\rceil$;
 6: Assign players to blocks $B_1, B_2, \cdots, B_{\hat{z}}$ such that
    $u \in B_i \Leftrightarrow \ell_u \in \left(\ell_{max}\left(2\Delta N^{2c+2}\right)^{-i}, \ell_{max}\left(2\Delta N^{2c+2}\right)^{-i+1}\right]$;
 7: foreach $u \in N$ do
 8:     set the player $u$ to play the strategy $s_u \leftarrow \mathcal{BR}_u(0)$;
 9: end for
10: for phase $i \leftarrow 1$ to $\hat{z}-1$ such that $B_i \neq \emptyset$ do
11:     while $\exists u \in B_i$ with a $p$-move w.r.t. the original cost $f$ or $\exists u \in B_{i+1}$ with a $q$-move
        w.r.t. to modified cost $f'$ do
12:         $u$ deviates to that best-response strategy $s_u \leftarrow \mathcal{BR}\left(s_1, \cdots, s_n\right)$.
13:     end while
14: end for
```

where

$$b_i := \ell_{max} \cdot \left(2\Delta N^{2c+2}\right)^{-i+1} \text{ and } b_{i+1} := \ell_{max} \cdot \left(2\Delta N^{2c+2}\right)^{-i}$$

define the boundaries of the block B_i, $\ell_u = C_u\left(\mathcal{BR}_u(0)\right)$ and $\ell_{max} = \max_{u\in\mathcal{N}}\ell_u$.

The players are partitioned to at most $\hat{z} = 1 + \lceil\log_{2\Delta N^{2c+2}}\left(\ell_{max}/\ell_{min}\right)\rceil \leq N$ blocks, where for any block B_i the ratio $\frac{b_i}{b_{i+1}} = 2\Delta N^{2c+2}$, $\Delta = \max_{e\in E}\frac{f_e(N)}{f_e(1)}$ and $\ell_{min} = \min_{u\in\mathcal{N}}\ell_u$. Observe that for cost functions which are polynomials of maximum degree d with non negative coefficients, Δ is polynomial in the number of players. As part of the initialization step, the algorithm forces every player $u \in \mathcal{N}$ to play their optimistic strategy $\mathcal{BR}_u(0)$, thus each player incurs a cost at most Δb_i. This describes the initial state of the game denoted by s^0, where s^i denotes the state of the game after the phase i. The sequence of moves in the game is divided into multiple phases determined by the player blocks. The phases of the game progresses from $1 \to \hat{z}-1$. During a phase i of the game, only players in the block B_i and B_{i+1} make moves. Particularly, players in block B_i make their p-move using the original cost function f and the players in block B_{i+1} make their q-move, but now using the modified cost functions f' that satisfies Lemma 2.8 for some $\lambda > 0$.

For player $u \in \mathcal{N}$ and an arbitrary state $s \in S$, a deviation to a strategy $s'_u \in S_u$ is referred to as a p-move if

$$C_u(s'_u, s_{-u}) < \frac{C_u(s)}{p}.$$

Similarly, a q-move with respect to the modified cost functions f' is defined as a move with

$$C'_u(s'_u, s_{-u}) < \frac{C'_u(s)}{q}.$$

A phase i terminates in a state s^i, if for all $u \in B_i$,

$$C_u(s^i) \le p \cdot C_u(s'_u, s^i_{-u})$$

i.e., the players in B_i are in a p-equilibrium w.r.t. the original cost functions and for all $u \in B_{i+1}$,

$$C'_u(s^i) \le q \cdot C'_u(s'_u, s^i_{-u})$$

i.e., the players in B_{i+1} are in a q-equilibrium w.r.t. the modified cost functions. The remaining players, i.e., $u \in \mathcal{N} \setminus (B_i \cup B_{i+i})$ remain frozen to their strategy associated with the phase $i-1$. Also, note that players in a block B_i were frozen to their optimistic strategy $\mathcal{BR}_u(0)$ until phase $i-1$. Let us denote by R_i the players involved in phase i. Recall that during the phase i, only players in R_i are allowed to make their best-response p/q-moves. Let us denote by $f_e^{R_i}$ the cost introduced by these players on a resource $e \in E$ of the game. Furthermore, the players $\mathcal{N} \setminus R_i$ are frozen to their strategy in phase $i-1$. Therefore, the cost incurred by a player $u \in R_i$ using a resource $e \in E$ can be given as $f_e^{R_i}(n_e^{R_i}(s)) = f_e\left(n_e^{R_i}(s) + n_e^{\mathcal{N}\setminus R_i}(s)\right)$, where $n_e^{R_i}(s)$ denotes the number of players $u \in R_i$ in the state s using the resource e and $n_e^{\mathcal{N}\setminus R_i}(s)$ denotes the number of players on the resource e in the state s that do not participate in the phase i of the game. The potential among the players in R_i, i.e., the potential of the subgame induced by players in R_i will be denoted as Φ_{R_i}.

Here, we have to take into account that the game played by the players belonging to block $B_i \cup B_{i+1}$ in phase i is no longer a potential game as the players use different cost functions. However, we can show that the strong smoothness condition of Lemma 2.8 guarantees that the values of the modified cost functions f' can be conveniently bounded.

Lemma 2.10. *Let f' to be the set of modified cost functions satisfying strong $(\lambda, 0)$-smoothness condition for some $\lambda > 0$ and f to be the original cost functions. Then, for all $i \ge 1$,*

$$f_e(i) \le f'_e(i) \le \lambda f_e(i).$$

Proof. The strong smoothness condition of Lemma 2.8 gives us that

$$\lambda \cdot \sum_{j=z+1}^{m+z} f_e(j) - m f'_e(n+z+1) + n f'_e(n+z) \ge \sum_{j=z+1}^{n+z} f_e(j)$$

for all $0 \le (n+z), m \le N$.

Setting $n = 0$, $m = 1$, and $z = i-1$ gives,

$$\lambda f_e(i) \ge f'_e(i).$$

Furthermore, with $m = 0$, $n = 1$, and $z = i-1$ we have that,

$$f'_e(i) \ge f(i).$$

□

To bound the stretch of any (sub-) game in a q-approximate pure Nash equilibrium, the following lemma is useful. In its proof we handle the modified cost functions which then leads to value of

$$\theta(q) := \frac{\lambda}{1 + N\lambda\frac{1-q}{q}}$$

(cf. Algorithm 1) that depends on the stretch λ of the modified cost functions, instead of the original ones. We remark that for this lemma, the property that the induced subgames are also smooth (i.e., Lemma 2.8) is crucial.

Lemma 2.11. *Let s be any q-approximate pure Nash equilibrium with respect to the modified cost functions and s^* be a strategy profile with minimal potential. Then, for every set of players $F \subseteq \mathcal{N}$, $\Phi_F(s) \leq \theta(q) \cdot \Phi_F(s^*)$.*

Proof. Let C'_u and C_u denote the cost of a player $u \in F$ using the modified cost functions f' and the original cost functions f, respectively. From the definition of q-approximate equilibrium we have that,

$$C'_u(s) \leq q \cdot C'_u(s^*_u, s_{-u}).$$

Then, using Lemma 2.10,

$$C'_u(s^*_u, s_{-u}) - C'_u(s) \geq \frac{1-q}{q} C'_u(s) \geq \frac{1-q}{q} \lambda C_u(s) \geq \frac{1-q}{q} \lambda \Phi_F(s),$$

summing the above inequality for all players $u \in F$ gives,

$$\sum_{u \in F} \left(C'_u(s^*_u, s_{-u}) - C'_u(s) \right) \geq \frac{1-q}{q} N \lambda \Phi_F(s). \tag{2.2}$$

Then, by the smoothness condition of Lemma 2.8 and inequality (2.2) we have,

$$\begin{aligned} \Phi_F(s) &\leq \lambda \cdot \Phi_F(s^*) - \left(\sum_{u \in F} \left(C'_u(s^*_u, s_{-u}) - C'_u(s) \right) \right) \\ &\leq \lambda \cdot \Phi_F(s^*) - \frac{1-q}{q} N \lambda \Phi_F(s). \\ \left(1 + \frac{1-q}{q} N \lambda \right) \Phi_F(s) &\leq \lambda \cdot \Phi_F(s^*) \\ \Phi_F(s) &\leq \frac{\lambda}{1 + \frac{1-q}{q} N \lambda} \cdot \Phi_F(s^*). \end{aligned} \tag{2.3}$$

Setting $\theta(q) = \frac{\lambda}{1 + N\lambda \frac{1-q}{q}}$ and inequality (2.3) concludes the proof. □

The next claim bounds from above and below the value of the potential in an arbitrary state s of the game.

Claim 2.12 (Caragiannis et al. [CFGS11]). *For any state s of a congestion game with a set of players $\mathcal{N}$, a set of resource E and cost functions $(f_e)_{e \in E}$, it holds that*

$$\sum_{e \in E} f_e(n_e(s)) \leq \Phi(s) \leq \sum_{u \in \mathcal{N}} C_u(s).$$

The following lemmas give us useful bounds on the potential of a subgame in an arbitrary state $s \in S$.

Lemma 2.13 (Caragiannis et al. [CFGS11]). *Let s be a state of the congestion game $\mathcal{G}$ with a set of players $\mathcal{N}$ and let $F \subseteq \mathcal{N}$. Then,*

$$\Phi(s) \leq \Phi_F(s) + \Phi_{\mathcal{N} \setminus F}(s)$$

and $\Phi(s) \geq \Phi_F(s)$.

Lemma 2.14 (Caragiannis et al. [CFGS11]). *Let $C(u)$ denote the cost of player $u \in R_i$ just after making their last move within phase i. Then,*

$$\Phi_{R_i}(s^i) \leq \sum_{u \in R_i} C(u).$$

We now bound the potential of the set of players $R_i \subseteq B_i \cup B_{i+1}$ that move in any given phase i. Most importantly, the players of B_i, were in an q-approximate equilibrium with respect to C'_u at the end of the previous round. Hence, for every subset of B_i, we can exploit Lemma 2.11 to obtain a small upper bound on the potential among players R_i participating in a phase i at the beginning of the phase. Recall that for a phase i, $b_i := \ell_{max} \cdot \left(2\Delta N^{2c+2}\right)^{-i+1}$ and s^i denotes the state of the game after the execution of phase i.

Lemma 2.15. *For every phase $i \geq 2$, it holds that $\Phi_{R_i}(s^{i-1}) \leq \frac{b_i}{N^c}$.*

Proof. We prove by contradiction. Let us assume that the inequality does not hold, i.e.,

$$\Phi_{R_i}(s^{i-1}) > \frac{b_i}{N^c}.$$

Then, we show that the players $u \in R_i \cap B_i$ were not in a q-approximate equilibrium w.r.t. the modified cost functions f' in the phase $i-1$. However, this violates the termination condition of phase $i-1$ in the algorithm.

Observe that the players in the block B_{i+1} are not allowed to moved until the conclusion of phase $i-1$ and remain in their optimistic strategy $\mathcal{BR}_u(0)$ according to the initial settings of the algorithm. Moreover, the cost incurred by a player $u \in B_{i+1}$ is at most Δb_{i+1}. Therefore, the total cost incurred by the players in $R_i \cap B_{i+1}$, i.e.,

$$\sum_{u \in R_i \cap B_{i+1}} \Delta b_{i+1} \leq N\Delta b_{i+1}.$$

The potential among players in $R_i \cap B_{i+1}$ is bounded by,

$$\Phi_{R_i \cap B_{i+1}}(s^{i-1}) \leq N\Delta b_{i+1}. \tag{2.4}$$

Using Lemma 2.13, (2.4), and our assumption on $\Phi_{R_i}(s^{i-1})$ we get,

$$\begin{aligned}
\Phi_{R_i \cap B_i}(s^{i-1}) &\geq \Phi_{R_i}(s^{i-1}) - \Phi_{R_i \cap B_{i+1}}(s^{i-1}) \\
&> \frac{b_i}{N^c} - N\Delta b_{i+1} \\
&= \left(\frac{2\Delta N^{2c+2}}{N^c} - N\Delta\right) b_{i+1} \\
&\geq N^{c+1}\Delta b_{i+1}.
\end{aligned} \tag{2.5}$$

With a slight abuse of notation, let us denote by $C(u)$ the cost of a player $u \in R_i \cap B_i$ after they made their last move during the phase i. So, the change in potential contributed by the player u in the phase i is then at least $(p-1)C(u)$. Let us denote by ξ_i the decrease of potential due to the moves of the players in B_{i+1} in phase i. Note that ξ_i could be negative as the players of B_{i+1} use the modified cost functions. The change in potential due to the moves by all the players in R_i is given by $(p-1)\sum_{u \in R_i \cap B_i} C(u) + \xi_i$ and we can bound

$$(p-1) \sum_{u \in R_i \cap B_i} C(u)$$

$$\begin{aligned}
&\leq \Phi_{R_i}(s^{i-1}) - \Phi_{R_i}(s^i) - \xi_i \\
&\leq \Phi_{R_i \cap B_i}(s^{i-1}) + \Phi_{R_i \cap B_{i+1}}(s^{i-1}) - \Phi_{R_i}(s^i) - \xi_i \\
&\leq \Phi_{R_i \cap B_i}(s^{i-1}) + N\Delta b_{i+1} - \Phi_{R_i}(s^i) - \xi_i \\
&< \left(1 + \frac{1}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1}) - \Phi_{R_i}(s^i) - \xi_i. \qquad (2.6)
\end{aligned}$$

To account in the change of potential ξ_i from s^{i-1} to s^i due the players in B_{i+1}, we observe that the cost of a player $u \in R_i \cap B_{i+1}$ was at most $C_u(s^{i-1}) \leq \Delta b_{i+1}$ as the player u was put by the algorithm on the strategy $\mathcal{BR}(0)$. By Lemma 2.10, the player's cost with respect to the modified cost function on this strategy is $C'_u(s^{i-1}) \leq \lambda \Delta b_{i+1}$. Since, a player may always switch back to this strategy, their cost in s^i can be bounded by

$$C_u(s^i) \leq C'_u(s^i) < q\lambda\Delta b_{i+1}.$$

This yields a bound on the change of the potential of

$$\xi_i > -qN\lambda\Delta b_{i+1}.$$

Now we can bound the potential in s^i by the cost of the players. We then can use inequality (2.6) for the players in $R_i \cap B_i$. By Lemma 2.10, the cost of a player $u \in R_i \cap B_{i+1}$ after they made their last q-move during the phase i is at most $\lambda \Delta b_{i+1}$.

$$\begin{aligned}
\Phi_{R_i}(s^i) &\leq \sum_{u \in R_i} C(u) \\
&= \sum_{u \in R_i \cap B_{i+1}} C(u) + \sum_{u \in R_i \cap B_i} C(u) \\
&< N\lambda\Delta b_{i+1} + \frac{1}{p-1}\left(1 + \frac{1}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1}) \\
&\quad - \frac{1}{p-1}\Phi_{R_i}(s^i) - \frac{1}{p-1}\xi_i \\
&\leq \frac{\lambda}{N^c}\Phi_{R_i \cap B_i}(s^{i-1}) + \frac{1}{p-1}\left(1 + \frac{1}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1}) \\
&\quad - \frac{1}{p-1}\Phi_{R_i}(s^i) + \frac{q}{p-1}N\lambda\Delta b_{i+1} \\
&\leq \frac{\lambda}{N^c}\Phi_{R_i \cap B_i}(s^{i-1}) + \frac{1}{p-1}\left(1 + \frac{1}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1}) \\
&\quad - \frac{1}{p-1}\Phi_{R_i}(s^i) + \frac{q}{p-1}\frac{\lambda}{N^c}\Phi_{R_i \cap B_i}(s^{i-1}) \\
&\leq \frac{\lambda(p-1)}{(p-1)N^c}\Phi_{R_i \cap B_i}(s^{i-1}) + \frac{1}{p-1}\left(1 + \frac{1}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1}) \\
&\quad - \frac{1}{p-1}\Phi_{R_i}(s^i) + \frac{q}{p-1}\frac{\lambda}{N^c}\Phi_{R_i \cap B_i}(s^{i-1}) \\
&\leq \frac{1}{p-1}\left(1 + \frac{(p-1)\lambda + 1 + q\lambda}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1}) \\
&\quad - \frac{1}{p-1}\Phi_{R_i}(s^i)
\end{aligned}$$

equivalent to,

$$\frac{p}{p-1}\Phi_{R_i}(s^i) < \frac{1}{p-1}\left(1 + \frac{(p-1)\lambda + 1 + q\lambda}{N^c}\right) \Phi_{R_i \cap B_i}(s^{i-1})$$

$$
\begin{aligned}
&= \frac{p}{p-1}\left(\frac{1}{p} + \frac{(p-1)\lambda + 1 + q\lambda}{pN^c}\right)\Phi_{R_i\cap B_i}(s^{i-1})\\
&< \frac{p}{p-1}\left(\frac{1}{p} + \frac{\lambda}{N^c} + \frac{1+q\lambda}{pN^c}\right)\Phi_{R_i\cap B_i}(s^{i-1})\\
&< \frac{p}{p-1}\left(\frac{1}{p} + \frac{\lambda}{N^c} + \frac{1+q}{N^c}\right)\Phi_{R_i\cap B_i}(s^{i-1}).
\end{aligned}
$$

Therefore,

$$\Phi_{R_i}(s^i) < \left(\frac{1}{p} + \frac{1+q+\lambda}{N^c}\right)\Phi_{R_i\cap B_i}(s^{i-1}). \tag{2.7}$$

Observe that until the end of phase $i-1$, the players in the block $R_i \cap B_{i+1}$ have not deviated from their initial strategy of $\mathcal{BR}_u(0)$ in the state s^{i-1}. However, this cannot be guaranteed in the state s^i where the players in $R_i \cap B_{i+1}$ could have made their best-response q-moves. Therefore, it is important that the players in $R_i \cap B_{i+1}$ play the same strategy as they had in phase $i-1$, before we compare the potential among the players in $R_i \cap B_i$ in the state s^{i-1} and s^i.

Consider the following setting. Let $\hat{s}$ be a state where player in $R_i \cap B_i$ play their strategy in s^i and players $u \in \mathcal{N} \setminus (R_i \cap B_i)$ play their strategy in s^{i-1}. Since, the cost incurred by players in $R_i \cap B_{i+1}$ in the state s^i after deviating to their strategy in s^{i-1} is at most $N\lambda\Delta b_{i+1}$. We can bound the potential among the players in R_i in the state $\hat{s}$ as,

$$
\begin{aligned}
\Phi_{R_i}(\hat{s}) &\le \Phi_{R_i\cap B_i}(s^i) + \Phi_{R_i\cap B_{i+1}}(s^{i-1})\\
&\le \Phi_{R_i\cap B_i}(s^i) + N\lambda\Delta b_{i+1}\\
&\le \Phi_{R_i}(s^i) + N\lambda\Delta b_{i+1}. \qquad (2.8)
\end{aligned}
$$

Using Lemma 2.13 we get,

$$\Phi_{R_i\cap B_i}(\hat{s}) \le \Phi_{R_i}(\hat{s})$$

applying inequality (2.8) we get,

$$\le \Phi_{R_i}(s^i) + N\lambda\Delta b_{i+1}$$

then from inequalities (2.5) and (2.7),

$$
\begin{aligned}
&< \left(\frac{1}{p} + \frac{1+q+2\lambda}{N^c}\right)\Phi_{R_i\cap B_i}(s^{i-1})\\
&= \frac{1}{\theta(q)}\Phi_{R_i\cap B_i}(s^{i-1}).
\end{aligned}
$$

The last equality follows from the definition of p in Algorithm 1.

If s^* were to be the state in which the game attained its global minimum, then the last inequality implies that the potential among the players in $R_i \cap B_i$ in state s^* i.e., $\Phi_{R_i\cap B_i}(s^*)$ is strictly smaller than $\frac{1}{\theta(q)}\Phi_{R_i\cap B_i}(s^{i-1})$, i.e.,

$$
\begin{aligned}
\Phi_{R_i\cap B_i}(s^*) &\le \Phi_{R_i\cap B_i}(\hat{s})\\
&< \frac{1}{\theta(q)}\Phi_{R_i\cap B_i}(s^{i-1}).
\end{aligned}
$$

However, this violates the equilibrium condition in Lemma 2.11 to conclude that the players in $R_i \cap B_i$ were not in a q-equilibrium at the end of the phase $i-1$. This contradicts our assumption. □

To analyze the convergence of the algorithm, we have to take into account the fact that players use different cost functions and hence, convergence is no longer guaranteed by Rosenthal's potential function. However, it turns out that the Rosenthal potential with respect to the modified cost functions can serve as an approximate potential function, i.e., it also decreases for the p-moves of players using the original cost functions.

Lemma 2.16. *The Rosenthal potential $\widetilde{\Phi}$ with respect to the modified cost functions f' is a p-approximate potential function with respect to the original cost functions f. That is,*

$$C_u(s'_u, s_{-u}) < \frac{1}{p} C_u(s) \text{ implies } \widetilde{\Phi}(s'_u, s_{-u}) < \widetilde{\Phi}(s),$$

where

$$\widetilde{\Phi}(s) := \sum_{e \in E} \widetilde{\Phi}_e(n_e(s)) = \sum_{e \in E} \sum_{i=1}^{n_e(s)} f'_e(i).$$

Proof. To simplify notation let $n_e := n_e(s)$. From Lemma 2.10 we know that,

$$\widetilde{\Phi_e}(n_e + 1) - \widetilde{\Phi_e}(n_e) = f'(n_e + 1) \le \lambda f(n_e + 1),$$

and

$$f'(n_e + 1) \ge f(n_e + 1).$$

Therefore,

$$1 \le \frac{\widetilde{\Phi}_e(n_e + 1) - \widetilde{\Phi}_e(n_e)}{f_e(n_e + 1)} \le \lambda.$$

Using the above inequalities, the change in potential function $\widetilde{\Phi}$ due to player u making a p-move with respect to f, i.e.,

$$\begin{aligned}
\widetilde{\Phi}(s'_u, s_{-u}) - \widetilde{\Phi}(s) &= \sum_{e \in E} \widetilde{\Phi}_e(s'_u, s_{-u}) - \widetilde{\Phi}_e(s) \\
&= \sum_{e \in s'_u \setminus s_u} \widetilde{\Phi}_e(n_e + 1) - \widetilde{\Phi}_e(n_e) + \sum_{e \in s_u \setminus s'_u} \widetilde{\Phi}_e(n_e - 1) - \widetilde{\Phi}_e(n_e) \\
&\le \sum_{e \in s'_u \setminus s_u} \lambda \cdot f_e(n_e + 1) - \sum_{e \in s_u \setminus s'_u} f_e(n_e) \\
&\le \lambda \left(\sum_{e \in s'_u \setminus s_u} f_e(n_e + 1) + \sum_{e \in s'_u \cap s_u} f_e(n_e) \right) \\
&\quad - \left(\sum_{e \in s_u \setminus s'_u} f_e(n_e) + \sum_{e \in s'_u \cap s_u} f_e(n_e) \right) \\
&= \lambda \cdot C_u(s'_u, s_{-u}) - C_u(s) \\
&\le p \cdot C_u(s'_u, s_{-u}) - C_u(s).
\end{aligned}$$

The last inequality is due to the choice of p in Algorithm 1 such that it is slightly larger than λ. □

The following lemma exhibits an even stronger property. It shows that p-moves with respect to the original cost functions are q-moves with respect to the modified cost functions.

Lemma 2.17. *Let $u \in \mathcal{N}$ be a player that makes a p-move with respect to the original cost function f. Then,*

$$p \cdot C_u(s'_u, s_{-u}) - C_u(s) \geq q \cdot C'_u(s'_u, s_{-u}) - C'_u(s),$$

where C_u and C'_u are the cost of the player u with respect to f and f', respectively.

Proof. Let us recall the definition of p, q, and $\theta(q)$ in Algorithm 1,

$$p := \left(\frac{1}{\theta(q)} - \frac{1 + 2\lambda + q}{N^c}\right)^{-1}$$

$$q := \left(1 + \frac{1}{N^c}\right)$$

$$\theta(q) := \frac{\lambda}{1 + \frac{1-q}{q} N\lambda}$$

Observe that $p \geq \theta(q)$. Therefore,

$$\begin{aligned} p \cdot C_u(s'_u, s_{-u}) - C_u(s) &\geq \theta(q) \cdot C_u(s'_u, s_{-u}) - C_u(s) \\ &= \frac{\lambda}{1 + \frac{1-q}{q} N\lambda} \cdot C_u(s'_u, s_{-u}) - C_u(s) \\ &= \frac{q\lambda}{1 + \frac{1}{N^c}(1 - N\lambda)} \cdot C_u(s'_u, s_{-u}) - C_u(s) \\ &\geq q\lambda \cdot C_u(s'_u, s_{-u}) - C_u(s). \end{aligned}$$

Then, from Lemma 2.10 we have that,

$$p \cdot C_u(s'_u, s_{-u}) - C_u(s) \geq q \cdot C'_u(s'_u, s_{-u}) - C'_u(s).$$

□

Using Lemma 2.15 and 2.17, we can bound the runtime which depends on Δ to allow for arbitrary non-decreasing functions.

Lemma 2.18. *The algorithm terminates after at most $O(\lambda\Delta^3 N^{5c+5})$ best-response moves.*

Proof. The proof follows from Lemma 2.16 and Lemma 2.17. Again, we denote f' to be the modified cost functions and f to be the original cost functions.

Let us recall that the algorithm partitions the sequence of best-response moves in the game into $\hat{z} - 1$ phases, where $\hat{z} = 1 + \lceil \log_{2\Delta n^{2c+2}} (\ell_{max}/\ell_{min}) \rceil \leq N$. In a phase $i \in \{1, \dots, \hat{z} - 1\}$, players in block $R_i \cap B_i$ make their p-move with respect to f and players in block $R_i \cap B_{i+1}$ make their q-move with respect to f'. We will bound the number of p-moves and q-moves in any given phase i, using the potential function with respect to the modified cost function f'. Lemma 2.16 shows that when players in the block B_i make their p-moves, they also reduce the potential function with respect to the modified cost functions f'. Lemma 2.17 shows that the change in cost of a player due to a p-move with respect to the function f is at least the change in cost due to a q-move with respect to f'. Therefore, in order to bound total number of moves in any given phase i, it is sufficient to assume that players in block B_i make q-moves with respect to f' instead of p-moves.

Using these we now bound the maximum number of best response moves in a phase,
Phase $i = 1$:
Let us assume that all players in the phase R_1 have a q-move.

Define
$$\Delta' = \max_{e\in E, n\in N} \frac{f'_e(n)}{f_e(1)} \leq \max_{e\in E, n\in N} \frac{\lambda \cdot f_e(n)}{f_e(1)},$$
where the inequality follows from Lemma 2.10. Then, for any player $u \in R_1$, the maximum cost incurred by the player at the beginning of phase with respect to f' is at most $\Delta' \cdot b_1$. Observe that the maximum potential associated with the subgame in the phase R_1 with respect to the modified cost function f' is then at most $N\Delta' b_1$.

Also, observe that the minimum cost experienced by the players in R_1 is at least b_3. Therefore, when a player $u \in R_1$ makes a best-response q-move, they must reduce the potential by at least $(q-1)b_3$. Then, using the fact that $b_i = 2\Delta N^{2c+2} b_{i+1}$, we obtain the number of best response moves among the players in R_1 to be at most,
$$\frac{N\Delta' b_1}{(q-1)b_3} = \frac{N\Delta' \left(4\Delta^2 N^{5c+4}\right)}{N^c(q-1)} \leq 4\lambda\Delta^3 N^{5c+5}. \tag{2.9}$$
Phase $i \geq 2$: Again, let us assume that all players in the R_i have a q-move. Lemma 2.15 shows that for each phase $i \geq 2$, the potential among the players R_i participating in the phase i at the beginning of the phase i.e., $\Phi_{R_i}(s^{i-1})$ is at most $\frac{b_i}{N^c}$.

Therefore, due to Lemma 2.10 the potential with respect to the modified cost function is then at most $\frac{\lambda \cdot b_i}{N^c}$. By the definition of blocks in Algorithm 1, the minimum cost that a player would incur is at least b_{i+2}. This implies, when a player u makes a best-response q-move during phase i, they would reduce the potential of the subgame among the players in R_i by at least $(q-1)b_{i+2}$. Hence, using the fact that $b_i = 2\Delta N^{2c+2} b_{i+1}$, we obtain the number of best response moves among the players in R_i to be at most,
$$\frac{\lambda \cdot b_i}{N^c(q-1)b_{i+2}} = \frac{\lambda \left(4\Delta^2 N^{4c+4}\right)}{N^c(q-1)} \leq 4\lambda\Delta^2 N^{4c+4}. \tag{2.10}$$
Using (2.9) and (2.10) we can bound the number of best-response moves in the game to be at most $O(\lambda\Delta^3 N^{5c+5})$. □

The next lemma shows that when players involved in phases $i \geq 2$ make their moves, they do not increase the cost of players in the blocks $B_1, B_2, \cdots, B_{i-1}$ significantly.

Lemma 2.19. *Let u be a player that takes part in the phase $t \leq i$, then it holds that,*
$$C_u(s^{i+1}) \leq C_u(s^i) + \frac{b_{i+1}}{N^c} + qN\lambda\Delta b_{i+2}.$$
Proof. We derive the proof by showing that if the increase in cost of the player u is greater than $\frac{b_{i+1}}{N^c} + qN\lambda\Delta b_{i+2}$, then it violates the fact that $\Phi_{R_{i+1}}(s^i) \leq \frac{b_{i+1}}{N^c}$.

Now, let us assume that there exists player $u \in B_i$ for whom the lemma does not hold i.e.,
$$C_u(s^{i+1}) > C_u(s^i) + \frac{b_{i+1}}{N^c} + qN\lambda\Delta b_{i+2}. \tag{2.11}$$
This implies that there exists a set of resources $C \subseteq s_u$ such that for each $e \in C$ it is used by at least one player in R_{i+1} in the state s^{i+1}, thus contributing to the increase in cost of the player u. Then, (2.11) gives us that,
$$\sum_{e\in C} f_e(n_e(s^{i+1})) > \frac{b_{i+1}}{N^c} + qN\lambda\Delta b_{i+2}.$$

By Claim 2.12 we get,

$$\Phi_{R_{i+1}}(s^{i+1}) > \frac{b_{i+1}}{N^c} + qN\lambda\Delta b_{i+2}.$$

As the players in $R_{i+1} \cap B_{i+2}$ might have increased (or decreased) $\Phi_{R_{i+1}}$ by at most $qN\lambda\Delta b_{i+2}$ and the players $R_{i+1} \setminus B_{i+2}$ only decreased the potential, we know that

$$\begin{aligned}\Phi_{R_{i+1}}(s^i) &\geq \Phi_{R_{i+1}}(s^{i+1}) - qN\lambda\Delta b_{i+2} \\ &> \frac{b_{i+1}}{N^c} + qN\lambda\Delta b_{i+2} - qN\lambda\Delta b_{i+2} \\ &= \frac{b_{i+1}}{N^c}\end{aligned}$$

The last inequality violates Lemma 2.15. Hence, this contradicts our assumption and therefore, the lemma holds. □

Lemma 2.20 (Caragiannis et al. [CFGS11]). *Let u be a player that takes part in the phase $t \leq i$ of the congestion game and let s'_u be any strategy other than the one assigned by the algorithm during the phase t of the game, then it holds that,*

$$C_u(s'_u, s^i_{-u}) \leq C_u(s'_u, s^{i+1}_{-u}) + \frac{b_{i+1}}{N^c}.$$

Proof. We prove by contradiction. Assume the lemma does not hold for some player u, i.e.,

$$C_u(s'_u, s^i_{-u}) > C_u(s'_u, s^{i+1}_{-u}) + \frac{b_{i+1}}{N^c}.$$

This implies that during the phase i, there exists a subset of resources $C \subseteq s'_u$ such that for each $e \in C$, there exists a player $u' \in R_{i+1}$ who used the resource e in the state s^i but not in s^{i+1} and thus contributed to cost of the player u at end of the phase i, i.e.,

$$\sum_{e \in C} f_e(n_e(s'_u, s^i_{-u})) > \frac{b_{i+1}}{N^c}.$$

Furthermore, the cost of these resources yield a lower bound on the potential at the beginning of the phase, i.e,

$$\begin{aligned}\Phi_{R_{i+1}}(s^i) &\geq \sum_{e \in C} f_e(n_e(s'_u, s^i_{-u})) \\ &> \frac{b_{i+1}}{N^c}.\end{aligned}$$

The last inequality violates Lemma 2.15 and therefore, this contradicts our assumption. □

Lemma 2.21. *Let u be a player in the block B_t, where $t \leq \hat{z} - 2$. Let s'_u be a strategy different from the one assigned to u by the algorithm at the end of the phase t. Then, for each phase $i \geq t$, it holds that, $C_u(s^i) \leq p \cdot C_u(s'_u, s^i_{-u}) + \frac{2p+1}{N^c} \sum_{k=t+1}^{i} b_k$.*

Proof. For the proof we use Lemma 2.19 recursively to obtain the first inequality. The second inequality follows from the fact that there was no improving p-move to s'_u for the player in phase t to s'_u. The third inequality follows from Lemma 2.20. The fourth inequality from the definition of b_i.

$$\begin{aligned}
C_u(s^i) &\leq C_u(s^t) + \sum_{k=t+1}^{i} \left(\frac{b_k}{N^c} + qN\lambda\Delta b_{k+1} \right) \\
&\leq p \cdot C_u(s'_u, s^t_{-u}) + \sum_{k=t+1}^{i} \left(\frac{b_k}{N^c} + qN\lambda\Delta b_{k+1} \right) \\
&\leq p \left(C_u(s'_u, s^i_{-u}) + \sum_{k=t+1}^{i} \frac{b_k}{N^c} \right) + \sum_{k=t+1}^{i} \left(\frac{b_k}{N^c} + qN\lambda\Delta b_{k+1} \right) \\
&\leq p \left(C_u(s'_u, s^i_{-u}) + \sum_{k=t+1}^{i} \frac{b_k}{N^c} \right) + \sum_{k=t+1}^{i} \left(\frac{b_k}{N^c} + \frac{q\lambda}{2N^{2c+1}} b_k \right) \\
&\leq p \cdot C_u(s'_u, s^i_{-u}) + (2p+1) \sum_{k=t+1}^{i} \frac{b_k}{N^c}.
\end{aligned}$$

□

As no players' costs and alternatives is significantly influenced by moves in later blocks, they remain in an approximate equilibrium which can be used to finally prove the correctness of the algorithm.

Lemma 2.22. *The state computed by the algorithm is a* $p\left(1+\frac{5}{N^c}\right)$*-approximate equilibrium.*

Proof. We show that after a player $u \in B_i$ has made their final best-response move during a phase i, they would be in a $p\left(1+\frac{5}{N^c}\right)$-approximate equilibrium at the end of the game in the state $s^{\hat{z}-1}$, i.e., the cost incurred by them after the final phase of the game is,

$$C_u(s^{\hat{z}-1}) \leq p \left(1 + \frac{5}{N^c}\right) C_u(s'_u, s^{\hat{z}-1}_{-u}),$$

where $s'_u \in S_u$ is any arbitrary strategy. For players participating in the last phase $\hat{z}-1$ of the game i.e., $u \in (B_{\hat{z}-1} \cup B_{\hat{z}})$, observe that at the end of the phase $\hat{z}-1$, players in block $B_{\hat{z}-1}$ are in a p-approximate equilibrium with respect to the original cost functions f and players in the block $B_{\hat{z}}$ are in a q-approximate equilibrium with respect to the modified cost function f'. Furthermore, due to Lemma 2.17 players in the block $B_{\hat{z}}$ are also in a p-approximate equilibrium with respect to the original cost function f. Therefore, the lemma holds for players in $u \in (B_{\hat{z}-1} \cup B_{\hat{z}})$.

We show that after the final phase of the game, for the players $u \in B_t$ with $1 \leq t \leq \hat{z}-2$,

$$C_u(s^{\hat{z}-1}) \leq p \left(1 + \frac{5}{N^c}\right) C_u(s'_u, s^{\hat{z}-1}_{-u}).$$

As per the assignment of the players to blocks and the fact that the cost of a player cannot be less than $\ell_u = C_u\left(\mathcal{BR}_u\left(0\right)\right)$, it holds that for any player $u \in B_t$ after the final phase of the game,

$$C_u(s'_u, s^{\hat{z}-1}_{-u}) \geq b_{t+1}. \tag{2.12}$$

By the definition of b_i, we have,

$$\sum_{k=t+1}^{\hat{z}} b_k \leq 2b_{t+1}. \tag{2.13}$$

Using inequalities (2.12), (2.13), and Lemma 2.21 we get for $p \geq 1$,

$$\begin{aligned}
C_u(s^{\hat{z}-1}) &\leq p \cdot C_u(s'_u, s_{-u}^{\hat{z}-1}) + (2p+1) \sum_{k=t+1}^{\hat{z}-1} \frac{b_k}{N^c} \\
&\leq p \cdot C_u(s'_u, s_{-u}^{\hat{z}-1}) + \frac{2(2p+1)}{N^c} C_u(s'_u, s_{-u}^{\hat{z}-1}) \\
&\leq p \left(1 + \frac{5}{N^c}\right) C_u(s'_u, s_{-u}^{\hat{z}-1}).
\end{aligned}$$

□

Lemmas 2.18 and 2.22 conclude the proof of the Theorem 2.9 to show that for

$$q = \left(1 + \frac{1}{N^c}\right)$$

and

$$p = \left(\frac{1}{\theta\,(q)} - \frac{1 + 2\lambda + q}{N^c}\right)^{-1}$$

the algorithm computes a α-approximate pure Nash equilibrium in polynomial time, where

$$\begin{aligned}
\alpha &\leq \left(\frac{1}{\theta(q)} - \frac{1 + 2\lambda + q}{N^c}\right)^{-1} \left(1 + \frac{5}{N^c}\right) \\
&= \frac{1}{\left(\frac{1 - \frac{N\lambda}{N^c+1}}{\lambda} - \frac{1+2\lambda+q}{N^c}\right)} \left(1 + \frac{5}{N^c}\right) \\
&= \frac{\lambda}{\left(1 - \frac{N\lambda}{N^c+1} - \frac{\lambda(1+2\lambda+q)}{N^c}\right)} \left(1 + \frac{5}{N^c}\right) \\
&\leq \frac{\lambda}{\left(1 - \frac{N\lambda}{N^c+1} - \frac{\lambda(2\lambda+3)}{N^c}\right)} \left(1 + \frac{5}{N^c}\right),
\end{aligned}$$

choosing $c = 10 \log\left(\frac{\lambda}{\varepsilon}\right)$ we can easily bound

$$\begin{aligned}
\frac{\lambda}{\left(1 - \frac{N\lambda}{N^c+1} - \frac{\lambda(2\lambda+3)}{N^c}\right)} \left(1 + \frac{5}{N^c}\right) &\leq \frac{\lambda}{\left(1 - \frac{\varepsilon}{5} - \frac{\varepsilon}{5}\right)} \left(1 + \frac{\varepsilon}{5}\right) \\
&\leq \lambda(1 + \varepsilon).
\end{aligned}$$

This concludes the proof of correctness for Algorithm 1.

□

2.3.2 Improving the Approximation Factor

In Section 2.3.1 we proved that given a set of cost functions f' satisfying the strong smoothness condition of Lemma 2.8 for some $\lambda > 0$, Algorithm 1 computes a $\lambda(1+\varepsilon)$-approximate pure Nash equilibrium. We now study how one could optimally choose the modified cost functions f' such that value of λ is minimized. Observe that from Lemma 2.8, for any resource $e \in E$, the modified cost functions f'_e can be computed using a linear program. That is, for an objective function Φ_e and a bound on the number of players in $\mathcal{N}$, finding functions f'_e that minimize λ, can be easily solved by the following linear program LP_Φ with the variables $f'_e(0), \dots, f'_e(N+1)$, and λ_e.

$$\begin{aligned} &\min \lambda_e \\ \lambda_e \cdot \sum_{i=z+1}^{m+z} f_e(i) - m f'_e(n+z+1) + n f'_e(n+z) &\geq \sum_{i=z+1}^{n+z} f_e(i) \qquad \forall (n+z), m \in [N]_0 \\ f'_e(n+1) &\geq f'_e(n) \qquad \forall n \in [N]_0 \\ f'_e(n) &\geq 0 \qquad \forall n \in [N+1]_0 \end{aligned}$$

Choosing the output of the linear program LP_Φ as the modified cost functions f' and setting value of $\lambda := \max_{e \in E} \lambda_e$ in Algorithm 1, results in $\lambda(1+\varepsilon)$-approximate pure Nash equilibrium. Moreover, observe that LP_Φ is compact, i.e., the number of constraints and variables are polynomially bounded in the number of players. Hence, we state the following theorem.

Theorem 2.23. *Optimal resource cost functions f'_e for objective functions Φ_e can be computed in polynomial time.*

Linear and Polynomial Cost Functions

Observe that Theorem 2.9 holds for all congestion games with arbitrary non-decreasing cost functions. We now turn to the important class of polynomial cost functions with non-negative coefficients. We can use a standard trick to simplify the analysis for resources with polynomial cost functions of the form $f_e(x) = \sum_{i=0}^{d} a_i x^d$ by replacing such resources by $d+1$ resources with cost functions $a_0, a_1 x, \dots, a_d x^d$ and adjust the strategy sets accordingly. Hence, by an additional scaling argument it suffices to only consider cost functions of the form $f_e(x) = x^i$ and $i \in \{1, \dots, d\}$.

Furthermore, we can show that for polynomials of small degree, it is sufficient to restrict the attention to the first $K = 150$ values of the cost functions. Hence, we only need to solve a linear program of constant size. The following lemma states that for the larger values of n with appropriate values of λ_d and ν, we can easily obtain $(\lambda_d, 0)$-smoothness by choosing $f'(n) = \nu n^d$.

Lemma 2.24. *For $d \leq 5$ and $n \geq 150$, the function $f'(n) = \nu n^d$ with $\nu = \sqrt[d+1]{\lambda_d}$ is strong $(\lambda_d, 0)$-smooth with respect to the potential function Φ for an appropriate λ_d.*

Proof. For each degree d and its associated value of $\lambda_d = \rho_d$ (Table 2.1), we need to show that there exists a ν such that for the smoothness condition in Lemma 2.8, the following holds for all $x \geq 150$,

$$\lambda_d \sum_{i=z+1}^{m+z} i^d - m \cdot \nu \cdot (x+1)^d + (x-z) \cdot \nu \cdot x^d - \sum_{i=z+1}^{x} i^d \geq 0.$$

For all $\lambda_d > 0$,

$$\begin{aligned}
&\lambda_d \sum_{i=z+1}^{m+z} i^d - m\cdot\nu\cdot(x+1)^d + (x-z)\cdot\nu\cdot x^d - \sum_{i=z+1}^{x} i^d \\
&\geq \lambda_d \sum_{i=z+1}^{m+z} i^d - m\cdot\nu\cdot(x+1)^d + (x-z)\cdot\nu\cdot x^d - \int_{z+1}^{x+1} t^d dt \\
&\geq \lambda_d \sum_{i=z+1}^{m+z} i^d - m\cdot\nu\cdot(x+1)^d + (x-z)\cdot\nu\cdot x^d - \frac{(x+1)^{d+1}}{d+1} + \frac{(z+1)^{d+1}}{d+1} \\
&= \lambda_d \sum_{i=z+1}^{m+z} i^d - m\cdot\nu\cdot(x+1)^d + \nu\cdot x^{d+1} - \nu\cdot z\cdot x^d - \frac{(x+1)^{d+1}}{d+1} + \frac{(z+1)^{d+1}}{d+1} \\
&\geq \lambda_d \int_{z+1}^{m+z} t^d dt - m\cdot\nu\cdot(x+1)^d + \nu\cdot x^{d+1} - \nu\cdot z\cdot x^d - \frac{(x+1)^{d+1}}{d+1} + \frac{(z+1)^{d+1}}{d+1} \\
&= \lambda_d \frac{(m+z)^{d+1}}{d+1} - \lambda_d \frac{(z+1)^{d+1}}{d+1} - m\cdot\nu\cdot(x+1)^d + \nu\cdot x^{d+1} - \nu\cdot z\cdot x^d - \frac{(x+1)^{d+1}}{d+1} \\
&\quad + \frac{(z+1)^{d+1}}{d+1}.
\end{aligned}$$

The above expression is minimized for,

$$m = \sqrt[d]{\frac{\nu\cdot(x+1)^d}{\lambda_d}} - z.$$

Substituting for m we obtain,

$$\begin{aligned}
&\geq \lambda_d \frac{\left(\sqrt[d]{\frac{\nu\cdot(x+1)^d}{\lambda_d}}\right)^{d+1}}{d+1} - (\lambda_d - 1)\frac{(z+1)^{d+1}}{d+1} - \left(\sqrt[d]{\frac{\nu\cdot(x+1)^d}{\lambda_d}} - z\right)\cdot\nu\cdot(x+1)^d \\
&\quad + \nu\cdot x^{d+1} - \nu\cdot z\cdot x^d - \frac{(x+1)^{d+1}}{d+1} \\
&= \lambda_d \frac{\left(\sqrt[d]{\frac{\nu\cdot(x+1)^d}{\lambda_d}}\right)^{d+1}}{d+1} - (\lambda_d - 1)\frac{(z+1)^{d+1}}{d+1} - \sqrt[d]{\frac{\nu\cdot(x+1)^d}{\lambda_d}}\cdot\nu\cdot(x+1)^d + z\cdot\nu\cdot(x+1)^d \\
&\quad + \nu\cdot x^{d+1} - \nu\cdot z\cdot x^d - \frac{(x+1)^{d+1}}{d+1} \\
&= (x+1)^{d+1}\left(\left(\frac{1}{d+1} - 1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{\lambda_d}} - \frac{1}{d+1}\right) - \left((\lambda_d - 1)\frac{(z+1)^{d+1}}{d+1} - z\cdot\nu\left((x+1)^d - x^d\right)\right) \\
&\quad + \nu\cdot x^{d+1}.
\end{aligned}$$

The above expression is minimized for,

$$z = \sqrt[d]{\frac{\nu((x+1)^d - x^d)}{\lambda_d - 1}} - 1.$$

Substituting for z we get,

$$\geq (x+1)^{d+1}\left(\left(\frac{1}{d+1} - 1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{\lambda_d}} - \frac{1}{d+1}\right) - \left(\frac{1}{d+1} - 1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{\lambda_d - 1}}\left((x+1)^d - x^d\right)^{\frac{d+1}{d}}$$

$$+\nu \cdot x^{d+1} - \nu\left((x+1)^d - x^d\right).$$

For any $\nu \geq 0$ the above expression is,

$$\geq (x+1)^{d+1}\left(\left(\frac{1}{d+1}-1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{\lambda_d}} - \frac{1}{d+1}\right) + \nu\left(x^{d+1} - (x+1)^d + x^d\right).$$

Now, what is left to show is that there exists a $\nu \geq 0$ such that,

$$(x+1)^{d+1}\left(\left(\frac{1}{d+1}-1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{\lambda_d}} - \frac{1}{d+1}\right) + \nu\left(x^{d+1} - (x+1)^d + x^d\right) \geq 0$$

equivalent to,

$$\nu\left(x^{d+1} - (x+1)^d + x^d\right) \geq (x+1)^{d+1}\left(\left(1-\frac{1}{d+1}\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{\lambda_d}} + \frac{1}{d+1}\right). \tag{2.14}$$

For $\nu = \sqrt[d+1]{\lambda_d}$ and $\forall x \in \mathbb{N}$ such that,

$$\frac{(x+1)^{d+1}}{(x^{d+1} - (x+1)^d + x^d)} \leq \sqrt[d+1]{\lambda_d},$$

the inequality (2.14) holds. From the fact that,

$$\lim_{x\to\infty} \frac{(x+1)^{d+1}}{(x^{d+1} - (x+1)^d + x^d)} = 1,$$

and choice of $\lambda_d = \rho_d$ (Table 2.1), gives for $0 < d \leq 5$, that $x \geq 150$ is sufficient. □

We further note, that for a given $\lambda_d > 0$, for each n and z, we only need to consider a limited range for m.

Lemma 2.25. *For fixed n, z, if $\lambda_d \cdot \sum_{i=z+1}^{m+z} i^d - mf'(n+z+1) + nf'(n+z) \geq \sum_{i=z+1}^{n+z} i^d$ is true $\forall m \leq (n+z+1)^2(d+1)$, it also holds $\forall m > (n+z+1)^2(d+1)$.*

Proof. For any given $n, z \in \mathbb{N}$, and $\lambda_d > 0$, we show for all $m \geq (n+z+1)^2(d+1)$ that,

$$\lambda_d \sum_{i=z+1}^{m+z} i^d - mf'(n+z+1) + nf'(n+z) - \sum_{i=z+1}^{n+z} i^d \geq 0.$$

We first upper bound the feasible values for f'. From the smoothness condition in Lemma 2.8, observe that for $n = z = 0$ and $m = 1$ the inequality implies that $f'(1) \leq \lambda_d$. Furthermore, by the choice of $m = n$, we get

$$\begin{aligned} f'(n+z+1) &\leq (\lambda_d - 1)\sum_{i=1}^{n+z}\left(\frac{1}{i}\sum_{j=1}^{i} j^d\right) + \lambda_d \\ &\leq (\lambda_d - 1)\sum_{j=1}^{n+z} j^d + \lambda_d \\ &\leq (\lambda_d - 1)(n+z)^{d+1} + \lambda_d. \end{aligned}$$

This gives,

$$\begin{aligned}
&\lambda_d \sum_{i=z+1}^{m+z} i^d - mf'(n+z+1) + nf'(n+z) - \sum_{i=z+1}^{n+z} i^d \\
&\geq \lambda_d \sum_{i=z+1}^{m+z} i^d - m\left((\lambda_d - 1)(n+z)^{d+1} + \lambda_d\right) + nf'(n+z) - \sum_{i=z+1}^{n+z} i^d \\
&\geq \lambda_d \sum_{i=z+1}^{m+z} i^d - m\left((\lambda_d - 1)(n+z)^{d+1} + \lambda_d\right) - \sum_{i=z+1}^{n+z} i^d \\
&\geq \lambda_d \sum_{i=z+1}^{m+z} i^d - \lambda_d m(n+z)^{d+1} + m(n+z)^{d+1} - m\lambda_d - (n+z)^{d+1} \\
&= \lambda_d \sum_{i=z+1}^{m+z} i^d - \lambda_d m(n+z)^{d+1} + (m-1)(n+z)^{d+1} - m\lambda_d \\
&\geq \lambda_d \sum_{i=z+1}^{m+z} i^d - \lambda_d m(n+z)^{d+1} - m\lambda_d.
\end{aligned}$$

Now we can bound for $m \geq (d+1)(n+z+1)^2$.

$$\begin{aligned}
&\lambda_d \sum_{i=z+1}^{m+z} i^d - \lambda_d m(n+z)^{d+1} - m\lambda_d \\
&= \lambda_d \sum_{i=z+1}^{m+z} i^d - \lambda_d m((n+z)^{d+1} + 1) \\
&\geq \lambda_d \frac{m(m+z)^d}{d+1} - \lambda_d m((n+z)^{d+1} + 1) \\
&\geq \lambda_d \frac{m^{d+1}}{d+1} - \lambda_d m((n+z)^{d+1} + 1) \\
&\geq 0.
\end{aligned}$$

□

By Lemma 2.24 and 2.25 it remains to solve the following linear program LP_Φ^K to obtain our results ρ_d as listed in Table 2.1 for $d \leq 5$.

$$\begin{aligned}
&\min \rho_d \\
&\rho_d \cdot \sum_{i=z+1}^{m+z} i^d - mf'(n+z+1) + nf'(n+z) \geq \sum_{i=z+1}^{n+z} i^d \qquad \forall (n+z) \in [K-1]_0, \\
&\qquad\qquad \forall m \in [(K+1)^2(d+1)]_0 \\
&f'(n+1) \geq f'(n) \qquad \forall n \in [K-1]_0 \\
&f'(K) \leq \nu K^d \\
&f'(n) \geq 0 \qquad \forall n \in [K]_0
\end{aligned}$$

Corollary 2.26. *For every congestion game with polynomial cost functions of degree $d \leq 5$ and for every constant $\varepsilon > 0$, the algorithm computes a $(\rho_d + \varepsilon)$-approximate pure Nash equilibrium in polynomial time.*

Chapter 3

Inefficiency of Equilibria under Universal Taxation

In the previous chapter we presented an interesting extension to Roughgarden's [Rou15] (λ, μ)-smoothness framework to obtain cost functions that enabled us to compute improved approximate pure Nash equilibria in congestion games. In this chapter we study yet another interesting method based on our analysis from Chapter 2 to compute robust load dependent taxes that improves the price of anarchy (PoA) in congestion games using a constant size linear program and then using linear programming duality prove lower bounds on the PoA under universal taxation.

3.1 Introduction

We introduced congestion games and the notion of price of anarchy (PoA) [KP99] in Chapter 1. Recall that the PoA measures the inefficiency of equilibria in strategic games. The last two decades saw various authors, e.g., [CK05a, AAE05, ADG$^+$11, Rou15, RT00, Rou05] investigating the bounds on the price of anarchy for various non-cooperative games [NRTV07] and several attempts to improve the inefficiency of their self-emerging solutions. One of the many approaches used to improve the PoA is the introduction of taxes [CKK06, FKK10, FS07]. Influence of taxes on the selfish behavior of players to improve inefficiency of equilibria was first studied in non-atomic selfish routing games [NRTV07, p. 462] by Cole et al. [CDR06], where they investigate the influence of economic incentives and its ability to induce desirable solutions. The atomic [NRTV07, p. 465] variant of selfish routing games with unweighted instances are often referred to as congestion games. Recall that congestion games are cost-minimization games that constitute an important class of games to model resource allocation among non-cooperative users. Analyzing and improving the inefficiency of equilibria in these games has been of considerable interest in the scientific community [CKK06, FKK10, FS07, Swa12].

As described in Chapter 2, a congestion game denoted by the tuple

$$G = \left(\mathcal{N}, E, (S_u)_{u\in N}, (f_e)_{e\in E}\right)$$

consists of a finite set of players $\mathcal{N} = \{1, 2, \ldots, N\}$, who compete over a finite set of resources $E = \{e_1, e_2, \ldots, e_m\}$. Each player $u \in \mathcal{N}$ has a set of strategies denoted by $S_u \subseteq 2^E$. Each resource $e \in E$ has a non-negative and non-decreasing cost function

$f_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ associated with it. The cost contributed by a resource $e \in E$ to each player using it is denoted by $f_e(n_e(s))$, where $n_e(s)$ denotes the number of players on a resource $e \in E$ in the state s. Therefore, the cost of a player $u \in \mathcal{N}$ in a state $s = (s_1, \ldots, s_N) \in \times(S_u)_{u \in \mathcal{N}}$ of the game is given by

$$C_u(s) = \sum_{e \in E : e \in s_u} f_e(n_e(s)),$$

and the social cost of the game in the state s is denoted by

$$C(s) = \sum_{u \in \mathcal{N}} C_u(s).$$

A state $s \in S$ is called a pure Nash equilibrium if for all players $u \in \mathcal{N}$ and for all $s'_u \in S_u$,

$$C_u(s) \leq C_u(s'_u, s_{-u}),$$

where $C_u(s'_u, s_{-u})$ denotes the cost of player u when only u deviates.

For a set of resources E, the load dependent tax function $t : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$, is the excess cost incurred by the players on a resource $e \in E$ with cost $f_e(x)$, e.g.,

$$f'_e(x) = f_e(x) + t_e(x).$$

3.1.1 Our Contribution

We present load dependent tax functions for congestion games that are universal, i.e., robust against perturbations of the instance such as addition or removal of resources or players. Our approach in Chapter 2 yields a simple and distributed method to compute load dependent universal taxes that improves the inefficiency of equilibria in congestion games. In [Vij17] the author computes robust load dependent taxes that weakly improve the PoA in congestion games with cost functions that are polynomials of maximum degree d. Particularly, in [Vij17] the author shows that for polynomials of maximum degree $d = 1, 2$, and 3, the PoA under universal load dependent taxes is at most $2.1, 5.18$, and 17.3, respectively. They compute tax functions that achieve the aforementioned bounds using an LP formulation for which the constraints are derived based on an extention of a rather complicated technique presented in Feldotto et al. [FGS14]. Here, we extend their work to achieve much stronger bounds on the PoA under universal taxation for congestion games with polynomial cost functions of maximum degree d, using a simple and elegant technique compared to that in [Vij17].

In Section 3.2 we derive tax functions using an LP for which the necessary constraints are obtained from the $(\lambda, 0)$-smoothness condition described in Lemma 2.6, which is based on the famous (λ, μ)-smoothness framework introduced by Roughgarden [Rou15]. We prove necessary conditions required to compute the tax functions using only a constant size LP for resource cost functions that are polynomials of maximum degree $d \leq 5$. Table 3.1 lists our results for PoA under refundable taxation for resource cost functions that are bounded degree polynomials. The values for Ψ_d in Table 3.1 were obtained using the python program in Listing A.2. Furthermore, in Section 3.3 we prove lower bounds on the PoA with universal load dependent taxes, using a construction derived from dual of the LP. This construction shows that PoA under universal taxation is at least the PoA of selfish scheduling on identical machines. Numerically, our lower bounds match the upper bounds presented in Section 3.2.

We remark that the taxes we consider in Section 3.2 are refundable [CDR06] and do not contribute to the overall cost of the game. Bilò and Vinci [BV16] present an algorithm to compute load dependent taxes that improve the price of anarchy e.g., for linear congestion games from 2.5 to 2. Although our methods yield slightly weaker results, our tax functions are locally computable and in contrast to [BV16], are independent of the actual instance of the game. We would like to remark that our results for PoA were achieved independently of that in Paccagnan et al. [PCFM20] (preprint) that used a similar technique.

d	PoA without Taxes Aland et al. [ADG+11]	Optimal Taxes Bilò and Vinci [BV16]	**Universal Taxes** Ψ_d Local Search w.r.t. ζ_{SC}
1	2.5	2	2.012
2	9.583	5	5.10
3	41.54	15	15.56
4	267.6	52	55.46
5	1514	203	220.41

Table 3.1: PoA under taxation in congestion games with polynomial cost functions of degree at most d.

3.1.2 Related Work

Meyers and Schulz [MS12] study the complexity of computing an optimal solution in a congestion game and prove NP-hardness. Makarychev and Sviridenko [MS14] give the best known approximation algorithm using randomized rounding on a natural feasibility LP with an approximation factor of $\mathcal{B}_{d+1}$ which is the $(d+1)^{\text{th}}$ Bell number, where d is the maximum degree of the polynomial cost function. Interestingly, the same was later achieved using load dependent taxes by Bilò and Vinci [BV16], where they apply the *primal-dual* method [Bil18] to upper bound the PoA under refundable [CDR06] taxation in congestion games. They determine load specific tax functions and show that the PoA is at most $O\left(d/\log d\right)^{d+1}$ with respect to ε-approximate equilibrium. However, we remark that the load dependent taxes computed in [BV16] are not universal, i.e., they are sensitive to the instance of the game.

3.2 Load Dependent Universal Taxes

We now look at an extension of Lemma 2.6 for computing load dependent universal taxes in congestion games. We give a rather simple approach to locally (on resource) compute load dependent universal taxes. Table 3.1 lists the improved PoA bounds under refundable taxation using our technique for congestion games with resource cost functions that are bounded degree polynomials of maximum degree d. By the smoothness argument (Theorem 2.4, [Rou15]) the new bounds immediately extends to mixed, (coarse) correlated equilibria, and outcome generated by no-regret sequences. Moreover, since the linear program that computes the cost or tax function does only depend on the original cost function of that resource, the computed taxes are robust against perturbations of the instance such as adding or removing of resources or players.

We seek to compute universal load dependent taxes that minimize the PoA under refundable taxation. Therefore, we consider the following optimization problem. For an objective function $h(s) = \sum_{e\in E} n_e(s) \cdot f_e(n_e(s))$, find functions f'_e that satisfies Lemma 2.6 minimizing λ. For a resource objective function $h_e(n_e(s)) = n_e(s) \cdot f_e(n_e(s))$ and a bound on the number of players in $\mathcal{N}$, this can be easily solved by the following linear program $\mathrm{LP}_{\mathrm{SC}}$ with the variables $f_e'(0), \ldots, f_e'(N+1)$, and λ_e.

$$\begin{aligned} & \min \lambda_e \\ \lambda_e \cdot h_e(m) - m f_e'(n+1) + n f_e'(n) &\geq h_e(n) && \text{for all } n \in [N]_0, m \in [N]_0 \\ f_e'(n+1) &\geq f_e'(n) && \text{for all } n \in [N]_0 \\ f_e'(n) &\geq 0 && \text{for all } n \in [N+1]_0 \end{aligned}$$

Observe that, we can solve $\mathrm{LP}_{\mathrm{SC}}$ locally for each resource with cost function f_e. For the LP solution λ_e and $f_e'(n)$, define the tax function as $t_e(n) := f_e'(n) - f_e(n)$. The resulting price of anarchy under taxation is then $\lambda := \max_{e \in E} \lambda_e$.

For any (distributed) local search algorithm (such as Bjelde et al. [BKS17]) that seeks to minimize the social cost $C(s) = \sum_{e \in E} n_e(s) \cdot f_e(n_e(s))$, we define

$$\zeta_{\mathrm{SC}}(s) := \sum_{e \in E} \sum_{i=1}^{n_e(s)} f_e'(i)$$

as a pseudo-potential function. Recall that a pure Nash equilibrium is a local optimum of the potential function and similarly, a local optimum to ζ_{SC} is a pure Nash equilibrium of a game in which we change the resource cost functions to f'. Then, from Lemma 2.6 it is guaranteed that every local optimum with respect to ζ_{SC} has an approximation factor of at most $\lambda := \max_{e \in E} \lambda_e$ with respect to the social cost $C(s)$. Using approximate local search by Orlin et al. [OPS04] w.r.t. ζ_{SC}, we can compute a solution close to that in polynomial time and more so to state the following.

Corollary 3.1. *For every congestion game the ε-local search algorithm using ζ_{SC}, produces a $\lambda(1+\varepsilon)$ local optimum in running time polynomial in the input length, and $1/\varepsilon$.*

Linear and Polynomial Cost Functions

For the interesting case of polynomial resource cost functions of maximum degree d, similar to Section 2.3, we show that for polynomials of small degree, it is sufficient to restrict the attention to the first $K = 1154$ values of the cost functions. Hence, we only need to solve a linear program of constant size. The following lemma states that for the values of n greater than K and an appropriate value of ν and λ_d, we can easily obtain $(\lambda_d, 0)$-smoothness by choosing $f'(n) = \nu n^d$.

Lemma 3.2. *For $d \leq 5$ and $n \geq 1154$, the function $f'(n) = \nu n^d$ with $\nu = \sqrt[d+1]{(d+1)\lambda_d}$ is $(\lambda_d, 0)$-smooth with respect to $h(n) = n^{d+1}$ and an appropriate λ_d.*

Proof. For each degree d and an appropriate value of $\lambda_d = \Psi_d$ (Table 3.1), we need to show that there exist a ν such that the the smoothness condition in Lemma 2.6 holds for all $n \geq 1154$, i.e.,

$$\lambda_d m^{d+1} - m\nu(n+1)^d + n\nu n^d - n^{d+1} \geq 0.$$

For all $\lambda_d > 0$,

$$\begin{aligned} & \lambda_d m^{d+1} - m\nu(n+1)^d + n\nu n^d - n^{d+1} \\ &= \lambda_d m^{d+1} - m\nu(n+1)^d + n^{d+1}(\nu - 1). \end{aligned}$$

The above expression is minimized at,

$$m = \sqrt[d]{\frac{\nu(n+1)^d}{(d+1)\lambda_d}}.$$

Substituting for m gives,

$$\begin{aligned}
&= \lambda_d \left(\sqrt[d]{\frac{\nu(n+1)^d}{(d+1)\lambda_d}}\right)^{d+1} - \left(\sqrt[d]{\frac{\nu(n+1)^d}{(d+1)\lambda_d}}\right)\nu(n+1)^d + n^{d+1}(\nu-1) \\
&= \left(\frac{1}{d+1}\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{(d+1)\lambda_d}}(n+1)^{d+1} - \frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{(d+1)\lambda_d}}(n+1)^{d+1} + n^{d+1}(\nu-1) \\
&= \left(\frac{1}{d+1} - 1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{(d+1)\lambda_d}}(n+1)^{d+1} + n^{d+1}(\nu-1).
\end{aligned}$$

We need to show that there exists a $\nu \geq 0$ such that,

$$\left(\frac{1}{d+1} - 1\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{(d+1)\lambda_d}}(n+1)^{d+1} + n^{d+1}(\nu-1) \geq 0$$

equivalent to,

$$n^{d+1}(\nu-1) \geq \left(1 - \frac{1}{d+1}\right)\frac{\nu^{\frac{d+1}{d}}}{\sqrt[d]{(d+1)\lambda_d}}(n+1)^{d+1}. \tag{3.1}$$

For $\nu = \sqrt[d+1]{(d+1)\lambda_d}$ and $\forall n \in \mathbb{N}_{\geq 0}$ such that,

$$\frac{(n+1)^{d+1}}{n^{d+1}}\left(1 - \frac{1}{d+1}\right) \leq \left(\sqrt[d+1]{(d+1)\lambda_d} - 1\right),$$

the inequality (3.1) holds. Also, using the fact that,

$$\lim_{n\to\infty} \frac{(n+1)^{d+1}}{n^{d+1}} = 1,$$

and choice of $\lambda_d = \Psi_d$ (Table 3.1), gives for $0 < d \leq 5$, that $n \geq 1154$ is sufficient. □

We further note, that for a fixed $\lambda_d > 0$ and for each n, we only need to consider a limited range for m in the LP$_{\text{SC}}$.

Lemma 3.3. *For a fixed n, if $\lambda_d \cdot m^{d+1} - mf(n+1) + nf(n) \geq n^{d+1}$ is true for all $m \leq (n+1)^2$, it also holds for all $m > (n+1)^2$.*

Proof. For any given $n \in \mathbb{N}$ and $\lambda_d > 0$, we show for all $m \geq (n+1)^2$ that,

$$\lambda_d m^{d+1} - mf'(n+1) + nf'(n) - \sum_{i=1}^{n} n^{d+1} \geq 0.$$

We first upper bound the feasible values for $f'(n)$. From the smoothness condition in Lemma 2.6, note that for $n = 0$ and $m = 1$, implies that $f'(1) \leq \lambda_d$. Furthermore, by the choice of $m = n$, we get $f'(n+1) \leq f'(n) + \frac{\lambda_d - 1}{n} n^{d+1}$. By recursion, we obtain

$f'(n+1) \leq (\lambda_d-1)\sum_{i=1}^n i^d+\lambda_d$ which we can simply bound by $f'(n+1) \leq (\lambda_d-1)\cdot n^{d+1}+\lambda_d$. Now we can bound for $m \geq (n+1)^2$.

$$\begin{aligned}
&\lambda_d m^{d+1} - mf'(n+1) + nf'(n) - n^{d+1}\\
&\geq \lambda_d m^{d+1} - m\left((\lambda_d-1)n^{d+1} + \lambda_d\right) - n^{d+1}\\
&= \lambda_d m^{d+1} - m\lambda_d n^{d+1} + mn^{d+1} - m\lambda_d - n^{d+1}\\
&\geq \lambda_d m^{d+1} - m\lambda_d n^{d+1} - m\lambda_d\\
&\geq 0
\end{aligned}$$

□

As a consequence of Lemma 3.2 and 3.3 it only remains to solve the following linear program of constant size for each $d \leq 5$ to obtain our results Ψ_d (listed in Table 3.1). Our results match the recent results that were obtained independently by Paccagnan et al. [PCFM20].

$$\begin{aligned}
&\min \Psi_d\\
\Psi_d m^{d+1} - mf'(n+1) + nf'(n) &\geq n^{d+1} && \forall n \in [K-1]_0, m \in [(K+1)^2]_0\\
f'(K) &\leq \nu K^d\\
f'(n+1) &\geq f'(n) && \forall n \in [K-1]_0\\
f'(n) &\geq 0 && \forall n \in [K]_0
\end{aligned}$$

Corollary 3.4. *For every congestion game with polynomial cost functions of degree $d \leq 5$, each cost function f'_e can be computed in constant time and the resulting game is $(\Psi_d, 0)$-smooth with respect to social cost.*

3.3 Lower Bound

We now present an interesting connection between the PoA of congestion games under universal load dependent taxation and the PoA of selfish scheduling games. Let us recall that Lemma 2.6 gives rise to the following optimization problem, i.e., given an objective function

$$h(s) = \sum_{e\in E} h_e(n_e(s)),$$

find functions f'_e that minimize λ. For a resource objective function h_e and a bound on the number of players i.e., N, this can be computed using the following linear program LP$_h$ with the variables $f'_e(1), \dots, f'_e(N+1)$ and λ_e.

$$\begin{aligned}
&\min \lambda_e\\
\lambda_e \cdot h_e(m) - mf'_e(n+1) + nf'_e(n) &\geq h_e(n) && \text{for all } n \in [N]_0, m \in [N]_0\\
f'_e(n+1) &\geq f'_e(n) && \text{for all } n \in [N]_0\\
f'_e(n) &\geq 0 && \text{for all } n \in [N+1]_0
\end{aligned}$$

Any feasible solution to the linear program LP$_h$ emerging from Lemma 2.6 are cost functions $f'_e : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ that guarantees that the objective value associated with the function h is at most $\lambda := \max_{e\in E} \lambda_e$.

We now prove a lower bound on the PoA of congestion games with universal taxation. To that end, we consider the dual program to the relaxed LP_h, i.e., we omit the constraints that the functions f' must be non-decreasing and show that for every feasible solution of the dual, we can construct an instance of a selfish scheduling game with an objective value that is equal to the value of the dual LP solution, regardless of the actual cost function of the resources in the game.

We now construct an instance of selfish scheduling game on identical machines [CFK+06]. This is a congestion game in which players' strategies are singletons. Furthermore, each resource has the same cost function and hence, players only seek to choose a resource with minimal load. Obviously, every equilibrium in the scheduling game is an equilibrium in a congestion game in which the resource cost function is an arbitrary non-decreasing function. The dual program LPD_h is as follows:

$$\max \sum_{n=0}^{N}\sum_{m=0}^{N} h(n)\cdot y_{n,m} \tag{3.2}$$

$$\sum_{n=0}^{N}\sum_{m=0}^{N} h(m)\cdot y_{n,m} \le 1 \tag{3.3}$$

$$\sum_{m=0}^{N} n\cdot y_{n,m} - \sum_{m=0}^{N} m\cdot y_{n-1,m} \le 0 \qquad \text{for all } n\in[N]$$

$$y_{n,m} \ge 0 \qquad \text{for all } n,m\in[N]_0$$

Lemma 3.5. *Every optimal solution of LPD_h with objective value λ can be turned into an instance of selfish scheduling on identical machines with an objective value of $\lambda-\varepsilon$ for an arbitrary $\varepsilon>0$.*

Proof. Let $\tilde{y}$ be a feasible solution to LPD_h with $\lambda=\sum_{n=0}^{N}\sum_{m=0}^{N} h(n)\cdot\tilde{y}_{n,m}$. We round down each $\tilde{y}_{n,m}$ to rational numbers $y_{n,m}\ge(1-\frac{\varepsilon}{\lambda})\tilde{y}_{n,m}$ and let M be a sufficiently large scaling factor such that each $y_{n,m}\cdot M$ is an integer. We construct a congestion game as follows. The game $G=(\mathcal{N},R,\{S_u\}_{u\in\mathcal{N}},\{c_r\}_{r\in R})$ consist of a set of players $\mathcal{N}=\bigcup_{n\in[0,N]}\mathcal{N}_n$, where each set $\mathcal{N}_n$ consists of $n\cdot\sum_{m=0}^{N} y_{n,m}\cdot M$ many players. The set of resources is $R=\bigcup_{n\in[0,N],m\in[0,N]} R_{n,m}$, where $R_{n,m}$ represents a pool consisting of $y_{n,m}\cdot M$ identical machines.

We will make sure that in the game G, there is an equilibrium s^* in which on each machine in each set $R_{n,m}$ there are exactly n many players. Hence, using (3.2) the total cost over all resources in s^* is

$$\sum_{n=0}^{N}\sum_{m=0}^{N} h(n)\cdot y_{n,m}\cdot M \ge \lambda(1-\varepsilon/\lambda)M.$$

Additionally, there is a state s in which on each machine in each set $R_{n,m}$ there are exactly m many players. Hence, using (3.3) the total cost over all resources in s is

$$\sum_{n=0}^{N}\sum_{m=0}^{N} h(m)\cdot y_{n,m}\cdot M \le M.$$

Each player from a set $\mathcal{N}_n$ in the game has two strategies, which we call an equilibrium strategy and an optimal strategy. The equilibrium strategy consists of one particular resource $r\in\bigcup_{m\in[N]_0} R_{n,m}$ and the optimal strategy of a particular resource

$r \in \bigcup_{m\in[N]_0} R_{n-1,m}$. The assignment of resources to strategies is such that each resource $r \in R_{n,m}$ belongs to exactly n equilibrium strategies of players from $\mathcal{N}_n$ and at most m optimal strategies of players from $\mathcal{N}_{n+1}$. Note that the existence of such an assignment is guaranteed by the feasibility of y and hence,

$$\sum_{m=0}^{N} n \cdot y_{n,m} \leq \sum_{m=0}^{N} m \cdot y_{n-1,m}.$$

If each player chooses the equilibrium strategy, we obtain a state s^* as described above which is a pure Nash equilibrium as switching to the optimal strategy yields exactly the same load. If each player chooses the optimal strategy we obtain a state s as described above. Therefore, the objective value is at least $\lambda - \varepsilon$. □

Evidently our upper bound of 2.012 for congestion games with linear cost functions matches the price of anarchy bound for selfish scheduling games on identical machines in Caragiannis et al. [CFK$^+$06]. Numerically, the optimal solution to LPD$_h$ matches the optimal solution to LP$_h$. That is, LP$_h$ is not only optimizing the smoothness inequality, but also that there exists no other resource cost function that can guarantee a smaller objective value than λ. Based on this observation we conjecture that the taxes computed by LP$_{\text{SC}}$ are in fact optimal.

Chapter 4

Scheduling with Machine-Dependent Priority Lists

We now introduce a fairly new variant of a scheduling problem on parallel machines based on fixed order local scheduling rules. To the best of our knowledge, our variant is closely related to many of the existing models with local precedence constraints in the literature [BFO$^+$19, AGA99, ANCK08, All15, BV20]. In this chapter we study the model from an optimization perspective. In later chapters we extend this variant of the scheduling problem to a strategic model and investigate various algorithmic aspects from a decentralized perspective.

4.1 Introduction

Scheduling problems are considered to be a fundamental and well studied area in theoretical computer science. A typical scheduling problem instance involves assigning a set of n independent jobs $1, 2, \ldots, n$ to m machines $M_1, M_2, \ldots, M_m$, such that the assignment minimizes a predefined objective, e.g., makespan of the schedule, total job completion time, etc. The underlying problem then essentially involves two steps:

1. Allocating jobs to machines optimally.
2. Determining the optimal sequence in which jobs assigned to a machine are processed.

We refer the readers to [Pin08] for a comprehensive literature review on various models of scheduling problems. Here, we study a simple, yet challenging variant of the parallel machine scheduling problem that often arises in the area of transportation and infrastructure allocation. We consider priority based scheduling on parallel machines. In priority based scheduling the request of some jobs are preferred over others, i.e., each machine imposes a preferential order over the jobs assigned to it. The order may be machine dependent and need not be restricted to general rules such as fair cost sharing, first-in first-out, shortest processing time first, etc. Many real-world applications such as airline boarding, vehicle parking, distributed computing, etc., can be succinctly modeled using the aforementioned variant. To the best of our knowledge, there is very little known in the literature about the model we consider from a theoretical perspective.

A priority based scheduling problem is given by a set of jobs $\mathcal{N}$ and a set of machines M, each with a priority list π that determines its preference on the order in which jobs assigned to it are processed. Each job $j \in \mathcal{N}$ has a processing time of p_{ij} on machine M_i and the goal is to determine an assignment of jobs to machines that minimize the total completion time, such that the schedule respects the ordering of jobs on individual machines. A fixed order scheduling problem can be seen as a special case of scheduling problem with sequence dependent setup times, where for each pair of jobs there is a machine dependent change over cost c_{ikj}, i.e., the additional cost paid by job j if it must be scheduled after job k on machine i. Then, our problem can be succinctly described with $c_{ijk} \in \{0, \infty\}$.

When every machine has an identical (global) priority ordering over the jobs, then the problem can also be viewed as determining m disjoint subsets $J_1, J_2, \dots, J_m$ of jobs that minimizes total sum of completion times. Interestingly, in case of identical machines the problem can also be formulated as the following clustering problem. For every machine M_i, let G_i be an undirected complete graph in which $V = \mathcal{J}$ and $E = V \times V$ (including self-loops). The edge (u, v) has weight p_u if $\pi_i(u) \preceq \pi_i(v)$ and weight p_v, otherwise. For example, Figure 4.1(i) illustrates the graph for $\mathcal{J} = \{a, b, c, d\}$, with processing times $p = (5, 6, 1, 2)$ and a global priority list $\pi = (a, b, c, d)$.

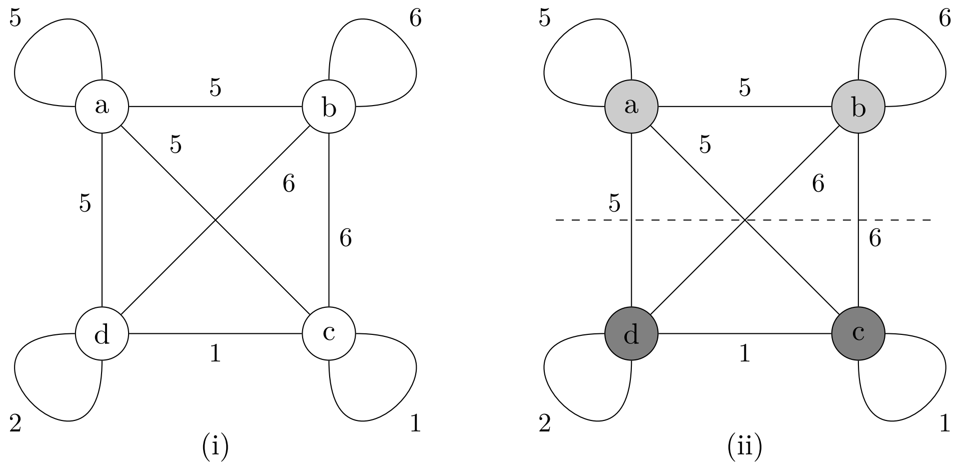

Figure 4.1: Clustering Example.

An assignment on m machines corresponds to a coloring $\chi : V \to \{1, \dots, m\}$ of the vertices, such that the vertices colored i, correspond to jobs assigned on machine i. Then, the contribution of machine i to the total completion time is given by the total weight of the (complete) subgraph of the vertices for which $\chi(v) = i$ in G_i. With a global priority list, all the graphs are identical, and an assignment is just a clustering of nodes in G to m partitions. For example, the optimal solution for the above instance and $m = 2$ is given in Figure 4.1(ii).

4.1.1 The Model

An instance of *scheduling with machine-dependent priority lists* is given by the tuple

$$(\mathcal{N}, M, (p_{ij})_{i \in M, j \in \mathcal{N}}, (\pi_i)_{i \in M}),$$

where $\mathcal{N}$ is a finite set of $n \geq 1$ jobs, M is a finite set of $m \geq 1$ parallel unrelated machines, $p_{ij} \in \mathbb{R}_{\geq 0}$ is the processing time of job $j \in \mathcal{N}$ on machine $i \in M$, and $\pi_i : \mathcal{N} \mapsto \{1, \ldots, n\}$ is the priority list of machine $i \in M$. A schedule $\sigma = (\sigma_j)_{j \in \mathcal{N}} \in M^{\mathcal{N}}$ assigns a machine $\sigma_j \in M$ to every job $j \in \mathcal{N}$. The jobs must be processed without preemption. We denote the problem using the standard three field notation as $R|\pi| \sum C_j$. Also, $R|\pi_{global}| \sum C_j$ denotes the instance when all machines have an identical priority list.

Given a schedule σ, the jobs are processed according to their order in the machines' priority lists. The completion time of a job $j \in \mathcal{N}$ assigned to machine $i \in M$ is given by,

$$C_j := \sum_{\substack{k \in \mathcal{N} \\ \sigma_j = \sigma_k \ \wedge \ \pi_i(k) \preceq \pi_i(j)}} p_{ik}.$$

The total completion time of schedule σ is then given by,

$$C(\sigma) = \sum_{j \in \mathcal{N}} C_j.$$

Let J_i be the set of jobs assigned to machine i in a given schedule σ. Then, the completion time of a job is equal to the total processing time of the jobs preceding it, and its own processing time. Therefore, the total completion time on M_i is given by

$$\sum_{j \in J_i} \sum_{k \in J_i : \pi_i(k) \preceq \pi_i(j)} p_{ik}.$$

As a consequence, we can formulate $R|\pi| \sum C_j$ as the following optimization problem. The variable $x_{ij} \in \{0, 1\}$ indicates whether job j is assigned to machine i.

$$\begin{aligned} \min \sum_{j \in \mathcal{N}} \sum_{i \in M} \sum_{k : \pi_i(k) \preceq \pi_i(j)} & x_{ij} \cdot x_{ik} \cdot p_{ik} & \\ \sum_{i \in M} x_{ij} &= 1 & \text{for all } j \in \mathcal{N} \\ x_{ij} &\in \{0, 1\} & \text{for all } i \in M, j \in \mathcal{N} \end{aligned}$$

Since the priority lists can be arbitrary, approximation techniques such as [Sku01] which additionally restrict a machine's priority list to the shortest processing time first order, are not immediately applicable on the above quadratic program. We would like to remark that unlike the traditional scheduling problem with precedence constraints [LK78], e.g., $P|prec| \sum C_j$, our model only imposes local precedence constraints to the subset of jobs assigned to a machine.

4.1.2 Our Contribution

The traditional scheduling problem that minimizes the total completion time can be solved optimally. For parallel machines, optimal solutions to these problems schedule jobs on any given machine according to the shortest processing time first rule. For unrelated machines, the problem can be solved by means of a matching problem. We study the computational impact due to imposing a fixed order on each machine.

Table 4.1 lists our main results. In Section 4.2 we show computational hardness of scheduling with machine-dependent priority list. For identical machines, we show that the problem is NP-hard when each machine has an arbitrary priority list. However, when

Instance class \ Type	Global List	Arbitrary List
$P\|\pi\|\sum C_j$	QPTAS	NP-hard/APX-hard
$R\|\pi\|\sum C_j$	NP-hard/APX-hard	NP-hard/APX-hard
$R2\|\pi\|\sum C_j$	P	–

Table 4.1: Hardness of scheduling with machine-dependent priority lists.

machines have an identical priority list, we show that a QPTAS exists by proving that the problem can be reduced to an instance of a problem minimizing the weighted completion time, where all jobs have unit processing time. For unrelated parallel machines, we show that the problem is NP-hard even when all machines have identical priority lists. Furthermore, in Section 4.3 we study the non-approximability of the aforementioned problem. We show that problem is APX-hard. In case of unrelated machines, we show that the problem is APX-hard even if all machines have identical priority lists. In case of two unrelated machines and identical priority lists, we show that the problem can be solved optimally using dynamic programming.

4.1.3 Related Work

To the best of our knowledge, the impact of machine-dependent priority list on scheduling problems has only been considered marginally in the literature. The problem without priority list can be solved in polynomial time. The shortest processing time first (SPT) rule is optimal for $P||\sum C_j$ [Smi56, Pin08]. For $R||\sum C_j$ Horn [Hor73] and Bruno et al. [BCS74] show that the problem can solved in polynomial time using bipartite matching. In case of identical priority lists, the problem can also be modeled as a special case of scheduling with sequence dependent setup time. We refer the readers to [AGA99, ANCK08, All15] for a comprehensive survey on scheduling with setup times. Bosman et al. [BFO+19] study fixed order scheduling minimizing weighted total completion time, where all machines have an identical priority lists. They show a QPTAS for the problem when all jobs have unit processing time. For the general case they show a LP based randomized rounding algorithm that has a constant factor approximation guarantee of 29.1.

4.2 Computational Hardness

Identical Machines

We prove the hardness of solving $P|\pi|\sum C_j$ optimally using a reduction from the Maximum Bounded 3-Dimensional Matching (Max-3DM-3) problem. Max-3DM-3 is known to be NP-hard [Kan91].

Maximum Bounded 3-Dimensional Matching:
The input to the Max-3DM-3 problem is a set of triplets $T \subseteq X \times Y \times Z$, where $|T| \geq n$ and $|X| = |Y| = |Z| = n$. The number of occurrences of every element of $X \cup Y \cup Z$ in T is at most 3. The goal is to decide whether T has a 3-dimensional matching of size n, i.e., there exists a subset $T' \subseteq T$, such that $|T'| = n$, and every element in $X \cup Y \cup Z$ appears exactly once in T'.

Theorem 4.1. *The problem $P|\pi|\sum C_j$ is NP-hard.*

Proof. The hardness proof is by a reduction from Max-3DM-3. Given an instance of MAX-3DM-3 matching and $0 < \varepsilon < 1/9n$, we construct the following instance of $P|\pi|\sum C_j$. There are $m = |T|$ identical machines, $M_1, M_2, \ldots, M_{|T|}$. The set of jobs $\mathcal{N}$ consists of:

1. $3n$ jobs with processing time ε—one for each element in $X \cup Y \cup Z$.
2. A set D of $m - n$ jobs with processing time 1.
3. A set U of m jobs with processing time 2.

The priority lists are defined as follows. For every triplet $e = (x_i, y_j, z_k) \in T$, the priority list of the machine M_e is $(D, x_i, y_j, z_k, U, X \setminus \{x_j\}, Y \setminus \{y_j\}, Z \setminus \{z_j\})$. Note that a set in the priority list implies that all the jobs in the set appear in a fixed arbitrary order. We now show that for every $0 < \varepsilon < 1/9n$, it is NP-hard to decide whether the scheduling problem instance has an assignment with sum of completion times less than $4m - 2n + 1$ or at least $4m - 2n + 1 + 3n\varepsilon$.

Claim 4.2. *If a* $3D$*-matching of size* n *exists, then the sum of completion times is less than* $4m - 2n + 1$.

Proof. Let T' be a matching of size n. For every $e \in T'$, assign the three jobs of e on M_e. Also, assign one job from D on each of the remaining $m - n$ machines. Finally, assign one job from U on every machine.

Observe that, every machine with a triplet contributes $(1 + 2 + 3)\varepsilon + 2 + 3\varepsilon = 2 + 9\varepsilon$ to the total completion time. Every other machine contributes $1 + 3 = 4$ to the total completion time. All together we get that for $0 < \varepsilon < 1$,

$$\sum_{j \in \mathcal{N}} C_j = 4m - 2n + 9n\varepsilon < 4m - 2n + 1.$$

□

Claim 4.3. *If a* $3D$*-matching of size* n *does not exist, then the sum of completion times is at least* $4m - 2n + 1 + 3n\varepsilon$.

Proof. Let us first consider the schedule only with the m U-jobs of length 2 and the $m - n$ D-jobs of length 1. Their contribution to the total completion time is at least $4m - 2n$, since in an optimal SPT order of these jobs there are n machines with only one U-job and $m - n$ machines with one U- and one D-job. Also, in any schedule in which some machine processes two U-jobs or two D-jobs, the contribution of the U-jobs and the D-jobs to the total completion time is at least $4m - 2n + 1$.

Since each of the jobs originating from $X \cup Y \cup Z$ contributes at least its own length to the total completion time, in order to have total completion time less than $4m - 2n + 1 + 3n\varepsilon$, it must be that no machine processes two U- or two D-jobs. Thus, $m - n$ machines process exactly one D-job each, and every machine processes exactly one U-job.

Given that a 3D-matching does not exist, in every schedule, the jobs originated from $X \cup Y \cup Z$, either spread on more than n machines, or at least one of them is processed after a U-job on some machine. In the first case, some machine processes both a D-job and some ε-job. By the aforementioned priority lists, the ε-job is processed after the D-job, and thus contributes at least $1 + \varepsilon$ to the total completion time. In the second case, some ε-job is processed after a U-job, and thus contributes at least $2 + \varepsilon$ to the total completion time. We get that the total completion time is at least $4m - 2n + 1 + 3n\varepsilon$. □

Theorem 4.1 follows immediately from Claims 4.2 and 4.3. □

Next, we show that for the case when all machines have an identical priority list, the problem $P|\pi_{global}|\sum C_j$ exhibits a QPTAS. Let us denote by $P|p_j = 1, \pi_{global}|\sum w_j C_j$ a weighted instance of the problem $P|\pi_{global}|\sum C_j$, where all jobs have unit processing time. Also, denote by $\widetilde{\pi}$ the priority list π in a reversed order. Then we show that the problem $P|\pi_{global}|\sum C_j$ can be reduced to an instance of the problem $P|p_j = 1, \pi_{global}|\sum w_j C_j$ in polynomial time.

Theorem 4.4. *$P|\pi_{global}|\sum C_j =_p P|p_j = 1, \pi_{global}|\sum w_j C_j$.*

Proof. An instance $(p_1, \ldots, p_n)$ and π of $P|\pi_{global}|\sum C_j$ has a polynomial time reduction to an instance $(w_1, \ldots, w_n)$ and $\widetilde{\pi}$ of $P|p_j = 1, \pi_{global}|\sum w_j C_j$ as follows: $w_j = p_j$ for all $j \in \mathcal{N}$ and $\widetilde{\pi}$ is the reverse of π. Then,

$$\sum_{j\in\mathcal{N}} C_j = \sum_{j\in\mathcal{N}} n_j \cdot p_j = \sum_{j\in\mathcal{N}} w_j C'_j,$$

where for any feasible assignment σ, $n_j = |\ \{j' \in \mathcal{N} \mid \sigma_j = \sigma_{j'} \text{ and } \pi(j) \preceq \pi(j')\}\ |$ and C'_j is the completion time of job j in $P|p_j = 1, \pi_{global}|\sum w_j C_j$ with $\widetilde{\pi}$. □

Bosman et al. [BFO+19] present a quasi-polynomial time approximation scheme (QPTAS) for $P|p_j = 1, \pi_{global}|\sum w_j C_j$. As a consequence, we state the following result.

Corollary 4.5. *For every scheduling instance of $P|\pi_{global}|\sum C_j$ and every $\varepsilon > 0$, there exists a ε-approximation scheme with running time $n^{O(\log_{1+\varepsilon} n)}$.*

Unrelated Machines

We now show that even with identical priority lists on every machine the scheduling problem is NP-hard in the case of unrelated machines.

Theorem 4.6. *The problem $R|\pi_{global}|\sum C_j$ is NP-hard.*

Proof. The hardness proof is by a reduction from Max-3DM-3. Given an instance I of Max-3DM-3 matching and $0 < \varepsilon < 1/6n$, we construct the following instance of $R|\pi_{global}|\sum C_j$. There are $m = |T|$ machines, $M_1, M_2, \ldots, M_{|T|}$. All machines have the same priority list, $\pi = (D, X, Y, Z)$.

The set of jobs consists of:

1. $3n$ element-jobs, one for each element in $X \cup Y \cup Z$. Let $e_\ell = (x_i, y_j, z_k)$ be the ℓ^{th} triplet in T. Then, the processing times of the three element jobs corresponding to x_i, y_j, z_k on M_ℓ is ε. All the other element-jobs have processing time 1 on M_ℓ.

2. A set D of $m - n$ jobs. The processing time of every job in D is 1, independent of the machine.

We prove that it is NP-hard to decide if the problem has a schedule with sum of completion time less than $m - n + 1$ or at least $m - n + 1$

Claim 4.7. *If a 3D-matching of size n exists, then the sum of completion times is less than $m - n + 1$.*

Proof. Let T' be a matching of size n. For every $e \in T'$, assign the three jobs of e on M_e. Also, assign one job from D on each of the remaining $m - n$ machines. Every machine with a triplet contributes $(1 + 2 + 3)\varepsilon = 6\varepsilon$ to the total sum of completion time. Every other machine contributes 1 to the total sum of completion time. All together we get that for $0 < \varepsilon < 1/6n$,

$$\sum C_j = m - n + 6n\varepsilon < m - n + 1.$$

□

Claim 4.8. *If a 3D-matching of size n does not exist, then the sum of completion times is at least $m - n + 1$.*

Proof. The contribution of the D-jobs to the total completion time is at least $m-n$. Given that a 3D-matching does not exist, in every schedule, the jobs originated from $X \cup Y \cup Z$, either spread on more than n machines, or at least one of them is processed on some different triplet machine. In the first case, some machine processes both a D-job and some ε-job. By the priority lists, the ε-job is processed after the D-job, and thus contributes at least $1 + \varepsilon$ to the total completion time. In the second case, some ε-job contributes at least 1 to the total completion time. Therefore, we get that the total completion time is at least $m - n + 1$. □

The proof of Theorem 4.6 follows from Claims 4.7 and 4.8. □

Scheduling on 2 Unrelated Machines

Although the problem $R|\pi_{global}|\sum_j C_j$ is NP-hard, if we restrict ourselves to the case of 2 machines, the problems can be solved efficiently.

Theorem 4.9. *$R2|\pi_{global}|\sum C_j$ is polynomial time solvable.*

Proof. We will solve $R2|\pi_{global}|\sum C_j$ by means of a dynamic program. W.l.o.g. let us assume $\pi = (n, n-1, \ldots, 2, 1)$. Let $OPT(k, r)$ denote the objective value of the optimal solution for jobs $1, \ldots, k$ given that the first machine processes r jobs and the second machine processes $k - r$ jobs. Observe that $OPT(0, 0) = 0$. Then for $k = 1, \ldots, n$ and r such that $1 \le r \le k$, we have

$$OPT(k, r) = \min\left\{(OPT(k-1, r-1) + r \cdot p_{k1}), (OPT(k-1, r-1) + (k - r + 1) \cdot p_{k2})\right\}.$$

The entire table $(OPT(i, j))_{i\in[n], j\in[i]}$ can be computed in $O(n^2)$ time.

To prove correctness, recall that the total completion time of a machine with n jobs scheduled to that machine can be expressed as

$$\sum C_j = np_{(1)} + (n-1)p_{(2)} + \ldots + p_{(n)},$$

where $p_{(j)}$ denotes the processing time of the job in the jth position in the sequence. Then, the partition of jobs minimizing the total sum completion time can be determined from the table $(OPT(i, j))_{i\in[n], j\in[i]}$. □

4.3 Inapproximability

In Section 4.2 we showed that the scheduling problem $P|\pi|\sum C_j$ and $R|\pi_{global}|\sum_j C_j$ are NP-hard. We now show that they also do not exhibit a polynomial time approximation scheme using an L-reduction from Maximum Bounded 3-Dimensional Matching (Max-3DM-3) which is known to be APX-hard [Kan92]. Let us first recall the definition of an L-reduction.

Definition 4.10 (Papadimitriou and Yannakakis [PY91])**.** Let Π_1 and Π_2 be two optimization problems. We say Π_1 L-reduces to Π_2 if there exist polynomial time computable functions f, h and constants $\alpha, \beta > 0$ such that for each instance $I \in \Pi_1$ the following holds,

1. $f(I) \in \Pi_2$ such that, $OPT(f(I)) \leq \alpha \cdot OPT(I)$.
2. Given any solution φ to $f(I)$, $h(\varphi)$ is a feasible solution to I such that,

$$| \mathit{cost}(h(\varphi)) - OPT(I) | \leq \beta \mid \mathit{cost}(\varphi) - OPT(f(I)) \mid .$$

Recall that in Max-3DM-3 we are given three disjoint sets $X = \{x_1, \dots, x_q\}, Y = \{y_1, \dots, y_q\}, Z = \{z_1, \dots, z_q\}$, and a set of triplets $T \subseteq X \times Y \times Z$ with $|T| = s$, such that any element of X, Y, Z occurs in exactly one, two or three triplets in T. The goal is to find a subset $T' \subseteq T$ of maximum cardinality such that no two triplets of T' agree in any coordinate. Max-3DM-3 is known to be APX-hard even if one only allows instances where the optimal solution consists of q triples [Pet94]. In the following we will only consider this additionally restricted version of Max-3DM-3.

Theorem 4.11. *The scheduling problem $P|\pi|\sum C_j$ is APX-hard.*

Proof. We present an L-reduction from Max-3DM-3 to show that $P|\pi|\sum C_j$ does not have a PTAS unless $\mathcal{P} = \mathcal{NP}$. Given an instance $I = (q, T)$ of Max-3DM-3, we construct an instance $f(I)$ of $P|\pi|\sum C_j$ with $4q + 2s$ jobs and $q + s$ machines as follows.

The set of jobs consists of:

E : $3q$ element-jobs with processing time 1—one for each element in $X \cup Y \cup Z$.

D : s dummy jobs with processing time 5.

U : $q + s$ dummy jobs with processing time 10.

The set of machines consists of:

S : s triplet machines.

Q : q dummy machines.

The priority lists of the machines are as follows. Note that a set in the priority list implies that all the jobs in the set appear in a fixed arbitrary order.

1. Let (x_u, y_v, z_w) be the ℓ-th triplet in T. The priority list of S_ℓ is

$$\pi_\ell = (D, x_u, y_v, z_w, U, X \setminus \{x_u\}, Y \setminus \{y_v\}, Z \setminus \{z_w\}).$$

2. For any machine $e \in Q$, $\pi_e = (D, X, U, Y, Z)$.

Observe that the transformation f : Max-3DM-3 $\mapsto P|\pi|\sum C_j$ is polynomial time computable and therefore, satisfies the condition in Definition 4.10. Next, we specify the function $h : P|\pi|\sum C_j \mapsto$ Max-3DM-3 as required in Definition 4.10. Let φ be a feasible schedule for $f(I)$. A triplet machine in φ is referred to as *good* if the first three jobs it processes compose a triplet in T. Let $h(\varphi)$ be the corresponding set of triplets—that are scheduled first on a good machine.

We proceed to show that condition 1 in Definition 4.10 is satisfied with $\alpha = 79$.

Claim 4.12. *If $OPT(I) = q$ then $OPT(f(I)) \leq 79q$.*

Proof. Let T' be a matching of size q. For every $e \in T'$, assign the three jobs of e on M_e. Also, assign one job from D on each of the remaining $s - q$ triplet machines, and each of the dummy machines. Finally, assign one job from U on every machine. Every machine with a triplet contributes $1 + 2 + 3 + 13 = 19$ to the total completion time. Every other machine processes one D-job and one U-job, and contributes $5 + 15 = 20$ to the total completion time. All together we get $OPT(f(I)) \leq 19q + 20s$. Since $s \leq 3q$, we have $OPT(f(I)) \leq 79q = 79 \cdot OPT(I)$. □

Next, we show that there exists $\beta > 0$ such that condition 2 in Definition 4.10 is valid. For a given schedule φ, let $cost(\varphi)$ be the total completion time of the jobs, and let g be the number of good machines. Recall that $h(\varphi)$ is the corresponding set of triplets—that are scheduled first on good machines. We show that for some $\beta > 0$ that

$$| \, cost\,(h(\varphi)) - OPT(I) \, | \leq \beta \, | \, cost(\varphi) - OPT(f(I)) \, | \, .$$

Observe that, since $OPT(I) = q$ and $cost\,(h(\varphi)) = g$, it is sufficient to show that there exists $\beta > 0$, such that $q - g \leq \beta\,(cost(\varphi) - OPT(f(I))$. Particularly, we show that this is true for $\beta = 1$.

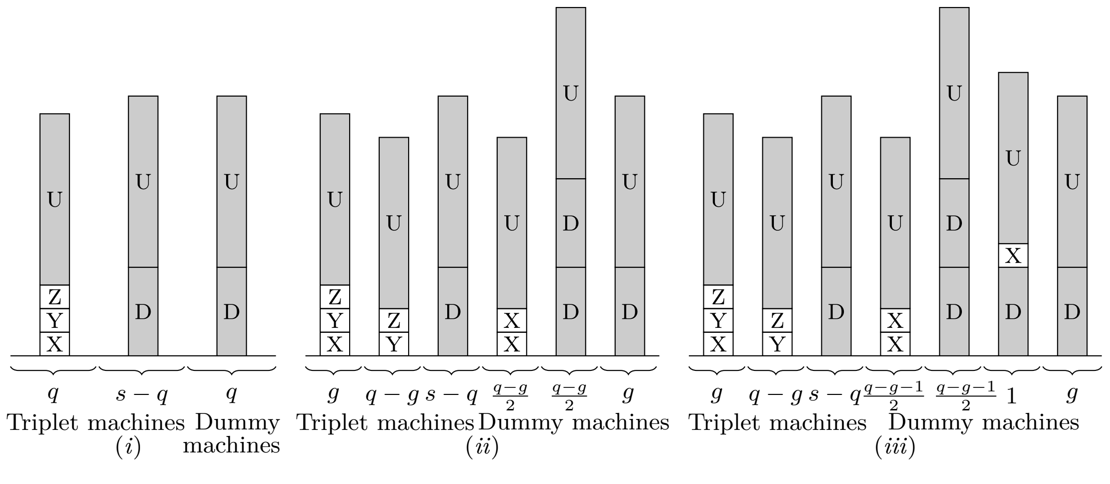

Figure 4.2: An optimal schedule (*i*), best schedule with g good machines when $q - g$ is even (*ii*) and odd (*iii*).

Claim 4.13. $q - g \leq cost(\varphi) - OPT(f(I))$

Proof. We first show that the lemma is valid for a nicely structured schedule, and then show that every other schedule with g good machines has a higher cost.

We distinguish between $q-g$ being even and odd. First, assume $q-g$ is even. Assume that φ, i.e., our schedule with g good machines has the following structure as illustrated in Figure 4.2(ii).

1. There are g good triplet machines, each processing jobs corresponding to one triplet and a U-job. Their contribution to $cost(\varphi)$ is $(1+2+3+13)g = 19g$.
2. There are $q-g$ triplet machines, each processing a job from Y, a job from Z, and a U-job. We remark that this is feasible as we assume that a matching of size q exists. Their contribution to $cost(\varphi)$ is $(1+2+12)(q-g) = 15q-15g$.
3. There are $s-q$ triplet machines, each processing a D-job and a U-job. Their contribution to $cost(\varphi)$ is $(5+15)(s-q) = 20s-20q$.
4. There are $(q-g)/2$ dummy machines, each processing two jobs from X, and a U-job. Their contribution to $cost(\varphi)$ is $(1+2+12)(q-g)/2 = 7.5(q-g)$.
5. There are $(q-g)/2$ dummy machines, each processing two D-jobs, and a U-job. Their contribution to $cost(\varphi)$ is $(5+10+20)(q-g)/2 = 17.5(q-g)$.
6. There are g dummy machines, each processing a D-job and a U-job. Their contribution to $cost(\varphi)$ is $(5+15)g = 20g$.

Summing up we obtain, $cost(\varphi) = 20q+20s-g$, which together with the fact that $OPT(f(I)) \leq 19q+20s$ implies that $cost(\varphi) - OPT(f(I)) \geq q-g$.

Next, assume $q-g$ is odd. Assume that φ, our schedule with g good machines has the following structure as illustrated in Figure 4.2(iii).

1. There are g good triplet machines, each processing jobs corresponding to one triplet and a U-job. Their contribution to $cost(\varphi)$ is $(1+2+3+13)g = 19g$.
2. There are $q-g$ triplet machines, each processing a job from Y, a job from Z, and a U-job. Again, notice that this is feasible as we assume that matching of size q exists. Their contribution to $cost(\varphi)$ is $(1+2+12)(q-g) = 15q-15g$.
3. There are $s-q$ triplet machines, each processing a D-job and a U-job. Their contribution to $cost(\varphi)$ is $(5+15)(s-q) = 20s-20q$.
4. There are $(q-g-1)/2$ dummy machines, each processing two jobs from X, and a U-job. Their contribution to $cost(\varphi)$ is $(1+2+12)(q-g-1)/2 = 7.5(q-g-1)$.
5. There are $(q-g-1)/2$ dummy machines, each processing two D-jobs, and a U-job. Their contribution to $cost(\varphi)$ is $(5+10+20)(q-g-1)/2 = 17.5(q-g-1)$.
6. There is 1 dummy machine, processing a D-job, a job from X, and a U-job. Its contribution to $cost(\varphi)$ is $(5+6+16) = 27$.
7. There are g dummy machines, each processing a D-job and a U-job. Their contribution to $cost(\varphi)$ is $(5+15)g = 20g$.

Summing up, $cost(\varphi) = 20q+20s-g+2$, which together with $OPT(f(I)) \leq 19q+20s$, implies that $cost(\varphi) - OPT(f(I)) \geq q-g+2 \geq q-g$.

In order to conclude the proof, we show that schedule φ is the optimal schedule given g good machines. In other words, we prove that for every schedule φ' with g good machines, we have $cost(\varphi') \geq cost(\varphi)$, as required.

Recall that the total completion time of a machine with n jobs scheduled to that machine can be expressed as

$$\sum C_j = np_{(1)} + (n-1)p_{(2)} + \ldots + p_{(n)},$$

where $p_{(j)}$ denotes the processing time of the job in the jth position in the sequence. Finding an optimal schedule boils down to optimally assigning coefficients to processing times conditional on the priorities of the machines.

Fix the $3g$ element jobs on the g good machines. Given that on each good machine 3 element jobs are fixed, the coefficient of each position on a good machine after the three element jobs is increased by an additive constant of 3—in order to make sure that the increase in coefficient of the three element jobs is also taken into account.

In order to minimize the sum of completion times, we assign $q+s-g$ U-jobs to the 1-coefficients. Since $10+3<2\cdot 10$, the remaining g U-jobs will be assigned a 1-coefficient on good machines (hence the additive constant of 3).

Similarly, we want to assign as many as possible D-jobs to 2-coefficients. On all $s-q+g$ machines with two jobs this is feasible. Since D-jobs always have the highest priority, they can never be assigned to a 2-coefficient when scheduled with an element job (recall that on every machine we also scheduled a U-job). So the remaining $q-g$ D-jobs should be assigned to machines in pairs so that $\lfloor (q-g)/2 \rfloor$ D-jobs are assigned a 2-coefficient and the remaining $\lceil (q-g)/2 \rceil$ D-jobs are assigned a 3-coefficient.

Finalizing the proof, we assign $q-g+\lceil (q-g)/2 \rceil$ element jobs to the remaining 2-coefficients, and the remaining $q-g+\lfloor (q-g)/2 \rfloor$ element jobs to a 3-coefficient.

Since schedule φ fulfills the properties of the optimal schedule given g good machines described above, the proof is complete.

□

The proof of Theorem 4.11 then follows from Claims 4.12 and 4.13. □

Next, we consider hardness of approximating $R|\pi_{global}|\sum C_j$. Although the construction in Theorem 4.11 can be easily adapted for the case of unrelated machines, we prove APX-hardness using a slightly different construction.

Theorem 4.14. *The scheduling problem $R|\pi_{global}|\sum C_j$ is APX-hard.*

Proof. We present an L-reduction from Max-3DM-3 to show that $R|\pi_{global}|\sum C_j$ does not have a PTAS unless $\mathcal{P}=\mathcal{NP}$. Given an instance I of Max-3DM-3, we construct an instance $f(I)$ of $R|\pi_{global}|\sum C_j$ with $m+2q$ jobs and $m=|T|$ machines as follows. All machines have the same priority list, $\pi=(D,X,Y,Z)$.

The set of jobs consists of:

1. $3q$ element-jobs, one for each element in $X\cup Y\cup Z$. Let $e_\ell=(x_i,y_j,z_k)$ be the ℓ^{th} triple in T. The processing times of the three element-jobs corresponding to x_i,y_j,z_k on M_ℓ is 1. All the other element-jobs have processing time ∞ on M_ℓ.

2. A set D of $m-q$ dummy jobs. The processing time of every job in D is 3, independent of the machine.

Next we specify the function h as required in Definition 4.10. Let φ be a feasible schedule for an instance $f(I)$ of $R|\pi_{global}|\sum C_j$. Every machine M_ℓ processes at most 3 jobs whose processing time on M_ℓ is 1, specifically, the jobs corresponding to e_ℓ. A triple e_ℓ is included in $h(\varphi)$ if M_ℓ processes exactly these three corresponding jobs. Note that f and h are polynomial-time computable. We turn to show that they fulfill the requirements specified in Definition 4.10.

Claim 4.15. *If $OPT(I)=q$ then $OPT(f(I))\leq 12\cdot OPT(I)$.*

Proof. Recall that I is an instance of Max-3DM-3 that has an optimal solution with q triples, that is, $OPT(I) = q$. Consider the following schedule for instance $f(I)$: For each triple $e_\ell = (x_i, y_j, z_k)$ in the optimal matching of I, assign the three element-jobs corresponding to x_i, y_j, z_k on M_ℓ. Next, assign the $m - q$ dummy jobs on the remaining $m - q$ machines, one dummy job on each machine. In the resulting schedule there are q machines with three unit-length jobs and $m - q$ machines with a single job of length 3. Thus, $\sum C_j = (1 + 2 + 3)q + 3(m - q) = 3(q + m)$. Note that $m \le 3q$, since any element of X, Y, Z occurs in at most three triples in T, and $m = |T|$. We conclude that $OPT(f(I)) \le 3(q + m) \le 12q = 12 \cdot OPT(I)$. That is, condition 1 in Definition 4.10 is fulfilled with $\alpha = 12$. □

We now show that the condition 2 is fulfilled with $\beta = 1$. Consider a schedule φ for an instance $f(I)$. Without loss of generality we may assume that in φ every job has a finite processing time; otherwise, $cost(\varphi)$ is infinite and condition 2 is clearly satisfied.

Claim 4.16. *W.l.o.g., no machine processes three or more dummy jobs.*

Proof. Assume that some machine M_u processes at least three dummy jobs. If there is a machine M_v in φ with a single job, then φ can be improved by moving a dummy job from M_u to M_v. The total completion time on these two machines is decreased by at least 9 and is increased by 6.

If all the machines have at least two jobs, then there must be a machine M_v with two element-jobs. Suppose not, then since there are only $m - q$ dummy jobs, there are more than q machines without a dummy job. So we need $3q$ element-jobs to fill those machines plus $m - q$ element-jobs to fill the machines with a dummy job yielding a total of more than $3q + m - q \ge 3q$ element-jobs, contradicting the fact that there are only $3q$ element-jobs in I.

Furthermore, by moving a dummy job from M_u to M_v, the total completion time is reduced by at least 9 and is increased by 9. □

For $i = 1, 2$, let d_i denote the number of machines in φ that process i dummy jobs, and let m_3 denote the number of *good* machines that process exactly 3 element-jobs. The remaining $m - m_3 - d_1 - d_2$ machines process only element-jobs, and there are at most two element-jobs on each of these machines.

We conclude that out of the $3q$ element-jobs, $3m_3$ are processed on good machines, at most $2(m - m_3 - d_1 - d_2)$ are processed on machines that process exactly two element-jobs, and the remaining jobs are processed after at least one dummy job, and thus have completion time at least 4.

The total completion time of the element-jobs is therefore at least

$$(1 + 2 + 3)m_3 + (1 + 2)(m - m_3 - d_1 - d_2) + 4(3q - 3m_3 - 2m + 2m_3 + 2d_1 + 2d_2).$$

In addition, the total completion time of the dummy jobs is $3d_1 + 9d_2$.

Summing up, we get

$$\begin{aligned}\sum C_j &\ge 6m_3 + 3m - 3m_3 - 3d_1 - 3d_2 + 12q - 12m_3 - 8m + 8m_3 + 8d_1 + 8d_2 + 3d_1 + 9d_2 \\ &= 12q - m_3 - 5m + 8d_1 + 14d_2.\end{aligned}$$

We now use the fact that $d_1 + 2d_2 = m - q$ to get

$$\sum C_j \ge 12q - m_3 - 5m + 8d_1 + 14d_2$$

$$\begin{aligned} &> 12q - m_3 - 5m + 7(d_1 + 2d_2) \\ &= 12q - m_3 - 5m + 7m - 7q \\ &= 12q - m_3 + 2m. \end{aligned}$$

Recall that $OPT(f(I)) \leq 3(q+m)$. Thus,

$$\begin{aligned} \sum C_j - OPT(f(I)) &\geq 12q - m_3 + 2m - 3q - 3m \\ &= 9q - m_3 - m \\ &\geq 6q - m_3. \end{aligned}$$

The last inequality follows from the fact that $m \leq 3q$.

Observe that given any solution φ to $f(I)$, $h(\varphi)$ is a feasible solution to I such that, $\mid cost(h(\varphi)) - OPT(I) \mid = q - m_3 < 6q - m_3 \leq \mid cost(\varphi) - OPT(f(I)) \mid$, i.e., condition 2 is fulfilled with $\beta = 1$. □

Chapter 5

Scheduling Games with Machine-Dependent Priority Lists

In the last chapter we introduced a variant of scheduling problem on parallel machines, where each machine uses an individual priority list to decide on the order according to which the jobs on the machine are processed. Surprisingly, the side constraint of local precedence makes the problem computationally challenging even with identical machines. In this chapter we investigate this scheduling variant as a strategic game on related machines, where individual jobs are allowed to selfishly pick a machine that minimize their completion time.

5.1 Introduction

Scheduling problems have traditionally been studied from a centralized point of view in which the goal is to find an assignment of jobs to machines so as to minimize some global objective function. Two of the classical results are that Smith's rule, i.e., schedule jobs in decreasing order according to their ratio of weight over processing time, is optimal for single machine scheduling with the sum of weighted completion times as objective [Smi56] and list scheduling, i.e., greedily assign the job with the highest priority to a free machine, yields a 2-approximation for identical machines with the minimum makespan as objective [Gra66]. Many modern systems provide service to multiple strategic users, whose individual payoff is affected by the decisions made by others. As a result, non-cooperative game theory has become an essential tool in the analysis of job-scheduling applications. The jobs are controlled by selfish users who independently choose which resources to use. The resulting job-scheduling games have by now been widely studied and many results regarding the efficiency of equilibria in different settings are known.

A particular focus has been placed on finding coordination mechanisms [CKN04], i.e., local scheduling policies, that induce a good system performance. In these works, it is common to assume that ties are broken in a consistent manner (see, e.g., Immorlica et al. [ILMS09]), or that there are no ties at all (see, e.g., Cole et al. [CCG$^+$15]). In practice, there is no real justification for this assumption, except that it avoids subtle difficulties in the analysis. We relax this restrictive assumption and consider the more general setting in which machines have arbitrary individual priority lists. That is, each machine schedules those jobs that have chosen it according to its priority list. The priority lists are publicly known to the jobs.

We introduced scheduling games with machine-dependent priority lists in Chapter 1. In this chapter we analyze the effect of having machine-dependent priority lists on the corresponding job-scheduling game. We study the existence of pure Nash equilibria (NE), the complexity of identifying whether an NE profile exists, the complexity of computing an NE, in particular a good one, and the equilibrium inefficiency. We begin by once again familiarizing ourselves with the model description and preliminaries. However, we remark that the readers may choose to skip the following subsection.

5.1.1 The Model

An instance of a *scheduling game with machine-dependent priority lists* is given by a tuple

$$G = (\mathcal{N}, M, (p_i)_{i\in\mathcal{N}}, (s_j)_{j\in M}, (\pi_j)_{j\in M}),$$

where $\mathcal{N}$ is a finite set of $n \geq 1$ jobs, M is a finite set of $m \geq 1$ related machines, $p_i \in \mathbb{R}_{\geq 0}$ is the processing time of job $i \in \mathcal{N}$, $s_j \in \mathbb{R}_{\geq 0}$ denotes the speed of the machine $j \in M$, and $\pi_j : \mathcal{N} \to \{1, \ldots, n\}$ is the priority list of machine $j \in M$.

A strategy profile $\sigma = (\sigma_i)_{i\in\mathcal{N}} \in M^{\mathcal{N}}$ assigns a machine $\sigma_i \in M$ to every job $i \in \mathcal{N}$. Given a strategy profile σ, the jobs are processed according to their order in the machines' priority lists. The set of jobs that delay $i \in \mathcal{N}$ in σ is denoted by

$$B_i(\sigma) = \{i' \in \mathcal{N} \mid \sigma_{i'} = \sigma_i \wedge \pi_{\sigma_i}(i') \leq \pi_{\sigma_i}(i)\}.$$

Note that job i itself also belongs to $B_i(\sigma)$. Let $p_i(\sigma) = \sum\limits_{i'\in B_i(\sigma)} p_{i'}$, then the cost of job $i \in \mathcal{N}$ is equal to its completion time in σ, given by

$$C_i(\sigma) = \frac{p_i(\sigma)}{s_{\sigma_i}}.$$

Each job chooses a strategy so as to minimize its costs. A strategy profile σ is a *pure Nash equilibrium (NE)* if for all $i \in \mathcal{N}$ and all $\sigma'_i \in M$, we have that

$$C_i(\sigma) \leq C_i(\sigma'_i, \sigma_{-i}).$$

Let $\mathcal{E}(G)$ denote the set of Nash equilibria for a given instance G. We would like to remark that $\mathcal{E}(G)$ may be empty.

For a strategy profile σ, let $C(\sigma)$ denote the cost of σ. The cost is defined with respect to some objective, e.g., the makespan, i.e.,

$$C_{\max}(\sigma) \coloneqq \max_{i\in\mathcal{N}} C_i(\sigma),$$

or the sum of completion times, i.e.,

$$\sum_{i\in\mathcal{N}} C_i(\sigma).$$

It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of the society as a whole. For a game G, let $P(G)$ be the set of feasible profiles of G. We denote by $OPT(G)$ the cost of a social optimal solution, i.e., $OPT(G) = \min_{\sigma\in P(G)} C(\sigma)$. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [KP99], and *price of stability* (PoS) [ADK$^+$08]. Let us recall the definitions of PoA and PoS in Chapter 1. The PoA is the worst-case inefficiency of a pure Nash equilibrium, while the PoS measures the best-case inefficiency of a pure Nash equilibrium.

Definition 5.1. Let $\mathcal{G}$ be a family of games, and let G be a game in $\mathcal{G}$. Let $\mathcal{E}(G)$ be the set of pure Nash equilibria of the game G. Assume that $\mathcal{E}(G) \neq \emptyset$.

- The *price of anarchy* of G is the ratio between the *maximum* cost of an NE and the social optimum of G, i.e., $\text{PoA}(G) = \max\limits_{\sigma \in \mathcal{E}(G)} C(\sigma)/OPT(G)$. The *price of anarchy* of $\mathcal{G}$ is $\text{PoA}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{PoA}(G)$.
- The *price of stability* of G is the ratio between the *minimum* cost of an NE and the social optimum of G, i.e., $\text{PoS}(G) = \min\limits_{\sigma \in \mathcal{E}(G)} C(\sigma)/OPT(G)$. The *price of stability* of $\mathcal{G}$ is $\text{PoS}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{PoS}(G)$.

5.1.2 Our Contribution

In Section 5.2 we first show that for scheduling games with machine-dependent priority lists a pure Nash equilibrium in general need not exist, and use this to show that it is NP-complete to decide whether a particular game has a pure Nash equilibrium. We then provide a characterization of instances in which a pure Nash equilibrium is guaranteed to exist. Specifically, existence is guaranteed if the game belongs to at least one of the following four classes:

$\mathcal{G}_1$: all jobs have unit processing time.

$\mathcal{G}_2$: there are two machines.

$\mathcal{G}_3$: all machines have the same speed.

$\mathcal{G}_4$: all machines have the same priority list.

For all four of these classes, there is a polynomial time algorithm that computes a pure Nash equilibrium. In fact, for all four classes we prove that better-response dynamics converge to a pure Nash equilibrium. This characterization is tight in a sense that our inexistence example disobeys it in a minimal way, i.e., it describes a game on three machines, two of them having the same speed and the same priority list. We also show that the result for $\mathcal{G}_2$ cannot be extended for games with two unrelated machines. Another characterization we consider is the number of jobs in the instance. We present a game of 4 jobs that has no pure Nash equilibrium, and show that every game of 3 jobs admits a pure Nash equilibrium.

In Section 5.3 we analyze the efficiency of Nash equilibria by means of two different measures of efficiency: the makespan, i.e., the maximum completion time of a job and the sum of completion times. For all four classes of games with a guaranteed pure Nash equilibrium, we provide tight bounds for the price of anarchy and the price of stability with respect to both measures. Our results are summarized in Table 5.1.

1. If jobs have unit processing times, we show that the price of anarchy is equal to 1, which means that selfish behavior is optimal.
2. For two machines with speed 1 and $s \leq 1$ respectively, we prove that the PoA and the PoS are at most $s+1$ if $s \leq \frac{\sqrt{5}-1}{2}$, and $\frac{s+2}{s+1}$ if $s \geq \frac{\sqrt{5}-1}{2}$. Moreover, our analysis is tight for all $s \leq 1$. The maximal inefficiency, listed in Table 5.1, is achieved for $s = \frac{\sqrt{5}-1}{2}$. In case the sum of completion times is considered as an objective, the price of anarchy can grow linearly in the number of jobs.

Instance class \ Objective	Makespan PoA/PoS	Sum of Comp. Times PoA/PoS
$\mathcal{G}_1$: Unit jobs	1	1
$\mathcal{G}_2$: Two machines	$(\sqrt{5}+1)/2$	$\Theta(n)$
$\mathcal{G}_3$: Identical machines	$2 - 1/m$	$\Theta(n/m)$
$\mathcal{G}_4$: Global priority list	$\Theta(m)$	$\Theta(n)$

Table 5.1: Our results for the equilibrium inefficiency.

3. If machines have identical speeds, but potentially different priority lists, the price of anarchy with respect to the makespan is equal to $2 - 1/m$. The upper bound follows because every Nash equilibrium can be seen as an outcome of Graham's List-Scheduling algorithm. This generalizes a similar result by Immorlica et al. [ILMS09] for priorities based on shortest processing times first. The lower bound example shows the bound is tight, even with respect to the price of stability. For the sum of completion times objective, we show that the price of anarchy is at most $O(n/m)$, and provide a lower bound example for which the price of stability grows in the order of $O(n/m)$.

4. If there is a global priority list and machines have arbitrary speeds, we show that the $\Theta(m)$-approximation of List-Scheduling carry over for the makespan inefficiency, and the results for two machines carry over for the sum of completion times.

We conclude with results regarding the complexity of computing a good NE. While a simple greedy algorithm can be used to compute an NE for an instance with identical machines (the class $\mathcal{G}_3$), in Section 5.4 we show that it is NP-hard to compute an NE schedule that approximates the best NE of a game in this class. Specifically, it is NP-hard to approximate the best NE with respect to the minimum makespan within a factor of $2 - 1/m - \varepsilon$ for all $\varepsilon > 0$, and it is NP-hard to approximate the best NE with respect to the sum of completion times within a factor of r for any constant $r > 1$.

5.1.3 Related Work

Scheduling games were initially studied in the setting in which each machine processes its jobs in parallel so that the completion time of each job is equal to the makespan of the machine. The goal of these papers were to characterize the inefficiency of selfish behavior as measured by the price of anarchy [KP99]. Most attention has been given to the makespan as a measure of efficiency. Czumaj and Vöcking [CV07] gave tight bounds on the price of anarchy for related machines, whereas Awerbuch et al. [AART06] and Gairing et al. [GLMM10] provided tight bounds for restricted machine settings. We refer to Vöcking [Vöc07] for an overview. These tight bounds grow with the number of machines and that is why Christodoulou et al. [CKN04] introduced the idea of using coordination mechanisms, i.e., local scheduling policies, to improve the price of anarchy. They studied the price of anarchy with priority lists based on longest processing times first. Immorlica et al. [ILMS09] generalized their results and studied several different scheduling policies, among which longest and shortest processing times first, in multiple scheduling settings. Both these policies guarantee the existence of a pure Nash equilibrium in the related machine setting. These results are a special case of our result, as we prove the existence of a pure Nash equilibrium if there is a global priority list. For shortest processing times first, a pure Nash equilibrium is also guaranteed in the unrelated machines setting. Here, the set of Nash equilibria corresponds to the set of solutions of the Ibarra-Kim algorithm. A result that is also proven in Heydenreich et al. [HMU07]. Other (in)existence results are Dürr and Nguyen [DN09], who proved that a Nash equilibrium exists for two machines with a random order and balanced jobs, Azar et al. [AJM08], who showed that for unrelated machines with priorities based on the ratio of a job's processing time to its fastest processing time, a Nash equilibrium need not exist, Lu and Yu [LY12], who proved that group-makespan mechanisms guarantees the existence of a Nash equilibrium, and Kollias [Kol13], who showed that non-preemptive coordination mechanisms need not induce a pure Nash equilibrium.

For the sum of weighted completion times, Correa and Queyranne [CQ12] proved a tight upper bound of 4 for restricted related machines with priority lists derived from Smith's rule. Cole et al. [CCG$^+$15] generalized the bound of 4 to unrelated machines with Smith's rule and proposed better scheduling policies. Hoeksma and Uetz [HU19] gave a tighter bound for the more restricted setting in which jobs have unit weights and machines are related. Caragiannis et al. [CGV17] proposed a framework that uses price of anarchy results of Nash equilibria in scheduling games to come up with combinatorial approximation algorithms for the centralized problem.

5.2 Equilibrium Existence and Computation

In this section we give a precise characterization of scheduling game instances that are guaranteed to have an NE. The conditions that we provide are sufficient but not necessary. A natural question is to decide whether a given game instance that does not fulfill any of the conditions has an NE. We show that answering this question is an NP-complete problem.

We first show that an NE may not exist, even with only three machines, two of which have the same speed and the same priority list.

Example 5.2. Consider the game G^* with 5 jobs, $\mathcal{N} = \{a, b, c, d, e\}$, and three machines, $M = \{M_1, M_2, M_3\}$, with $\pi_1 = (a, b, c, d, e)$, and $\pi_2 = \pi_3 = (e, d, b, c, a)$. The first machine has speed $s_1 = 1$ while the two other machines have speed $s_2 = s_3 = 1/2$. The job processing times are $p_a = 5, p_b = 4, p_c = 4.5, p_d = 9.25$, and $p_e = 2$.

Job a is clearly on M_1 in every NE. Therefore job e is not on M_1 in an NE, as job e is first on M_2 or M_3. Since these two machines have the same priority list and the same speed, we can assume w.l.o.g., that if an NE exists, then there exists an NE in which job e is on M_3. We distinguish two different cases for job d, as given that e is on M_3, d prefers M_2 over M_3.

1. Job d is on M_1. Then as job b has the highest remaining priority among b and c on all machines, job b picks the machine with the lowest completion time, which is M_2, and job c lastly is then on M_1. As a result, d prefers M_2 (since $18.5 < 18.75$) over M_1.

2. Job d is on M_2. Then as job b has the highest remaining priority among b and c on all machines, job b picks the machine with the lowest completion time, which is M_1, and job c lastly is then on M_3. As a result, d prefers M_1 (since $18.25 < 18.5$) over M_2.

Thus, the game G^* has no pure Nash equilibrium.

We can use the above example to show that deciding whether a game instance has an NE is NP-complete by using a reduction from 3-bounded 3-dimensional matching.

Theorem 5.3. *Given an instance of a scheduling game, it is NP-complete to decide whether the game has an NE.*

Proof. Given a game and a profile σ, verifying whether σ is an NE can be done by checking for every job whether its current assignment is also its best-response, therefore the problem is in NP.

The hardness proof is by a reduction from Maximum Bounded 3-Dimensional Matching (Max-3DM-3). Recall that the input to the Max-3DM-3 problem is a set of triplets $T \subseteq X \times Y \times Z$, where $|T| \geq n$ and $|X| = |Y| = |Z| = n$. The number of occurrences of every element of $X \cup Y \cup Z$ in T is at most 3. The goal is to decide whether T has a 3-dimensional matching of size n, i.e., there exists a subset $T' \subseteq T$, such that $|T'| = n$, and every element in $X \cup Y \cup Z$ appears exactly once in T'. Max-3DM-3 is known to be NP-hard [Kan91].

Given an instance T of Max-3DM-3 matching and $\varepsilon > 0$, we construct the following scheduling game G_T. The set of jobs consists of:

1. The 5 jobs $\{a, b, c, d, e\}$ from the game G^* in Example 5.2.
2. A single dummy job f, with processing time 2.
3. A set D of $|T| - n$ dummy jobs with processing time 3.
4. A set U of $|T| + 1$ dummy jobs with processing time 20.
5. $3n$ jobs with processing time 1—one for each element in $X \cup Y \cup Z$.

There are $m = |T| + 4$ machines, $M_1, M_2, \ldots, M_{|T|+4}$. All the machines except for M_2 and M_3 have speed $s_j = 1$. For M_2 and M_3, $s_2 = s_3 = 1/2$.

The heart of the reduction lies in determining the priority lists. The first three machines will mimic the no-NE game G^* from Example 5.2. Note that if job e is missing from G^* then there exists an NE of $\{a, b, c, d\}$ on M_1, M_2, M_3. The idea is that if a Max-3DM-3 matching exists, then job e would prefer M_4 and leave the first three machines for $\{a, b, c, d\}$. However, if there is no Max-3DM-3, then some job originated from the elements in $X \cup Y \cup Z$ will precede job e on M_4, and e's best-response would be on M_2 or M_3, where it is guaranteed to have completion time 4, and the no-NE game G^* would come to life. The dummy jobs in U are long enough to guarantee that each of the jobs $\{a, b, c, d\}$ prefers the first three machines over the last $|T| + 1$ machines.

The priority lists are defined as follows. When the list includes a set, it means that the set elements appear in arbitrary order. For the first machine,

$$\pi_1 = (a, b, c, d, e, f, U, X, Y, Z, D).$$

For the second and third machines,

$$\pi_2 = \pi_3 = (e, d, b, c, a, f, U, X, Y, Z, D).$$

For the fourth machine, we have priority list

$$\pi_4 = (f, X, Y, Z, e, U, D, a, b, c, d).$$

The remaining $|T|$ machines are *triplet-machines*. For every triplet $t = (x_i, y_j, z_k) \in T$, the priority list of the triplet-machine corresponding to t is

$$(D, x_i, y_j, z_k, U, f, X \setminus \{x_j\}, Y \setminus \{y_j\}, Z \setminus \{z_j\}, a, b, c, d, e).$$

Observe that in any NE, the dummy job f with processing time 2 is assigned as the first jobs on M_4. Also, the dummy jobs in D have the highest priority on the triplet-machines, thus, in every NE, there are $|D| = |T| - n$ triplet-machines on which the first job is from D. Finally, it is easy to see that in every NE there is exactly one dummy job from U on each of the last $|T| + 1$ machines.

Figure 5.1 provides an example for $n = 2$ and $|T| = 3$.

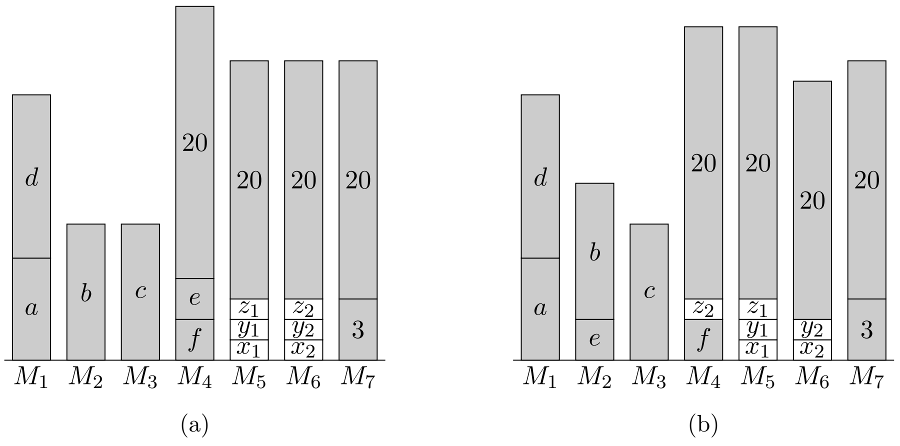

Figure 5.1: (a) Let $T = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_1, y_2, z_2)\}$. A matching of size 2 exists. Job e is assigned on M_4, an NE exists. (b) Let $T = \{(x_1, y_1, z_1), (x_2, y_2, z_1), (x_1, y_2, z_2)\}$. A matching of size 2 does not exist. Job e is not assigned on M_4 and the no-NE game G^* is induced on the first three machines.

In order to complete the proof we prove the following two claims that relate the existence of a NE in the game G_T to the existence of a perfect matching in the 3DM-3 instance T. We first show that if the MAX-3DM-3 instance has a perfect matching, then the game induced due to our construction has a pure Nash equilibrium.

Claim 5.4. *If a 3D-matching of size n exists in T, then the game G_T has an NE.*

Proof. Let $T' \subseteq T$ be a matching of size n. Assign the jobs of $X \cup Y \cup Z$ on the triplet-machines corresponding to T' and the jobs of D on the remaining triplet-machines. Assign f and e on M_4. Also, assign a single job from U on all but the first 3 machines. We are left with the jobs a, b, c, d that are assigned on the first three machines: a and d on M_1, b on M_2 and c on M_3. It is easy to verify that the resulting assignment is an NE. The crucial observation is that all the jobs originated from $X \cup Y \cup Z$ completes at time at most 3, and therefore have no incentive to select M_4. Thus, job e completes at time 4 on M_4 and therefore, has no incentive to join the no-NE game on the first three machines. □

The next claim shows that if the MAX-3DM-3 instance does not have a perfect matching, then as a consequence of our construction, the no-NE subgame G^* is triggered, and G_T has no NE.

Claim 5.5. *If a 3D-matching of size n does not exist, then the game G_T has no NE.*

Proof. Since a matching does not exist, at least one job from $X \cup Y \cup Z$, is not assigned on its triplet machine, and thus prefers M_4, where its completion time is 3. Thus, job e prefers to be first on M_2 or M_3, where its completion time is 4. The long dummy jobs guarantee that machines M_1, M_2 and M_3 attracts exactly the 5 jobs $\{a, b, c, d, e\}$ and the no-NE game G^* is played on the first three machines. □

The proof of Theorem 5.3 then immediately follows from Claims 5.4 and 5.5. □

We now introduce four classes of games for which an NE is guaranteed to exist. This characterization is tight in a sense that our inexistence example disobeys it in a minimal way. For classes $\mathcal{G}_3$ ($s_j = 1$ for all $j \in M$) and $\mathcal{G}_4$ ($\pi_j = \pi$ for all $j \in M$), a simply greedy algorithm shows that an NE always exists. We refer the readers to Correa and Queyranne [CQ12], and Farzad et al. [FOV08], respectively, for a formal proof.

The following algorithm computes an NE for instances in the class $\mathcal{G}_1$, i.e., $p_i = 1$ for all $i \in \mathcal{N}$. It assigns the jobs greedily, where in each step, a job is added on a machine on which the cost of a next job is minimized.

Algorithm 2 Calculating an NE of unit jobs on related machines

```
1: Let $\ell_j$ denote the number of jobs assigned on machine $j$. Initially, $\ell_j = 0$ for all
   $1 \leq j \leq m$.
2: repeat
3:     Let $j^\star = \arg\min_j \; (\ell_j + 1)/s_j$.
4:     Assign on machine $j^\star$ the first unassigned job on its priority list.
5:     $\ell_{j^\star} = \ell_{j^\star} + 1$.
6: until all jobs are scheduled
```

Theorem 5.6. *If $p_i = 1$ for all jobs $i \in \mathcal{N}$, then Algorithm 2 calculates an NE.*

Proof. Let $\sigma^\star$ be the schedule produced by Algorithm 2. We show that $\sigma^\star$ is an NE. Note that the jobs are assigned one after the other according to their completion time in $\sigma^\star$. That is, if j_1 is assigned before j_2 then $C_{j_1}(\sigma^\star) \leq C_{j_2}(\sigma^\star)$. Assume by contradiction that $\sigma^\star$ is not an NE, and let i be a job that can migrate from its current machine to machine j and reduce its completion time. Assume that if it migrates, then i would be assigned as the k-th job on machine j. This contradicts the choice of the algorithm when the k-th job on machine i is assigned, since j should have been selected. If no job is k-th on machine i, then we get a contradiction to the assignment of i. □

The following algorithm produces an NE for instances in the class $\mathcal{G}_2$, i.e., $m = 2$.

Algorithm 3 Calculating an NE schedule on two related machines

```
1: Assign all the jobs on $M_1$ (fast machine) according to their order in $\pi_1$.
2: For $1 \leq k \leq n$, let the job $i$ for which $\pi_2(i) = k$ perform a best-response move (migrate
   to $M_2$ if this reduces its completion time).
```

Theorem 5.7. *If $m = 2$, then an NE exists and can be calculated efficiently.*

Proof. Assume w.l.o.g. that $s_1 = 1$ and $s_2 = s \leq 1$. Consider Algorithm 3, which initially assigns all the jobs on the fast machine. Then, the jobs are considered according to their order in π_2, and every job gets an opportunity to migrate to M_2.

Let us denote by $\widehat{\sigma}$ the schedule after the first step of the algorithm (where all the jobs are on M_1), and let σ denote the schedule after the algorithm terminates. The following two claims show that after the termination of the algorithm, no job has a unilateral deviation that improves its cost, i.e., σ is an NE.

Claim 5.8. *No job for which $\sigma_i = M_1$ has a beneficial migration.*

Proof. Assume by contradiction that job i is assigned on M_1 and has a beneficial migration. Assume that $\pi_2(i) = k$. Job i was offered to perform a migration in the k-th iteration of step 2 of the algorithm, but chose to remain on M_1. The only migrations that took place after the k-th iteration are from M_1 to M_2. Thus, if migrating is beneficial for i after the algorithm completes, it should have been beneficial also during the algorithm, contradicting its choice to remain on M_1. □

Claim 5.9. *No job for which $\sigma_i = M_2$ has a beneficial migration.*

Proof. Assume by contradiction that the claim is false and let i be the first job on M_2 (first with respect to π_2) that may benefit from returning to M_1. Recall that, $\widehat{\sigma}$ denotes the schedule before job i migrates to M_2—during the second step of the algorithm. Recall that $C_i(\sigma)$ is the completion time of job i on M_2 and $C_i(\widehat{\sigma})$ is its completion time on M_1 before its migration.

Since the jobs are activated according to π_2 in the second step of the algorithm, no jobs are added before job i on M_2. Job i may be interested in returning to M_1 only if some jobs that were processed before it on M_1, move to M_2 after its migration. Denote by Δ the set of these jobs, and let δ be their total processing time. Let i' be the last job from Δ to complete its processing in σ. Since job i' performs its migration out of M_1 after job i, and jobs do not join M_1 during step 2 of the algorithm, the completion time of i' when it performs the migration is at most $C_{i'}(\widehat{\sigma})$. The migration from M_1 to M_2 is beneficial for i', thus, $C_{i'}(\sigma) < C_{i'}(\widehat{\sigma})$.

The jobs in Δ are all before job i in π_1 and after job i in π_2. Therefore, $C_{i'}(\widehat{\sigma}) < C_i(\widehat{\sigma})$, and $C_{i'}(\sigma) \geq C_i(\sigma) + \delta/s$. Finally, we assume that σ is not stable and i would like to return to M_1. By returning, its completion time would be $C_i(\widehat{\sigma}) - \delta$. Given that the migration is beneficial for i, and that i is the first job who likes to return to M_2, we have that $C_i(\widehat{\sigma}) - \delta < C_i(\sigma)$.

Combining the above inequalities, we get

$$\begin{aligned} C_i(\widehat{\sigma}) &< C_i(\sigma) + \delta \leq C_{i'}(\sigma) - (1/s - 1)\delta \\ &< C_{i'}(\widehat{\sigma}) - (1/s - 1)\delta < C_i(\widehat{\sigma}) - (1/s - 1)\delta, \end{aligned}$$

which contradicts the fact that $s \leq 1$ and $\delta \geq 0$. □

By combining the Claims 5.8 and 5.9, we conclude that no player has a beneficial deviation and σ is an NE. □

Additionally, a class for which we show that an NE is guaranteed to exist is the class of games with at most 3 jobs. Consider an instance consisting of m machines with arbitrary priority lists, and 3 jobs $a, b,$ and c. Let M_1 be a machine with the highest speed. Assume $\pi_1 = (a, b, c)$. Clearly, job a is on M_1 in every NE. An NE can be computed by adding jobs b and c greedily one after the other. If job b picks M_1 then the resulting schedule is an NE. If job b picks M_2 and job c is then added before it on M_2, then job b may migrate, to get a final NE. The above characterization is tight, as there exists a game with only 4 jobs that has no NE.

Example 5.10. Consider the game $\hat{G}$ with 4 jobs, $\mathcal{N} = \{a, b, c, d\}$, and three machines, $M = \{M_1, M_2, M_3\}$, with $\pi_1 = (a, b, c, d)$, and $\pi_2 = \pi_3 = (d, b, c, a)$. The speed of machine j is $s_j = 1/j$. The job processing times are $p_a = 5, p_b = 4, p_c = \frac{13}{3} \approx 4.33$ and $p_d = 9.25$.

Job a is clearly on M_1 in every NE. Since $s_2 > s_3$, job d is not on M_3 in any NE. We distinguish two different cases for job d.

1. Job d is on M_1. Then as job b has the highest remaining priority among b and c on all machines, job b picks the machine with the lowest completion time, which is M_2, and job c lastly is then on M_1. As a result, d prefers M_2 over M_1 (since $18.5 < 18.58$).
2. Job d is on M_2. Then as job b has the highest remaining priority among b and c on all machines, job b picks the machine with the lowest completion time, which is M_1, and job c lastly is then on M_3 (since $13 < 13.33$). As a result, d prefers M_1 (since $18.25 < 18.5$) over M_2.

Thus, the game $\hat{G}$ has no pure Nash equilibrium.

A possible generalization of our setting considers unrelated machines, that is, for every job i and machine j, p_{ij} is the processing time of job i if processed on machine j. We conclude this section with an example demonstrating that an NE need not exist in this environment already with only two unrelated machines.

Example 5.11. Consider a game G with 3 jobs, $\mathcal{N} = \{a, b, c\}$, and two machines, $M = \{M_1, M_2\}$ with $\pi_1 = (a, b, c)$ and $\pi_2 = (c, a, b)$. The job processing times are $p_{a1} = 5$, $p_{a2} = 4$, $p_{b1} = 7$, $p_{b2} = 4$, $p_{c1} = 1$ and $p_{c2} = 7$. We show that G has no NE. Specifically, we show that no assignment of job c can be extended to a stable profile.

First, assume that job c is on M_1. Then job a has the highest remaining priority on the two machines and picks M_2. Given that job a is on M_2, job b prefers M_1 over M_2. However, job c now prefers to pick M_2 as its completion time there is 7, which is smaller than 8.

Second, assume that job c is on M_2. Then job a has the highest remaining priority on the two machines and picks M_1. Given that job a is on M_1, job b prefers M_2. However, job c now prefers to pick M_1 as its completion time there is 6, which is smaller than 7.

5.2.1 Convergence of Best-Response Dynamics

In this section we consider the question whether natural dynamics such as better-responses are guaranteed to converge to an NE. Given a strategy profile σ, a strategy σ'_i for job $i \in N$ is a better-response if $C_i(\sigma'_i, \sigma_{-i}) < C_i(\sigma)$.

We show that every sequence of best-response converge to an NE for every instance $G \in \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \cup \mathcal{G}_4$.

Theorem 5.12. *Let G be a game instance in $\mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \cup \mathcal{G}_4$. Any best-response sequence in G converges to an NE.*

Proof. The proof has the same structure for all four classes. Assume that best-response dynamics (BRD) does not converge. Since the number of different profiles is finite, this implies that the sequence of profiles contains a loop. That is, the sequence includes a profile σ_0, starting from which jobs migrate and eventually return to their strategy in σ_0. Let Γ denote the set of jobs that perform a migration during this loop. For each of the four classes we identify a job $i \in \Gamma$ such that once job i migrates, it cannot have an additional beneficial move.

Consider first the case $G \in \mathcal{G}_1$, that is, a game with unit jobs. Let $C_{\min}$ be the lowest cost of a job in Γ during the BR-cycle. Let M_1 be a machine on which $C_{\min}$ is achieved. Let i be the job achieving cost $C_{\min}$ on M_1 with the highest priority on M_1 among the jobs in Γ. Once i achieves cost $C_{\min}$, its cost does not increase, as no job is added to M_1 before it. Job i cannot have an additional beneficial move, as this will contradict the definition of $C_{\min}$.

We turn to consider games in $\mathcal{G}_2$, that is, G is played on two machines. W.l.o.g., assume $s_1 = 1$ and $s_2 = s \leq 1$. Let i be the job in Γ with highest priority in π_2. Given that BRD loops and that $i \in \Gamma$, it holds that during the BR sequence i migrates from M_1 to M_2 and then back from M_2 to M_1.

We show that once i moves from M_1 to M_2, moving back to M_1 cannot be beneficial for it. Let σ' denote the schedule before job i migrates from M_1 to M_2. Assume by contradiction that i may benefit from returning to M_1. Let L_1 be the total processing time of jobs on M_1 that precede i on π_1 in σ'. We have that $C_i(\sigma') = L_1 + p_i$. Let L_2 be the the total processing time of jobs in $\mathcal{N} \setminus \Gamma$ that precede i on π_2. Since i has the highest priority among Γ on M_2, its cost while on M_2 is $(L_2 + p_i)/s$, independent of other jobs leaving and joining M_2. The migration of i from M_1 to M_2 is beneficial, thus, $L_1 + p_i > (L_2 + p_i)/s$. Migrating back to M_1 may become beneficial only if the total processing time of job that would precede it on M_1 is less than L_1, thus, at least one job that precedes i on π_1 migrates out of M_1 when i is on M_2. Let k be the last job, for which $\pi_1(k) < \pi_1(i)$ that have left M_1 when i is on M_2. Following k's migration the processing time of jobs on M_1 that precede i in π_1 is L'_1. Migrating back is beneficial for i, thus, $L'_i + p_i < (L_2 + p_i)/s$ (additional jobs may join M_1 after k leaves it, but this only makes M_1 less attractive for i). Since $\pi_2(k) > \pi_2(i)$, the cost of k after its migrating to M_2 is at least $(L_2 + p_i + p_k)/s$. k's migration from M_1 to M_2 is beneficial, thus, $L'_1 + p_k > (L_2 + p_i + p_k)/s$. By combining the above inequalities we reach a contradiction. Specifically, $L'_i + p_i < (L_2 + p_i)/s = (L_2 + p_i + p_k)/s - p_k/s < L'_i + p_k - p_k/s \leq L'_i$. We conclude that job i cannot benefit from returning to M_1 and thus, cannot be involved in the BRD-cycle.

Assume next that $G \in \mathcal{G}_3$, that is, machines have identical speeds. Let t be the lowest start time of a job in Γ during the BR-cycle. Let M_1 be a machine on which t is achieved. Let i be the job in Γ with highest priority on M_1. Clearly, once i achieves start time, t, it cannot have an additional beneficial move, as this will contradict its choice.

Finally, if $G \in \mathcal{G}_4$, that is, when machines share a global priority list, then once the job in Γ with the highest priority migrates, it selects the machine with the lowest total processing time of jobs in $\mathcal{N} \setminus \Gamma$ that precedes it, and cannot have an additional beneficial move later.

□

5.3 Equilibrium Inefficiency

Two common measures for evaluating the quality of a schedule are the makespan, given by $C_{\max}(\sigma) = \max_{i \in \mathcal{N}} C_i(\sigma)$, and the sum of completion times, given by $\sum_{i \in \mathcal{N}} C_i(\sigma)$. In this section we analyze the equilibrium inefficiency with respect to each of the two objectives, for each of the four classes for which an NE is guaranteed to exist.

We begin with $\mathcal{G}_1$, the class of instances with unit jobs. For this class we show that allowing arbitrary priority lists does not hurt the social cost, even on machines with different speeds.

Theorem 5.13. *$PoA(\mathcal{G}_1) = PoS(\mathcal{G}_1) = 1$ for both the min-makespan and the sum of completion times objective.*

Proof. Let σ be a schedule of unit jobs. The quality of σ is characterized by the vector $(n_1(\sigma), n_2(\sigma), \dots, n_m(\sigma))$ specifying the number of jobs on each machine. The makespan of σ is given by $\max_j n_j(\sigma)/s_j$, and the sum of completion times in σ is $\sum_j \frac{n_j(\sigma)(n_j(\sigma)+1)}{2s_j}$.

Theorem 5.6 shows that assigning the jobs greedily, where on each step a job is added on a machine on which the cost of the next job is minimized, yields an NE. Let $\sigma^\star$ denote the resulting schedule, and let $C_1(\sigma^\star) \leq C_2(\sigma^\star) \leq \ldots \leq C_n(\sigma^\star)$ be the sorted vector of jobs' completion times in $\sigma^\star$. The proof proceeds by showing that this vector corresponds to schedules that minimize the makespan, as well as the sum of completion times. Also, we show that every NE schedule induces the same cost vector as $\sigma^\star$.

First, we show that $\sigma^\star$ achieves the minimum makespan. Assume that there exists a schedule σ' such that $\max_j n_j(\sigma')/s_j < \max_j n_j(\sigma^\star)/s_j$. Let $M_1 = \text{argmax}_j n_j(\sigma^\star)/s_j$. It must be that $n_{M_1}(\sigma') < n_{M_1}(\sigma^\star)$. Since $\sum_j n_j(\sigma') = \sum_j n_j(\sigma^\star) = n$, there must be a machine M_2 such that $n_{M_2}(\sigma^\star) < n_{M_2}(\sigma')$. Thus, the last job on machine M_1 in $\sigma^\star$ can benefit from migrating to machine M_2, as its cost will be at most $(n_{M_2}(\sigma^\star)+1)/s_{M_2} \leq n_{M_2}(\sigma')/s_{M_2} \leq \max_j n_j(\sigma')/s_j < \max_j n_j(\sigma^\star)/s_j$. This contradicts the assumption that $\sigma^\star$ is an NE.

Second, we analyze the sum of completion times objective. For a schedule σ, the sum of completion times is $\sum_j (1+\ldots+n_j)/s_j$. Using similar arguments, if $\sigma^\star$ is not optimal with respect to the sum of completion times, there exists a beneficial migration from a machine whose contribution to the sum is maximal, to a machine with a lower contribution.

Now, let σ be an NE schedule with sorted cost vector and let $C_1(\sigma) \leq C_2(\sigma) \leq \ldots \leq C_n(\sigma)$, and assume by contradiction that it has a different cost vector than σ^*. Let i be the minimal index such that $C_i(\sigma^\star) \neq C_i(\sigma)$. Since σ and $\sigma^\star$ agree on the costs of the first $i-1$ jobs, and since $\sigma^\star$ assigns the i-th job on a minimal-cost machine, it holds that $C_i(\sigma^\star) < C_i(\sigma)$. We get a contradiction to the stability of σ, since some job can reduce its cost to $C_i(\sigma^\star)$. The first and the second step concludes the proof of theorem. $\square$

In Theorem 5.7 it is shown that an NE exists for any instance on two related machines. We now analyze the equilibrium inefficiency of this class. Let $\mathcal{G}_2^s$ denote the class of games played on two machines with speeds $s_1 = 1$ and $s_2 = s \leq 1$.

Theorem 5.14. *For the min-makespan objective, $PoA(\mathcal{G}_2^s) = PoS(\mathcal{G}_2^s) = s+1$ if $s \leq \frac{\sqrt{5}-1}{2}$, and $PoA(\mathcal{G}_2^s) = PoS(\mathcal{G}_2^s) = \frac{s+2}{s+1}$ if $s > \frac{\sqrt{5}-1}{2}$.*

Proof. Let $G \in \mathcal{G}_2^s$. Let $W = \sum_i p_i$ be the total processing time of all jobs. Assume first that $s \leq \frac{\sqrt{5}-1}{2}$. For the minimum makespan objective, $OPT(G) \geq W/(1+s)$. Also, for any NE σ, we have that $C_{\max}(\sigma) \leq W$, since every job can migrate to be last on the fast machine and have completion time at most W. Thus, PoA $\leq s+1$.

Assume next that $s > \frac{\sqrt{5}-1}{2}$. Let job a be a last job to complete in a worst Nash equilibrium σ, p_1 be the total processing time of all jobs different from a on machine 1, and p_2 be the total processing time of all jobs different from a on machine 2 in σ. Then since σ is a Nash equilibrium, $C_{\max}(\sigma) \leq p_1 + p_a$ and $C_{\max}(\sigma) \leq (p_2 + p_a)/s$. Combining these two inequalities yields

$$C_{\max}(\sigma) \leq \frac{W + p_a}{1+s} \leq \frac{s+2}{s+1} \cdot OPT(G),$$

where for the inequality we use that $OPT(G) \geq W/(1+s)$ and $OPT(G) \geq p_a$, and thus PoA$\leq \frac{s+2}{s+1}$.

For the PoS lower bound, assume first that $s < \frac{2}{\sqrt{5}+1}$. Consider an instance consisting of two jobs, a and b, where $p_a = 1$ and $p_b = 1/s$. The priority lists are $\pi_1 = \pi_2 = (a, b)$. The unique NE is that both jobs are on the fast machine. $C_a(\sigma) = 1, C_b(\sigma) = 1+1/s$. For every $s < \frac{\sqrt{5}-1}{2}$, it holds that $1 + 1/s < 1/s^2$, therefore, job b does not have a beneficial migration. An optimal schedule assigns job a on the slow machine, and both jobs complete at time $1/s$. The corresponding PoS is $s + 1$. For $s = \frac{\sqrt{5}-1}{2}$, by taking $p_b = 1/s + \varepsilon$, the PoS approaches $(s + 2)/(s + 1)$ as $\varepsilon \to 0$.

Assume now that $s > \frac{\sqrt{5}-1}{2}$. Consider an instance consisting of three jobs, x, y and z, where $p_x = 1, p_y = s^2 + s - 1$, and $p_z = s + 1$. The priority lists are $\pi_1 = \pi_2 = (x, y, z)$. Note that $p_y \geq 0$ for every $s \geq \frac{\sqrt{5}-1}{2}$. In all NE, job x is on the fast machine, and job y is on the slow machine. Indeed, job y prefers being alone on the slow machine since $s^2 + s > \frac{s^2+s-1}{s}$. Job z is indifferent between joining x on the fast machine or y on the slow machine, since $1 + p_z = (p_y + p_z)/s = s + 2$. In an optimal schedule, job z is alone on the fast machine, and jobs x and y are on the slow machine. Both machines have the same completion time $s + 1$. The PoS is $\frac{s+2}{s+1}$. □

Theorem 5.15. *For the sum of completion times objective, PoA($\mathcal{G}_2^s$) $= \Theta(n)$ and PoS($\mathcal{G}_2^s$) $= \Theta(n)$ for all $s \leq 1$.*

Proof. For the upper bound, note that $OPT(G) \geq \sum_i p_i$ and in every NE schedule σ, $C_i(\sigma) \leq \sum_i p_i$. This implies PoA $= \Theta(n)$. For the PoS lower bound, consider an instance consisting of a set Z of $n - 2$ jobs with processing time ε, and two jobs, a and b, where $p_a = 1$ and $p_b = s$. The priority lists are $\pi_1 = \pi_2 = (a, b, Z)$. Note that $p_a + p_b > p_b/s$, therefore, in every NE, job a is first on M_1 and job b is first on M_2. Thus, every ε-job has completion time at least 1. The sum of completion times is at least $n + O(n^2)\varepsilon$. An optimal schedule assigns a and b on M_1 and all the ε-jobs on M_2. The sum of completion times is at most $3 + O(n^2)\varepsilon/s$. For small enough ε, we get that the PoS is $\Theta(n)$. □

We turn to analyze the equilibrium inefficiency of the class $\mathcal{G}_3$, consisting of games played on identical-speed machines, having machine-dependent priority lists. The proof of the following theorem is based on the observation that every NE schedule is a possible outcome of Graham's *List-scheduling* (LS) algorithm [Gra66].

Theorem 5.16. *For the min-makespan objective, PoA($\mathcal{G}_3$) =PoS ($\mathcal{G}_3$) $= 2 - \frac{1}{m}$.*

Proof. Let σ be an NE schedule. We claim that σ is a possible outcome of Graham's *List-scheduling* algorithm [Gra66]. Indeed, assume that List-scheduling is performed and the jobs are considered according to their start time in σ. Every job selects its machine in σ, as otherwise, we get a contradiction to the stability of σ. Since List-scheduling provides a $2 - \frac{1}{m}$ approximation to the makespan, we get the upper bound of the PoA.

For the lower bound, given $m > 1$, the following is an instance for which PoS$= 2 - \frac{1}{m}$. The instance consists of a single job with processing time m and $m(m-1)$ unit jobs. In all priority lists, the heavy job is last and the unit jobs are prioritized arbitrarily. It is easy to verify that in every NE profile the unit jobs are partitioned in a balanced way among the machines, and the heavy job is assigned as last on one of the machines. Thus, the completion time of the heavy job is $2m - 1$. On the other hand, an optimal assignment assign the heavy job on a dedicated machine, and partition the unit job in a balanced way among the remaining $m - 1$ machines. In this profile, all the machines have load m. The corresponding PoS is $\frac{2m-1}{m} = 2 - \frac{1}{m}$. □

Theorem 5.17. *For the sum of completion times objective, PoA($\mathcal{G}_3$) $\leq \frac{n-1}{m} + 1$, and for every $\varepsilon > 0$, PoS($\mathcal{G}_3$) $\geq \frac{n}{m} - \varepsilon$.*

Proof. For the upper bound of the PoA, note that, independent of the number of machines, the sum of completion times is at least $\sum_i p_i$. Also, for every job a, if a is not assigned on any machine, then there exists a machine with load at most $\frac{\sum_{i \neq a} p_i}{m}$, therefore, in every NE profile, the completion time of job a is at most $\frac{\sum_{i \neq a} p_i}{m} + p_a$. Summing this equation for all the jobs, we get that the sum of completion times of any NE is at most

$$\frac{n \sum_i p_i - \sum_i p_i}{m} + \sum_i p_i = \sum_i p_i \frac{n-1}{m} + 1.$$

We conclude that the PoA is at most $\frac{n-1}{m} + 1$.

For the PoS lower bound, given m, let $\varepsilon \to 0$, and consider an instance with $n = km$ jobs, out of which, m jobs $j_1, \ldots, j_m$ have length 1 and the other $(k-1)m$ jobs have length ε. Assume that π_i gives the highest priority to j_i then to all the ε-jobs, and then to the other $m-1$ unit jobs.

In every NE, machine i processes first the unit-job j_i, followed by $k-1$ ε-jobs. Thus, every job has completion time at least 1. The sum of completion times is $n + O(mk^2)\varepsilon$. On the other hand, an optimal solution assigns on machine i a set of $k-1$ ε-jobs followed by one unit-job j_k for $k \neq i$, resulting in a sum of completion times of $m + O(mk^2)\varepsilon$. The PoS tends to $\frac{n}{m}$ as ε decreases. $\square$

The last class of instances for which an NE is guaranteed to exist includes games with a global priority list, and is denoted by $\mathcal{G}_4$. It is easy to verify that for this class, the only NE profiles are those produced by List-Scheduling algorithm, where the jobs are considered according to their order in the priority list. Different NE may be produced by different tie-breaking rules. Thus, the equilibrium inefficiency is identical to the approximation ratio of LS [CS80]. Since the analysis of LS is tight, this is also the PoS.

Theorem 5.18. *For the min-makespan objective, PoS($\mathcal{G}_4$) =PoA($\mathcal{G}_4$) $= \Theta(m)$.*

For the sum of completion times objective, we note that the proof of Theorem 5.15 for two related machines uses a global priority list. The analysis of the PoA is independent of the number and speeds of machines.

Theorem 5.19. *For the sum of completion times objective, PoA($\mathcal{G}_4$) $= \Theta(n)$ and PoS($\mathcal{G}_4$) $= \Theta(n)$.*

5.4 Hardness of Computing an NE with Low Social Cost

Correa and Queyranne [CQ12] showed that if all the machines have the same speeds, but arbitrary priority lists, then an NE is guaranteed to exist, and can be calculated by a simple greedy algorithm.

In this section we discuss the complexity of computing a good NE in this setting. We refer to both objectives of minimum makespan and minimum sum of completion times. For both objectives, our results are negative. Specifically, not only that it is NP-hard to compute the best NE, but it is also hard to approximate it, and to compute an NE whose social cost is better than the one guaranteed by the PoA bound.

Starting with the minimum makespan, in Theorem 5.16, we have shown that the PoA for this objective is at most $2-\frac{1}{m}$. We show that we cannot hope for a better algorithm than the simple greedy algorithm. More formally, we prove that it is NP-hard to approximate the best NE within a factor of $2 - \frac{1}{m} - \varepsilon$ for all $\varepsilon > 0$.

Theorem 5.20. *If for all machines $s_j = 1$, then it is NP-hard to approximate the best NE w.r.t. the makespan objective within a factor of $2 - \frac{1}{m} - \varepsilon$ for all $\varepsilon > 0$.*

Proof. We show that for every $\varepsilon > 0$, there is an instance on m identical machines for which it is NP-hard to decide whether the game has an NE profile with makespan at most $m + 3\varepsilon$ or at least $2m - 1$.

The hardness proof is by a reduction from Maximum Bounded 3-Dimensional Matching (Max-3DM-3). Recall that the input of the Max-3DM-3 problem is a set of triplets $T \subseteq X \times Y \times Z$, where $|T| \geq n$ and $|X| = |Y| = |Z| = n$, and the number of occurrences of every element of $X \cup Y \cup Z$ in T is at most 3. The goal is to decide whether T has a 3-dimensional matching of size n.

Given an instance of Max-3DM-3 and $\varepsilon > 0$, consider the following game on $m = |T| + 2$ machines, $M_1, M_2, \ldots, M_{|T|+2}$. The set of jobs includes job a with processing time m, job b with processing time $m - 1$, a set D of $|T| - n$ dummy jobs with processing time 3ε, two dummy jobs d_1, d_2 with processing time 2ε, a set U of $(m-1)^2$ unit jobs, and $3n$ jobs with processing time ε—one for each element in $X \cup Y \cup Z$.

We turn to describe the priority lists. We remark that when the list includes a set, it means that the set elements appear in arbitrary order. The symbol ϕ means that the remaining jobs appear in arbitrary order. For the first machine, $\pi_1 = (d_1, b, a, U, \phi)$. For the second machine, $\pi_2 = (d_2, X, Y, Z, b, U, a, d_1)$. The remaining $m - 2$ machines are *triplet-machines*. For every $t = (x_i, y_j, z_k) \in T$, the priority list of the triplet-machine corresponding to t is $(D, x_i, y_j, z_k, U, \phi)$.

The heart of the reduction lies in determining the priority lists. The idea is that if a 3D-matching exists, then job b would not prefer M_1 over M_2. This will enable job a to be assigned early on M_1. However, if a 3D-matching does not exist, then some job originated from the elements in $X \cup Y \cup Z$ will precede job b on M_2, and b's best-response would be on M_1. The jobs in U have higher priority than job a on all the machines except for M_1, thus, unless job a is on M_1, it is assigned after $|U|/(m-1)$ unit-jobs from U, inducing a schedule with high makespan.

Observe that in any NE, the two dummy jobs with processing time 2ε are assigned as the first jobs on M_1 and M_2. Also, the dummy jobs in D have the highest priority on the triplet-machines, thus, in every NE, there are $|D| = |T| - n$ triplet-machines on which the first job is from D.

Figure 5.2 provides an example for $m = 5$.

In order to complete the gap reduction, we need the following two claims for the upper and lower threshold. First, we show that if a perfect matching exists, then this guarantees an NE in the associated scheduling problem instance with makespan at most $m + 3\varepsilon$.

Claim 5.21. *If a 3D-matching of size n exists, then there is an NE with makespan $m+3\varepsilon$.*

Proof. Let T' be a matching of size n. Assign the jobs of $X \cup Y \cup Z$ on the triplet-machines corresponding to T' and the jobs of D on the remaining triplet-machines. Also, assign d_1 and d_2 on M_1 and M_2 respectively. M_1 and M_2 now have load 2ε while the triplet machines have load 3ε. Next, assign job a on M_1 and job b on M_2. Finally, add the unit-jobs as balanced as possible: m jobs on each triplet-machine and a single job after job b on M_2. It is easy to verify that the resulting assignment is an NE. Its makespan is $m + 3\varepsilon$. □

The next claim proves the other direction of the reduction. That is, any NE with makespan less than $2m - 1$ induces a perfect matching.

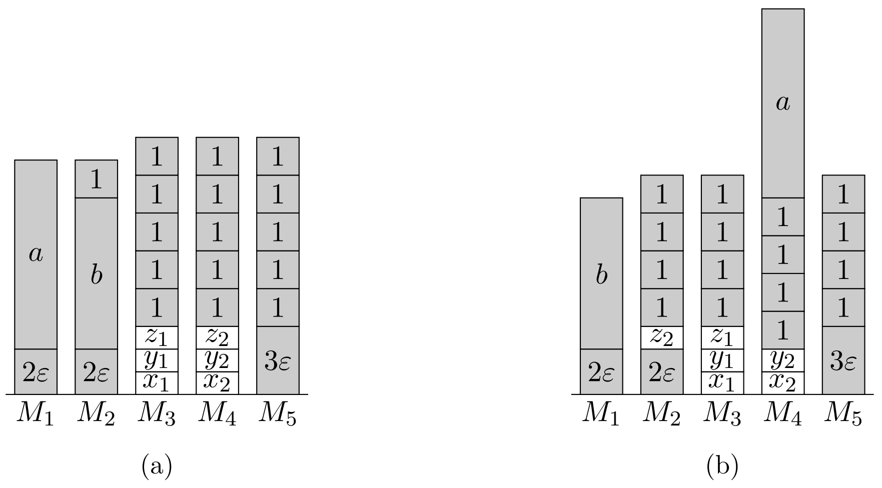

Figure 5.2: Let $n = 2$ and $T = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_1, y_2, z_2)\}$. (a) an NE given the matching $T' = \{(x_1, y_1, z_1), (x_2, y_2, z_2)\}$. The makespan is $5+3\varepsilon$. (b) an NE if a matching of size 2 is not found. Job z_2 is stable on M_2, thus, job b prefers M_1 over M_2. The makespan is $9 + 2\varepsilon$.

Claim 5.22. *If there is an NE with makespan less than $2m - 1$, then there exists a $3D$-matching of size n.*

Proof. Let σ be an NE whose makespan is less than $2m-1$. Since $p_a = m$ and $p_b = m-1$, this implies that a is not assigned after b on M_1 or on M_2. Also, since jobs of U have higher priority than a on all the machines except for M_1, it holds that a is not assigned after $m - 1$ unit-jobs. Thus, it must be that job a is processed on M_1 and job b is not on M_1. Job b does not prefer M_1 over M_2, only if it starts its processing right after job d_2 on M_2. Since the jobs of $X \cup Y \cup Z$ have higher priority than job b on M_2, they are all assigned on triplet-machines and starts their processing after jobs of total processing time at most 2ε. Thus, every triplet machine processes at most three jobs of $X \cup Y \cup Z$—the jobs corresponding to the triplet, whose priority is higher than the priority of the unit-jobs of U. Moreover, since the jobs of D have higher priority on the triplet-machines, there are $|T| - n$ triplet-machines on which the jobs of D are first and exactly n machines each processing exactly the three jobs corresponding to the machine's triplet. Thus, the assignment of the jobs from $X \cup Y \cup Z$ on the triplet-machines induces a matching of size n. □

Claims 5.21 and 5.22 conclude the proof of the theorem. □

We turn to analyze the complexity of computing the best NE with respect to the sum of completion times. Traditionally, this objective is simpler than minimizing the makespan, as the problem can be solved efficiently by SPT-rule if there are no priorities. We show that even in the simple case of identical machines, in which an NE is guaranteed to exists [CQ12], it is NP-hard to approximate the solution's value.

Theorem 5.23. *If for all machines $s_j = 1$, then for any $r > 1$, it is NP-hard to approximate the best NE w.r.t. the sum of completion times within factor r.*

Proof. Given m, r, let k be a large integer such that $\frac{m+k}{m+1} > r$. We show that for every $r > 1$, there is an instance on m identical machines for which it is NP-hard to decide whether the game has an NE profile with sum of completion times at most $m+1$ or more than $m+k$.

The hardness proof is again by a reduction from Maximum Bounded 3-Dimensional Matching (Max-3DM-3). Given an instance of Max-3DM-3 and $r>1$, consider the following game on $m=|T|+2$ machines, $M_1, M_2, \ldots, M_{|T|+2}$. Recall that k satisfies $\frac{m+k}{m+1} > r$. Let $\varepsilon > 0$ be a small constant, such that $(k^2+3k+6m)\varepsilon < 1$. The set of jobs includes job a with processing time ε, job b with processing time 1, a set D of $|T|-n$ dummy jobs with processing time 3ε, two dummy jobs d_1, d_2 with processing time 2ε, a set U of $m-1$ unit jobs, $3n$ jobs with processing time ε—one for each element in $X \cup Y \cup Z$, and a set K of k jobs with processing time ε. Note that there are exactly m unit jobs (the job b and the jobs of U), while all other jobs have $O(\varepsilon)$ processing time.

We turn to describe the priority lists. Note that, when the list includes a set, it means that the set elements appear in arbitrary order. The symbol ϕ means that the remaining jobs appear in arbitrary order. For the first machine, $\pi_1 = (d_1, b, a, K, U, \phi)$. For the second machine, $\pi_2 = (d_2, X, Y, Z, b, U, a, K, \phi)$. The remaining $m-2$ machines are *triplet-machines*. For every $t=(x_i, y_j, z_k) \in T$, the priority list of the triplet-machine corresponding to t is $(D, x_i, y_j, z_k, a, U, K, \phi)$.

The heart of the reduction lies in determining the priority lists. The idea is that if a 3D-matching exists, then job b would not prefer M_1 over M_2. This will enable job a and all the tiny jobs of K to be assigned early on M_1 each having completion time at most $(k+3)\varepsilon$. However, if a 3D-matching does not exist, then some job originated from the elements in $X \cup Y \cup Z$ will precede job b on M_2, and b's best-response would be on M_1. The jobs in U have higher priority than a and K on M_2, thus, on every machine there would be at least one job of length 1 that precedes the jobs of K, implying that the sum of completion times will be more than $m+k$.

Observe that in any NE, the two dummy jobs with processing time 2ε are assigned as the first jobs on M_1 and M_2. Also, the dummy jobs in D have the highest priority on the triplet-machines, thus, in every NE, there are $|D| = |T|-n$ triplet-machines on which the first job is from D.

Figure 5.3 provides an example for $m=5$.

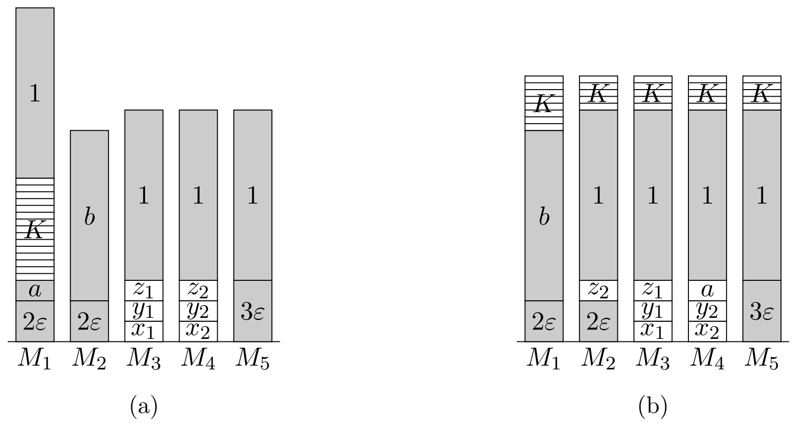

Figure 5.3: Let $n=2$ and $T=\{(x_1,y_1,z_1),(x_2,y_2,z_2),(x_1,y_2,z_2)\}$. (a) an NE given the matching $T'=\{(x_1,y_1,z_1),(x_2,y_2,z_2)\}$. The jobs of K start their processing at time 3ε. (b) an NE if a matching of size 2 does not exist. Job z_2 selects M_2, thus, job b prefers M_1 over M_2. The jobs of K start their processing at time at least $1+2\varepsilon$.

The following claims prove the lower and the upper threshold in the gap instance of the scheduling problem.

Claim 5.24. *If a $3D$-matching of size n exists, then there is an NE with sum of completion time at most $m + 1$.*

Proof. Let T' be a matching of size n. Assign the jobs of $X \cup Y \cup Z$ on the triplet-machines corresponding to T' and the jobs of D on the remaining triplet-machines. Also, assign d_1 and d_2 on M_1 and M_2 respectively. M_1 and M_2 now have load 2ε while the triplet machines have load 3ε. Next, assign job a and the jobs of K on M_1, and job b on M_2. Finally, add one unit-job on each triplet-machine and on M_1. It is easy to verify that the resulting assignment is an NE. The jobs of K are not delayed by unit jobs, so each of them completes at time at most $(k + 3)\varepsilon$. The other jobs with processing time $O(\varepsilon)$ contributes at most 6ε to the sum of completion times on every machine, and every unit job completes at time $1 + O(\varepsilon)$. Thus, the sum of completion times is $m + f(\varepsilon)$, where ε was chosen such that $f(\varepsilon) < 1$. □

Claim 5.25. *If there is an NE with sum of completion times less than $m + k$, then there exists a $3D$-matching of size n.*

Proof. Let σ be an NE whose sum of completion times is less than $m + k$. There are m unit jobs, and in any NE, each is processed on a different machine, as otherwise, some machine has load $f(\varepsilon)$, and the second unit job on a machine has a beneficial migration. In order to have sum of completion times less than $m + k$, at least one job from K is not assigned after a unit job. The only machine on which jobs from K may precede a unit job is M_1, where jobs of K may precede a job from U. This is possible only if job b is not processed on M_1. Job b does not prefer M_1 over M_2, only if it starts its processing right after job d_2 on M_2. Since the jobs of $X \cup Y \cup Z$ have higher priority than job b on M_2, they are all assigned on triplet-machines and starts their processing after jobs of total processing time at most 2ε. Thus, every triplet machine processes at most three jobs of $X \cup Y \cup Z$—the jobs corresponding to the triplet, whose priority is higher than the priority of the unit-jobs of U. Moreover, since the jobs of D have higher priority on the triplet-machines, there are $|T| - n$ triplet-machines on which the jobs of D are first, and exactly n machines each processing exactly the three jobs corresponding to the machine's triplet. Thus, the assignment of the jobs from $X \cup Y \cup Z$ on the triplet-machines induces a matching of size n. □

Claims 5.24 and 5.25 conclude the proof of theorem. □

Note that, given m, r, the game built in the reduction has $n < (r + 3)m$ jobs. That is, $r > \frac{n}{m} - 3$. Also, as shown in Theorem 5.17, for the sum of completion times objective, $\text{PoA}(\mathcal{G}_3) \leq \frac{n-1}{m} + 1$. Thus, the above analysis shows that up to a small additive constant, it is NP-hard to compute an NE that approximates the optimal sum of completion time better than the PoA.

Chapter 6

Weighted Congestion Games with Resource-Dependent Priority Lists

In the last chapter we studied scheduling games with machine-dependent priority lists and analyzed the effect of fixed order scheduling on the corresponding job-scheduling game. We analyzed algorithmic aspects such as existence of pure Nash equilibria, the complexity of identifying whether an NE profile exists, the complexity of computing an NE, and bounded the equilibrium inefficiency in these games. We now study a natural generalization of the selfish scheduling game by allowing for arbitrary sized strategy sets to the users in the game. We believe that our extension could succinctly model and allow for analyzing many of the socio-economic problems arising in areas such as transportation networks.

6.1 Introduction

The recent past saw a rising demand on the road transportation networks across the world. The steady increase in the production of motor vehicles due to the continuous demand for logistics and mobility, necessitated well planned transportation networks. However, the non-cooperative behavior among the users of the network resulted in sub-optimal traffic flows and avoidable congestion. This selfish behavior led the scientific community to study the problem from a game theoretic perspective. In particular, to represent the problem as a strategic resource allocation game. Easley and Kleinberg [EK12] model the traffic network as a congestion game [Ros73a]. Although, congestion games remain to be a common and well studied model for traffic networks, it has no element of time dependency and assumes that all users on a resource experience the same latency. In a road network, a vehicle can only be delayed by the vehicles ahead of it. Farzad et al. [FOV08] introduced a simple modification that allows to incorporate some element of time dependency. They proposed a static model in which resources have priorities over the users, for example depending on when a player arrives at that resource, so as to model that a user will only delay other users on the resource that have a lower priority.

In Chapter 5, we studied selfish scheduling with machine-dependent priority list. We now extend this model to allow for non-singleton strategy space. Recall that we introduced weighted congestion games with resource-dependent priority lists in Chapter 1 as a natural generalization to scheduling games with machine-dependent priority lists. Our extended model is primarily motivated by its application to traffic networks. We assume that each resource has a fixed priority list and then provide a detailed analysis in terms of the existence and inefficiency of NE. Let us begin by recollecting the model definition and preliminaries. We remark that the readers may choose to skip the following subsection.

6.1.1 The Model

An instance of a *weighted congestion game with resource-dependent priority lists* in which the strategy space of a player consists of *subsets of resources* is given by a tuple

$$G = (\mathcal{N}, E, (S_i)_{i\in\mathcal{N}}, (w_i)_{i\in\mathcal{N}}, (c_e)_{e\in E}, (\pi_e)_{e\in E}),$$

where $\mathcal{N} := \{1, \dots, N\}$ is a finite set of players, E is a finite set of resources, $S_i \subseteq 2^E$ is the set of feasible strategies for player $i \in \mathcal{N}$, $w_i \in \mathbb{R}_{\geq 0}$ is the weight of player $i \in \mathcal{N}$, $c_e \in \mathbb{R}_{\geq 0}$ is the cost coefficient of resource $e \in E$, and $\pi_e : \mathcal{N} \mapsto \{1, \dots, N\}$ is the priority list of resource E that defines its preference over the players using it.

Observe that scheduling games described in Chapter 5 are symmetric singleton congestion games in which the strategy space of each player is the set of all resources. For the general setting, the players' costs are defined as follows.

Given a strategy profile $s = (s_i)_{i\in\mathcal{N}} \in S := S_1 \times \ldots \times S_N$, for every player $i \in \mathcal{N}$ and resource $e \in s_i$, let us denote by

$$B_{ie}(s) = \{i' \in \mathcal{N} \mid e \in s_{i'} \wedge \pi_e(i') \leq \pi_e(i)\}$$

the set of players on resource e that delay player $i \in \mathcal{N}$ in the strategy profile s and define $w_e^i(s) = \sum\limits_{i'\in B_{ie}(s)} w_{i'}$.

The cost of a player $i \in \mathcal{N}$ is given by

$$C_i(s) = w_i \cdot \sum_{e\in s_i} c_e \cdot w_e^i(s).$$

Notice that here we assume that players' costs are multiplied by their weight, whereas we did not make such assumption for scheduling games in Chapter 5. However, we remark that this has no implications for the existence of pure Nash equilibria, but only affects the efficiency results. Each player chooses a strategy so as to minimize their cost. A strategy profile $s \in S$ is a pure Nash equilibrium if for all $i \in \mathcal{N}$ and all $s_i' \in S_i$, we have $C_i(s) \leq C_i(s_i', s_{-i})$. For a strategy profile s, let $C(s)$ denote the social cost of s and is given by

$$C(s) = \sum_{i\in\mathcal{N}} C_i(s).$$

The inefficiency of equilibria arising due to the non-cooperative behavior of the players is measured according to the *price of anarchy* (PoA) [KP99]. Recall that the PoA is the worst-case inefficiency of a pure Nash equilibrium w.r.t. the social optimum.

6.1.2 Our Contribution

We generalize the model of scheduling games with machine dependent priority list to allow for arbitrary strategy sets. In Section 6.2 we show that in general, even with unweighted players, a pure Nash equilibrium need not exist by making use of the famous Condorcet paradox [MdC85]. We then use this example to prove that the question whether a pure Nash equilibrium exists is NP-hard, even with unit weight players. Additionally, we show that it is NP-hard to determine if a $(3/2-\varepsilon)$-approximate pure Nash equilibrium exists for any $\varepsilon > 0$. On the positive side, in Section 6.2.1 we show that matroid congestion games with unit weight players always possess a pure Nash equilibrium. Finally in Section 6.3 we study the price of anarchy with respect to the sum of weighted costs and show that the upper bound of 4 proven by Cole et al. [CCG+15] for unrelated machine scheduling with Smith's rule also extends to congestion games with resource-dependent priority lists. This ratio is smaller than the price of anarchy of the atomic game with priorities defined by Farzad et al. [FOV08]. We then extend our analysis to games with polynomial cost functions of some fixed degree d.

6.1.3 Related Work

Rosenthal [Ros73a] proved that congestion games are potential games and thus always exhibit a pure Nash equilibrium. Awerbuch et al. [AAE13] and Christodoulou and Koutsoupias [CK05b] proved a tight bound of 5/2 for affine congestion games, and 2.618 for weighted affine congestion games. Fotakis et al. [FKS05] showed that weighted congestion games need not have a pure Nash equilibrium, unless we restrict cost functions to be linear. Ackermann et al. [ARV09] proved that as long as strategy spaces are restricted to be matroidal, then a Nash equilibrium always exists.

Ackermann et al. [AGM+08] were the first to study a congestion game with priorities. They proposed a model in which users with higher priority on a resource displace users with lower priorities such that the latter incur infinite cost. Closest to our model is Farzad et al. [FOV08], who studied priority based selfish routing for non-atomic and atomic users and analyzed the inefficiency of equilibria. Recently, Biló and Vinci [BV20] studied a congestion game with a global priority classes that can contain multiple jobs and characterize the price of anarchy as a function of the number of classes. Gourvès et al. [GMMT15] studied capacitated congestion games to characterize the existence of pure Nash equilibria and computation of an equilibrium when they exist. Pilouras et al. [PNS16] assumed that the priority lists are unknown to the players a priori and consider different risk attitudes towards having a uniform at random ordering.

In the following sections we investigate various algorithmic aspects of our generalization. First, we show that a Nash equilibrium need not exist and in fact, the question whether a Nash equilibrium exists is NP-complete, even for unit weight players. Recall that in unit weight singleton games an NE is guaranteed to exist [FKK+09]. Second, we prove that an NE is guaranteed to exist if we consider matroid congestion games with unweighted players. Third, we show a tight bound on the price of anarchy for the sum of weighted costs in case of affine cost functions. Finally, we bound the PoA for games with non-decreasing cost functions that are polynomials of maximum degree d.

6.2 Equilibrium Existence

In this subsection, we restrict ourselves to congestion games with priority lists and unit weights, i.e., $w_i = 1$ for all $i \in \mathcal{N}$. We first provide an example that shows that a Nash equilibrium need not exist. Farzad et al. [FOV08] give a different example with two players for which an NE need not exist. Our example describes a symmetric game.

Example 6.1. The game $G^\star$ contains 3 players, $w_i = 1$ for all $i \in \mathcal{N}$, and 6 resources. Each players $i \in N$ has two pure strategies: $\{e_1, e_2, e_3\}$ and $\{e_4, e_5, e_6\}$. The delays are equal to 1 for all resources, and the priority lists are $\pi_j(i) = i + j - 1$ (modulo 3) for all $j \in E$ and $i \in \mathcal{N}$. Observe that there is no Nash equilibrium if all three players choose the same three resources. Also, due to the Condorcet paradox [MdC85], there is no Nash equilibrium in which two players choose one subset of resources and the other player chooses the other. Specifically, one of these two players has cost 5 and the other has cost 4. By deviating to the other triplet of resources, the player whose cost is 5 can reduce its cost to 4.

A natural question is to decide whether a game instance with unit weight players has an NE profile. Our next result shows that this is an NP-hard problem. The hardness proof is different from the one in Theorem 5.3, since this proof considers unit weight players and multiple-resources strategies, while that proof is for weighted players and singleton strategies.

Theorem 6.2. *Given an instance of a congestion game with priority lists, it is NP-complete to decide whether the game has an NE profile. This is valid also for unit weight players.*

Proof. Given a profile s, verifying that it is a NE can be done in polynomial time by considering the players one after the other and checking for each player, whether its current strategy is its best-response to the other players' strategies in s.

The hardness proof is by a reduction from 4-exact cover. The input to the 4-exact cover problem for a given n, is a set $U = \{a_1, \ldots, a_{4n}\}$ of elements, and a set Q of subsets of U. Every element in Q is a set of 4 elements from U. The goal is to decide whether there exists a subset $Q' \subseteq Q$, such that $|Q'| = n$, and every element in U appears exactly once in Q'. For example, let $U = \{a, b, c, d, e, f, g, h\}$, and $Q = \{\{a, b, c, d\}, \{c, d, e, f\}, \{e, f, g, h\}\}$. Then the answer is positive for $Q' = \{\{a, b, c, d\}, \{e, f, g, h\}\}$. 3-exact cover is one of Karp's 21 NP-complete problems [Kar72]. By adding dummy elements to an instance of 3-exact cover, it is easy to extend the hardness to cover by 4-sets.

Recall that $G^\star$ is our no-NE game described above. The idea is to construct a game consisting of n copies of $G^\star$, and additional resources and strategies. Recall that $G^\star$ is a 3-player game. In the game we construct, if a matching of size n exists, then no copy of $G^\star$ attracts three players, while if the maximum matching has size less than n, then at least one copy of $G^\star$ attracts three players and no NE exists.

Given an instance of 4-exact cover, we construct a game with $3n$ players and $10n$ resources as follows. First, for every element in U we have one resource. In addition, for every $1 \leq k \leq n$, we have a copy of the 6 resources of $G^\star$, specifically, $E^\star_k = \{e^k_1, e^k_2, e^k_3, e^k_4, e^k_5, e^k_6\}$. All the resources have the same cost function $c_j = 1$.

For every $1 \leq k \leq n$, we have three players. The first is denoted P^k_1, and its strategy space consists of all sets of four resources from U that form an element in Q, and the two strategies of player 1 in $G^\star$, played over the resources $E^\star_k$, that is, $\{e^k_1, e^k_2, e^k_3\}$ and $\{e^k_4, e^k_5, e^k_6\}$. The two additional players are denoted P^2_k and P^3_k, and their strategy space consists of the two strategies of Players 2 and 3 in $G^\star$, played over the resources E_k, that is, $\{e^k_1, e^k_2, e^k_3\}$ and $\{e^k_4, e^k_5, e^k_6\}$.

We turn to specify the resources' priority lists: The resources of $E^\star_k$ have the same priority lists as in $G^\star$, where the players P^1_k, P^2_k, P^3_k correspond to players $1, 2, 3$ respectively. The priority lists of the resources in U are arbitrary.

We show that G has an NE if and only if an exact cover of size n exists. First, if an exact cover Q' of size n exists, then the following profile is an NE: For every $1 \leq k \leq n$, player P^k_1 selects the four resources corresponding to subset k in Q', player P^k_2 selects $\{e^k_1, e^k_2, e^k_3\}$ and player P^k_3 selects $\{e^k_4, e^k_5, e^k_6\}$. The cost of every player P^k_1 is 4 - since it is the only user of four resources. The cost of every player P^k_2 or P^k_3 is 3 - since each of them is the only user of three resources. It is easy to verify that this profile is an NE.

Assume that no exact cover of size n exists. Thus, in every profile, at least one resource from U is used by at least two players $P_1^{k_1}$ and $P_1^{k_2}$. The cost of one of these players, say $P_1^{k_1}$, is therefore at least 5, and deviating to a strategy in $G^\star_{k_1}$ is beneficial for it. Specifically, by deviating to $\{e_1^{k_1}, e_3^{k_1}, e_3^{k_1}\}$, its cost reduces to 4. Thus, if no matching of size n exists, then for at least one $1 \leq k \leq n$, the player P_1^k is trapped in our no-NE game $G^\star_k$ and the whole game has no NE. □

Inapproximability

We now extend the construction in Farzad et al. [FOV08] to show that it is NP-complete to decide whether the game has a $(3/2-\varepsilon)$-approximate pure Nash equilibrium for any $\varepsilon > 0$.

Example 6.3. The game $G^\star$ contains 2 unit weight players, and 4 resources $\{e_1, \dots, e_4\}$. Player 1 has two pure strategies: $\{e_1, e_2\}$ and $\{e_3, e_4\}$. Player 2 has two pure strategies: $\{e_1, e_4\}$ and $\{e_2, e_3\}$. The cost coefficient is equal to 8/9 for all resources, and the priority lists are: $e_1 = e_3 := 2 \succ 1$ and $e_2 = e_4 := 1 \succ 2$. Observe that there is no exact pure Nash equilibrium in this game. Since, one of these two players has cost 24/9 and the other has cost 16/9. By deviating to the other pair of resources, the player whose cost is 24/9 can reduce its cost to 16/9. Therefore, the game also does not have a $(3/2-\varepsilon)$-approximate PNE for any $\varepsilon > 0$.

Theorem 6.4. *Given an instance of a congestion game with priority lists, it is NP-complete to decide whether the game has a $(3/2-\varepsilon)$-approximate pure Nash equilibrium for any $\varepsilon > 0$.*

Proof. Given a strategy profile, one can verify in polynomial time if it is an approximate pure Nash equilibrium. The hardness proof is by a reduction from 3-exact cover. The input to the 3-exact cover problem for a given n, is a set $U = \{a_1, \dots, a_{3n}\}$ of elements, and a set Q of subsets of U. Every element in Q is a set of 3 elements from U. The goal is to decide whether there exists a subset $Q' \subseteq Q$, such that $|Q'| = n$, and every element in U appears exactly once in Q'.

Given an instance of 3-exact cover, we construct a game with $2n$ players and $7n$ resources as follows. First, for every element in U we have one resource with cost coefficient $c_j = 2.01/3$. In addition, for every $1 \leq k \leq n$, we have a copy of the 4 resources of $G^\star$ (Example 6.3), i.e, $E^\star_k = \{e_1^k, e_2^k, e_3^k, e_4^k\}$. All the resources in $E^\star_k$ have the same cost coefficient $c_j = 8/9$.

For every $1 \leq k \leq n$, we have two players. The first is denoted P_1^k, and its strategy space consists of all sets of three resources from U that form an element in Q, and the two strategies of player 1 in $G^\star$, played over the resources $E^\star_k$, that is, $\{e_1^k, e_2^k\}$ and $\{e_3^k, e_4^k\}$. The second players are denoted P_2^k, and their strategy space consists of the two strategies of Players 2 in $G^\star$, played over the resources E_k, i.e, $\{e_1^k, e_4^k\}$ and $\{e_2^k, e_3^k\}$. The resources of $E^\star_k$ have the same priority lists as in $G^\star$, where the players P_1^k, P_2^k correspond to players $1, 2$ respectively.

The priority lists of the resources in U are arbitrary.

We show that G has an NE if and only if an exact cover of size n exists. First, if an exact cover Q' of size n exists, then the following profile is an NE: For every $1 \leq k \leq n$, player P_1^k selects the three resources corresponding to subset k in Q'. Player P_2^k selects arbitrarily picks one of their strategy. It is easy to verify that this profile is a 3/2-approximate NE.

Assume that no exact cover of size n exists. Thus, in every profile, at least one resource from U is used by at least two players $P_1^{k_1}$ and $P_1^{k_2}$. The cost of one of these players, say $P_1^{k_1}$, is therefore at least 8.04/3, and deviating to a strategy in $G^\star_{k_1}$ can improve by a factor greater than 3/2. Thus, if no matching of size n exists, then for at least one $1 \leq k \leq n$, the player P_1^k is trapped in our no-NE game $G^\star_k$ and the whole game has no $(3/2 - \varepsilon)$-approximate NE. $\square$

6.2.1 Matroid Congestion Games

If we restrict the class of allowable strategy sets to be bases of a matroid, we obtain some positive results. A *matroid* is a tuple $M = (E, \mathcal{I})$, where E is a finite set and $\mathcal{I} \subseteq 2^E$ is a non-empty family of subsets of E such that:

1. $\emptyset \in \mathcal{I}$,
2. if $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$, and
3. if $X, Y \in \mathcal{I}$ with $|X| > |Y|$, then there exists an $e \in X \setminus Y$ such that $Y \cup \{e\} \in \mathcal{I}$.

We call a set $X \in \mathcal{I}$ an *independent set*. An inclusion-wise maximal independent set of $\mathcal{I}$ is called a basis of matroid M. It is well-known that all bases have the same cardinality. We denote this cardinality by the rank $rk(M)$. We refer the readers to Korte and Vygen [KV12] for a detailed discussion on matroids.

Matroid congestion games were introduced in Ackermann et al. [ARV08]. A *matroid congestion game with priority lists* is a congestion game with priority lists denoted by a tuple

$$G = (\mathcal{N}, E, (S_i)_{i\in\mathcal{N}}, (w_i)_{i\in\mathcal{N}}, (c_e)_{e\in E}, (\pi_e)_{e\in E}),$$

if for all $i \in \mathcal{N}$, $M_i = (E, \mathcal{I}_i)$ with $\mathcal{I}_i = \{I \subseteq \Sigma \mid \Sigma \in S_i\}$ is a matroid and S_i is the set of bases of M_i.

Theorem 6.5. *If $w_i = 1$ for all $i \in \mathcal{N}$ in a matroid congestion game, then an NE exists and can be calculated efficiently.*

Proof. The following algorithm finds a pure Nash equilibrium. For all resources $e \in E$, set a counter $n_e = 1$, and for all players $i \in \mathcal{N}$, assume their strategy set is empty. We recursively add a resource to a strategy set of a player. Find a resource $e \in E$ with $c_e \cdot n_e \leq c_{e'} \cdot n_{e'}$ for all $e' \in E$. Consider player $i \in \mathcal{N}$ with the highest priority on resource e. If resource e is feasible for player i, add e to the strategy set of player i, delete i from the priority list of e and increase the counter of resource e by 1. If resource e is not feasible for player i, delete i from the priority list of e.

Since the game is a matroid congestion game, the above algorithm yields a basis and thus a feasible strategy set for each player $i \in \mathcal{N}$. Let s denote the induced strategy sets. We show that s is a Nash equilibrium. Suppose not, then there is a player $i \in \mathcal{N}$ and strategy sets $s_i = \{e_1, \ldots, e_r\}$ and $s'_i = \{e'_1, \ldots, e'_r\}$, where resources are ordered in increasing costs given s_{-i}, so that $C_i(s_i, s_{-i}) > C_i(s'_i, s_{-i})$. This implies that there exists a smallest $k > 1$ such that $c_{e_k} \cdot n_{e_k}(s_i, s_{-i}) > c_{e'_k} \cdot n_{e'_k}(s'_i, s_{-i})$. Consider the two sets $\{e_1, \ldots, e_{k-1}\}$ and $\{e'_1, \ldots, e'_{k-1}, e'_k\}$. Both sets are independent sets and thus by property (3) of matroids, there is some e'_j for $1 \leq j \leq k$ so that the set $\{e_1, \ldots, e_{k-1}, e'_j\}$ is also an independent set. Since $c_{e'_j} \cdot n_{e'_j}(s'_i, s_{-i}) \leq c_{e'_k} \cdot n_{e'_k}(s'_i, s_{-i}) < c_{e_k} \cdot n_{e_k}(s_i, s_{-i})$, we obtain a contradiction as the algorithm must have picked resource e'_j instead of resource e_k. $\square$

6.3 Equilibrium Inefficiency

We consider the sum of weighted players' costs as a measure of the quality of a strategy profile, i.e.,

$$C(s) = \sum C_i(s).$$

Our analysis below is for linear cost functions and can be trivially extended to affine cost functions. As discussed in Section 2.2, a cost-minimization game G is said to be (λ, μ)-*smooth* if for all strategy profiles s, s' we have

$$\sum_{i \in N} C_i(s'_i, s_{-i}) \leq \lambda \cdot C(s') + \mu \cdot C(s).$$

It is well-known that if a game G is (λ, μ)-smooth with $\lambda > 0$ and $\mu < 1$, then PoA(G) is at most $\frac{\lambda}{1-\mu}$. Moreover, these bounds are valid for mixed, correlated and coarse-correlated equilibria [Rou15].

Theorem 6.6. *Every congestion game with resource-dependent priority lists is* $\left(2, \frac{1}{2}\right)$-*smooth. Hence, the price of anarchy is at most* 4.

Proof. Given a strategy profile s, define $w_e(s) = \sum\limits_{i' \in \mathcal{N} : e \in s_{i'}} w_{i'}$. Then, for all s, s',

$$\begin{aligned}
\sum_{i \in \mathcal{N}} C_i(s'_i, s_{-i}) &\leq \sum_{i \in \mathcal{N}} \sum_{e \in s'_i} w_i \cdot c_e \cdot (w_e(s) + w_i) \\
&= \sum_{e \in E} c_e \cdot \left(w_e(s') \cdot w_e(s) + \sum_{i \in N : e \in s'_i} w_i^2 \right) \\
&\leq \sum_{e \in E} c_e \cdot \left(w_e(s')^2 + \frac{1}{4} \cdot w_e(s)^2 + \sum_{i \in \mathcal{N} : e \in s'_i} w_i^2 \right) \\
&\leq 2 \cdot C(s') + \frac{1}{2} \cdot C(s),
\end{aligned}$$

where the second inequality follows from $\left(w_e(s') - \frac{1}{2} \cdot w_e(s)\right)^2 \geq 0$ and the third inequality from $C(s) = \sum_{e \in E} \frac{1}{2} \cdot c_e \cdot \left(w_e(s)^2 + \sum_{i \in N : e \in s_i} w_i^2\right)$ for all s. □

Correa and Queyranne [CQ12] give an example that shows that the bound of 4 is tight for restricted singleton congestion games with priority lists derived from Smith's rule. In fact, their example can be adjusted so that the equilibrium is unique and thus yields a lower bound for the price of stability.

6.3.1 Polynomial Costs

Until now we considered games with affine cost functions and proved a tight bound on the PoA. We now prove an asymptotically tight bound for polynomial costs function with maximum degree d. The cost of a player $i \in \mathcal{N}$ is given by,

$$C_i(s) = w_i \cdot \sum_{e \in s_i} (w_e^i(s))^d.$$

Theorem 6.7. *Every congestion game with resource-dependent priority and cost functions that are polynomials of maximum degree d has PoA $\in \Theta(d)^{d+1}$.*

Proof. We first prove an upper bound on the price of anarchy in weighted congestion games with priority lists with non-decreasing cost functions that are polynomials of maximum degree d. We then prove a lower bound using unweighted players.

We begin the proof with the following observation.

Observation 6.8. *For all $i \in [k]$ such that $m_i \geq 1$ and $\sum_{j=1}^{k} m_j = d+1$,*

$$\binom{d+1}{m_1,\ldots,m_i,\ldots,m_k}\prod_{j=1}^{k} w_j^{m_j} \leq (d+1)\cdot w_i \cdot \binom{d}{m_1,\ldots,m_i-1,\ldots,m_k}\prod_{j=1,j\neq i}^{k} w_j^{m_j}\cdot w_i^{m_i-1}.$$

Claim 6.9. *For all $w_1, w_2, \ldots, w_k \in \mathbb{R}_{\geq 0}$, and $k \geq 1$,*

$$(w_1+w_2+\ldots+w_k)^{d+1} \leq (d+1)\cdot\Big(w_1\cdot w_1^d + w_2\cdot(w_1+w_2)^d + \ldots + w_k(w_1+w_2+\ldots+w_k)^d\Big)$$

Proof. The proof follows using the multinomial theorem and Observation 6.8, i.e.,

$$\begin{aligned}
&(w_1+w_2+\ldots+w_k)^{d+1} = \sum_{m_1+\ldots+m_k=d+1}\binom{d+1}{m_1,\ldots,m_k}\prod_{j=1}^{k} w_j^{m_j}\\
&= \binom{d+1}{d+1}\cdot w_1^{d+1} + \ldots + \sum_{m_1+\ldots+m_k=d+1, m_k\geq 1}\binom{d+1}{m_1,\ldots,m_k}\prod_{j=1}^{k} w_j^{m_j}\\
&\leq (d+1)\cdot w_1\cdot\binom{d}{d}\cdot w_1^d + \ldots + (d+1)\cdot w_k\cdot\sum_{m_1+\ldots+m_k=d}\binom{d}{m_1,\ldots,m_k}\prod_{j=1}^{k} w_j^{m_j}\\
&= (d+1)\cdot\Big(w_1\cdot w_1^d + \ldots + w_k(w_1+\ldots+w_k)^d\Big).
\end{aligned}$$

□

Theorem 6.10 (Hölder's inequality)**.** *Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ be positive real numbers and $a, b > 1$ be such that $1/a + 1/b = 1$. Then,*

$$\sum_{i=1}^{n} x_i\cdot y_i \leq \left(\sum_{i=1}^{n} x_i^a\right)^{1/a} + \left(\sum_{i=1}^{n} y_i^b\right)^{1/b}.$$

Lemma 6.11. *Every congestion game with resource-dependent priority and cost functions that are polynomials of maximum degree d has a PoA of at most φ^{d+1}, where φ is the unique solution to $x^{d+1} = (d+1)(x+1)^d$.*

Proof. Let s be a pure Nash equilibrium and s^* a social optimum. Then, from the Nash inequality we have,

$$\begin{aligned}
C(s) &\leq \sum_{i\in\mathcal{N}} C_i(s_i^*, s_{-i})\\
&\leq \sum_{i\in\mathcal{N}} w_i\cdot\sum_{e\in s_i^*} w_e(s_i^*, s_{-i})^d = \sum_{i\in\mathcal{N}} w_i\cdot\sum_{e\in s_i^*}(w_e(s_{-i})+w_i)^d\\
&\leq \sum_{i\in\mathcal{N}} w_i\cdot\sum_{e\in s_i^*}(w_e(s)+w_e(s^*))^d = \sum_{e\in E} w_e(s^*)\cdot(w_e(s)+w_e(s^*))^d
\end{aligned}$$

for some $0 < \gamma < 1$,

$$\begin{aligned}
&\leq \sum_{e\in E}(1-\gamma)^{1-d}\cdot w_e(s^*)^{d+1} + \gamma^{1-d}\cdot w_e(s^*)\cdot w_e(s)^{d+1} \\
&\leq (1-\gamma)^{1-d}\cdot \sum_{e\in E} w_e(s^*)^{d+1} + \gamma^{1-d}\cdot \left(\sum_{e\in E} w_e(s^*)^{d+1}\right)^{1/(d+1)} \cdot \left(\sum_{e\in E} w_e(s)^{d+1}\right)^{d/(d+1)} \\
&\leq (1-\gamma)^{1-d}\cdot (d+1)\cdot C(s^*) + \gamma^{1-d}\cdot (d+1)\cdot C(s^*)^{1/(d+1)}\cdot C(s)^{d/(d+1)},
\end{aligned}$$

where the fourth inequality follows by convexity (c.f. Farzad et al. [FOV08]), the fifth inequality from Hölder's inequality and the last inequality follows from Claim 6.9.

In particular, define $z = \left(\frac{C(s)}{C(s^*)}\right)^{1/(d+1)}$, we have

$$z^{d+1} \leq (1-\gamma)^{1-d}\cdot (d+1) + \gamma^{1-d}\cdot (d+1)\cdot z^d.$$

Setting $\gamma = \frac{\varphi}{\varphi+1}$, where φ solves $z^{d+1} = (d+1)(z+1)^d$, and $z = \varphi$, we have

$$z^{d+1} = (1-\gamma)^{1-d}\cdot (d+1) + \gamma^{1-d}\cdot (d+1)\cdot z^d,$$

and thus

$$\frac{C(s)}{C(s^*)} \leq \varphi^{d+1}.$$

□

Lemma 6.12. *The PoA of every congestion game with resource-dependent priority and cost functions that are polynomials of maximum degree d is at least $(d+1)^{d+1}$.*

Proof. To prove a lower bound we construct the following game.

Let $k \in \mathbb{Z}$. The game contains $n = (d+2)k$ players and $m = (d+2)k$ resources. Each player $i \in \mathcal{N}$ has $w_i = 1$. Each player $i \in \mathcal{N}$ has two pure strategies: $\{e_i, \dots, e_{i+k-1}\}$ and $\{e_{i+k}, \dots, e_{i+(d+2)k-1}\}$ (all indices are modulo $(d+2)k$). The cost functions are $c_e(x) = x^d$ for all $e \in E$, and for all $e_j \in E$ and $i \in \mathcal{N}$ the priority lists are defined as

$$\pi_{e_j}(i) := (j + (d+1)k + 1 - i) \mod n,$$

where $\pi_{e_j}(i)$ is priority of player i on resource e_j. With a slight abuse of notation, if $\pi_{e_j}(i) = 0$ in the above expression, then set $\pi_{e_j}(i) = n$.

Observe that, the strategy profile s with $s_i = \{e_{i+k}, \dots, e_{i+(d+2)k-1}\}$ for all $i \in \mathcal{N}$ is a pure Nash equilibrium, since the cost of each player $i \in \mathcal{N}$ is given by

$$\sum_{j=1}^{(d+1)k} j^d \leq k\cdot ((d+1)k+1)^d,$$

where the r.h.s. is the cost incurred by the player i if they unilaterally deviate.

Alternatively, the strategy profile s^* such that, $s_i^* = \{e_i, \dots, e_{i+k-1}\}$ for all $i \in \mathcal{N}$ yields a cost per player of

$$\sum_{j=1}^{k} j^d.$$

Hence,

$$\frac{C(s)}{C(s^*)} \geq \frac{\sum_{j=1}^{(d+1)k} j^d}{\sum_{j=1}^{k} j^d} \to (d+1)^{d+1} \text{ as } k \to \infty.$$

□

Lemma 6.11 and 6.12 together conclude the proof of the theorem. □

Part II

Decentralized Overlay Networks

Chapter 7

Overlay Networks under Adversarial Churn

In this chapter we study yet another interesting model, where its efficacy is influenced by the uncertainty in the user behavior. This model is motivated by a communication networking concept known as Peer-to-Peer (P2P) overlay network, which has become increasingly popular over the past few decades. P2P overlay networks are highly dynamic communication networks in which a relatively large number of users join or leave the network over a period of time. A predominant problem in such a network with no admission control is to ensure that the users in the network remain connected to each others at any point in time. Moreover, the problem is quite challenging in the presence of peer dynamics, i.e., users joining and leaving the network continuously in an uncoordinated manner. In this chapter, we investigate the maintenance of such a dynamic network to guarantee that the active users remain connected to one another. To that extend, we present a robust overlay network called Linearized DeBruijn Swarm—a highly churn resistant overlay topology.

7.1 Introduction

Peer-to-peer (P2P) networking has proven to be an useful technique to construct fault resilient decentralized systems. Many applications such Bittorent, Gnutella, KaZaA, and numerous other file sharing frameworks rely on such decentralized networking principles [ECP$^+$05]. In a P2P architecture the users in the network are often referred to as peers or nodes and are connected to each others over virtual overlay links built over an underlay network, e.g., the Internet [Sch01]. Together, they form a logical network topology also known as an overlay network [ECP$^+$05, see, Figure 1]. Nodes provide computational and storage infrastructure. Within the overlay, each node has a logical address and logical links that allows it to search and store information in the network. P2P overlay networks are highly dynamic networks with nodes joining and leaving the network continuously. This uncertainty in the network due to unpredictable churn (join/leave) of nodes makes information storage and retrieval a challenging problem.

A key requirement for all applications that rely on P2P networks is reliable communication between all nodes, i.e., each node should be able to send a message to another node at all times. This is complicated by the fact that in every large-scale system, errors and attacks are considered as a rule rather than an exception. At the same time there is usually no or very little admission control for new participants. This implies a massive amount of churn, i.e., nodes joining and leaving the network at any given time. In fact, empirical studies have shown that 50% of all nodes are subjected to churn over the course of an hour [SR06]. This alludes for robust and distributed protocols that maintain connected overlays in spite of heavy churn.

In this chapter we deal with the problem of maintaining a routable overlay under adversarial churn. A overlay is routable, if each node in the network is able to send a message to any given logical address in the network at any point in time. Note that it is convenient to model unpredictability in the node dynamics using an adversary [AS21, APR+15, DGS16]. The adversary picks a set of nodes that leave the network and proposes a set nodes that join the network, continuously. The topology of the overlay plays a crucial role in determining the severity of the problem. P2P overlay networks have been mainly classified into three types: *Structured*, *Unstructured* and *Hybrid* [ECP+05, Dar05]. However, the strict structural requirements in the topology of structured P2P overlays make them the most susceptible to faults due to unpredictable node dynamics [ECP+05]. Choosing the node set cleverly allows the adversary to destroy the structural properties and disconnect the network. Therefore, we focus our attention to the interesting case of structured P2P overlay networks.

Note that an adversary that knows all virtual links between the nodes can simply partition the network by churning out the neighborhood of a node [GRS19]. Previous literature therefore considered models with an additional restriction, where the adversary has slightly outdated information about the nodes' logical links [AS21, APR+15, DGS16]. In particular, the adversary could access all information that is at least $O(\log\log n)$ rounds old, where $n \in \mathbb{Z}_+$ is the minimal number of nodes in the network at any point in time. This includes the nodes' logical links, internal states, random decisions, the contents of all messages, etc. Within these $O(\log\log n)$ rounds the nodes in the network execute a distributed algorithm that completely rearranges the network topology. However, we remark that the techniques presented in [AS21, APR+15, DGS16] cannot be used if one wants to grant the adversary access to even more recent information.

To overcome above mentioned restriction, we propose a trade-off in the form of a (a, b)-late omniscient adversary that has an almost up-to-date information about the network topology, but is more outdated with regard to all other aspects. In particular, it has full knowledge of the topology such as logical links of the nodes, after a rounds and complete knowledge of messages, internal states, etc., after b rounds. We believe that our trade-off is quite a natural assumption and that in a real network, an adversary with similar properties could e.g., be an agency eavesdropping on an Internet exchange point. Although they monitor who communicate based on the involved Internet Protocol (IP) addresses, but are unable to decrypt the content of the messages immediately.

7.1.1 The Model

We briefly introduced the overlay model in Chapter 1. We now present a detailed description of the model we consider. Recall that we assume time proceeds in synchronous rounds and observe a dynamic set of nodes $\mathcal{V} := \big(V_0, V_1, \dots\big)$ such that V_t is the set of nodes in round t. We remark that synchronicity is a standard assumption in the literature to allow for the nodes to react to the adversary's actions periodically. Each node u is identified by a unique and immutable identifier denoted by $\mathrm{id}(u)$ of size $O(\log n)$ bits. In a real-world network these identifiers could e.g., be the nodes' IP addresses. At the beginning of each round t, a node $u \in V_t$ may store the identifiers of other nodes in the node set V_t, as its neighbors in round t. The choice of neighbors form the logical links in the overlay.

For all $u, v \in V_t$, there exists a directed edge (u, v) in the network, if the node u knows $\mathrm{id}(v)$. A node $u \in V_t$ can send a message to a node $v \in V_t$, if and only if it knows the identifier of v, i.e., $\mathrm{id}(v)$. Let us denote by D_t^u the neighborhood of node u in round t, i.e.,

$$D_t^u := \{v \in V_t \mid u \text{ knows } \mathrm{id}(v) \text{ in round } t\},$$

then $(D_t^u)_{u \in V_t}$ completely defines the logical links and therefore, the structure of the overlay network in round t.

Exchange of messages in each round results in series of directed communication graphs $\mathcal{G} := \big(G_0, G_1, \dots\big)$ with $G_t = (V_t, E_t)$ and

$$E_t := \{(u, v) \mid v \text{ receives a message from } u \text{ in round } t\}.$$

Creating an edge in the communication graph may be compared to sending a UDP (User Datagram Protocol [1]) message to the desired receiver or establishing a TCP (Transmission Control Protocol [2]) connection between two nodes. We assume that a node can send messages to $O(\log n)$ different nodes in each round with message length at most $O(\operatorname{polylog} n)$ bits.

Our model assumes that the node set $\mathcal{V}$ is determined by an adversary. This implies, in every round t the adversary can propose a set $O_t \subset V_{t-1}$ that leaves the network and a set $J_t \subset V_t$ that joins the network, i.e., $V_t := \{V_{t-1} \setminus O_t\} \cup J_t$. In particular, the adversary has to comply to the following assumptions:

1. **Lateness**.

   We consider the adversary to be $(2, O(\log n))$-late omniscient, i.e., the adversary has slightly outdated knowledge of the topology, i.e., the series of graphs $\mathcal{G} := (G_0, G_1, \dots)$ created through the communication between nodes. In particular, in round t the adversary only has full knowledge of all graphs until G_{t-2}. Furthermore, it has no knowledge of the nodes' internal states, the position of the node, virtual links, contents of messages, etc., for $O(\log n)$ rounds, i.e., the adversary learns the content of message sent in round t only in round $t + O(\log n)$.

2. **Churn Rate**.

   For all $V_t \in \mathcal{V}$ it holds that $|V_t| \in [n, \kappa n]$, where $\kappa \geq 1$ is a small constant. This implies that the number of nodes in any round stays within $\Theta(n)$. For a suitable value $T \in O(\log n)$, we assume that $|V_{t+T} \cap V_t| \geq (1 - \alpha)n$, where $\alpha \in [0, 1)$ is a fixed constant. This allows the churn to be $O(n)$ in each round as long as there is a stable set of size $\Theta(n)$ that remains in the network for at least T rounds.

3. **Bootstrap Phase**.

   We assume that until a round $B \in O(\log^2 n)$ the adversary is inactive and no churn occurs, this phase is often referred to as the bootstrap phase. We would like to remark that this is a standard assumption to prepare the network to allow for topological reconfiguration (see, e.g. [AS21, APR+15, DGS16]). Only after the conclusion of the bootstrap phase, is the adversary allowed to begin churning nodes in or out of the network.

4. **Restricted Join**.

[1] https://tools.ietf.org/html/rfc768.
[2] https://tools.ietf.org/html/rfc793.

We assume that a new node $v \in V_t \setminus V_{t-1}$ can only join the network via a node $w \in V_t \cap V_{t-2}$, i.e., the node v joins only via nodes that were in the network for at least 2 rounds. Finally, we assume the number of nodes that join the network via the same node $v \in V_t$ in a round t is a fixed constant.

5. **Uniform Hash Function**.

   We assume that each node is seeded with a fixed uniform hash function $h : \mathbb{R}^2 \mapsto [0, 1)$ prior to being administered into the network. The adversary always remains oblivious to this hash function. The nodes use the function h to uniformly at random pick position in the $[0, 1)$-interval. Seed knowledge is a standard assumption in the literature [AS21].

We remark that our model incorporates observations from Stutzbach and Reza [SR06] that new nodes join and leave very frequently, but there is a (relatively) stable set of nodes that remain committed to the network for a longer period of time. To the best of our knowledge, this is a significantly more flexible model compared to other related work. Given the above mentioned constraints on the adversary, a round t consists of the following four steps:

1. **Active Node Set**.

   At the beginning of each round t, the adversary can select a set of nodes $O_t \subset V_{t-1}$ that leave the network in round t. These nodes are not allowed to send or receive any messages and leave the network immediately. Furthermore, the adversary may propose a set of nodes J_t that joins the network in round t. For each node $v \in J_t$ the adversary selects a bootstrap node $w \in V_t \setminus J_t$ (satisfying the necessary conditions for Restricted Join) that receives a reference to v. Recall that each node can only bootstrap a constant number of new nodes into the network.

2. **Receiving Messages**.

   All active nodes receive all messages sent to them in the previous round. Note that this even holds for messages that were sent by nodes that were churned out in the current round.

3. **Local Processing**.

   After receiving all messages, a node can perform arbitrary calculations on its local variables and the received messages. This is a standard assumption, since network operations are considered to be significantly slower than local computation.

4. **Sending Messages**.

   Finally, each node may send messages to other nodes. Recall that sending a message to another node implicitly creates an edge in the graph G_{t+1}. Every message sent in round t is received in the round $t + 1$. Furthermore, due to the lateness condition these edges can only be seen by the adversary at the beginning of the round $t + 3$.

7.1.2 Our Contribution

We present a distributed overlay maintenance algorithm that can handle a $(2, O(\log n))$-late adversary by completely rearranging the structure of the network every 2 rounds. The algorithm executes over a dynamic set of nodes $\mathcal{V} := \big(V_0, V_1, \dots\big)$ chosen by a $(2, O(\log n))$-late adversary, to create a sequence of mutually independent overlays that allow for a series of communication graphs $\mathcal{G} := \big(G_0, G_1, \dots\big)$ with $G_i := (V_i, E_i)$. Furthermore, the algorithm allows for routing a message to a logical address $p \in [0,1)$ within $O(\log n)$ rounds. The overlay we consider in this work is an extension of the Linearized DeBruijn Graph presented in Richa et al. [RS11a], which by itself is based on the De Bruijn Graph [de 46] and draws ideas from Naor and Wieder [NW07] that uses quorums of logarithmic size to send and receive messages. The latter is adapted from Fiat et al. [FSY05], where the authors use this approach for the chord overlay [SMK$^+$01]. Our approach uses several structural properties of the overlay as well as a careful analysis of non-independent events to ensure fast reconfiguration of the network.

In Section 7.2 we introduce a graph topology based on the Linearized DeBruijn Graph called Linearized DeBruijn Swarm (LDS). In Section 7.3 we present Alg-Routing—a routing algorithm that can route a message to any point in $[0,1)$-interval in $O(\log n)$ rounds with high probability. In Section 7.5 we presents 3 algorithms each of which are executed concurrently by every node in the network. Algorithm Alg-LDS—rearranges the graph topology such that it is completely rebuilt every 2 rounds, while ensuring efficient routing, Algorithm Alg-Random—handles addition of new nodes into the overlay, and Algorithm Alg-Sampling—enables a node in the overlay to pick a random node in the overlay. The overall message complexity due to these algorithms is then at most $O(\log^3 n)$ messages per node and round with high probability.

7.1.3 Related Work

Table 7.1: Overview of different models in the literature.

Paper	**Lateness***	**Churn Rate**†	**Immediate**‡
[AS21]	$(O(\log\log n), O(\log\log n))$	$(\alpha n, O(\log\log n))$	Yes
[DGS16]	$(O(\log\log n), O(\log\log n))$	$(n - \frac{n}{\log n}, O(\log\log n))$	No*
[APR$^+$15]	$(O(\log n), O(\log n))$	$\left(O\left(\frac{n}{\log n}\right), O(\log n)\right)$	Yes
Our Results	$(2, O(\log n))$	$(\alpha n, O(\log n))$	Yes

* An adversary is (a,b)-late if it has full knowledge of the topology after a rounds and complete knowledge after b rounds.
† The churn rate is (C,T) if the adversary can perform C join/leaves in T rounds.
‡ Churned out nodes leave the network immediately.
* Nodes remain in the network for additional $O(\log\log n)$ rounds.

The last two decades has seen extensive amount of work in analyzing overlay networks under high adversarial churn. As already mentioned in the introduction, these works had a variety of different model assumptions. We refer the readers to [ECP$^+$05, AS21] for a comprehensive survey on previous results. In the following, we only concentrate on models closely related to ours. The initial series of papers [Sch05, FSY05, AS07] assumed only a subset of nodes are subjected to adversarial churn. However, these nodes could also behave malicious and try to sabotage the overlay's structure and the routing mechanism by sending corrupted messages. A general assumption was that up to a constant fraction of nodes would be malicious. Scheideler [Sch05] present a protocol that spreads these nodes over the network such that, each connected subset of logarithmic size contains a constant fraction of non-malicious nodes. Fiat et al. [FSY05] build upon this work and present a full overlay maintenance algorithm that provided a robust DHT. In their approach, each virtual address $p \in [0, 1)$ is maintained by a committee of $O(\log n)$ nodes. More recent work [AMM+13, AS21, DGS16] considered all nodes to be susceptible to adversarial churn. However, they usually do not consider malicious behavior. The adversaries in these papers can be almost succinctly described using three properties that we mentioned earlier, i.e., the lateness, the churn rate, and if it is immediate. Recall that we say an adversary is (a, b)-late if it has full knowledge of the topology after a rounds and complete knowledge of all sent messages, internal states, etc. after b rounds, the churn rate is (C, T) if the adversary can perform C join/leaves in T rounds, and an adversary is immediate if churned out nodes have to leave the network immediately and without the possibility to send and receive more messages. Table 7.1 shows an overview over the different models. Note that the table is only for comparison as it simplifies some of the models and does not depict all of their respective nuances. However, we remark that these simplifications do not weaken the adversary. Augustine et al. [APR+15] present an algorithm that builds and maintains an overlay in the presence of a nearly completely oblivious adversary. Here, the overlay no longer has a fixed structure, rather is an unstructured expander graph of constant degree. Note that this overlay has no virtual addressing. However, in [AMM+13] the authors present a scheme that allows to quickly search for data in these networks, as long as only one look-up request arrives each round. Drees et al. [DGS16] build a semi-structured expander called H_d-Graph, which is the union of d random rings. Their adversary is not only $O(\log \log n)$-late with regard to communication, it also has access to all nodes' memory and all sent messages after $O(\log \log n)$. Nodes that are churned out in round t may remain in the network for an additional $O(\log \log n)$ rounds, i.e., it does not satisfy the property of immediateness. The SPARTAN framework presented in [AS21] probably has the greatest resemblance with our work. In SPARTAN the nodes maintain a logical overlay resembling a butterfly network. To ensure robustness each of the butterfly's virtual nodes is simulated by $O(\log n)$ nodes. The key difference between our work and SPARTAN is the adversary's lateness. Similar to [DGS16], the SPARTAN framework assumes the adversary to be $(O(\log \log n), O(\log \log n))$-late, but in return allows the churn to be as high as αn in $O(\log \log n)$ rounds. However, unlike [DGS16], SPARTAN allows the adversary to be immediate.

7.2 Definitions and Preliminaries

In this section we present definitions and results from probability theory that are used extensively in the analysis of our algorithms.

Definition 7.1 (Negative Correlation [Sch00, p. 31])**.** A set of random variables $(X_i)_{i\in[n]}$ are *negatively correlated (NC)*, if for every $S \subseteq [n]$ it holds that,

$$\mathbf{E}\left[\prod_{i\in S} X_i\right] \leq \prod_{i\in S} \mathbf{E}[X_i].$$

Definition 7.2 (Negative Association [JDP83, Waj17])**.** A set of random variables $(X_i)_{i\in[n]}$ are *negatively associated (NA)*, if for any two functions f, g, both monotonically increasing (or both monotonically decreasing) defined on disjoint subsets of X, it holds that,

$$\mathbf{E}[f(X) \cdot g(X)] \leq \mathbf{E}[f(X)] \cdot \mathbf{E}[g(X)].$$

Note that all independent and (multivariate) hypergeometric random variables are negatively associated [JDP83, DR98].

Corollary 7.3 (NA implies NC [Waj17]). *Let $X_1, X_2, \ldots, X_n$ be a set of NA random variables. Then, it holds that for every $S \subseteq [n]$,*

$$\mathbf{E}\left[\prod_{i \in S} X_i\right] \leq \prod_{i \in S} \mathbf{E}[X_i].$$

The following propositions from Joag-Dev and Proschan [JDP83] will be extensively used in many of our proofs.

Proposition 7.4 ([JDP83, DR98]). *If $X \coloneqq (X_1, \ldots, X_n)$ and $Y \coloneqq (Y_1, \ldots, Y_m)$ are negatively associated sets of random variables that are mutually independent, then the vector $(X, Y) \coloneqq (X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ are also negatively associated.*

Proposition 7.5 ([JDP83, DR98]). *Let $(X_1, \ldots, X_n)$ be negatively associated random variables. For some $k \leq n$, let $I_1, \ldots, I_k \subseteq [n]$ be disjoint index sets. For $j \in [k]$, let $f_j : \mathbb{R}^{|I_j|} \mapsto \mathbb{R}$ be functions that are all non-decreasing or all non-increasing. Define $Y_j \coloneqq f_j(X_i : i \in I_j)$. Then the random variables $(Y_1, \ldots, Y_k)$ are negatively associated.*

Lemma 7.6 (Zero-One Lemma [DR98]). *If $Y_1, \ldots, Y_n$ are zero-one random variables such that $\sum_i Y_i = 1$, then $Y_1, \ldots, Y_n$ are negatively associated.*

Furthermore, we make use of the following Chernoff bounds.

Lemma 7.7 (Chernoff-Hoeffding Bounds [DR98, MU05]). *Let $X \coloneqq \sum_{i \in [n]} X_i$ be the sum of n negatively correlated random variables with $X_i \in \{0, 1\}$ for each $i \in [n]$. Then, it holds that for any $0 < \gamma < 1$,*

$$\mathbf{Pr}[X \geq (1 + \gamma)\mathbf{E}[X]] \leq \exp\left(-\frac{\gamma^2 \cdot \mathbf{E}[X]}{2}\right)$$

and

$$\mathbf{Pr}[X \leq (1 - \gamma)\mathbf{E}[X]] \leq \exp\left(-\frac{\gamma^2 \cdot \mathbf{E}[X]}{3}\right).$$

Also, for any $\gamma \geq 1$,

$$\mathbf{Pr}[X \geq (1 + \gamma)\mathbf{E}[X]] \leq \exp\left(-\frac{\gamma \cdot \mathbf{E}[X]}{3}\right).$$

Throughout this chapter we assume that each node in the network is aware of n and κ, i.e., the lower and upper bound on the number of nodes currently in the network. We make this simplification due to Stutzbach and Reza [SR06], that the number of nodes stays relatively stable. Furthermore, in order to simplify notations, we define $\lambda \coloneqq 2\log(\kappa n)$. For convenience we assume that λ is an integer. We would like to remark that all our algorithms may be adapted to work with close estimates of λ and $\frac{\lambda}{n}$ using approaches presented in [RS11a, FSY05, KS04, KLSY07].

DeBruijn Swarm

We now present the *Linearized DeBruijn Swarm* (LDS), which is a combination of a well-analyzed network overlay of low degree, i.e., the Linearized DeBruijn Graph (LDG) presented in [RS11a, FS17] and techniques from robust overlays, i.e., the usage of logarithmic sized quorums that simulate a single node [FSY05]. Note that the LDG is inspired

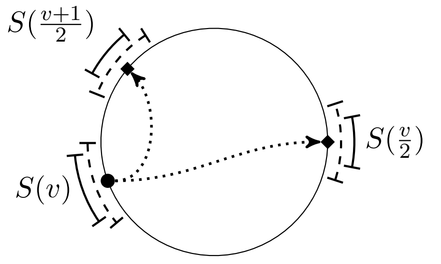

Figure 7.1: A node v is connected each node in the dashed areas.

by, but not equivalent to the classical De Bruijn Graph. The notion of swarm was also described in Fiat et al. [FSY05].

In the remainder of this section we present the LDS's topology and show some of its basic properties. Each node $v \in V$ chooses a position $p_v \in [0,1)$ *uniformly* and *independently* at random. Note that for the sake of convenience, the position of a node $v \in V$, we just write v instead of p_v. It should always be clear from the context if we refer to the node, its identifier, or its (current) position. However, we will make the context sufficiently clear to avoid ambiguity, if necessary.

Nodes can calculate their distance to one another node using the distance function $d : V^2 \to [0,1)$. Given two nodes $v, w \in V$ the distance function d returns the shortest distance between v and w in the $[0,1)$-torus.

Formally, the function $d : V \times V \mapsto [0,1)$ is defined as follows:

$$d(v,w) := \begin{cases} |v-w| & \text{if } |v-w| \leq \frac{1}{2} \\ 1-|v-w| & \text{otherwise.} \end{cases} \tag{7.1}$$

Furthermore, the distance function also satisfies the triangular inequality, i.e.,

$$d(v,w) \leq d(v,z) + d(z,w) \qquad \text{for all } v,w,z \in V. \tag{7.2}$$

For convenience we introduce the following notions for the relation between two nodes $u, v \in V$. If $|u-v| \leq \frac{1}{2}$, then u is *left* (clockwise) of v if $u < v$ and *right* (clockwise) otherwise. For $|u-v| > \frac{1}{2}$ the relation is reversed. Furthermore, the set $\langle u, v\rangle \subset V$ contains all nodes which are right of u and left of v. Given a node w, we say that a node u is closer to w than v if and only if $d(u,w) < d(v,w)$. Lastly, we call a node u the closest neighbor of v if there exists no other node $u' \in V$ closer to v than u, i.e., $u := \underset{k\in V}{\operatorname{argmin}}\, d(k,v)$.

In the LDG presented by Richa et al. [RS11a], each node v connects to exactly six other nodes. Namely, the two closest nodes left and right of p_v and the two closest node left and right of the points $\frac{p_v}{2}$ and $\frac{p_v+1}{2}$, respectively. We extend this structure such that, each node connects to $O(\log n)$ closest neighbors. For a given point $p \in [0,1)$ we call $S(p) \subset V$ the *swarm* of p. It holds that a node $v \in S(p)$ if and only if $d(v,p) \leq \frac{c\lambda}{n}$. Here, $c > 1$ is a robustness parameter which should be chosen as small as possible. These swarms (and not the nodes) form the building blocks of LDS. Denote by $|S(p)|$ the number of nodes in the swarm $S(p)$. We call the swarms $S(p)$ adjacent to $S(p')$ if there is an edge (v,w) between every node $v \in S(p)$ and $w \in S(p')$. Note that each swarm $S(p)$ spans an interval of length $\frac{2c\lambda}{n}$, as it consists of two intervals of length $\frac{c\lambda}{n}$ to the left and right of p, respectively. Sometimes it will be necessary to distinguish between the nodes in the left and right interval of p, so we define

$$S^L(p) := \{v \in S(p) \mid v \text{ is left of } p\}$$

and
$$S^R(p) := \{v \in S(p) \mid v \text{ is right of } p\}$$
as the left and right side of $S(p)$. Given this notion of a swarm, we can now formally define the LDS as follows:

Definition 7.8 (Linearized DeBruijn Swarm)**.** Let V be a set of nodes with $|V| \geq n$ positioned on the $[0,1)$-interval and $\lambda := 2\log(\kappa n)$. Then, the LDS $D := (V, E_L \cup E_{DB})$ with parameter $c \in \mathbb{N}_+$ has the following properties:

- $(v,w) \in E_L \iff w \in V$ and $d(v,w) \leq \frac{2c\lambda}{n}$.
- $(v,w) \in E_{DB} \iff w \in V$ and $d\left(\frac{v+i}{2}, w\right) \leq \frac{3c\lambda}{2n}$ with $i \in \{0,1\}$.

A Linearized DeBruijn Swarm is illustrated in Figure 7.1. Over the course of this chapter we will refer to the edges in E_L as *list* edges, whereas the edges in E_{DB} as *DeBruijn* edges.

From Definition 7.8 we state the following lemma.

Lemma 7.9 (Swarm Property)**.** *Consider any point $p \in [0,1)$ and its swarm $S(p) \subseteq V$ in the LDS. Then, $S(p)$ is adjacent to $S\left(\frac{p}{2}\right)$ and $S\left(\frac{p+1}{2}\right)$.*

Proof. Let $p \in [0,1)$ be any point and $v \in S(p)$ be a node in p's swarm. From Definition 7.8 this implies that $d(p,v) \leq \frac{c\lambda}{n}$. We now show that node v has a connection to every node in $S\left(\frac{p}{2}\right)$ and $S\left(\frac{p+1}{2}\right)$. For the proof we only analyze adjacency to $S\left(\frac{p}{2}\right)$, since the other case is analogous. We distinguish between the following two scenarios.

1. If $|p-v| \leq \frac{1}{2}$, then from (7.1) it holds that,
$$d\left(\frac{p}{2}, \frac{v}{2}\right) := \left|\frac{p}{2} - \frac{v}{2}\right| = \frac{1}{2}|p-v| \leq \frac{1}{2}\frac{c\lambda}{n}. \tag{7.3}$$
Let u be any node in $S(\frac{p}{2})$, then $d\left(u, \frac{p}{2}\right) \leq \frac{c\lambda}{n}$. Then using (7.2) and (7.3) we get,
$$\begin{aligned} d\left(u, \frac{v}{2}\right) &\leq d\left(u, \frac{p}{2}\right) + d\left(\frac{p}{2}, \frac{v}{2}\right) \\ &\leq \frac{c\lambda}{n} + \frac{c\lambda}{2n} \\ &= \frac{3c\lambda}{2n}. \end{aligned} \tag{7.4}$$
From Definition 7.8, since node v has a DeBruijn edge to each node $w \in V$ with $d\left(\frac{v}{2}, w\right) \leq \frac{3c\lambda}{2n}$, then using (7.4) the lemma follows.

2. Otherwise, $|p-v| > \frac{1}{2}$.

   Observe that this only occurs if either $p \in \left[0, \frac{c\lambda}{n}\right]$ or $p \in \left[1 - \frac{c\lambda}{n}, 1\right)$, i.e., the point p is close to 0 or 1 and v lies on the opposite site of the interval. We distinguish between the following two cases.

   (a) If $p \in \left[0, \frac{c\lambda}{n}\right]$, then it also holds that $\frac{p}{2} \in [0,p]$. Implies,
   $$d\left(v, \frac{p}{2}\right) \leq d(v,p) \leq \frac{c\lambda}{n}. \tag{7.5}$$

Then, by the triangle inequality and inequality (7.5), it holds that for every node $u \in S\left(\frac{p}{2}\right)$,

$$d(u, v) \leq d\left(u, \frac{p}{2}\right) + d\left(\frac{p}{2}, v\right) \leq \frac{2c\lambda}{n}.$$

From Definition 7.8, the node u is then a list neighbor of v and the lemma follows.

(b) Otherwise, if $p \in \left[1 - \frac{c\lambda}{n}, 1\right)$ then it holds that $\frac{p}{2} \in \left[\frac{1}{2} - \frac{c\lambda}{2n}, \frac{1}{2}\right)$. Now consider the distance between $\frac{p}{2}$ and $\frac{v+1}{2}$. Here, it holds that,

$$d\left(\frac{v+1}{2}, \frac{p}{2}\right) := \left|\frac{v+1}{2} - \frac{p}{2}\right| = \frac{1}{2}|v + 1 - p|.$$

Observe that since $v < p$ and $p < v + 1$, it holds that $|(1 + v) - p|$ is equivalent to $1 - |v - p|$. Therefore, the inequality simplifies to

$$d\left(\frac{v+1}{2}, \frac{p}{2}\right) = \frac{1}{2}(1 - |v - p|) = \frac{1}{2}d(v, p).$$

Since $\frac{1}{2}d(v, p) \leq \frac{c\lambda}{2n}$, applying the triangle inequality we get that for every node $u \in S(\frac{p}{2})$,

$$\begin{aligned} d\left(u, \frac{v+1}{2}\right) &\leq d\left(u, \frac{p}{2}\right) + d\left(\frac{v+1}{2}, \frac{p}{2}\right) \\ &\leq \frac{c\lambda}{n} + \frac{c\lambda}{2n} \\ &= \frac{3c\lambda}{2n}. \end{aligned} \tag{7.6}$$

From Definition 7.8, since node v has a DeBruijn edge to each node $w \in V$ with $d\left(\frac{v+1}{2}, w\right) \leq \frac{3c\lambda}{2n}$, then using (7.6) the lemma follows.

□

The next lemma shows that if nodes are assigned to the $[0, 1)$-interval uniformly and independently at random, then all swarms have roughly the same size w.h.p. Throughout this chapter w.h.p. means with probability $\left(1 - \frac{1}{n^k}\right)$, where n is the number of nodes and k is a tunable constant.

Lemma 7.10 (Swarm Size)**.** *Let the swarm length to be $\frac{c\lambda}{n}$ with $c \geq 12k$ and assume all nodes pick their positions uniformly and independently at random. Then, for any point $p \in [0, 1)$ it holds that,*

$$\mathbf{Pr}\left[\frac{1}{2}c\lambda < |S(p)| < 2c\lambda\right] \geq 1 - \frac{2}{n^k},$$

where $|S(p)|$ denotes the number nodes in $S(p)$.

Proof. The proof is analogous to [RS11a, Lemma 3]. Given that each node picks its position uniformly and independently at random in the $[0, 1)$-interval, the probability that a node chooses a point in an interval of length $\frac{c\lambda}{n}$ is exactly $\frac{c\lambda}{n}$. Note that since c is a constant and $\lambda \in O(\log n)$, for big enough n, $\frac{c\lambda}{n} \leq 1$.

Let X be a random variable that counts the number of nodes in $S(p)$. For each $v \in V$, let X_v be a $\{0, 1\}$ random variable such that, $X_v = 1$ if v picked a position in a swarm $S(p)$ and 0, otherwise. Thus, it holds that $X := \sum_{v \in V} X_v$ and $c\lambda \leq \mathbf{E}[X] \leq \kappa c\lambda$ for every

p. Furthermore, it holds that $X := \sum_{v \in V} X_v$ is the sum of independent random variables. Hence, using Lemma 7.7 it holds that for $c \geq 12 \cdot k$,

$$\mathbf{Pr}\left[X \leq \frac{1}{2}c\lambda\right] \leq \exp\left(-\frac{\mathbf{E}[X]}{12}\right) \leq \exp\left(-k\lambda\right) = n^{-k}. \tag{7.7}$$

Moreover, it holds that for $c \geq 8 \cdot k$,

$$\begin{aligned}\mathbf{Pr}[X \geq 2c\lambda] &\leq \mathbf{Pr}\left[X \geq \left(1 + \tfrac{1}{2}\right)\kappa c\lambda\right] \\ &\leq \exp\left(-\frac{\mathbf{E}[X]}{8}\right) \\ &\leq \exp\left(-k\lambda\right) = n^{-k}. \end{aligned} \tag{7.8}$$

Then, using (7.7) and (7.8) the union bound yields the desired result.

□

Routing in a Linearized DeBruijn Graph

The LDS routing algorithm Alg-Routing we present in Section 7.3 is an adaptation of the classical LDG's routing algorithm presented in [RS11a, FS17]. Before we go into the details of Alg-Routing, we will first recall the classical LDG's routing algorithm. Routing in the LDG works by a bitwise adaption of the target address. Recall that our model assumes each node is aware of λ. Therefore, given any destination $p \in [0,1)$, a node can calculate the first λ bits $(p_1, \dots, p_\lambda)$ of p's binary representation. Then, starting with the least significant bit p_λ, the node v sends the message to the node closest to $x_1 := \frac{v+p_\lambda}{2}$. For this it uses the corresponding DeBruijn edge. After that, the message is sent to the node closest to $x_2 := \frac{x_1+p_{\lambda-1}}{2}$. This goes on until the first bit p_1. Finally, as a consequence of Lemma 7.10 there are w.h.p. only $O(\log n)$ hops over the list edges to p.

Definition 7.11 (Trajectory [RS11a]). Let $v \in V$ be a node and $p \in [0,1)$ be an arbitrary point. Furthermore, let $(v_1, \dots, v_\lambda) \in \{0,1\}^\lambda$ and $(p_1, \dots, p_\lambda) \in \{0,1\}^\lambda$ be the λ most significant bits of v and p, respectively. Then, the trajectory $\tau(v,p) := x_0, \dots, x_{\lambda+1} \in [0,1)^{\lambda+2}$ is a series of points in the $[0,1)$-interval defined as follows:

$$x_i := \begin{cases} v & i = 0 \\ (p_{\lambda-i+1}, \dots, p_\lambda, v_1, \dots, v_{\lambda-i}) & i \leq \lambda \\ p & i = \lambda + 1 \end{cases}$$

For each point x_i in the trajectory, the message is forwarded to the node closest to it. Then, forward the message along list edges until it reaches the node closest to the target.

Impossibility Results

Götte et al. [GRS19] show that it is impossible to maintain a connected overlay network under high adversarial churn in the presence of a $(0,\infty)$-late omniscient adversary, i.e., the adversary has up-to-date information on the network topology each round, but is completely oblivious to all other information stored in the nodes.

Lemma 7.12 (Götte et al. [GRS19]). *A $(0,\infty)$-late adversary with churn rate $(\alpha n, O(\log n))$ for some $\alpha \in (0,1)$, can disconnect any overlay in $O(\log n)$ rounds.*

Furthermore, Götte et al. also show that an adversary that is both oblivious to the topology and to the information stored in the nodes can disconnect an overlay network, if the new nodes are allowed to join via nodes that have been in the network for at most 1 round.

Lemma 7.13 (Götte et al. [GRS19]). *Let $v \in V$ be a node that joined in round t. Now assume a model where in round $t+1$ a new node $w \in V$ can join the network via v. Then, a (∞, ∞)-late adversary with churn rate $(\alpha n, O(\log n))$ for some $\alpha \in (0,1)$ can disconnect any overlay after $O(\log n)$ rounds.*

Lemma 7.13 proves that our model assumption on restricted join is justified. The formal proofs of Lemmas 7.12 and 7.13 are to appear in the Ph.D. thesis of Thorsten Götte [Gö21].

7.3 Routing in Overlay under Churn

In this section, we present a low-congestion routing algorithm Alg-Routing that routes messages to logical positions in a dynamic overlay network using the structural properties of the LDS. The algorithm delivers each message w.h.p. even in the presence of churn and a changing communication structure.

Alg-Routing must perform routing over a dynamic series of overlays $\mathcal{D} := (D_i)_{i \geq 0}$ where each D_i is an LDS. However, there are two challenges to be addressed, i.e.,

1. the churn orchestrated by the adversary and
2. the dynamic reconfiguration of the overlays.

The obvious solution would be to send the message not only to the closest node of each trajectory point, but to the whole swarm. However, observe that this trivial adaption to the LDG routing algorithm fails in the presence of churn. Given that any node on a message's trajectory can be churned out, a fraction of routing requests may never reach their destinations. In particular, if the adversary is aware of the topology, it could even churn out the whole swarm for a given trajectory point. Therefore, we introduce the notion of a *good* swarm adapted from Fiat et al. [FSY05]. In their work, a swarm is good if at least a fixed constant fraction of its nodes take part in the next round and hence, refer to such nodes as *good*.

Here, we need a slightly stronger notion as we require the good nodes to be relatively well spread over the swarm to enable our fast construction. To be precise, we say the left (right) side of a swarm is good if a constant fraction of its nodes is good and the full swarm is good, if both its left and right side are good. Furthermore, an LDS D_i is good if all its swarms are good. This property implies that there is always at least a constant fraction of good nodes in each swarm that can forward the message.

Besides the churn there is the problem of the dynamically rearranging overlay. In particular, the main algorithm we later introduce in Section 7.5 will create a series of overlays $D_1, D_2, \ldots$ which will persist for only 2 rounds each. That means a node changes its position every 2 rounds. Now if every node would keep all its routing messages and forward them from its new position, they would lose all the progress they made so far. Therefore, we define a *handover* procedure using a helper (handover) graph H_i. For any point $p \in [0, 1)$, let $S_i(p)$ be the swarm of p in D_i and $S_{i+1}(p)$ be the swarm of p in D_{i+1}. We assume that during the change from D_i to D_{i+1}, each node from $S_i(p)$ can send a message to any set of nodes from $S_{i+1}(p)$, i.e., the nodes form a helper graph H_i, where the swarms $S_i(p)$ and $S_{i+1}(p)$ are adjacent. Formally, it is defined as follows:

Algorithm 4 ALG-ROUTING: Routing Algorithm.

Desc: This algorithm is executed on a series of routable graphs $\mathcal{D} := (D_i, H_i)_{i\geq 0}$. The algorithm routes a message m from any node $u \in V_t$ to swarm $S(p)$ in $2\lambda + 2$ rounds.

Note: The following code is executed by each node $u \in V_t$ every round t. W.l.o.g. the forwarding step is executed in even rounds and the handover step is executed in odd rounds. The initial step is executed by the source node u. Messages initiated in even/odd rounds are processed by nodes in their target swarms in even/odd rounds, respectively.
Set $r = 16$, $\lambda = 2\log(\kappa n)$.

```
———————— Initial Step ————————
1: Sending a message m to $p$
       $(d_1, \dots, d_\lambda,) \longleftarrow \lambda$ most significant bits of $p$
       RND $:= t$
       if $t$ is even:
           Send $m := \big(p, \lambda - 1, (d_1, \dots, d_\lambda), \text{RND}\big)$ to all $w \in S(x_1)$ in $D_{t/2}$.
       else:
           Send $m := \big(p, \lambda, (d_1, \dots, d_\lambda), \text{RND}\big)$ to all $w \in S(x_0)$ in $D_{\lfloor t/2 \rfloor + 1}$.
———————— Forwarding Step ————————
2: Upon receiving $m := \big(p, k, (d_1, \dots, d_\lambda), \text{RND}\big)$
       if $k > 1$:
           $x \longleftarrow \frac{v+d_k}{2}$
           $(w_1, \dots, w_r) \longleftarrow r$ nodes chosen u.i.r from $S(x)$ in $D_{t/2}$
           Forward $m := \big(p, k-1, (d_1, \dots, d_\lambda), \text{RND}\big)$ to all $(w_i)_{i \in [r]}$.
       else:
           Deliver $m$ to all nodes $w \in S(x)$ in $D_{t/2}$.
———————— Handover Step ————————
3: Upon receiving $m := \big(p, k, (d_1, \dots, d_\lambda), \text{RND}\big)$
       if $k > 1$
           $(w_1, \dots, w_r) \longleftarrow r$ nodes chosen u.i.r from $S(x)$ in $D_{\lfloor t/2 \rfloor + 1}$.
           Forward $m$ to all $(w_i)_{i \in [r]}$.
       else:
           Deliver $m$ to all nodes $w \in S(x)$ in $D_{\lfloor t/2 \rfloor + 1}$.
```

Definition 7.14 (Handover Graph)**.** Let $(D_i)_{i\geq 0}$ be a series of LDS with $D_i = (V_{2i}, E_i)$. Then, the helper graph $H_i := (V_{2i+1}, E_i^H)$ is defined as follows:

$$(v, w) \in E_i^H \iff \text{exists } p \in [0,1) : v \in S_i(p) \wedge w \in S_{i+1}(p).$$

Given the definition of a handover graph, we state the following lemma.

Lemma 7.15 (Handover Property)**.** *Let $p \in [0,1)$ be an arbitrary point and let $S_{i+1}(p)$ be its swarm in D_{i+1}. Then, in H_i every node in $S_i^R(p)$ knows every other node in $S_{i+1}^R(p)$. The same holds for the left side.*

Proof. The property follows almost directly from the definition of swarms and the handover graph. We will prove the lemma only for the right side as the proof for the left side is completely analogous. Let v be any node in $S_{i+1}^R(p)$. By the definition of the Handover graph every node in $S_i(p_v)$ knows v, as clearly $v \in S_{i+1}(p_v)$. Since v is right of p and within distance $\frac{c\lambda}{n}$ of p, as it is in $S(p)$, it holds that $S_i^R(p) \subset S_i(v)$. Thus, by combining it with the definition of the Handover Graph, we get that every node $S_i^R(p) \subset S_i(v)$ must know v. Since this holds for all nodes $v \in S_{i+1}^R(p)$, all nodes in $S_i^R(p)$ must know all nodes in $S_{i+1}^R(p)$ and the lemma follows. □

Later, in Section 7.5 we will see how to implement such a handover graph, whereas here we just treat it as a property that is given. Note that we call a helper graph H_i good, if for each $p \in [0,1)$ a $3/4$-fraction of all nodes in $S_{i+1}(p)$ is not churned out in the next round.

We summarize our observations in following definition for a *routable* series of graphs.

Definition 7.16 (Routable Graphs)**.** Let $\mathcal{D} := (D_i, H_i)_{i\geq 0}$ be series of graphs defined on nodes $\mathcal{V} := (V_0, V_1, \dots)$, such that, each D_i consists of nodes in V_{2i} and each H_i consists of nodes in V_{2i+1}. Then, we call $\mathcal{D}$ routable, if

1. each D_i is an LDS,
2. each H_i enables a handover from each D_i to D_{i+1},
3. each D_i is good, i.e., it holds $|S_i(p) \cap V_{2i+1}| \geq 3/4 \cdot |S_i(p)|$ for all $p \in [0,1)$,
4. each H_i is good, i.e., it holds that for each $p \in [0,1)$ there exists a set of nodes $U(p) \subset V_{2i+1}$ such that, $S_{i+1}(p) \subseteq U(p)$ and
$$|S_{i+1}(p)| \geq 3/4 \cdot |U(p)|.$$

7.3.1 The Routing Algorithm

We now present the routing algorithm Alg-Routing for a dynamic series of routable graphs $\mathcal{D} := (D_i, H_i)_{i\geq 0}$. A trivial extension of the LDG routing algorithm would send each message to the whole swarm of each trajectory point. However, forwarding a message to a whole swarm would require $O(\log^2 n)$ messages to be sent in each step. In order to limit this to $O(\log n)$ messages [3], we adapt the approach as follows. Assume a node $v \in V_t$ wants to route a message m to target in $[0,1)$-interval. We first forward m to all nodes in $S(v)$. Then, each node in $S(v)$ picks $r \in \Theta(1)$ nodes uniformly and independently at random from the next swarm $S(x_1)$ in the trajectory and forwards m to them. Then, each node that received m at least once, forwards it to r nodes in $S(x_2)$ and so on. Only in the last step, the message is forwarded to all nodes of the target swarm to ensure that the whole swarm receives the message. We present Alg-Routing as Algorithm 4. Observe that in each even round $2i$, the message is forwarded along the trajectory step, i.e., from the swarm $S_i(x_j)$ to $S_i(x_{j+1})$ for some $j \in [\lambda]_0$, whereas in every odd round $2i+1$, the message is handed over from swarm $S_i(x_j)$ to $S_{i+1}(x_j)$ at some trajectory step $j \in [\lambda]_0$. Recall that for $p \in [0,1)$, $S_i(p)$ is the swarm of p in LDS D_i.

Analysis

In this section, we analyze Alg-Routing. In particular, we prove that w.h.p. all messages reach their target and further analyze the *dilation*, i.e., the number of steps until a message reaches its target and the *congestion*, i.e., the number of messages handled by each node in a round. Note that the latter depends on how many messages are sent each round and how their destinations are chosen. We would like to remark that we assume that each node sends exactly the same number of messages and chooses their destinations independently and uniformly at random.

[3]Given that αn nodes may fail in single round and we want to route each message on the first try, it is reasonable that one needs $O(\log n)$ copies of a message each round to ensure the survival of at least one w.h.p.

Theorem 7.17. *Let $\mathcal{D}$ be a routable series of LDS defined on nodes $\mathcal{V} := (V_0, V_1, \dots)$. Furthermore, let each node $v \in \mathcal{V}$ start $\varphi \in \mathbb{Z}_+$ messages to random targets $p \in [0, 1)$. Then* Alg-Routing *delivers each message with dilation exactly $2\lambda + 2$ and congestion $O(\varphi \log n)$ w.h.p.*

Proof. We begin the proof with the following lemma where we show that each message reaches its target swarm after exactly $2\lambda + 2$ rounds.

Lemma 7.18. *Let v be any node in $V_{2t} \in \mathcal{V}$ with $t \geq 0$, which sends a message to point $p \in [0, 1)$ along the trajectory $\tau(v, p)$ using* Alg-Routing*. Then, it holds that the message arrives at a node in $S_{t+\lambda+1}(p)$ in exactly $2\lambda + 2$ rounds.*

Proof. The proof follows by an induction over the trajectory steps $i = 0, \dots, \lambda$ and due to the fact that $S(x_\lambda)$ and $S(p)$ are adjacent. W.l.o.g. assume the message is initiated in round $2t = 0$. The proof follows analogously for message initiated in round $2t + 1$.

For the induction, observe that in round $2i$ the message is forwarded along the trajectory and therefore, moves from $S_i(x_i)$ to $S_i(x_{i+1})$, whereas in each round $2i + 1$ the message is handed over and therefore, moves from $S_i(x_{i+1})$ to $S_{i+1}(x_{i+1})$. Lemma 7.9 and the handover property imply that the nodes have necessary connections for each step but the last. We now prove by induction that for each $i \in [\lambda]_0$, that each copy of the message is stored at a node $u \in S_i(x_i)$ in round $2i$.

(IB) Consider step $i = 0$, i.e., the round in which the message is started. In this round, the message is at $v = x_0$ and therefore, is known by $v \in S_0(x_0)$.

(IS) Now suppose that the induction hypothesis holds for any arbitrary step $i \in [\lambda - 1]_0$ along the trajectory. Implying that in round $2i$ any copy of the message is at a node in $S_i(x_i)$. Now since rounds $2i$ are even rounds, Alg-Routing performs a forwarding step in D_i along the trajectory, i.e., every copy of the message is sent to some node in $S_i(x_{i+1})$. Observe that, the swarm property in Lemma 7.9 ensures that each node in $S_i(x_i)$ has a connection to $S_i(\frac{x_i}{2})$ and $S_i(\frac{x_i+1}{2})$. Now since x_{i+1} is either $\frac{x_i}{2}$ or $\frac{x_i+1}{2}$, it holds that each node in $S_i(x_i)$ has an edge to each node in $S_i(x_{i+1})$.

Next, in round $2i + 1$, Alg-Routing performs a handover operation on every copy of the message in $S_i(x_{i+1})$ (overlay H_i). Now, we use the Handover Property and observe that each node in $S_i(x_{i+1})$ has by the definition of H_i, an edge to each node in $S_{i+1}(x_{i+1})$. Therefore, every copy of the message can successfully be sent to a node in $S_{i+1}(x_{i+1})$ and therefore, available in round $2i + 2$. This concludes the induction.

Particularly, for $i = \lambda - 1$, i.e., in round $2\lambda - 2$ the message transits from $S_{\lambda-1}(x_{\lambda-1})$ to nodes in $S_{\lambda-1}(x_\lambda)$. Now since x_λ is either $\frac{x_{\lambda-1}}{2}$ or $\frac{x_{\lambda-1}+1}{2}$, it holds that each node $S_{\lambda-1}(x_{\lambda-1})$ has an edge to each node in $S_{\lambda-1}(x_\lambda)$. Therefore, every copy of the message can be successfully forwarded to each node in $S_{\lambda-1}(x_\lambda)$. Next, in round $2\lambda - 1$, Alg-Routing performs a handover operation on every copy of the message in $S_{\lambda-1}(x_\lambda)$ (overlay $H_{\lambda-1}$). Using the Handover Property and the observation that each node in $S_{\lambda-1}(x_\lambda)$ has by the definition of $H_{\lambda-1}$, an edge to each node in $S_\lambda(x_\lambda)$, the message can be successfully sent to a node in $S_\lambda(x_\lambda)$. The induction above implies that the message is known by all nodes in $S_\lambda(x_\lambda)$ in round 2λ. Now recall that x_λ and p are equal in their first λ bits. This implies that the distance between x_λ and p is then at most

$$d(x_\lambda, p) \leq \frac{1}{2^\lambda},$$

using $\lambda \coloneqq 2\log(\kappa n)$ we get,

$$\begin{aligned}
&= \frac{1}{e^{\log(2)\lambda}} = \frac{1}{\left(e^{\lambda}\right)^{\log(2)}} \\
&= \frac{1}{\left(\kappa n\right)^{2\log(2)}} \\
&\leq \frac{1}{\kappa n} \leq \frac{1}{n}.
\end{aligned}$$

Therefore, the swarms $S_\lambda(x_\lambda)$ and $S_\lambda(p)$ are adjacent and the message can be forwarded and handed over to nodes in $S_{\lambda+1}(p)$ in round $2\lambda+2$ as described in the induction step. This proves the lemma.

□

Notice that in the proof of Lemma 7.18, we assumed that each intermediate node along the trajectory forwarded the received message and omitted the fact that not all nodes in a swarm forward the message, as they may be churned out. Observe that, if an entire swarm is churned out, the message is discarded. However, as stated in Definition 7.16, we assume that all swarms are *good*, i.e., only a constant fraction of each swarm is malicious and does not forward the message. In the following lemma we show that there exists a good fraction of nodes in every swarm that are not churned out.

Lemma 7.19. *Consider a set of nodes S with $|S| \geq \frac{c\lambda}{2}$ picked at random from the set of all nodes in any given round. Let $G \subseteq S$ be the set of good nodes in S, then for churn parameters $\alpha = \frac{1}{16}$ and $\kappa = \left(1+\frac{1}{16}\right)$ with $c \geq 510k$ it holds that,*

$$\mathbf{Pr}\left[|G| \leq \frac{14}{17}|S|\right] \leq \frac{1}{n^k}.$$

Proof. Observe that by definition, for a churn rate of α, there are at least $(1-\alpha)n$ nodes that would survive into the next round. Therefore, there are at least $\frac{15}{16}n$ good nodes in any given round. Also, since there could be at most κn nodes, there are at most $\frac{17}{16}n$ nodes in any given round. Therefore, the fraction of good nodes in the $[0,1)$-interval in any given round is then at least $\frac{15}{17}$. Furthermore, nodes pick their position uniformly and independently at random in the $[0,1)$-interval.

Let $S \subset V_t$ be a set of nodes in round t picked at random from the $[0,1)$-interval such that, $|S| \geq \frac{c\lambda}{2}$. Let the random variables $(X_v)_{v\in S}$ be such that $X_v = 1$ if the node v is good and 0, otherwise. The random variables $(X_v)_{v\in S}$ observe random sampling without replacement and therefore, are negatively associated [JDP83]. Then, the random variable $X_S \coloneqq \sum_{v\in S} X_v$ denotes the number of good nodes in the set S. Moreover, observe that X_S is hypergeometrically distributed in $|V_t|$, the number of good nodes in V_t i.e., (at least) $\frac{15}{16}n$, and $|S|$ [Sch00, p. 44]. This implies,

$$\mathbf{E}[X_S] \geq |S|\frac{15}{17}.$$

Then applying the Chernoff bound on NA random variables with $c \geq 510 \cdot k$ we get,

$$\begin{aligned}
\mathbf{Pr}\left[X_S \leq \frac{14}{17}|S|\right] &\leq \mathbf{Pr}\left[X_S \leq \left(1-\frac{1}{15}\right)\mathbf{E}[X_S]\right] \\
&\leq \exp\left(-\frac{|S|}{255}\right) \leq \exp\left(-\frac{c\lambda}{510}\right) \leq \exp\left(-k\lambda\right) = n^{-k}.
\end{aligned}$$

□

Thus, as long as we observe only $o(n^k)$ swarms and each side of each swarm has more than $\frac{c\lambda}{2}$ nodes w.h.p., a simple union bound implies that in all swarms both the left and right side are good w.h.p.

Using Lemma 7.19 we can now show that the messages reach their destination w.h.p.

Lemma 7.20. *Let m be a message initiated in round $2q$ that is routed along $\tau(v,p) := x_0, \dots, x_\lambda, p$ using* Alg-Routing. *Then, it holds that all nodes in $S_{q+\lambda+1}(p)$ receive m w.h.p after exactly $2\lambda + 2$ rounds.*

Proof. W.l.o.g. we assume the message was initiated in an even round and $q = 0$. We prove by induction that, for each $t \in [2\lambda + 2]_0$, it holds that if t is an even round then, at least half of all nodes in $S_{t/2}(x_{t/2})$ receive the message m and forward them w.h.p. Otherwise, if t is an odd round, at least half of all nodes in $S_{\lfloor t/2 \rfloor}(x_{\lfloor t/2 \rfloor + 1})$ receive the message m and forward them w.h.p.

(IB) Consider round $t = 0$, i.e., the round in which the message is initiated. If this is an even round, then observe that Alg-Routing forwards message m from node v, i.e., x_0, to all nodes in $S_0(x_1)$. Using Lemma 7.10 and 7.19 we can conclude that at least half of all good nodes in the swarm $S_0(x_1)$ received the message m and survive until the next round. Therefore, the induction hypothesis holds.

(IS) Now assume the induction hypothesis holds for any arbitrary $t \in [2\lambda+1]_0$. Consider round $2b + 2$ for $b \in [\lambda]_0$, we show that at least half of all nodes in the swarm $S_{b+1}(x_{b+1})$ receive the message m w.h.p. from nodes in $S_b(x_{b+1})$.

By the induction's hypothesis, at least half of all nodes in $S_b(x_{b+1})$ received message m w.h.p. and therefore, each of these nodes forward r copies of the message m to nodes picked uniformly and independently at random from the swarm $S_{b+1}(x_{b+1})$. We now show that at least half of all nodes in the swarm $S_{b+1}(x_{b+1})$ receive the message w.h.p. in the round $2b + 2$.

From Lemma 7.10, it holds that $|S_{b+1}(x_{b+1})| \leq 4|S_b(x_{b+1})|$ w.h.p. Furthermore, by the induction hypothesis we know that at least half of all nodes in $S_b(x_{b+1})$ received m and therefore, forward r copies of m. Thus, in total there are at least $r/8 \cdot |S_{b+1}(x_{b+1})|$ copies of m sent to $S_{b+1}(x_{b+1})$. The probability that any of these messages is sent to a given node $v' \in S_{b+1}(x_{b+1})$ is $\frac{1}{|S_{b+1}(x_{b+1})|}$, since the destinations are chosen uniformly at random. Observe that one can view the forwarding of messages from $S_b(x_{b+1})$ to uniformly and independently picked nodes in $S_{b+1}(x_{b+1})$ as a balls-into-bins experiment, where (at least) $r/8 \cdot |S_{b+1}(x_{b+1})|$ balls are thrown into $|S_{b+1}(x_{b+1})|$ bins. Using Propositions 7.4 and 7.5 one can show that the number of nodes in $S_{b+1}(x_{b+1})$ that receive at least one ball is NA [DR98]. For $i \in \{1, \dots, |S_{b+1}(x_{b+1})|\}$ and $j \in \{1, \dots, r/8 \cdot |S_{b+1}(x_{b+1})|\}$, let $X_{i,j}$ be an indicator random variable such that, $X_{i,j} = 1$ if message j is sent to node i (picked uniformly and independently at random) by Alg-Routing in round $2b + 1$, and $X_{i,j} = 0$ otherwise.

For any fixed $j \in \{1, \dots, r/8 \cdot |S_{b+1}(x_{b+1})|\}$, let $Y_i := X_{i,j}$ for all $i \in \{1, \dots, |S_{b+1}(x_{b+1})|$. Then, from Lemma 7.6 we know that the random variables $Y_1, \dots, Y_{|S_{b+1}(x_{b+1})|}$ are NA. Since each message $j \in \{1, \dots, r/8 \cdot |S_{b+1}(x_{b+1})|\}$ is destined to a node that is picked uniformly and independently at random, using Proposition 7.4, we can conclude that the set of random variables $(X_{i,j})_{i \in \{1,\dots,|S_{b+1}(x_{b+1})|\}, j \in \{1,\dots,r/8 \cdot |S_{b+1}(x_{b+1})|\}}$ are NA.

For each $i \in \{1, \ldots, |S_{b+1}(x_{b+1})|\}$ consider a non-decreasing function as follows,

$$X_i = \begin{cases} 1 & \sum_{j \in \{1,\ldots, r/8 \cdot |S_{b+1}(x_{b+1})|} X_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, for each $i \in \{1, \ldots, |S_{b+1}(x_{b+1})|\}$,

$$\begin{aligned} \mathbf{Pr}[X_i = 1] &= 1 - \left(1 - \frac{1}{\mid S_{b+1}(x_{b+1}) \mid}\right)^{|S_{b+1}(x_{b+1})| \cdot r/8} \\ &\geq 1 - \left(\frac{1}{e}\right)^{\frac{r}{8}}. \end{aligned}$$

From Proposition 7.5, we know that the random variables $X_1, \ldots, X_{|S_{b+1}(x_{b+1})|}$ are NA.

Let $G_{b+1}(x_{b+1}) \subseteq S_{b+1}(x_{b+1})$ denote the set of good nodes in $S_{b+1}(x_{b+1})$. For each $v \in G_{b+1}(x_{b+1})$, let G_v be a $\{0, 1\}$ random variable such that, $G_v = 1$ if node v received at least one copy of m from some node in $S_{b+1}(x_b)$ and $G_v = 0$, otherwise. Observe that X_i denotes if a node in $S_{b+1}(x_{b+1})$ received at least one message in round $2b+2$ when exactly $|S_{b+1}(x_{b+1})| \cdot r/8$ messages are sent to $S_{b+1}(x_{b+1})$ uniformly and independently at random. Since $|S_{b+1}(x_{b+1})| \cdot r/8$ is a lower bound on the number of message being sent, for each $v \in G_{b+1}(x_b + 1)$ we have that,

$$\mathbf{Pr}[G_v = 1] \geq \mathbf{Pr}[X_v = 1].$$

Moreover, the random variables $(G_v)_{v \in G_{b+1}(x_{b+1})}$ are also NA as they can be seen as a subset of $(X_v)_{v \in S_{b+1}(x_{b+1})}$. To see this, first recall that the random variables $(X_i)_{i \in \{1,\ldots,|S_{b+1}(x_{b+1})|\}}$ are NA. In particular, this fact is independent of the number of messages and nodes in $S_{b+1}(x_{b+1})$. Now observe that for each $v \in G_{b+1}(x_{b+1})$, the random variable G_v is a non-decreasing function of its associated random variable X_v. Therefore, we can conclude that the random variables $(G_v)_{v \in G_{b+1}(x_{b+1})}$ are also NA.

Let $G \coloneqq \sum_{v \in G_{b+1}(x_{b+1})} G_v$ be a random variable that counts the number of good nodes in $S_{b+1}(x_{b+1})$ that received at least one copy of m in round $2b + 2$. From Lemma 7.19, we know that there are $14/17$ fraction of good nodes in $S_{b+1}(x_{b+1})$ w.h.p. Therefore, the expected number of good nodes that receive at least one message is given by,

$$\begin{aligned} \mathbf{E}[G] &= \sum_{v \in G_{b+1}(x_{b+1})} \mathbf{E}[G_v] \\ &\geq \sum_{v \in G_{b+1}(x_{b+1})} \left(1 - \frac{1}{e^{r/8}}\right) \\ &= \left(1 - \frac{1}{e^{r/8}}\right) \frac{14}{17} |S_{b+1}(x_{b+1})|. \end{aligned}$$

To complete the induction it suffices to show that,

$$\mathbf{Pr}\left[G \leq \frac{1}{2} |S_{b+1}(x_{b+1})|\right] \leq \frac{1}{n^k}.$$

Using $\gamma = \frac{2}{7}$ in Lemma 7.7 with $r = 16$ we get,

$$\mathbf{Pr}\left[G \leq \tfrac{5}{7}\left(1 - \tfrac{1}{e^2}\right)\tfrac{14}{17}|S_{b+1}(x_{b+1})|\right] \leq \exp\left(-\tfrac{4}{49\cdot 3}\left(1 - \frac{1}{e^2}\right)\tfrac{14}{17}|S_{b+1}(x_{b+1})|\right).$$

Lemma 7.10 then implies,

$$\leq \exp\left(-\tfrac{4}{49\cdot 3}\left(1 - \frac{1}{e^2}\right)\frac{14}{17}\frac{c\lambda}{2}\right)$$

$$= \exp\left(-\tfrac{4}{357}\left(1 - \frac{1}{e^2}\right)c\lambda\right)$$

for $c \geq 510 \cdot k$ we get,

$$\leq n^{-k}.$$

Particularly, setting $b = \lambda$ gives us that at least half of all nodes in $S_\lambda(x_{\lambda+1})$ forward the message in round $2\lambda + 1$ and therefore, at least half of all nodes in swarm $S_{\lambda+1}(x_{\lambda+1})$ receive the message in round $2\lambda + 2$. □

We conclude our analysis by observing each node's congestion. The following lemma bounds the expected number of trajectories that cross an interval in each round. Recall that a trajectory is defined on points in $[0, 1)$ and not on actual nodes (except the source and destination).

Lemma 7.21. *Assume η nodes choose their position independently and uniformly at random in the $[0, 1)$-interval. Moreover, let each node send $\varphi \in \mathbb{Z}_+$ messages to targets picked independently and uniformly at random from the $[0, 1)$-interval. Then, for every $I \subset [0, 1)$ it holds that,*

1. *X_I^j is the sum of independent $\{0, 1\}$ random variables,*
2. *$\mathbf{E}\left[X_I^j\right] = \varphi\eta|I|$,*

where X_I^j is a random variable that counts the number of trajectories that have their j^{th} step in the interval I and $|I|$ denotes the size of the interval I.

Proof. W.l.o.g. assume that $I := [a, b]$ with $0 \leq a \leq b < 1$.

1. Let $X_{I:(v,i)}^j$ be a $\{0, 1\}$ random variable such that, $X_{I:(v,i)}^j = 1$ if the trajectory of a message $i \in [\varphi]$ started by a node v crosses the interval I in its j^{th} step and $X_{I:(v,i)}^j = 0$, otherwise. Then the number of messages with their j^{th} step in the interval I is given by,

$$X_I^j = \sum_{v \in V}\sum_{i \leq \varphi} X_{I:(v,i)}^j.$$

Observe that a message's trajectory is uniquely defined by the starting node v and the end point p. Since for each message the target node is chosen uniformly and independently at random from $[0, 1)$-interval, we conclude that the set of random variables $\left(X_{I:(v,i)}^j\right)_{v \in V, i \in [\varphi]}$ are independent.

2. We prove by induction that for each step $j \in [\lambda + 1]_0$ along the trajectory, it holds that,

$$\mathbf{E}\left[X_I^j\right] = \varphi\eta|I|.$$

(IB) Consider step $j = 0$ i.e., the step where the messages are at their starting node at x_0. For each $v \in V$, let X_v be a $\{0, 1\}$ random variable such that, $X_v = 1$ if node $v \in I$ and $X_v = 0$, otherwise. Observe that the nodes pick their positions uniformly and independently at random in the $[0, 1)$-interval. Therefore,

$$\mathbf{Pr}[X_v = 1] = |I|.$$

This implies,

$$\mathbf{E}\left[\sum_{v \in V} X_v\right] = \sum_{v \in V} \mathbf{Pr}[X_v = 1] = \eta|I|.$$

Since each node initiates $\varphi \in \mathbb{Z}_+$ messages to randomly picked targets in the $[0, 1)$-interval,

$$\mathbf{E}\big[X_I^0\big] = \mathbf{E}\left[\sum_{v \in V} \varphi \cdot X_v\right] = \eta\varphi|I|.$$

(IS) Now assume the induction hypothesis holds for every $j \in [\lambda]_0$, this implies,

$$\mathbf{E}\Big[X_I^j\Big] = \varphi\eta|I|.$$

Let $I_0 := I \cap [0, 1/2)$ and $I_1 := I \cap [1/2, 1)$ be the parts of I that lie in the first and second half of $[0, 1)$-interval, respectively. Then, for all $k \in [\lambda + 1]_0$,

$$\mathbf{E}\Big[X_I^k\Big] = \mathbf{E}\Big[X_{I_0}^k\Big] + \mathbf{E}\Big[X_{I_1}^k\Big].$$

Observe that for each $i \in \{0, 1\}$, the bit representation of each point on I_i begins with i. Therefore, for any message m destined to a uniformly and independently picked target $p \in [0, 1)$-interval, it's trajectory crosses interval I_i in the j^{th} step of the trajectory if and only if $p_j = i$, where p_j is the j^{th} most significant bit of p in the binary representation. W.l.o.g. we only analyze I_0 to show that,

$$\mathbf{E}\Big[X_{I_0}^{j+1}\Big] = \varphi\eta|I_0|.$$

The proof for $\mathbf{E}\Big[X_{I_1}^{j+1}\Big] = \varphi\eta|I_1|$ is analogous and follows using similar arguments.

Consider an arbitrary trajectory $\tau := x_0, \ldots, x_{\lambda+1} \in [0, 1)^{\lambda+2}$ with $x_{j+1} \in I_0 := [a, b_0]$ for some $j \in [\lambda]_0$. From Definition 7.11 it must then hold that $x_j = 2x_{j+1}$. Therefore, for all trajectories that cross the interval $I_0 \in [0, 1/2)$ it holds that $x_j = 2x_{j+1}$ and hence, the j^{th} step must be in the interval $J := [2a, 2b_0]$. The size of the interval is then,

$$|J| = 2|I_0|.$$

By the induction hypothesis we know that,

$$\mathbf{E}\Big[X_J^j\Big] = \varphi n|J| = 2\varphi n|I_0|.$$

Note that since for every message it's target is picked uniformly and independently at random from the $[0, 1)$-interval, this is equivalent to the thought experiment of flipping a fair coin for each bit of the target address. Therefore, the probability that a trajectory τ in interval J points towards the interval I_0 in it's $(j+1)^{\text{th}}$ step is then,

$$\mathbf{Pr}[\tau_{j+1} \in I_0] = \frac{1}{2},$$

where τ_{j+1} denotes the position of the trajectory τ in step $j+1$.

Then, the expected number of trajectories that point from the interval J to the interval I_0 is given by,

$$\begin{aligned}
\mathbf{E}\Big[X_{I_0}^{j+1}\Big] &= \sum_{\ell=0}^{\infty} \mathbf{E}\Big[X_{I_0}^{j+1} | X_J^j = \ell\Big] \cdot \mathbf{Pr}\Big[X_J^j = \ell\Big] \qquad \text{(Law of Total Expectation)}\\
&= \sum_{\ell=0}^{\infty} \sum_{\tau \in [1,\dots,\ell]} \mathbf{Pr}[\tau_{j+1} \in I_0] \cdot \mathbf{Pr}\Big[X_J^j = \ell\Big]\\
&= \sum_{\ell=0}^{\infty} \frac{\ell}{2} \cdot \mathbf{Pr}\Big[X_J^j = \ell\Big]\\
&= \frac{1}{2} \sum_{\ell=0}^{\infty} \ell \cdot \mathbf{Pr}\Big[X_J^j = \ell\Big]\\
&= \frac{1}{2} \mathbf{E}\Big[X_J^j\Big] = \varphi\eta |I_0|.
\end{aligned}$$

Since $|I| = |I_0| + |I_1|$ we get,

$$\mathbf{E}\Big[X_I^{j+1}\Big] = \varphi\eta|I_0| + \varphi\eta|I_1| = \varphi\eta|I|.$$

This completes the induction. In particular, for $j = \lambda$,

$$\mathbf{E}\Big[X_I^{\lambda+1}\Big] = \varphi\eta|I_0| + \varphi\eta|I_1| = \varphi\eta|I|.$$

□

Using Lemma 7.21 we can bound from above the congestion for Alg-Routing as $O(\varphi \log n)$.

Lemma 7.22. *If each node picks its position and the $\varphi \in \mathbb{Z}_+$ target nodes uniformly and independently at random, then Alg-Routing has congestion at most $27r\varphi\kappa c\lambda$ w.h.p.*

Proof. Let $v \in V$ be any node and let $\left[v \pm \frac{c\lambda}{n}\right] =: I_v \subset [0,1)$ be an interval that contains all points p with $v \in S(p)$. Observe that a message may be routed via v only if its trajectory passes the interval I_v. From Lemma 7.21 we know that for any given round j the expected number of trajectories that cross the interval I_v is then,

$$2\varphi c\lambda \le \mathbf{E}\Big[X_{I_v}^j\Big] \le 2\varphi\kappa c\lambda,$$

where $X_{I_v}^j$ is a random variable that counts the number of trajectories that have their j^{th} step in the interval I_v. Using the Chernoff bound we get,

$$\mathbf{Pr}\Big[X_{I_v}^j \ge 3\varphi\kappa c\lambda\Big] \le \exp\left(-\frac{1}{4}\varphi c\lambda\right),$$

$$\text{for } c \geq 510 \cdot k,$$
$$\leq n^{-k}.$$

Therefore, the total number of trajectories that pass interval I_v in any round j is at most $3\varphi\kappa c\lambda$ w.h.p. We know from Definition 7.11, that a trajectory passing interval I_v in step j, had it's step $j-1$ in either interval $J^0 := \left[2\left(v \pm \frac{c\lambda}{n}\right)\right]$ with $p_j = 0$, or in the interval $J^1 := \left[2\left(v \pm \frac{c\lambda}{n}\right) - 1\right]$ with $p_j = 1$. Observe that the size of these intervals i.e., $|J^0| = |J^1| = 2|I_v|$. From Lemma 7.10 we know that the number of nodes in each of these interval i.e., J^0 and J^1, are at most $8c\lambda$ w.h.p. Therefore, the total number of messages that will be forwarded to the interval I_v in any given round is then at most $M = (r \cdot 8c\lambda) \cdot (3\varphi\kappa c\lambda)$ w.h.p.

Let $(X_i^v)_{i\in[M]}$ be a set of $\{0,1\}$ random variables such that, $X_i^v = 1$ if message m_i is sent to node v and $X_i^v = 0$, otherwise. From Lemma 7.10 we know that, any interval of size $\frac{2c\lambda}{n}$ has at least $c\lambda$ nodes w.h.p. Then,

$$\mathbf{Pr}[X_i^v = 1] \leq \frac{1}{c\lambda}.$$

Let $(\widetilde{X}_i^v)_{i\in[M]}$ be a set of $\{0,1\}$ random variables such that,

$$\mathbf{Pr}\left[\widetilde{X}_i^v = 1\right] = \frac{1}{c\lambda}.$$

Then,

$$\mathbf{E}\left[\sum_{i\in[M]} \widetilde{X}_i^v\right] = \frac{1}{c\lambda}(r \cdot 8c\lambda) \cdot (3\varphi\kappa c\lambda) = 24r\varphi\kappa c\lambda.$$

For $r = 16$ and $c = 510k$,

$$\begin{aligned}\mathbf{Pr}\left[\sum_{i\in[M]} X_i^v \geq 27r\varphi\kappa c\lambda\right] &\leq \mathbf{Pr}\left[\sum_{i\in[M]} \widetilde{X}_i^v \geq \left(1+\frac{1}{9}\right)24r\varphi\kappa c\lambda\right] \\ &\leq \exp\left(-\frac{24r\varphi\kappa c\lambda}{9^2\cdot 2}\right) \\ &\leq n^{-k}.\end{aligned}$$

□

Theorem 7.17 follows directly from Lemmas 7.18, 7.20 and 7.22. □

7.4 The Random Sampling Algorithm

The algorithm Alg-Routing can also be extended to send a message to a node (and not swarm) chosen uniformly at random. We call this algorithm Alg-Sampling presented as Algorithm 5. The underlying approach is adapted from King and Saia [KS04] and King et al. [KLSY07] and works as follows.

A node first picks a value $p \in [0,1)$ uniformly at random and routes the message to the swarm $S(p)$ using algorithm Alg-Routing. The message is then delivered to a randomly

Algorithm 5 ALG-SAMPLING: Random Sampling Algorithm.

```
Desc: This algorithm is executed on a routable graph $\mathcal{D} := (D_i, H_i)_{i \geq 0}$. It routes a
      message $m$ from any node $u \in V_{2t}$ to a node $v$ (almost) uniformly picked from $V_{2t+2\lambda+2}$.
                              Round $t$
1: Send a message m to a random node in $V_{2t+2\lambda+2}$
       $p \longleftarrow$ Uniformly chosen from $[0,1)$
       $\Delta \longleftarrow$ Uniformly chosen from $[0, 2c\lambda]$
       Route message $(m, p, \Delta)$ to target $S(p)$ using ALG-ROUTING
2: Upon receiving $(m, p, \Delta)$ from ALG-ROUTING
       $P \longleftarrow \{w \in S(p) \mid p_w \in S^R(p)\}$
       Choose $w$ such that $|\{u \mid u \in \langle p, w\rangle\}| = \Delta \mod |P|$
       Deliver $m$ to $w$
```

chosen node $w \in S(p)$ by including a uniformly random chosen number $\Delta \in [0, 2c\lambda]$ in the message. The message is then delivered to a node w for which it holds that,

$$|\{u \mid u \in \langle p, w\rangle\}| = \Delta \mod |S(p)|.$$

Since all nodes in $S(p)$ that received the message are aware of $|S(p)|$ and Δ, this can be checked locally without further messages. The following lemma bounds the sampling probability.

Lemma 7.23. *Let $\mathcal{D}$ be routable. Let node $v \in V_t$ start a message m_v using ALG-SAMPLING. Then, for all $u \in V_{t+2\lambda+2}$,*

$$\mathbf{Pr}[u \text{ receives } m_v] \in \left[\frac{1}{4n}, \frac{5}{n}\right].$$

Furthermore, for any two nodes $v, w \in V_t$ that start messages m_v and m_w, respectively, it holds that for all $u \in V_{t+2\lambda+2}$,

$$\mathbf{Pr}[u \text{ receives } m_v] = \mathbf{Pr}[u \text{ receives } m_w].$$

Proof. Let $Y(v, u)$ be a $\{0, 1\}$ random variable such that $Y(v, u) = 1$ in the event that v samples u i.e., the message m_v is delivered to u. Furthermore, let $p \in [0, 1)$ be the point that v chooses in step 1 of ALG-SAMPLING. Let us denote by $P^L(u)$ all points in the interval $\left[p_u - \frac{c\lambda}{n}, p_u\right]$, where p_u is the position of node u in the $[0, 1)$-interval. Then, we make the following observations:

1. As a necessary condition to sample the node u, node v must pick a point $p \in [0, 1)$ such that, $p \in P^L(u)$. Otherwise, u will never be considered in step 2 of ALG-SAMPLING. The probability that $p \in P^L(u)$ is then $\frac{c\lambda}{n}$.
2. Given that $p \in P^L(u)$, ALG-SAMPLING must pick u uniformly at random from all nodes in $S^R(p)$. The probability for this depends only on $|S^R(p)|$ and Δ. By Lemma 7.10 we know that w.h.p. it contains at most $2c\lambda$ nodes and at least $c\lambda/2$ nodes. Since $\Delta \leq 2c\lambda$ and $\frac{c}{2}\lambda \leq |S^R(p)| \leq 2c\lambda$, there are at least one and at most 4 choices of Δ that result in u being picked.

We now prove the first part of the lemma. For the lower bound we get that,

$$\mathbf{Pr}[Y(v, u)] = \mathbf{Pr}\left[p \in P^L(u)\right] \cdot \mathbf{Pr}\left[u \text{ is picked from } S^R(p)\right]$$

$$
\begin{aligned}
&\geq \mathbf{Pr}\big[p \in P^L(u)\big] \cdot \left(\mathbf{Pr}\big[|S^R(p)| \leq 2c\lambda\big]\frac{1}{2c\lambda} + \mathbf{Pr}\big[|S^R(p)| > 2c\lambda\big]\frac{1}{n}\right) \\
&= \frac{c\lambda}{n} \cdot \left(\left(1 - \frac{1}{n^k}\right)\frac{1}{2c\lambda} + \frac{1}{n^{k+1}}\right) \\
&\geq \frac{1}{2n} - \frac{1}{2n^{k+1}} \\
&\geq \frac{1}{4n}.
\end{aligned}
$$

The proof for the upper bound is analogous, we simply replace the upper bound on the swarm size with its lower bound, i.e.,

$$
\begin{aligned}
\mathbf{Pr}[Y(v,u)] &= \mathbf{Pr}\big[p \in P^L(u)\big] \cdot \mathbf{Pr}\big[u \text{ is picked from } S^R(p)\big] \\
&\leq \mathbf{Pr}\big[p \in P^L(u)\big] \cdot \left(\mathbf{Pr}\Big[|S^R(p)| \geq \frac{c}{2}\lambda\Big]\frac{1}{c\lambda} + \mathbf{Pr}\Big[|S^R(p)| \leq \frac{c}{2}\lambda\Big] \cdot 1\right) \\
&= \frac{2c\lambda}{n} \cdot \left(\left(1 - \frac{1}{n^k}\right)\frac{1}{c\lambda} + \frac{1}{n^k}\right) \\
&\leq \frac{c\lambda}{n} \cdot \left(\frac{4}{c\lambda} + \frac{1}{n^c}\right) \\
&= \frac{4}{n} + \frac{2c\lambda}{n^{k+1}}
\end{aligned}
$$

since $c\lambda \in O(\log n)$,

$$
\leq \frac{5}{n}.
$$

This concludes the first part of the lemma.

It remains to show that Alg-Sampling delivers messages of nodes v and w, i.e., m_v and m_w, respectively to a node $u \in V_{t+2\lambda+2}$ with the same probability. Let p_v and $p_w \in [0,1)$ be the points picked by these nodes for their respective messages. Furthermore, let Δ_v and $\Delta_w \in [0, 2c\lambda]$ be random numbers picked by Alg-Sampling.

Let $\ell(u,p) := |\{w \mid w \in \langle p, u\rangle\}|$ and $\mathcal{U}(u,p)$ be the number of possible choices of Δ that lead to u being picked given that the message is routed to p in the first step. Note that any point $p \in [0,1)$ and any $\Delta \in [0, 2c\lambda]$ is picked with equal probability by v and w. Then, it holds that,

$$
\begin{aligned}
\mathbf{Pr}[Y(v,u)] &= \mathbf{Pr}\big[p_v \in P^L(u)\big] \cdot \mathbf{Pr}\big[u \text{ is picked from } S^R(p_v)\big] \\
&= \mathbf{Pr}\big[p_v \in P^L(u)\big] \cdot \left(\sum_{p^* \in S^L(u)} \mathbf{Pr}[p_v = p^*] \cdot \mathbf{Pr}\big[\ell(u,p^*) = \Delta_v \mod |S^R(p^*)|\big]\right) \\
&= \mathbf{Pr}\big[p_v \in P^L(u)\big] \cdot \left(\sum_{p^* \in S^L(u)} \mathbf{Pr}[p_v = p^*]\frac{\mathcal{U}(p^*)}{|S^R(p^*)|}\right) \\
&= \mathbf{Pr}\big[p_w \in P^L(u)\big] \cdot \left(\sum_{p^* \in S^L(u)} \mathbf{Pr}[p_w = p^*]\frac{\mathcal{U}(p^*)}{|S^R(p^*)|}\right) \\
&= \mathbf{Pr}\big[p_w \in P^L(u)\big] \cdot \left(\sum_{p^* \in S^L(u)} \mathbf{Pr}[p_w = p^*] \cdot \mathbf{Pr}\big[\ell(u,p^*) = \Delta_w \mod |S^R(p^*)|\big]\right) \\
&= \mathbf{Pr}\big[p_w \in P^L(u)\big] \cdot \mathbf{Pr}\big[u \text{ is picked from } S^R(p_w)\big]
\end{aligned}
$$

$$= \mathbf{Pr}[Y(w,u)].$$

The fourth equality is due to the fact that,

$$\mathbf{Pr}\big[p_v \in P^L(u)\big] = \mathbf{Pr}\big[p_w \in P^L(u)\big] = \frac{c\lambda}{n},$$

and for any $p \in [0,1)$,

$$\mathbf{Pr}[p_v = p] = \mathbf{Pr}[p_w = p],$$

since all nodes use the same random hash function h to compute the positions.

This concludes the second part of the lemma. □

7.5 The Maintenance Algorithm

We now present algorithms for constructing an LDS every two rounds and handling addition of nodes to the overlay. Algorithms Alg-LDS and Alg-Random maintain a series of routable dynamic overlays $\mathcal{D} = (D_i, H_i)_{i \geq 0}$ with high probability. Before we present the algorithms, we first give an overview of our assumptions and choice of churn parameters.

We assume a $(2, 2\lambda+7)$-late adversary with a churn rate $({}^{n}/_{16}, 2\lambda+7)$. This implies that within $2\lambda+7$ (i.e., $O(\log n)$) rounds, a constant fraction of nodes can be subjected to churn. Furthermore, we assume that the number of nodes in any round is at most $(1+{}^{1}/_{16})n$. Note that the values $\alpha = {}^{1}/_{16}$ and $\kappa = (1+{}^{1}/_{16})$ are chosen for the sake of convenience in analysis. We require a bootstrap phase of length $B \coloneqq 2\lambda+7$ at the beginning of the algorithm. In this phase no churn occurs and this enables us to initialize our algorithm. Recall that such a bootstrap phase is a standard assumption in the area of churn-resistant overlays and is very likely necessary to construct a robust overlay.

We also assume that the system starts in an initial LDS D_0 in round 0. This assumption is made for convenience as the initial overlay can easily be constructed in the churn-free bootstrap phase using algorithms from [GHSS17, GHSW20]. Using their techniques this can be achieved in $O(\log^2 n)$ rounds with a deterministic algorithm or in $O(\log n)$ rounds w.h.p. with a randomized algorithm. Both these algorithms assume that the congestion and degree of each node is polylogarithmic and therefore, fit well into our model. We would like to remark that since our focus lies on fast reconfiguration and not on optimizing the bootstrap phase we omit the algorithmic details and the corresponding analysis. For ease of notation we will refer to round $t+B$ as t.

Let $\overline{V_t} \coloneqq V_t \cap V_{t-1}$ denote the set of all nodes except for the newly joined nodes in any given round t. Over the course of this section we distinguish between three types of nodes in each round t. Namely, the set of mature nodes $M_t \subseteq \overline{V}_t$, which are nodes that are in the network for at least $2\lambda+6$ rounds (or $2\lambda+7$ if they joined in an odd round), the set of fresh nodes $F_t \coloneqq \overline{V}_t \setminus M_t$ which are nodes that are at least one round, but less than $2\lambda+6$ rounds old, and the set of newly joined nodes i.e., $V_t \setminus \overline{V}_t$. Observe that $\overline{V}_t \coloneqq M_t \cup F_t$ and due to our choice of churn parameters, it holds $|M_t| \geq n(1-\frac{1}{16})$ and $|F_t| \leq {}^{n}/_{16}$. With a slight abuse of notation, for $x, y \in [0,1)$ and $y \leq x$, $[x \pm y]$ denotes the interval $[x-y, x+y]$.

The algorithms Alg-LDS and Alg-Random are executed concurrently. Alg-LDS ensures that all mature nodes build a routable overlay every even round and Alg-Random makes sure that all fresh nodes (which are not part of the routable overlay) stay connected to $\Theta(\log n)$ mature nodes until they mature themselves. This ensures that the mature nodes can route messages on behalf of the fresh nodes over the overlay. The main result of this chapter is stated in the following theorem.

Theorem 7.24. *Algorithms* Alg-LDS *and* Alg-Random *maintain a series of overlays $\mathcal{D}$ such that for $o(n^k)$ rounds w.h.p.,*

1. *the mature nodes form a routable series of graphs $\mathcal{D} := (D_0, H_0, D_1, \dots)$,*
2. *each fresh node is known by $\Theta(\log n)$ mature nodes, and*
3. *the congestion is $O(\log^3 n)$ per node and round.*

We would like to remark that both Alg-LDS and Alg-Random are heavily randomized and can possibly fail to create a connected and routable overlay if they are executed for too long. For example, the algorithm could fail if certain swarms are too small and/or too many messages are dropped by the routing algorithm. In these cases, the algorithm cannot construct the desired overlays and needs to be restarted along with another bootstrap phase. Given that the algorithm runs correctly w.h.p. i.e., failure happens with probability $O(n^{-k})$ for a tunable constant $k > 0$, we can only guarantee that the algorithm runs smoothly for $o(n^k)$ rounds w.h.p. until some failure happens. This follows by a simple application of the union bound. Throughout this chapter we assume that $\frac{k\log(n)}{n} \ll 1$, i.e., both k and $\log(n)$ are very small compared to n. Therefore, our algorithms only become applicable for large values of n (say $n > 10^6k$). However, as our goal is to show that the messages per node stays logarithmic in n even under heavy churn, we believe that it is justified to only consider very high values of n. In particular, we do not claim that we make the optimal choice of constants.

Alg-LDS and Alg-Random exchange four types of messages between the nodes to build a series of overlay.

1. The message Connect(id(v)) is sent by a fresh node v to advertise itself to a mature node in the overlay. It only contains v's identifier, i.e., id(v).
2. The message Create(id(v), p_v^t) is used to introduce a node v to its neighbors in D_t. The message contains the node's identifier id(v) and its position $p_v^t \in [0, 1)$ in the overlay D_t.
3. The message Join(id(v), p_v^t) is used to introduce a node v to nodes in D_{t-1}. It is routed from its origin to position p_v^t in overlay D_{t-1}. It contains the identifier of v and its position $p_v^t \in [0, 1)$ in D_t.
4. The message Token(id(v)) is sent by a mature node v to a point in the $[0, 1)$-interval picked (almost) uniformly at random. It only contains v's identifier.

Building a Routable Overlay

Alg-LDS is presented as Algorithm 6. After the bootstrap phase, the algorithm Alg-LDS creates a series of overlays $\mathcal{D} = (D_i, H_i)_{i\geq 0}$ that contain all *mature* nodes in any given round. In particular, in every even round $t = 2i$, the algorithm creates an LDS D_i which consists of all mature nodes M_t. In each odd round $t = 2i + 1$, the algorithm creates a handover graph H_i in which for each $p \in [0, 1)$ it holds that, $S_i(p)$ and $S_{i+1}(p)$ are adjacent (Lemma 7.9), where $S_i(p)$ and $S_{i+1}(p)$ are the swarms of point p in D_i and D_{i+1}, respectively.

To construct a series of overlays, in every even round, mature nodes continuously choose new positions $p^k, p^{k+1}, \dots$ for the corresponding overlays $D_k, D_{k+1}, \dots$ and use Alg-Routing to find their neighbors. The nodes pick a random position in the $[0, 1)$-interval using a uniform hash function $h : V \times \mathbb{N} \to [0, 1)$ known to all nodes. Recall that we assume that all nodes know h and the adversary does not have access to it. This hash function takes the node's id and the current round as an input and computes a random value p. Although the hash function h excludes some points in $[0, 1)$ from being picked as all values need to be encoded in $O(\log n)$ bits, we remark that this does not impact the correctness of our algorithms. We handle the values returned by h as continuous values, as it is a standard assumption [KS04, KLSY07] for the usage in overlay networks. The construction of D_i begins in round $2i - (2\lambda + 2)$. Every node $v \in \overline{V}_{2i-(2\lambda+2)}$ picks a position $p_v^i \in [0, 1)$ uniformly at random and routes its id to the targets p_v^i, $\frac{p_v^i}{2}$, and $\frac{p_v^i+1}{2}$ along their respective trajectories, e.g., $(v, x_1, \cdots, x_\lambda, p_v^i)$. This message arrives 2λ rounds later, i.e., in round $2i - 2$ at the swarm $S_{i-1}(x_\lambda)$. Then, within two rounds, ALG-LDS first constructs the handover graph H_{i-1} and then a new LDS D_i based on these positions. This ensures that in the even rounds ALG-ROUTING can perform the forwarding step and the handover in the odd rounds. Thus, ALG-LDS maintains a routable overlay.

Algorithm 6 ALG-LDS: Overlay Maintenance Algorithm.

```
Desc: In every even round $2t$ the algorithm constructs an LDS $D_t$ consisting of all nodes
    that joined the network before round $2t - (2\lambda + 2)$. In every odd round $2t + 1$ the
    algorithm performs a handover from $D_t$ to $D_{t+1}$ by constructing a helper graph $H_t$.
Note: The following code is executed by each node $u \in M_t$ every even and odd round
    respectively. The messages are handled in the given order. The last block of commands
    in each phase is executed after all messages have been handled.
    ——————————— Round $(2t)$ ———————————
 1: Upon receiving CREATE($\mathrm{id}(v), p_v^t$) from $u'$
        $D_t^u \longleftarrow D_t^u \cup \{(v, p_v^t)\}$        ▷ $u$ creates edges to these nodes.
 2: Upon receiving JOIN($\mathrm{id}(v), p_v^{t+1}$) from ALG-ROUTING
        Send JOIN($\mathrm{id}(v), p_v^{t+1}$) to all nodes $w \in D_t^u$ with
            $p_w^t \in \left[p_v^{t+1} - \frac{2c\lambda}{n}, p_v^{t+1} + \frac{2c\lambda}{n}\right]$
 3: Finally
        Perform Forwarding Step from ALG-ROUTING using edges created from $D_t$.
        $F^u \longleftarrow$ All fresh nodes known by $u$ (provided by ALG-RANDOM)
        $\forall v \in F^u \cup \{u\}$ do:
            $p_v^{t+\lambda+1} \longleftarrow h(v, t)$
            Route message JOIN($\mathrm{id}(v), p_v^{t+\lambda+1}$) to targets $\left\{p_v^{t+\lambda+1}, \frac{p_v^{t+\lambda+1}}{2}, \frac{p_v^{t+\lambda+1}+1}{2}\right\}$
            using ALG-ROUTING.
    ——————————— Round $(2t + 1)$ ———————————
 4: Upon receiving JOIN($\mathrm{id}(v), p_v^{t+1}$) from a node $u'$
        $H_t^u \longleftarrow H_t^u \cup \{(v, p_v^{t+1})\}$        ▷ $u$ creates edges to these nodes
 5: Finally:
        Perform Handover Step from ALG-ROUTING using edges created from $H_t^u$.
        $\forall (v, p_v^{t+1}) \in H_t^u$ do:
            Send CREATE($\mathrm{id}(v), p_v^{t+1}$) to all nodes $w$ such that, $(w, p_w^{t+1}) \in H_t^u$ with
            $p_w^{t+1} \in \left[p_v^{t+1} - \frac{2c\lambda}{n}, p_v^{t+1} + \frac{2c\lambda}{n}\right]$.
```

We now describe ALG-LDS in detail. The algorithm proceeds in rounds. Each mature node has two variables D_t^u and H_t^u. D_t^u stores u's neighborhood in D_t, whereas H_t^u stores the references for the handover. Both variables may be reset at the end of each round. In every even round, nodes in $\overline{V}_{2i-(2\lambda+2)}$ pick a random position $p \in [0, 1)$ and routes their id using message JOIN(id, p) to the respective target points using ALG-ROUTING i.e., step 3 of ALG-LDS. This is the position the node will occupy in D_i in round $2i$. Particularly, it sends its id to the swarms $S_{i-1}(p)$, $S_{i-1}(p/2)$, and $S_{i-1}((p+1)/2)$, thereby creating the handover graph H_{i-1} in round $2i-1$. Starting from the handover graph H_{i-1}, the overlay D_i can then be created in a single additional round through local introductions.

We will now describe the construction of the overlays H_{i-1} and D_i in detail given that the algorithm worked correctly until that round. We assume the system is currently in an even round $t = 2i - 2$ and in all previous rounds the mature nodes formed a routable overlay $\mathcal{D} := D_0, H_0, \ldots, D_{i-1}$. In other words, all messages of the form $\textsc{Join}(\mathrm{id}(v), p_v^i)$, where $\mathrm{id}(v)$ is an identifier and p_v^i is its position in D_i, are one trajectory step away from reaching their target. Then, the construction of H_{i-1} and D_i is as follows:

1. In round $2i - 2$ Alg-LDS executes step 2. This implies, all messages are routed to their target location. In particular, each message with target point p_v^i will be received by all nodes in $S_{i-1}(p_v^i)$. Recall that this is ensured by the fact that the message is sent to all nodes in $S_{i-1}(p_v^i)$ in the last step of the trajectory. Particularly, Alg-LDS additionally ensures that the message is delivered to all nodes in $\left[p_v^i \pm \frac{2c\lambda}{n}\right]$, $\left[\frac{p_v^i}{2} \pm \frac{3c\lambda}{2n}\right]$, and $\left[\frac{p_v^i+1}{2} \pm \frac{3c\lambda}{2n}\right]$ in D_{i-1}.

2. In round $2i - 1$, each node $w \in \left[p_v^i \pm \frac{2c\lambda}{n}\right] \cup \left[\frac{p_v^i}{2} \pm \frac{3c\lambda}{2n}\right] \cup \left[\frac{p_v^i+1}{2} \pm \frac{3c\lambda}{2n}\right]$ receives messages of the form $\textsc{Join}(v, p_v^i)$. This implies, each node in $u \in S_{i-1}(p_v^i) \cup S_{i-1}(\frac{p_v^i}{2}) \cup S_{i-1}(\frac{p_v^i+1}{2})$ knows every node $u' \in S_i(p_v^i)$. Therefore, as consequence of step 4 of Alg-LDS and using Definition 7.14, we claim that the construction of the handover graph H_{i-1} is complete. In the remainder of round $2i - 1$, the algorithm initiates construction of D_i.

   That is, all nodes must learn their neighbors in D_i. For this, the nodes iterate over all received messages of the form $\textsc{Join}(v, p_v^i)$ and *introduces* v to all its neighbors at $p_v^i, \frac{p_v^i}{2}$, and $\frac{p_v^i+1}{2}$ in D_i. By introduction, we mean that the neighbor's id and position is sent to v, i.e, step 5 of Alg-LDS. These messages arrive in round $2i$.

3. Finally, at the beginning of round $2i$, each node knows all its neighbors in new overlay D_i, i.e., step 1 of Alg-LDS.

Observe that correctness of Alg-LDS implies, step 1 ensures all mature nodes form the overlay D_i and can perform the forwarding step for Alg-Routing and step 4 ensures that for any $p \in [0, 1)$, Alg-Routing can handover message from D_{i-1} to D_i.

Note that after round $2i - 1$ the nodes' positions in D_{i-1} and D_i are in no relation with each other and the edges in D_{i-1} are independent of D_i. Therefore, the adversary stays oblivious of all nodes' current positions.

Furthermore, observe that our approach requires that both fresh and mature nodes send out the join requests and that all messages take exactly the same time to reach its destination. The latter is ensured by Alg-Routing. For the former, we assume that each fresh node $v \in F_{2i}$ is known by least one mature node which is part of D_i. Recall that the fresh nodes are not part of the overlay. Therefore, the mature nodes send out Join requests on behalf of each fresh node $u \in F_{2i}$ known to them. Note that each node can compute $h(\mathrm{id}(u), 2i)$, if it knows $\mathrm{id}(u)$. The ids of these nodes are stored in the variable F^u. However, this is ensured by Alg-Random and explained in the later part of the section.

7.5.1 Analysis of Alg-LDS

We now analyze the correctness of Alg-LDS. Throughout this section we assume that Alg-Random works correctly and each fresh node is connected to $\Theta(\log n)$ mature nodes at any time. This enables each node in $\overline{V_t}$ to start a join request in every even round.

Lemma 7.25. *Let $\mathcal{D}_t := (D_0, H_0 \dots, D_i)$ be routable graph until round $t = 2i$. Then it holds $\mathcal{D}_{t+2} := (D_0, H_0 \dots, D_i, H_i, D_{i+1})$ is a routable graph until round $t + 2$ w.h.p.*

Proof. W.l.o.g. we assume that the algorithm is currently in round $t = 2i$ and the overlays $\mathcal{D}_t := (D_0, H_0 \dots, D_i)$ were routable. This implies that the mature nodes know all their neighbors in D_i, all join requests (v, p_v^{i+1}) started 2λ rounds ago are delivered to the penultimate point in the trajectory, and the nodes are ready to perform the final forwarding step of the messages. Due to Lemma 7.19 we know that D_i is good, and at least $3/4$-fraction of each swarm in D_i survives until $2i + 1$. We will now show that Alg-LDS maintains the following three properties:

1. Alg-LDS successfully constructs H_i in round $2i + 1$,
2. constructs a new LDS D_{i+1} in round $2i + 2$, and
3. overlay H_i and D_{i+1} are good w.h.p.

Together, these three properties imply that $\mathcal{D}_{t+2} := (D_0, H_0 \dots, D_i, H_i, D_{i+1})$ satisfy Definition 7.16 and therefore, are series of routable overlays.

The following lemma shows that Alg-LDS constructs H_i in round $2i+1$, i.e., we show that for any $p \in [0, 1)$, every node in $S_i(p)$ knows the id of every node in $S_{i+1}(p)$. The proof essentially follows using correctness of Alg-Routing and Lemma 7.9.

Lemma 7.26 (Correctness of the Handover Construction)**.** *Let $\mathcal{D} := (D_0, H_0 \dots, D_i)$ be routable graph until round $t = 2i$. Then, in round $2i + 1$, each node in $\left[p_v^{i+1} \pm \frac{2c\lambda}{n}\right] \cup \left[\frac{p_v^{i+1}}{2} \pm \frac{3c\lambda}{2n}\right] \cup \left[\frac{p_v^{i+1}+1}{2} \pm \frac{3c\lambda}{2n}\right]$ receives Join(v, p_v^{i+1}) w.h.p. This implies that in round $2i+1$ the nodes form the Handover graph H_i.*

Proof. The proof follows directly from the correctness of Alg-Routing and the overlay's topology. Since $\mathcal{D}$ is routable until (and including) round $2i$, all messages that were started in round $2i - 2\lambda$ are correctly routed to nodes in the swarm corresponding to the penultimate step i.e., x_λ in it's trajectory w.h.p. via Alg-Routing. This includes all Join(v, p^{i+1}) messages that were started in round $2i - 2\lambda$. Due to Lemma 7.19 we know that D_i is good w.h.p. Moreover, $S_i(x_\lambda)$ are $S_i(p_v^{i+1})$ are adjacent. Thus, in round $2i+1$ every node in $\left[p_v^{i+1} \pm \frac{2c\lambda}{n}\right] \cup \left[\frac{p_v^{i+1}}{2} \pm \frac{3c\lambda}{2n}\right] \cup \left[\frac{p_v^{i+1}+1}{2} \pm \frac{3c\lambda}{2n}\right]$ received Join(v, p^{i+1}) and therefore, knows v w.h.p. Particularly, the following is true for every $p \in [0, 1)$

1. Each $u \in S_i^R(p)$ receives Join(v, p') for every $v \in S_{i+1}(p) \cup S_{i+1}(p + \frac{c\lambda}{n})$. Therefore, every $u \in S_i^R(p)$ knows $v \in S_{i+1}(p)$.
2. Each $u \in S_i^L(p)$ receives Join(v, p') for every $v \in S_{i+1}(p) \cup S_{i+1}(p - \frac{c\lambda}{n})$. Therefore, every $u \in S_i^L(p)$ knows $v \in S_{i+1}(p)$.

The proof then follows from the definition of the Handover graph H_i. □

We continue with the construction of D_{i+1}. In particular, we show that every mature node $v \in V_{2i+2}$ creates edge to:

1. the list neighbors left and right of p_v^{i+1}, and
2. the DeBruijn neighbors left and right of $\frac{p_v^{i+1}}{2}$ and $\frac{p_v^{i+1}+1}{2}$.

The following lemma show that for neighbor v and w in D_{i+1}, w.h.p. there exists at least one node u that receives the messages $\textsc{Join}(v, p_v^{t+1})$ and $\textsc{Join}(w, p_w^{t+1})$ in round $2i+1$.

Lemma 7.27. *Let v, w be any two neighbors in D_{i+1}, then w.h.p.*

$$|\{u \in G_{2i+1} \mid u \text{ receives } (v, p_v^{i+1}) \text{ and } (w, p_w^{i+1})\}| \geq \frac{14}{17}c\lambda,$$

where $G_{2i+1} \subseteq V_{2i+1}$ are the set of good nodes in the round $2i+1$.

Proof. Consider two nodes v and w with $d(p_v^{i+1}, p_w^{i+1}) \leq \frac{2c\lambda}{n}$, i.e., neighbors in D_{i+1}. W.l.o.g. we assume that p_w^{i+1} is right of p_v^{i+1} and $d(p_v^{i+1}, p_w^{i+1}) = \frac{2c\lambda}{n}$. We make these simplifying assumptions since the proof is analogous for the left and right side and any closer point can only have more nodes for the introduction.

Observe that the last step of the trajectory is traversed in round $2i$. Particularly, Alg-LDS ensures that the message $\textsc{Join}(v, p_v^{i+1})$ and $\textsc{Join}(w, p_w^{i+1})$ is forwarded to every node in the interval $\left[p_v^{i+1}, p_v^{i+1} + \frac{2c\lambda}{n}\right]$ and $\left[p_w^{i+1}, p_w^{i+1} - \frac{2c\lambda}{n}\right]$, respectively. This is possible due to the topology of D_i. This implies there is an interval $I \in [0,1)$ of length $\frac{2c\lambda}{n}$ such that all nodes belonging to this interval receive both $\textsc{Join}(v, p_v^{i+1})$ and $\textsc{Join}(w, p_w^{i+1})$ in round $2i+1$. The claim then follows using Lemma 7.10 that interval I has at least $c\lambda$ nodes w.h.p. and Lemma 7.19 that at least $\frac{14}{17}$ of those nodes in the interval I are good nodes and remain until round $2i+1$ w.h.p. □

Using similar arguments as in Lemma 7.27, one can guarantee that there are a good fraction of nodes in the intervals

$$\left[\frac{p_v^{i+1}}{2}, \frac{p_v^{i+1}}{2} + \frac{3c\lambda}{2n}\right], \left[\frac{p_v^{i+1}}{2}, \frac{p_v^{i+1}}{2} - \frac{3c\lambda}{2n}\right], \left[\frac{p_v^{i+1}+1}{2}, \frac{p_v^{i+1}+1}{2} + \frac{3c\lambda}{2n}\right], \text{ and}$$

$\left[\frac{p_v^{i+1}+1}{2}, \frac{p_v^{i+1}+1}{2} - \frac{3c\lambda}{2n}\right]$ in D_i that received $\textsc{Join}(v, p_v^{i+1})$. As a consequence of Lemma 7.27 and step 5 in Alg-LDS, D_{i+1} is constructed.

Let $U(p) \subset V_{2i-2\lambda}$ be the set of all nodes such that, for each $v \in U(p)$, it holds that $p_v^{i+1} \in \left[p \pm \frac{c\lambda}{n}\right]$. Now since nodes pick their position in D_{i+1} uniformly at random, using Lemma 7.10 we know that $|U(p)| \geq c\lambda$. Furthermore, due to our choice of churn rate, i.e., $(n/16, 2\lambda+7)$ and using Lemma 7.19, at least $\frac{14}{17}|U(p)|$ survive until round $2i+2$. Therefore, both H_i and D_{i+1} are good.

The next lemma shows that a $(2, 2\lambda+7)$-late adversary effectively reduces the adversarial churn to a randomized churn as the adversary is oblivious to which nodes belong to which swarm in any given round.

Lemma 7.28. *A $(2, 2\lambda+5)$-late adversary enables Alg-LDS construct D_{i+1} independent of D_i.*

Proof. The proof follows from the correctness of Alg-Routing and Alg-LDS. Recall that in every even round, each mature node in $v \in M_{2i-(2\lambda+2)}$ picks a position p_v^i in D_i for itself and also for the fresh nodes that are connected to them. This is the position the node v occupies in round $2i$ i.e., LDS D_i. $\textsc{Join}(v, p_v^i)$ is routed using Alg-Routing and arrives at p_v^i in round $2i-1$ i.e., H_{i-1} and then within a round Alg-LDS constructs D_i. Therefore, a $(2, 2\lambda+5)$-late adversary is oblivious to the position of node v until round $2i+2$. However, observe that ALG-LDS ensures that the node is at the position p_v^{i+1} picked uniformly at random from $[0,1)$-interval in round $2i+2$ and not known to the adversary. This implies that the position of node v in D_{i+1} is independent of D_i. □

Finally, Lemma 7.25 follows from Lemma 7.26, 7.27, and 7.28. This concludes the analysis of the maintenance algorithm. □

Handling New and Fresh Nodes

Algorithm 7 ALG-RANDOM: Handling New & Fresh Nodes.

Desc: In each round t each fresh node connects to δ mature nodes that joined at least $t-(2\lambda+4)$ rounds ago.

Note: The following code is executed by each node $u \in \overline{V}_t$ every round t.

```
———————————————— Round $t$ ————————————————
1: Set $\delta = 60k\lambda$, $\tau = 26000k\lambda$.
2: **Upon receiving** CONNECT($v$) from node $v$:
       if $\exists i \in [0, 2\delta]$ with $c_i = \perp$
           $i \longleftarrow$ number chosen uniformly from $i \in [0, 2\delta]$ with $c_i = \perp$
           $c_i \longleftarrow v$ ▷accept at most $2\delta$ connections from fresh nodes.
3: **Upon $v$ joining**:
       $(w_1, \ldots, w_\delta) \longleftarrow \delta$ distinct tokens chosen randomly from $T$
           Send CONNECT($v$) to all $w_1, \ldots, w_\delta$          ▷ $u$ sends on behalf of $v$
       $(w_1, \ldots, w_\delta) \longleftarrow \delta$ distinct tokens chosen randomly from $T$
           Send TOKEN($w_1$),...,TOKEN($w_\delta$) to $v$       ▷ Supply $v$ with tokens.
4: **Upon receiving** TOKEN($v$):
       if $u$ is mature:
           $x \longleftarrow$ uniformly chosen from $\{0, 1\}$
           if $x = 0$:
               $c_i \longleftarrow$ random element from $c_i, \ldots, c_{2\delta}$
               Send TOKEN($v$) to $c_i$ (or discard if $c_i = \perp$) ▷Supply $c_i$ with token of $v$.
           else:
               $T \longleftarrow T \cup \{v\}$                      ▷ Tokens ready to be used
       else if $u$ is fresh:
           $T \longleftarrow T \cup \{v\}$                      ▷ Tokens ready to be used.
5: **Finally**:
       if $u$ is fresh:
           $(w_1, \ldots, w_\delta) \longleftarrow \delta$ distinct tokens chosen randomly from $T$
           Send CONNECT($u$) to all $w_1, \ldots, w_\delta$
       else if $u$ is mature:
           Send TOKEN($v$) to $\tau$ random nodes using Algorithm ALG-SAMPLING.
6: $C := (c_1, \ldots, c_{2\delta}) \longleftarrow (\perp, \ldots, \perp)$                ▷ Reset ids.
7: $T \longleftarrow \emptyset$                              ▷ Drop unused tokens.
```

We now present ALG-RANDOM as Algorithm 7. This algorithm ensures that each fresh node is known by $\delta \in \Theta(\log n)$ randomly chosen *good* mature nodes each round w.h.p. It uses two types of messages—TOKEN(v) and CONNECT(v). Both messages only contain a node's id. The former is used to spread the mature nodes' ids, the latter is used to advertise a fresh node's id to mature nodes. The algorithm is executed in rounds on nodes set F_t and M_t corresponding to the fresh and mature nodes of round t, respectively. Recall that every fresh node joins the network via a node which has been in the network for at least two rounds. This enables the bootstrapping node to update the newly joined node with ids of $\Omega(\log n)$ mature nodes and also advertise the id of the newly joined node to $\Omega(\log n)$ mature nodes in the overlay in the subsequent round, i.e, line 3 of Alg-Random. In line 5, each fresh node $f \in F_t$ which is at least one round old, advertises its own id to $O(\log n)$ mature nodes in the overlay. In line 2, every such unique advertisement a mature node receives, is associated with a unique key in $[0, O(\log n)]$ and stored in its memory. In line 5, each mature node $m \in M_t$, uniformly and independently at random samples, i.e., sends Token(id(m)) to $O(\log n)$ other mature nodes in the overlay using Alg-Sampling. Each sample id of a matured node received, is either sent to a newly joined node (i.e. less than a round old and bootstrapped via m) with probability $p = 1/2$ or is with probability $1 - p$ forwarded to the id of a fresh node, if available, whose key is picked uniformly at random from $[0, O(\log n)]$, i.e., line 4. Nodes store the received tokens in the variable T. Furthermore, the array $(c_1, \dots, c_{2\delta})$ stores the assignment of tokens to ids of fresh nodes. It holds $c_i = v$ if v's id is assigned to i. If no id is assigned to i we set $c_i = \perp$. The set C consists of all $c_i \neq \perp$.

Note that at the end of round t a node forgets all its incoming connections from fresh nodes and the assignment of numbers to ids is reset.

7.5.2 Analysis of Alg-Random

In this section we show that every fresh node is able to send its id to δ mature nodes each round w.h.p. and thus stays connected to the network. In particular, we set $\delta = 60k\lambda$, $\tau = 26000k\lambda$ and prove the following lemma.

Lemma 7.29. *Assume that until round $t-1$ each fresh node was connected to at least $\frac{\delta}{2}$ good nodes each round. Then, it holds w.h.p. that each $v \in F_t$ successfully connects to $\frac{\delta}{2}$ good nodes.*

We prove the lemma in several steps. First, we show that each node receives $\Omega(\log n)$ tokens w.h.p. Recall that each mature node starts τ tokens in round $t - (2\lambda + 3)$ that reach their random destination in round t.

Lemma 7.30. *Assume Lemma 7.29 held until round $t-1$. Furthermore, let $X(\theta, v)$ denote the event that any token θ reaches v in round t. Then, the following statements hold:*

1. *Any token reaches $v \in V_t$ with the same probability, i.e.,*

$$\mathbf{Pr}[X(\theta, v) = 1] = \mathbf{Pr}\big[X(\theta', v) = 1\big].$$

2. *For each token θ, it holds*

$$\mathbf{Pr}[X(\theta, v) = 1] \geq \frac{1}{32n}.$$

Proof. We prove each of the statements separately.

1. We extend Lemma 7.23 to fresh nodes and show all token reach a node $v \in V_t$ with the same (but not necessarily uniform) probability. Consider a token θ independently of its source node and let $X(\theta, u)$ indicate that θ reaches u. Furthermore, let $Z_{t-1} \subset M_{t-1}$ be the set of all nodes that know v's id. If v receives θ in round t, then the following two events must occur:

(a) The token must be delivered to a mature node $z \in Z_{t-1}$ using Alg-Sampling. We denote this event as $Y(\theta, z)$.

(b) Given any $z \in Z_{t-1}$ received θ, it must forward it to v in round $t-1$. We denote this event as $Y^z(\theta, v)$.

From Lemma 7.23 we know that for any two tokens θ and θ',

$$\mathbf{Pr}[Y(\theta, z) = 1] = \mathbf{Pr}\big[Y(\theta', z) = 1\big].$$

Note that for two different $z, z' \in Z_{t-1}$, the probabilities $\mathbf{Pr}[Y(\theta, z)]$ and $\mathbf{Pr}[Y(\theta, z')]$ may differ. Moreover, due to Alg-Random for any arbitrary token θ received by $z \in Z_{t-1}$,

$$\mathbf{Pr}[Y^z(\theta, v) = 1 \mid Y(\theta, z) = 1] = \frac{1}{4\delta}.$$

To finalize the proof, consider two nodes $u, w \in V_{t-(2\lambda+3)}$ and let θ^u and θ^w be tokens sent by v and w, respectively. Then,

$$\begin{aligned}
\mathbf{Pr}[X(\theta^u, v) = 1] &= \sum_{z \in Z_{t-1}} \mathbf{Pr}[Y(\theta^u, z) = 1 \cap Y^z(\theta^u, v) = 1] \\
&= \sum_{z \in Z_{t-1}} \mathbf{Pr}[Y(\theta^u, z) = 1] \cdot \mathbf{Pr}[Y^z(\theta^u, v) = 1 \mid Y(\theta^u, z) = 1] \\
&= \sum_{z \in Z_{t-1}} \mathbf{Pr}[Y(\theta^w, z) = 1] \cdot \mathbf{Pr}[Y^z(\theta^w, v) = 1 \mid Y(\theta^w, z) = 1] \\
&= \mathbf{Pr}[X(\theta^w, v) = 1].
\end{aligned}$$

Here, the first equality is due to the law of total probability and second equality is due to Lemma 7.23 and the fact that each mature node z forwards a token to a fresh node with probability exactly $\frac{1}{4\delta}$.

2. The fact that $\mathbf{Pr}[X(v, w) = 1] \in \Omega(\frac{1}{n})$ then follows from three facts:

(a) A token reaches a given mature node with probability at least $\frac{1}{4n}$. This follows directly from Lemma 7.23.

(b) Each fresh node is connected to at least $\frac{\delta}{2}$ mature nodes w.h.p. This follows because we assume that Lemma 7.29 holds true in round $t-1$.

(c) A mature node forwards a token to a connected node with probability $\frac{1}{4\delta}$. Recall that a mature node picks a token from the set of available tokens in T with probability $\frac{1}{2\delta}$ and then forwards it to a newly connected node with probability $\frac{1}{2}$.

Combining these three facts yields the result. Formally:

$$\begin{aligned}
\mathbf{Pr}\big[X(\theta, v) = 1 \mid \big|Z_{t-1}\big| \geq \delta/2\big] &= \sum_{z \in Z_{t-1}} \mathbf{Pr}[Y(\theta, z) = 1] \cdot \mathbf{Pr}[Y^z(\theta, v) = 1 \mid Y(\theta, z) = 1] \\
&\geq \sum_{z \in Z_{t-1}} \frac{1}{4n} \frac{1}{4\delta} \\
&\geq \frac{\delta}{2} \frac{1}{4n} \frac{1}{4\delta} \\
&\geq \frac{1}{32n}.
\end{aligned}$$

□

Lemma 7.31. *Assume Lemma 7.29 held until round $t-1$. Furthermore, let $X(u,v)$ denote the event that any token sent by u reaches v in round t. Then, for $\tau = 26000k\lambda$ the following statements hold:*

1. *Each node sends at least one token to node $v \in V_t$ with the same probability, i.e., $\forall u, w \in V_{t-(2\lambda+3)}$*
$$\mathbf{Pr}[X(u,v)=1] = \mathbf{Pr}[X(w,v)=1].$$
2. *For each $u \in V_{t-(2\lambda+3)}$ it holds*
$$\mathbf{Pr}[X(u,v)=1] \geq \frac{\tau}{33n}.$$

Proof. We prove the statements separately.

1. Recall that Alg-Random ensures both u and w send τ tokens. We denote these tokens as $\theta_1^u, \dots, \theta_\tau^u$ and $\theta_1^w, \dots, \theta_\tau^w$. Let $X(\theta, v)$ be defined as in Lemma 7.30. The probability that any of these tokens reach v is given by:
$$\mathbf{Pr}\left[\bigcup_{i=1,\dots,\tau} X(\theta_i^u, v) = 1\right] = 1 - \mathbf{Pr}\left[\bigcap_{i=1,\dots,\tau} X(\theta_i^u, v) = 0\right].$$
Since all these tokens are independent, it holds that:
$$\mathbf{Pr}\left[\bigcap_{i=1,\dots,\tau} X(\theta_i^u, v) = 0\right] = \prod_{i=1,\dots,\tau} \left(1 - \mathbf{Pr}[X(\theta_i^u, v) = 1]\right).$$
The same holds respectively for $\mathbf{Pr}\left[\bigcap_{i=1,\dots,\tau} X(\theta_i^w, v) = 0\right]$. Putting these observations together we get that,
$$\begin{aligned}\mathbf{Pr}[X(u,v)=1] &= \mathbf{Pr}\left[\bigcup_{i=1,\dots,\tau} X(\theta_i^u, v) = 1\right] = 1 - \mathbf{Pr}\left[\bigcap_{i=1,\dots,\tau} X(\theta_i^u, v) = 0\right]\\ &= 1 - \prod_{i=1,\dots,\tau} \left(1 - \mathbf{Pr}[X(\theta_i^u, v) = 1]\right)\\ &= 1 - \prod_{i=1,\dots,\tau} \left(1 - \mathbf{Pr}[X(\theta_i^w, v) = 1]\right)\\ &= 1 - \mathbf{Pr}\left[\bigcap_{i=1,\dots,\tau} X(\theta_i^w, v) = 0\right] = \mathbf{Pr}\left[\bigcup_{i=1,\dots,\tau} X(\theta_i^w, v) = 1\right]\\ &= \mathbf{Pr}[X(w,v)=1].\end{aligned}$$
This was to be shown.
2. For any pair of v and u, the probability is lower bounded by
$$\begin{aligned}\mathbf{Pr}[X(u,v)=1] &= 1 - \mathbf{Pr}[X(u,v)=0]\\ &= 1 - \prod_{i=1,\dots,\tau} \left(1 - \mathbf{Pr}[X(\theta_i^u, v) = 1]\right)\end{aligned}$$

$$
\begin{aligned}
&= 1 - \left(1 - \frac{1}{32n}\right)^{\tau} \\
&\geq 1 - \exp\left(-\frac{\tau}{32n}\right) \\
&\geq 1 - \left(1 - \frac{\tau}{32n} + \left(\frac{\tau}{32n}\right)^2\right) \\
&\geq \frac{\tau}{32n} - \left(\frac{\tau}{32n}\right)^2 \\
&\geq \frac{\tau}{33n}
\end{aligned}
$$

where for the first inequality we use that for $m \geq 1$ and $|x| \leq m$,

$$\left(1 + \frac{x}{m}\right)^m \leq \exp(x),$$

for the second inequality we use the fact that for all $x \leq 1$,

$$\exp(x) \leq 1 + x + x^2,$$

which are standard exponential function inequalities, and the last inequality follows from fact that $\tau \in O(\log n)$ and thus $\frac{\tau}{32n}$ can be made arbitrarily small for a big enough n.

□

Together with our assumptions on the churn rate, we state the following lemma.

Lemma 7.32. *Let $c \geq 280k$. Let each mature node start $\tau \geq 20c\lambda$ token, then each fresh node receives at least $\frac{\tau}{100}$ distinct token with probability at least $1 - \frac{1}{n^k}$.*

Proof. Recall that at least $\frac{15}{16}n$ mature nodes in round $t-(2\lambda+3)$ that start τ tokens each. Hence, the minimal number of nodes that start tokens is at least $K := \frac{15n}{16}$. Fix a node v and let $X_1, \dots, X_K$ be the indicator variables that a nodes has a token that reaches v. Then the expected number of distinct tokens received by node v is given by,

$$\mathbf{E}\left[\sum_{i=1}^{K} X_i\right] \geq \sum_{i=1}^{K} \frac{\tau}{33n} \geq \frac{15\tau n}{16 \cdot 33 \cdot n} \geq \frac{\tau}{50},$$

where we use $\mathbf{Pr}[X_i = 1] \geq \frac{\tau}{33n}$ due to Lemma 7.31. Given that all mature nodes send their tokens independent of one another, the Chernoff Bound is applicable and the lemma follows for a big enough c. In particular, it holds for $c \geq 280k$,

$$
\begin{aligned}
\mathbf{Pr}\left[X \leq \frac{\tau}{100}\right] &= \mathbf{Pr}\left[X \leq (1 - 1/2)\frac{\tau}{50}\right] \\
&\leq \exp\left(-\frac{\tau}{4 \cdot 50 \cdot 3}\right) \\
&\leq \exp\left(-k\lambda\right) = n^{-k}.
\end{aligned}
$$

□

Therefore, choosing τ bigger than $20c\lambda$ ensures that each fresh node receives roughly $\Omega(\log n)$ distinct tokens w.h.p., which it can then use to advertise itself and the new nodes connected to it. Observe that for each fresh node, Alg-Random connects to δ many mature nodes. Lemma 7.32 then gives us an estimate on the appropriate number of tokens τ that each mature node sends, so that each fresh node has sufficient number of tokens. Then, in the following we can conveniently assume that each fresh node sends a connection request to δ mature nodes. However, these requests can still fail for two reasons:

1. First, the id of the token used for the connections belongs to a node that has already been churned out.
2. Second, the target has received more than 2δ connection requests and refuses the connection.

We begin by showing that only a small fraction of connection request are sent to churned out nodes. In the following lemma we say a node v is θ-good in round t if and only if $v \in V_{t-(2\lambda+3)} \cap V_{t+2}$.

Lemma 7.33. *Suppose that $\tau \geq 26000k\lambda$ and $\delta \geq 60k\lambda$, then each fresh node has at least $\frac{\delta}{2}$ connections to good nodes with probability at least $1 - \frac{3}{n^k}$.*

Proof. Fix a node $v \in V_t$ that advertises (i.e., connects to a mature node from which it received a token) itself or a newly joined node. Let the random variable Y denote the number of all distinct tokens that v received. We have already established that the number of distinct tokens that a node receives can be subjected to the Chernoff bound. The same holds for the number of tokens sent by θ-good nodes. Due to its lateness the adversary cannot anticipate where a node will send its tokens. Thus, the tokens of θ-good are randomly spread to the fresh nodes. Let the random variable G denote the number of θ-good nodes that sent its token to v. Then, G is the sum independent binary random variables i.e., for each good node $w \in V_{t-(2\lambda+3) \cap V_{t+2}}$, let $G_w \in \{0,1\}$ be the indicator for the event that w sends at least one token with its identifier to v. Then, it holds that $G := \sum_{w \in V_{t-(2\lambda+3)} \cap V_{t+2}} G_w$ and all G_w's are independent.

Recall that at least $n\left(1 - \frac{1}{16}\right)$ and at most $n\left(1 + \frac{1}{16}\right)$ nodes started tokens $2\lambda + 3$ rounds ago. Since at most $n/16$ nodes that started a token are churned out until round $t+2$, it holds that $\left(\frac{15}{16}\right)^2 n$ is a lower bound on the number of θ-good nodes that started tokens $2\lambda + 3$ rounds ago.

Fix a mature node w that sent a token $2\lambda + 3$ rounds ago. From Lemma 7.31 we know that $p \in [\frac{\tau}{33n}, \frac{\tau}{n}]$ is the probability that at least one token from w reaches v. Then,

$$\mathbf{E}[G] \geq p \left(\frac{15}{16}\right)^2 n.$$

Furthermore, let $\left(\widetilde{Y}_i\right)_{i \in [\frac{17n}{16}]}$ be $\{0,1\}$ random variables such that,

$$\mathbf{Pr}\left[\widetilde{Y}_i = 1\right] = \frac{\tau}{n}$$

and

$$\widetilde{Y} := \sum_{i=1}^{n(1+\frac{1}{16})} \widetilde{Y}_i.$$

Then,

$$\mathbf{E}\left[\sum_{i=1}^{n(1+\frac{1}{16})} \widetilde{Y}_i\right] = \tau\left(1+\frac{1}{16}\right).$$

We bound from above and below Y and G, respectively. Assume $pn \geq 768k\lambda$, the Chernoff bound gives us that,

$$\begin{aligned}\mathbf{Pr}\left[Y \geq \frac{9}{8}pn\right] &\leq \mathbf{Pr}\left[\widetilde{Y} \geq (1+1/17)(1+1/16)\tau\right] \\ &\leq \exp\left(-\frac{\tau}{2\cdot 17\cdot 16}\right) \\ &\leq \exp(-k\lambda) \\ &\leq n^{-k}. \end{aligned} \tag{7.9}$$

$$\begin{aligned}\mathbf{Pr}\left[G \leq \frac{13}{16}pn\right] &\leq \mathbf{Pr}\left[G \leq (1-1/15)\left(\frac{15}{16}\right)^2 pn\right] \\ &\leq \exp\left(-\frac{15^2\cdot pn}{3\cdot 16^2\cdot 15^2}\right) \\ &\leq \exp(-k\lambda) \\ &\leq n^{-k}. \end{aligned} \tag{7.10}$$

Therefore, it remains to show that we can choose pn big enough for (7.9) and (7.10) to hold. Recall that $\frac{\tau}{n} \geq p \geq \frac{\tau}{33n}$, then for $\tau \geq 26000k\lambda$, we have that $pn \geq 768k\lambda$.

Now, we condition on (7.9) and (7.10) being false and denote this event as $\mathcal{Z}$. Node v picks $\delta' = \min\{\delta, Y\}$ of the received tokens at random without replacement. Let $X_1, \ldots, X_{\delta'}$ be binary random variables such that X_j denotes if j^{th} advertisement by v is successful, i.e., its identifier is advertised to a good mature node. Moreover, $(X_i)_{i\in\{1,\ldots,\delta'\}}$ are negatively associated [JDP83]. Then, the random variable $X = \sum_i X_i$ denotes the number of tokens drawn corresponding to θ-good nodes. Observe that X is hypergeometrically distributed in Y, G and δ', i.e., the outcome of X depends on the overall number of tokens that v received and the number of identifiers of θ-good nodes. Therefore,

$$\begin{aligned}\mathbf{E}[X \mid \mathcal{Z}] &\geq \sum_{g=\frac{13}{16}pn}^{n(1+\frac{1}{16})} \sum_{y=0}^{\frac{9}{8}pn} \mathbf{Pr}[G=g, Y=y \mid \mathcal{Z}]\cdot\delta'\frac{g}{y} \\ &\geq \sum_{g=\frac{13}{16}pn}^{n(1+\frac{1}{16})} \sum_{y=0}^{\frac{9}{8}pn} \mathbf{Pr}[G=g, Y=y \mid \mathcal{Z}]\cdot\delta'\frac{\frac{13}{16}pn}{\frac{9}{8}pn} \\ &= \delta'\frac{\frac{13}{16}pn}{\frac{9}{8}pn}\sum_{g=\frac{13}{16}pn}^{n(1+\frac{1}{16})} \sum_{y=0}^{\frac{9}{8}pn} \mathbf{Pr}[G=g, Y=y \mid \mathcal{Z}] \\ &= \delta'\frac{\frac{13}{16}pn}{\frac{9}{8}pn} \\ &= \delta'\frac{13}{18}.\end{aligned}$$

The random variable X is negatively associated [JDP83]. Note that for our choice of τ, Lemma 7.32 implies that node v receive at least $\frac{\tau}{100}$ distinct tokens. Then, choosing δ' large enough, in particular, by choosing $\delta \geq 60k\lambda$ we get,

$$\begin{aligned}\mathbf{Pr}\left[X \leq \frac{\delta}{2} \mid \mathcal{Z}\right] &\leq \mathbf{Pr}\left[X \leq (1 - {}^{5}\!/\!{}_{18})\frac{13}{18}\delta\right] \\ &\leq \exp\left(-\frac{5^2 \cdot 13\delta}{3 \cdot 18^3}\right) \leq \exp\left(-\frac{\delta}{60}\right) \\ &\leq \exp\left(-k\lambda\right) = n^{-k}.\end{aligned}$$

Now since $\mathcal{Z}$ holds w.h.p., we get,

$$\begin{aligned}\mathbf{Pr}\left[X \leq \frac{\delta}{2}\right] &= \mathbf{Pr}[\mathcal{Z}] \cdot \mathbf{Pr}\left[X \leq \frac{\delta}{2} \mid \mathcal{Z}\right] + \mathbf{Pr}[\neg\mathcal{Z}] \cdot \mathbf{Pr}\left[X \leq \frac{\delta}{2} \mid \neg\mathcal{Z}\right] \\ &\leq \mathbf{Pr}[\mathcal{Z}] \cdot \mathbf{Pr}\left[X \leq \frac{\delta}{2} \mid \mathcal{Z}\right] + \mathbf{Pr}[\neg\mathcal{Z}] \cdot 1 \\ &= \mathbf{Pr}[\mathcal{Z}] \cdot \mathbf{Pr}\left[X \leq \frac{\delta}{2} \mid \mathcal{Z}\right] + \mathbf{Pr}\left[\left(Y \geq \frac{9}{8}pn\right) \cup \left(G \leq \frac{13}{16}pn\right)\right] \\ &\leq \left(1 - \frac{1}{n^k}\right)\frac{1}{n^k} + \frac{2}{n^k} \leq \frac{3}{n^k}.\end{aligned}$$

□

The next lemma shows that for our lateness parameters, the adversary cannot anticipate the destination of a token.

Lemma 7.34. *A $(2, 2\lambda + 7)$-late adversary enables* Alg-Random *ensure that every fresh node is connected to $\frac{\delta}{2}$ mature nodes in each round.*

Proof. The proof follows using the correctness of Alg-Routing and Alg-Random. Note that the adversary is oblivious of the random edges, as they only persist for 2 rounds. Each mature node disseminates tokens to random positions in the $[0, 1)$ interval. The tokens arrive at their target node for being sampled after $2\lambda + 2$ rounds. The mature nodes that receive the tokens forward them to fresh nodes which in turn connect to the mature nodes to stay connected in the network until they mature themselves. The fresh nodes then receive new tokens from these connections. The entire process takes $2\lambda + 5$ rounds in total. Therefore, in any given round t, a $(2, 2\lambda + 7)$-late adversary is oblivious to any communication between the fresh nodes and mature nodes, since all connections established until round t are already defunct, i.e., the adversary is unable to anticipate which tokens reach a given fresh node. This in turn enables Alg-Random maintain the invariant every round. □

Lemma 7.34 and the churn parameters ensure that there exists a constant size set of good nodes $G_t \coloneqq V_{t-(2\lambda+3)} \cap V_{t+2}$ that send tokens in round $t - (2\lambda + 3)$ and are not churned out until round V_{t+2}. Therefore, if a node receives enough tokens of good nodes, it can successfully advertise its identifier w.h.p.

It remains to show that at most 2δ fresh nodes connect to a mature node in any given round. We first analyze the expected number of incoming connections.

Lemma 7.35. *Let $A(v, u)$ denote the event that v advertises itself to u. Then, $\forall u, w \in V_t$ whose respective tokens reach v it holds that,*

$$\mathbf{Pr}[A(v, u) = 1] = \mathbf{Pr}[A(v, w) = 1].$$

Proof. Fix a fresh node v. We analyze the process of node v advertising itself to a mature node u in two stages:

1. The token from a mature node u must reach v.
2. v picks the token associated with u for the advertisement.

Let binary random variables $Y(u,v)$ and $Y(w,v)$ denote the event that tokens of u and w reached v, respectively. By Lemma 7.30 we know that these events have the same probability, i.e.,

$$\mathbf{Pr}[Y(u,v)=1]=\mathbf{Pr}[Y(w,v)=1].$$

Next, observe that the actual choice of the nodes to which the advertisements are sent to, only depends on the number of distinct tokens available with v. In particular, given that a node received ℓ distinct tokens, the probability for one of these tokens to be used is $\min\left\{1,\frac{\delta}{\ell}\right\}$, i.e.,

$$\mathbf{Pr}[A(v,u)\mid Y(u,v)=1]=\min\left\{1,\frac{\delta}{\ell}\right\}.$$

This follows from the fact that we draw (up to) δ tokens at random or all tokens if we received less than δ. Thus, we draw without replacements and observe a multivariate hypergeometric distribution [Sch00, p. 44].

Let the random variable N_v denote the number of distinct tokens received by v in round $t+(2\lambda+3)$. Since all mature nodes send (at least) one token to v independently and with same probability $p:=\mathbf{Pr}[Y(u,v)=1]$, the value of N_v only depends on the number of nodes we observe, i.e.,

$$\begin{aligned}\mathbf{Pr}[N_v=\ell]&=\sum_{\substack{S\subset M_t\\|S|=\ell}}\mathbf{Pr}\left[\bigcap_{u\in S}Y(u,v)=1\right]\cdot\mathbf{Pr}\left[\bigcap_{u'\notin S}Y(u',v)=0\right]\\&=\sum_{\substack{S\subset M_t\\|S|=\ell}}\prod_{u\in S}\mathbf{Pr}[Y(u,v)=1]\cdot\prod_{u'\notin S}\mathbf{Pr}\big[Y(u',v)=0\big]\\&=\binom{|M_t|}{\ell}p^{\ell}(1-p)^{n-\ell}.\end{aligned}$$

Thus, if we condition on the fact that v already received a token of a certain node, the probability that this node receives $\ell-1$ additional tokens from different nodes remains the same, i.e.,

$$\begin{aligned}\mathbf{Pr}[N_v=\ell\mid Y(u,v)=1]&=\sum_{\substack{S\subset M_t\setminus\{u\}\\|S|=\ell-1}}\mathbf{Pr}\left[\bigcap_{u'\in S}Y(u',v)=1\right]\cdot\mathbf{Pr}\left[\bigcap_{u''\notin S}Y(u'',v)=0\right]\\&=\binom{|M_t|-1}{\ell-1}\cdot p^{\ell-1}\cdot(1-p)^{|M_t|-(\ell-1)}\\&=\sum_{\substack{S\subset M_t\setminus\{w\}\\|S|=\ell-1}}\mathbf{Pr}\left[\bigcap_{u'\in S}Y(u',v)=1\right]\cdot\mathbf{Pr}\left[\bigcap_{u''\notin S}Y(u'',v)=0\right]\\&=\mathbf{Pr}[N_v=\ell\mid Y(w,v)=1].\end{aligned}$$

Thus, considering all possible outcomes for N_v we get,

$$\begin{aligned}
\mathbf{Pr}[A(v,u)=1] &= \mathbf{Pr}[Y(u,v)=1]\cdot\mathbf{Pr}[A(v,u)\mid Y(u,v)=1]\\
&= \mathbf{Pr}[Y(u,v)=1]\cdot\left(\sum_{\ell=1}^{|M_t|}\mathbf{Pr}[N_v=\ell\mid Y(u,v)=1]\cdot\min\left\{1,\frac{\delta}{\ell}\right\}\right)\\
&= \mathbf{Pr}[Y(w,v)=1]\cdot\left(\sum_{\ell=1}^{|M_t|}\mathbf{Pr}[N_v=\ell\mid Y(w,v)=1]\cdot\min\left\{1,\frac{\delta}{\ell}\right\}\right)\\
&= \mathbf{Pr}[Y(w,v)=1]\cdot\mathbf{Pr}[A(v,w)\mid Y(w,v)=1]\\
&= \mathbf{Pr}[A(v,w)=1].
\end{aligned}$$

□

It remains to show that in any given round, a mature node does not receive too many advertisements. The following lemma bounds the expected number of incoming connections.

Lemma 7.36. *Fix a mature node w that started tokens $2\lambda+3$ rounds ago. Let X be a random variable that denotes the number of fresh nodes that advertise themselves to w. It holds:*

$$\mathbf{E}[X]\le\delta.$$

Proof. For a fixed node $v\in F_t$ and $w\in M_t$, let $A(v,w)$ be the binary random variable that denotes if v connects to w. Let $A_v := \sum_{w\in M_t}A(v,w)$ be the random variable that counts the total number of advertisements sent out by v. Since each node picks at most δ tokens, it must hold that,

$$\mathbf{E}[A_v]\le\delta.$$

Furthermore, linearity of expectation gives us that,

$$\mathbf{E}[A_v]=\sum_{w\in V}\mathbf{E}[A(v,w)]=\sum_{w\in V}\mathbf{Pr}[A(v,w)=1].$$

Using Lemma 7.35 we also know that for all $w,u\in M_t$,

$$\mathbf{Pr}[A(v,w)=1]=\mathbf{Pr}[A(v,u)=1].$$

Now we combine our observations to bound from above $\mathbf{Pr}[A(v,w)=1]$.

$$\begin{aligned}
\delta\ge\mathbf{E}[A_v] &= \sum_{w\in V}\mathbf{E}[A(v,w)]\\
&= \sum_{w\in V}\mathbf{Pr}[A(v,w)=1]\\
&= |M_t|\cdot\mathbf{Pr}[A(v,w)=1].
\end{aligned}$$

Therefore,

$$\mathbf{Pr}[A(v,w)=1]\le\frac{\delta}{|M_t|}.$$

Let X be a random variable that counts the number of incoming connections to the node w. Then, due to our choice of churn parameters α and κ,

$$\mathbf{E}[X] = \sum_{v \in F_t} \mathbf{E}[A(v,w)] \leq |F_t| \cdot \frac{\delta}{|M_t|} \leq \delta.$$

□

Using Lemma 7.36 the following lemma bounds from above the total number of advertisements received by a mature node in any given round.

Lemma 7.37. *Each mature node receives at most 2δ advertisements from fresh nodes w.h.p.*

Proof. Fix a mature node $w \in M_t$ and let the random variable $X_v \in \{0,1\}$ denote whether $v \in F_t$ sends an advertisement to w. Then, the random variable $X := \sum_{v \in F_t} X_v$ denotes the total number of fresh nodes that advertise themselves to w. Furthermore, let $A, B \subset F_t$ be two disjoint subsets of fresh nodes and define random variables $X_A := \prod_{v \in A} X_v$ and $X_B := \prod_{v \in B} X_v$. We prove the following:

a) $\mathbf{E}[X_A \cdot X_B] \leq \mathbf{E}[X_A] \cdot \mathbf{E}[X_B]$.

b) $(X_v)_{v \in F_t}$ are negatively correlated.

As a consequence of (b) and Lemma 7.36, we apply Chernoff bound on the random variable X to prove the lemma.

To prove (a), observe that,

$$\begin{aligned} &\mathbf{E}[X_A \cdot X_B] \leq \mathbf{E}[X_A] \cdot \mathbf{E}[X_B] \\ \iff &\mathbf{E}[X_A \cdot X_B] - \mathbf{E}[X_A] \cdot \mathbf{E}[X_B] \leq 0 \\ \iff &\mathbf{Cov}(X_A, X_B) \leq 0. \end{aligned}$$

Therefore, it is sufficient to prove that $\mathbf{Cov}(X_A, X_B) \leq 0$.

Consider the following scenario in which we denote all tokens sent by w as red tokens and the tokens sent by all other nodes as blue tokens. Let $R_v \in \{0,1\}$ be the random variable that indicates that $v \in F_t$ received a red token. Furthermore, let B_v count the number of distinct blue tokens not received by v. Thus, the set $Y := (R_v \cup B_v)_{v \in F_t}$ completely characterizes the distribution of tokens to nodes.

Using the closure properties of NA, one can show that the set $Y := (R_v \cup B_v)_{v \in F_t}$ is NA.

For each red token θ_i^w with $i \in [\tau]$, define indicator random variables $(Z(\theta_i^w, v))_{v \in F_t}$ such that, $Z(\theta_i^w, v) = 1$ if θ_i^w reaches v, and 0 otherwise. Observe that for any fixed token θ^w belonging to node w, there is at most one $v \in F_t$ such that, $X(\theta^w, v) = 1$ and all others are 0, as a token is delivered to at most one node. Therefore, Lemma 7.6 gives us that for each token θ_i^w with $i \in [\tau]$, the set $(Z(\theta_i^w, v))_{v \in V_t}$ is negatively associated. Using Proposition 7.4 we can conclude that the set of random variables $(Z(\theta_i^w, v))_{v \in V_t, i \in [\tau]}$ are negatively associated. The variable R_v is now defined as follows:

$$R_v := \begin{cases} 1 & \text{if } \sum_{i=1}^{\tau} Z(\theta_i^w, v) > 0 \\ 0 & \text{else} \end{cases}$$

Since R_v is monotonically increasing in $\sum_{i=1}^{\tau} Z(\theta_i^w, v)$, using Proposition 7.5 we remark that the random variables $(R_v)_{v \in F_t}$ are negatively associated.

For the blue tokens, we adapt to the fact that we count the distinct identifiers that were not delivered to a node v. Therefore, we define $Z'(\theta, v) := 1 - Z(\theta, v)$ as a monotone decreasing function. In particular, we set $Z'(\theta, v) = 0$ if θ reaches v, and 1 otherwise. Since for any $u \in V_t \setminus \{w\}$, the random variables $(Z(\theta_i^u, v))_{v \in V_t, i \in [\tau]}$ are negatively associated, so are $(Z'(\theta_i^u, v))_{v \in V_t, i \in [\tau], u \in V_t \setminus \{w\}}$. For each node $u \in V_t \setminus \{w\}$ define B_v^u to be binary variable indicating if any token of u reached v. Thus, B_v^u equals 1 if all τ tokens of u missed v. That is,

$$B_v^u := \begin{cases} 1 & \text{if } \sum_{i=1}^{\tau} Z'(\theta_i^u, v) = \tau \\ 0 & \text{else} \end{cases}$$

Again, since B_v^u is an increasing function on $\sum_{i=1}^{\tau} Z'(\theta_i^u, v)$ and $B_v := \sum_{u \in V_t} B_v^u$, using Proposition 7.4 and Proposition 7.5 we remark that the random variables $(B_v)_{v \in F_t}$ and therefore, Y are negatively associated.

Furthermore, observe that for any fixed distribution of tokens to fresh nodes, i.e., for any given realization of Y, the expected value $\mathbf{E}[X_v \mid R_v, B_v]$ of each X_v only depends on the variables R_v and B_v. In particular, each $\mathbf{E}[X_v \mid R_v, B_v]$ monotonically rises in both R_v and B_v and is given by,

$$\begin{aligned} \mathbf{E}[X_v \mid Y = (R_v, B_v)] &= \mathbf{Pr}[X_v \mid Y = (R_v, B_v)] \\ &= \frac{R_v}{R_v + (m - B_v)}, \end{aligned}$$

where m is the total number of nodes in round $t - (2\lambda + 3)$ that sent blue tokens.

Now observe that the random variables $(X_v)_{v \in F_t}$ are mutually independent given Y. Thus, given two disjoint subsets $A, B \subset F_t$, the functions $f(Y_A) := \mathbf{E}[X_A|Y]$ and $g(Y_B) := \mathbf{E}[X_B|Y]$ monotonically rise in disjoint subsets $Y_A = (R_A \cup B_A)$ and $Y_B := (R_B \cup B_B)$, respectively. That is,

$$f(Y_A) := \mathbf{E}[X_A \mid Y] = \mathbf{E}\left[\prod_{v \in A} X_v \mid Y\right] = \prod_{v \in A} \mathbf{E}[X_v \mid Y = (R_v, B_v)] = \prod_{v \in A} \frac{R_v}{R_v + (m - B_v)}.$$

By the law of total covariance (c.f. [Rud09]), it holds that,

$$\mathbf{Cov}(X_A, X_B) = \mathbf{E}[\mathbf{Cov}(X_A, X_B) \mid Y] + \mathbf{Cov}(\mathbf{E}[X_A \mid Y], \mathbf{E}[X_B \mid Y])$$

Note that $\mathbf{E}[\mathbf{Cov}(X_A, X_B) \mid Y] = 0$, since $(X_v)_{v \in F_t}$ are independent given Y [Par17]. Thus, it holds:

$$\mathbf{Cov}(X_A, X_B) = \mathbf{Cov}(\mathbf{E}[X_A \mid Y], \mathbf{E}[X_B \mid Y]).$$

It remains to show that this term is smaller than 0. Recall that $f(Y_A) := \mathbf{E}[X_A|Y]$ and $g(Y_B) := \mathbf{E}[X_B|Y]$ are monotonically increasing function on disjoint subsets Y_A and Y_B, respectively. Furthermore, we showed that the random variables Y are negatively associated. Therefore, using Proposition 7.4 its holds that $f(Y_A)$ and $g(Y_B)$ are negatively associated. Then, Definition 7.2 implies,

$$\mathbf{Cov}(f(Y_A), g(Y_B)) \leq 0.$$

Therefore,

$$\mathbf{Cov}(\mathbf{E}[X_A|Y], \mathbf{E}[X_B|Y]) = \mathbf{Cov}(f(Y_A), g(Y_B)) \leq 0.$$

The proof of (b) follows immediately from (a). Pick any set $S \subseteq V_t$. Given that (a) is true for any disjoint subsets of V_t, using induction we get,

$$\mathbf{E}\left[\prod_{v \in S} X_v\right] \leq \prod_{v \in S} \mathbf{E}[X_v].$$

This implies that $(X_v)_{v \in F_t}$ are negatively correlated [Sch00].

Using Lemma 7.36, we know that $\mathbf{E}[X] \leq \delta$. Therefore,

$$\mathbf{Pr}[X > 2\delta] = \mathbf{Pr}\left[X > \frac{2\delta}{\mathbf{E}[X]}\mathbf{E}[X]\right] = \mathbf{Pr}\left[X > \left(1 + \left(\frac{2\delta}{\mathbf{E}[X]} - 1\right)\right)\mathbf{E}[X]\right].$$

Note that $\frac{\delta}{\mathbf{E}[X]} \geq 1$. Therefore, we use the third bound from Lemma 7.7 for parameters bigger than 1. Thus, for our choice of δ, in particular for all $\delta \geq 3k\lambda$ we get that,

$$\begin{aligned}
\mathbf{Pr}[X > 2\delta] &\leq \exp\left(-\frac{\left(\frac{2\delta}{\mathbf{E}[X]} - 1\right)\mathbf{E}[X]}{3}\right) \\
&= \exp\left(-\frac{(2\delta - \mathbf{E}[X])}{3}\right) \\
&\leq \exp\left(-\frac{(2\delta - \delta)}{3}\right) \\
&= \exp\left(-\frac{\delta}{3}\right) \\
&\leq \exp\left(-k\lambda\right) \\
&= \frac{1}{n^k}.
\end{aligned}$$

□

7.5.3 Congestion

We now bound the total congestion per node per round due to Alg-LDS and Alg-Random.

Lemma 7.38. *Algorithms Alg-LDS and Alg-Random have congestion of $O(\log^3 n)$ per node and round w.h.p.*

Proof. We observe the number of messages due to Alg-LDS and Alg-Random invoking Alg-Routing. We observe the two subroutines Alg-LDS and Alg-Random separately.

1. In Alg-LDS each round every mature starts three routing requests for itself and three routing requests on behalf of each fresh node connected to it. Since there are at most 2δ fresh nodes connected to a mature node w.h.p., a given mature node starts $O(\log n)$ routing requests.
2. In Alg-Random each round every mature starts $O(\log n)$ tokens per round. Each token corresponds to one routing request.

Using Lemma 7.18 that each routing takes $O(\log n)$ rounds and Lemma 7.22 that for each routing request Alg-Routing has a congestion of $O(\log n)$ per round, Alg-LDS and Alg-Random together have congestion $O(\log^3 n)$ per round.

Now we observe the remaining operations performed each round.

1. Observe that each node in the swarm only receives message either from nodes in its own swarm or its two neighboring swarms. For any given routing instance using Alg-Routing, there are at most $O(\log n)$ trajectories that cross a node in any given round. This implies that there are at most $O(\log^3 n)$ messages that arrive at a node per round.
2. Recall that each swarm is of size $O(\log n)$ w.h.p. Thus, during the introduction step in Alg-LDS each mature node introduces $O(\log n)$ nodes to their $O(\log n)$ neighbors. Resulting in a congestion of $O(\log^2 n)$ additional messages per node.
3. In Alg-Random each node, w.h.p., receives $O(\log n)$ tokens through the sampling algorithm and forwards them to fresh nodes. Additionally, each fresh node sends out $O(\log n)$ advertisements. Thus, altogether each node exchanges $O(\log n)$ messages.

□

Theorem 7.24 now follows from lemmas 7.25, 7.29 and 7.38.

Chapter 8

Conclusion and Open Problems

We presented an algorithm that computes approximate pure Nash equilibria in congestion games with significantly improved approximation guarantee. The most interesting question which was the initial motivation for the algorithm we presented in Chapter 2 is the complexity of approximate pure Nash equilibria. We find it very surprising that our technique yields such a significant improvement, e.g., for linear congestion games from 2 to 1.61, by using essentially the same algorithm of Caragiannis et al. [CFGS11]. However, the algorithmic technique is only limited by the lower bound for approximation factor of the stretch implied in Roughgarden [Rou14]. Hence, further significant improvements may need new algorithmic ideas. On the lower bound side, not much is known for linear or polynomial congestion games. The only computational lower bound for approximate pure Nash equilibria is from Skopalik and Vöcking [SV08] using unnatural and arbitrarily steep cost functions.

In Chapter 3 we extended the technique discussed in Chapter 2 and proposed an elegant and simple mechanism to compute robust load dependent cost functions that improve the inefficiency of equilibria in congestion games. We believe that the technique of perturbing the instance of an (optimization) problem such that a simple local search heuristic (or an equilibrium) guarantees an improved approximation ratio, can be applied to other strategic games as well. It would be interesting to see, whether one can achieve similar results for variants and generalizations of congestion games such as weighted [AAE05], atomic- or integer-splittable [Ros73b, RS11b] congestion games, scheduling games [CQ12, GLMM10, CCG$^+$15], etc. Considering other heuristics such as greedy or one-round walks [CMS12, BFFM11, KST19, BV17] would be another natural direction.

To the best of our knowledge, the problem of computing the social optimum of an instance of a scheduling problem with machine dependent priority discussed in Chapter 4 is a relatively new variant of scheduling with precedence constraints, that has not been studied extensively. The main difference from classical scheduling with precedence constraints is that a priority list determines the scheduling priority for jobs on a specific machine, rather than for the entire schedule. Therefore, it is not possible to adopt known ideas and techniques. We showed that for arbitrary priority list and identical parallel machines the problem is both NP-hard and APX-hard. For global priority list we showed similar hardness results for unrelated parallel machines. In case of identical machines, we proved the existence of a QPTAS using a reduction to an instance of a fixed order scheduling problem mentioned in Bosman et al. [BFO$^+$19]. For the case of 2 machine scheduling with global priority list, we showed that the problem can be solved optimally using a dynamic programming approach. This leaves us with many open questions, e.g., the hardness for the case of identical machines and global priority list. Another interesting direction is to study a constant factor approximation algorithm for these problems, especially when the machines have arbitrary priority lists.

Traditional analysis of coordination mechanisms assumes that jobs assigned to some machine are processed according to some policy, such as shortest or longest processing time. In Chapter 5 we investigated the effects of having a different scheduling policy, given by an arbitrary priority list, for every machine. We showed that in general, a pure Nash equilibrium schedule may not exist, and it is NP-hard to identify whether a given game has a pure Nash equilibrium. On the other hand, for several important classes of instances, we showed that a pure Nash equilibrium exists and can be computed efficiently, and we bounded the equilibrium inefficiency with respect to the common measures of minimum makespan and sum of completion times. We also showed that natural dynamics converge to a pure Nash equilibrium for all these classes. In terms of computational complexity, we proved that even for the simple class of identical machines, for which a pure Nash equilibrium can be computed efficiently, it is NP-hard to compute a pure Nash equilibrium whose quality is better than the quality of the worst pure Nash equilibrium. Our work leaves open several interesting directions for future work. Particularly, since our game may not have a pure Nash equilibrium, it is natural to consider weaker notions of stability. The existence and complexity of an approximate pure Nash equilibrium profiles is still open. We also studied the natural generalization of scheduling games with priority list, i.e., weighted congestion games with priority list in which players have an arbitrary strategy space and the cost of a player is the sum of the cost for the resources used, where each resource has its own priority list. Rosenthal [Ros73a] proved using a potential function argument that traditional congestion games always exhibit a pure Nash equilibrium. In Chapter 6 we showed that with the inclusion of priority lists, the game does not posses a potential function. In fact, it is NP-complete to decide if an instance has a pure Nash equilibrium. An interesting future direction would then be to investigate existence of an approximate potential function.

When designing large scale distributed networks, it is imperative to consider the unpredictable user/node dynamics. Strategic users of the system may join and leave the network in a manner that could potentially disconnect the network. In Chapter 7 we modeled the unpredictable strategic situation using an adversary. We presented an algorithm that maintains a structured overlay in presence of a $(2, O(\log n))$-late adversary with high churn rate. We believe that there is a strong connection between an adversaries lateness with regard to the topology and the churn rate. This leaves many interesting future directions for research. For instance, one could consider finding an algorithm that tolerates a $(1, O(\log n))$-late adversary. Also, one could consider a hybrid model where the adversary has almost up-to-date information about some nodes, but is more outdated with regard to others.

Appendix A

Source Code

```
from gurobipy import *

N2=10
ZMAX=101


for Z in range(100,ZMAX):   #set subset size range Z
  for N in range (200,201): #set number of players N
    for d in range(3,4):  #set polynomial degree range ((d-1),d)
      f = {}
      primal=Model('Primal');
      g = primal.addVar(obj=1, vtype=GRB.CONTINUOUS, name="Gamma")
      c = primal.addVar(obj=0, vtype=GRB.CONTINUOUS, name="C")
      for n in range(0,N+1+N2+Z+1000):
        f[n] = primal.addVar(obj=0, vtype=GRB.CONTINUOUS, name="f[%s]"%n)

      primal.setParam('OutputFlag', 0)
      primal.setParam('NumericFocus',3)
      primal.update()

      primal.addConstr(g - f[1] >= 0)

      for n in range(0,N+1):
        for m in range(0,N+1):
          for z in range(0,Z+1):
            if(d==2):
              primal.addConstr((((m+z) * ((m+z)+1)*0.5) - (z * (z+1)*0.5))
              * g - m * f[n + z + 1] + n * f[n + z] >= (((n+z) * ((n+z)+1)*0.5)
              - (z * (z+1)*0.5)))
            if(d==3):
              primal.addConstr((((m+z) * ((m+z) + 1)*(2*(m+z)+1)/6)
              - (z * (z + 1)*(2*z+1)/6)) * g - m * f[n + z + 1] + n * f[n + z]
               >= (((n+z) * ((n+z) + 1)*(2*(n+z)+1)/6)
               - (z * (z + 1)*(2*z+1)/6)))

            if(d==4):
              primal.addConstr((g * (((m+z) **2 * ((m+z) + 1)**2 / 4)
              - (z **2 * (z + 1)**2 / 4))) - (m * f[n + z+  1]) + (n * f[n+ z])
              >= (((n+z) **2  * ((n+z) + 1)**2 / 4)
              - (z **2 * (z + 1)**2 / 4)))
            if(d==5):
              primal.addConstr ((g * ((((m+z)*((m+z)+1)*(2*(m+z)+1)
              *(3*((m+z)**2) + 3*(m+z) -1))/30)
              - ((z*(z+1)*(2*z+1)*(3*(z**2) + 3*z -1))/30)))
            - (m * f[n + z+1]) + (n * f[n+z])
            >= ((((n+z)*((n+z)+1)*(2*(n+z)+1)*(3*((n+z)**2)
            + 3*(n+z) -1))/30)-((z*(z+1)*(2*z+1)*(3*(z**2) + 3*z -1))/30)))
          if(d==6):
            primal.addConstr((g * ((((m+z)**2*(((m+z)+1)**2)*(2*((m+z)**2)
            + 2*(m+z) -1))/12)-((z**2*((z+1)**2)*(2*(z**2) + 2*z -1))/12))
            - (m * f[n + z +1]) + (n * f[n+z])
            >= ((((n+z)**2*(((n+z)+1)**2)*(2*((n+z)**2) + 2*(n+z) -1))/12)
            - ((z**2*((z+1)**2)*(2*(z**2) + 2*z -1))/12))))
          if(d==7):
            primal.addConstr((g * ((((m+z) * (1 + (m+z)) * (1 + 2*(m+z))
            * (1 - 3*(m+z) + 6*(m+z)**3 + 3*(m+z)**4))/42)-((z * (1 + z)
            * (1 + 2*z) * (1 - 3*z + 6*(z**3) + 3*(z)**4))/42))
            - (m * c* (n + z +1)**6) + (n * c*(n+z)**6)
            >= ((((n+z) * (1 + (n+z))* (1 + 2*(n+z))* (1 - 3*(n+z)
            + 6*(n+z)**3 + 3*(n+z)**4))/42) - ((z * (1 + z) *(1 + 2*z)
            * (1 - 3*z + 6*(z**3) + 3*(z)**4))/42))))

    for n in range(0,N+N2+1+Z+500):
      primal.addConstr(f[n+1] - f[n] >= 0)

    for n in range(N+1,N+N2+1+Z+600):
      primal.addConstr(f[n] - c*(n**(d-1)) >= 0)
      primal.addConstr(f[n] - c*(n**(d-1)) <= 0)

    primal.presolve()
    primal.optimize()

    print('d:', (d-1), 'Primal objective:', primal.objVal)
```

Listing A.1: Python source code to compute strong smooth cost functions.

```python
from gurobipy import *

N2=100
for N in range(200, 201): #set number of players N
  for d in range(2, 7): #set polynomial degree range
    f = {}
    primal = Model('Primal')
    g = primal.addVar(obj=1, vtype=GRB.CONTINUOUS, name="Gamma")
    c = primal.addVar(obj=0, vtype=GRB.CONTINUOUS, name="C")
    for n in range(0, N + 1 + N2 + 1):
      f[n] = primal.addVar(obj=0, vtype=GRB.CONTINUOUS, name="f[%s]" % n)

    primal.setParam('OutputFlag', 0)
    primal.setParam('NumericFocus', 0)
    primal.update()

    primal.addConstr(g - f[1] >= 0)

    for n in range(0, N + 1):
      for m in range(0, N+N2 + 1):
        primal.addConstr((m ** d) * g - m * f[n + 1] + n * f[n] >= (n ** d))

    for n in range(0, N + N2 + 1):
      primal.addConstr(f[n + 1] - f[n] >= 0)

    for n in range(N + 1, N + N2 + 1):
      primal.addConstr(f[n] - c * (n ** (d - 1)) >= 0)
      primal.addConstr(f[n] - c * (n ** (d - 1)) <= 0)

    primal.presolve()
    primal.optimize()


    print('d:', (d-1),  'Primal objective:', primal.objVal)
```
Listing A.2: Python source code to compute robust load dependent tax functions.

Bibliography

[AAE05] Baruch Awerbuch, Yossi Azar, and Amir Epstein. The price of routing unsplittable flow. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 57–66, New York, NY, USA, 2005. ACM.

[AAE13] Baruch Awerbuch, Yossi Azar, and Amir Epstein. The price of routing unsplittable flow. *SIAM Journal on Computing*, 42(1):160–177, 2013.

[AART06] Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. Tradeoffs in worst-case equilibria. *Theoretical Computer Science*, 361(2-3):200–209, 2006.

[ACG$^+$13] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer Publishing Company, Incorporated, 2013.

[ADG$^+$11] Sebastian Aland, Dominic Dumrauf, Martin Gairing, Burkhard Monien, and Florian Schoppmann. Exact price of anarchy for polynomial congestion games. *SIAM Journal on Computing.*, 40(5), September 2011.

[ADK$^+$08] Elliot. Anshelevich, Anirban. Dasgupta, Jon. Kleinberg, Eva. Tardos, Tom. Wexler, and Tim. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[AGA99] Ali Allahverdi, Jatinder N.D Gupta, and Tariq Aldowaisan. A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239, 1999.

[AGM$^+$08] Heiner Ackermann, Paul Goldberg, Vahab S Mirrokni, Heiko Röglin, and Berthold Vöcking. A unified approach to congestion games and two-sided markets. *Internet Mathematics*, 5(4):439–457, 2008.

[AJM08] Yossi Azar, Kamal Jain, and Vahab Mirrokni. (almost) optimal coordination mechanisms for unrelated machine scheduling. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 323–332. Society for Industrial and Applied Mathematics, 2008.

[All15] Ali Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378, 2015.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AMM$^+$13] John Augustine, Anisur Rahaman Molla, Ehab Morsy, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Storage and search in dynamic peer-to-peer networks. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, page 53–62, New York, NY, USA, 2013. Association for Computing Machinery.

[ANCK08] Ali Allahverdi, C.T. Ng, T.C.E. Cheng, and Mikhail Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, June 2008.

[APR$^+$15] John Augustine, Gopal Pandurangan, Peter Robinson, Scott Roche, and Eli Upfal. Enabling robust and efficient distributed computation in dynamic peer-to-peer networks. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 350–369, 2015.

[ARV08] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. *Journal of the ACM*, 55(6):25:1–25:22, December 2008.

[ARV09] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. Pure nash equilibria in player-specific and weighted congestion games. *Theoretical Computer Science*, 410(17):1552–1563, 2009.

[AS07] Baruch Awerbuch and Christian Scheideler. Towards scalable and robust overlay networks. In *6th International workshop on Peer-To-Peer Systems, IPTPS 2007, Bellevue, WA, USA, February 26-27, 2007*, 2007.

[AS21] John Augustine and Sumathi Sivasubramaniam. Spartan: Sparse robust addressable networks. *Journal of Parallel and Distributed Computing*, 150:121–138, 2021.

[Aum74] Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

[BCS74] John. L. Bruno, Edward. G. Coffman, and Ravi. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7):382–387, 1974.

[BFFM11] Vittorio Bilò, Angelo Fanelli, Michele Flammini, and Luca Moscardelli. Performance of one-round walks in linear congestion games. *Theory of Computing Systems*, 49(1):24–45, Jul 2011.

[BFO$^+$19] Thomas Bosman, Dario Frascaria, Neil Olver, René Sitters, and Leen Stougie. Fixed-order scheduling on parallel machines. In Andrea Lodi and Viswanath Nagarajan, editors, *Integer Programming and Combinatorial Optimization*, pages 88–100, Cham, 2019. Springer International Publishing.

[Bil18] Vittorio Bilò. A unifying tool for bounding the quality of non-cooperative solutions in weighted congestion games. *Theory of Computing Systems*, 62(5):1288–1317, July 2018.

[BKS17] Antje Bjelde, Max Klimm, and Daniel Schmand. Brief announcement: Approximation algorithms for unsplittable resource allocation problems with diseconomies of scale. In *Proceedings of the 29th ACM Symposium on Parallelism* *in Algorithms and Architectures*, SPAA ’17, pages 227–229, New York, NY, USA, 2017. ACM.

[BV16] Vittorio Bilò and Cosimo Vinci. Dynamic taxes for polynomial congestion games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC ’16, pages 839–856, New York, NY, USA, 2016. ACM.

[BV17] Vittorio Bilò and Cosimo Vinci. On the Impact of Singleton Strategies in Congestion Games. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BV20] Vittorio Bilò and Cosimo Vinci. Congestion games with priority-based scheduling. In *International Symposium on Algorithmic Game Theory*, pages 67–82. Springer, 2020.

[CCG$^+$15] Richard Cole, José Correa, Vasilis Gkatzelis, Vahab Mirrokni, and Neil Olver. Decentralized utilitarian mechanisms for scheduling games. *Games and Economic Behavior*, 92:306–326, 2015.

[CDR06] Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. How much can taxes help selfish routing? *Journal of Computer and System Sciences*, 72(3):444–467, May 2006.

[CFGS11] Ioannis Caragiannis, Angelo Fanelli, Nick Gravin, and Alexander Skopalik. Efficient computation of approximate pure nash equilibria in congestion games. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS ’11, pages 532–541, Washington, DC, USA, 2011. IEEE Computer Society.

[CFGS12] Ioannis Caragiannis, Angelo Fanelli, Nick Gravin, and Alexander Skopalik. Computing approximate pure nash equilibria in congestion games. *SIGecom Exchanges*, 11(1):26–29, 2012.

[CFK$^+$06] Ioannis Caragiannis, Michele Flammini, Christos Kaklamanis, Panagiotis Kanellopoulos, and Luca Moscardelli. Tight bounds for selfish and greedy load balancing. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2006.

[CGV17] Ioannis Caragiannis, Vasilis Gkatzelis, and Cosimo Vinci. Coordination mechanisms, cost-sharing, and approximation algorithms for scheduling. In *International Conference on Web and Internet Economics*, pages 74–87. Springer, 2017.

[CK05a] George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC ’05, pages 67–73, New York, NY, USA, 2005. ACM.

[CK05b] George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 67–73. ACM, 2005.

[CKK06] Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Taxes for linear atomic congestion games. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 184–195, 2006.

[CKN04] George Christodoulou, Elias Koutsoupias, and Akash Nanavati. Coordination mechanisms. In *International Colloquium on Automata, Languages, and Programming*, pages 345–357. Springer, 2004.

[CMS12] George Christodoulou, Vahab S. Mirrokni, and Anastasios Sidiropoulos. Convergence and approximation in potential games. *Theoretical Computer Science*, 438:13 – 27, 2012.

[Coo00] Stephen Cook. The p versus np problem. In *Clay Mathematical Institute; The Millennium Prize Problem*, 2000.

[CPM19] Rahul. Chandan, Dario. Paccagnan, and Jason. R. Marden. When smoothness is not enough: Toward exact quantification and optimization of the price-of-anarchy. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 4041–4046, 2019.

[CQ12] José Correa and Maurice Queyranne. Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. *Naval Research Logistics*, 59(5):384–395, 2012.

[CS80] Yookun. Cho and Sartaj. Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.

[CS11] Steve Chien and Alistair Sinclair. Convergence to approximate nash equilibria in congestion games. *Games and Economic Behavior*, 71(2):315–327, 2011.

[CV07] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 3(1):4, 2007.

[Dar05] Vasilios Darlagiannis. *21. Hybrid Peer-to-Peer Systems*, pages 353–366. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[de 46] Nicolaas Govert de Bruijn. A combinatorial problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 49(7):758–764, 1946.

[DGP06] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, page 71–78, New York, NY, USA, 2006. Association for Computing Machinery.

[DGS16] Maximilian Drees, Robert Gmyr, and Christian Scheideler. Churn- and dos-resistant overlay networks based on network reconfiguration. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 417–427, 2016.

[DN09] Christoph Dürr and Kim Thang Nguyen. Non-clairvoyant scheduling games. In *International Symposium on Algorithmic Game Theory*, pages 135–146. Springer, 2009.

[DR98] Devdatt Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Structures & Algorithms*, 13(2):99–124, 1998.

[ECP$^+$05] Eng Keong Lua, Jon. Crowcroft, Marcelo. Pias, Ravi. Sharma, and Steven. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys Tutorials*, 7(2):72–93, 2005.

[EK12] David Easley and Jon Kleinberg. Networks, crowds, and markets: Reasoning about a highly connected world. *Significance*, 9:43–44, 2012.

[FGS14] Matthias Feldotto, Martin Gairing, and Alexander Skopalik. Bounding the potential function in congestion games and approximate pure nash equilibria. In *Proceedings of the 10th International Conference on Web and Internet Economics (WINE)*, number 8877 in LNCS, pages 30–43. Springer International Publishing Switzerland, 14 - 17 December 2014.

[FKK$^+$09] Dimitris Fotakis, Spyros Kontogiannis, Elias Koutsoupias, Marios Mavronicolas, and Paul Spirakis. The structure and complexity of nash equilibria for a selfish routing game. *Theoretical Computer Science*, 410(36):3305–3326, 2009. Graphs, Games and Computation: Dedicated to Professor Burkhard Monien on the Occasion of his 65th Birthday.

[FKK10] Dimitris Fotakis, George Karakostas, and Stavros G. Kolliopoulos. On the existence of optimal taxes for network congestion games with heterogeneous users. In *Algorithmic Game Theory - Third International Symposium, SAGT 2010, Athens, Greece, October 18-20, 2010. Proceedings*, pages 162–173, 2010.

[FKS05] Dimitris Fotakis, Spyros Kontogiannis, and Paul Spirakis. Selfish unsplittable flows. *Theoretical Computer Science*, 348(2-3):226–239, 2005.

[FOV08] Babak Farzad, Neil Olver, and Adrian Vetta. A priority-based model of routing. *Chicago Journal of Theoretical Computer Science*, 1, 2008.

[FPT04] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 604–612, New York, NY, USA, 2004. ACM.

[FS07] Dimitris Fotakis and Paul G. Spirakis. Cost-balancing tolls for atomic network congestion games. In *Proceedings of the 3rd International Conference on Internet and Network Economics*, WINE'07, pages 179–190, Berlin, Heidelberg, 2007. Springer-Verlag.

[FS17] Michael Feldmann and Christian Scheideler. A self-stabilizing general de bruijn graph. In *Proceedings of the 19th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 10616 of *Lecture Notes in Computer Science*, page 250–264. Springer, Cham, 2017.

[FSY05] Amos Fiat, Jared Saia, and Maxwell Young. Making chord robust to byzantine attacks. In *Proceedings of the 13th Annual European Conference on Algorithms*, ESA'05, page 803–814, Berlin, Heidelberg, 2005. Springer-Verlag.

[Gö21] Thorsten Götte. *Ph.D. Thesis*. Unpublished, Private Communication, Paderborn University, Germany, 2021.

[GHSS17] Robert Gmyr, Kristian Hinnenthal, Christian Scheideler, and Christian Sohler. Distributed monitoring of network properties: The power of hybrid networks. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 137:1–137:15, 2017.

[GHSW20] Thorsten Götte, Kristian Hinnenthal, Christian Scheideler, and Julian Werthmann. Time-optimal construction of overlay networks. *CoRR*, abs/2009.03987, 2020.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., USA, 1979.

[GLMM10] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. Computing nash equilibria for scheduling on restricted parallel links. *Theory of Computing Systems*, 47(2):405–432, 2010.

[GMMT15] Laurent Gourvès, Jérôme Monnot, Stefano Moretti, and Nguyen Kim Thang. Congestion games with capacitated resources. *Theory of Computing Systems*, 57(3):598–616, 2015.

[GMS04] Christos. Gkantsidis, Milena. Mihail, and Amin. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM 2004*, volume 1, page 130, 2004.

[Gra66] Ronald Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[GRS18] Thorsten Götte, Vipin Ravindran Vijayalakshmi, and Christian Scheideler. Always be two steps ahead of your enemy. *CoRR*, abs/1810.07077, 2018.

[GRS19] Thorsten. Götte, Vipin. Ravindran Vijayalakshmi, and Christian. Scheideler. Always be two steps ahead of your enemy. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1073–1082, 2019.

[HMU07] Birgit Heydenreich, Rudolf Müller, and Marc Uetz. Games and mechanism design in machine scheduling-an introduction. *Production and Operations Management*, 16(4):437–454, 2007.

[Hor73] William. A. Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21(3):846–847, 1973.

[HPQ17] Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM Journal on Computing*, 46(6):1712–1744, 2017.

[HSW01] Han Hoogeveen, Petra Schuurman, and Gerhard J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. *INFORMS Journal on Computing*, 13(2):157–168, 2001.

[HU19] Ruben Hoeksma and Marc Uetz. The price of anarchy for utilitarian scheduling games on related machines. *Discrete optimization*, 31:29–39, 2019.

[ILMS09] Nicole Immorlica, Li Erran Li, Vahab Mirrokni, and Andreas Schulz. Coordination mechanisms for selfish scheduling. *Theoretical computer science*, 410(17):1589–1598, 2009.

[JC10] Xing Jin and S.-H. Gary Chan. *Unstructured Peer-to-Peer Network Architectures*, pages 117–142. Springer US, Boston, MA, 2010.

[JDP83] Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, 11(1):286–295, 03 1983.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[Kan91] Viggo Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37(1):27–35, 1991.

[Kan92] Viggo Kann. On the approximability of np-complete optimization problems. *Ph. D. Dissertation, Royal Institute of Technology 1992*, 01 1992.

[Kar72] Richard Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[KLSY07] Valerie King, Scott Lewis, Jared Saia, and Maxwell Young. Choosing a random peer in chord. *Algorithmica*, 49(2):147–169, 2007.

[Kol13] Konstantinos Kollias. Nonpreemptive coordination mechanisms for identical machines. *Theory of Computing Systems*, 53(3):424–440, 2013.

[KP99] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science*, STACS'99, pages 404–413, Berlin, Heidelberg, 1999. Springer-Verlag.

[KS04] Valerie King and Jared Saia. Choosing a random peer. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, page 125–130, New York, NY, USA, 2004. Association for Computing Machinery.

[KST19] Max Klimm, Daniel Schmand, and Andreas Tönnis. The online best reply algorithm for resource allocation problems. In Dimitris Fotakis and Evangelos Markakis, editors, *Algorithmic Game Theory*, pages 200–215, Cham, 2019. Springer International Publishing.

[KV12] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012.

[LK78] Jan. K. Lenstra and Alexander. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.

[LY12] Pin-Yan Lu and Chang-Yuan Yu. Worst-case nash equilibria in restricted routing. *Journal of Computer Science and Technology*, 27(4):710–717, 2012.

[MdC85] Marie Jean Antoine Marquis de Condorcet. *Essai sur l'application de l'analyse a la probabilite des decisions: rendues a la pluralite de voix.* De l'Imprimerie royale, 1785.

[MS96] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[MS06] Peter Mahlmann and Christian Schindelhauer. Distributed random digraph transformations for peer-to-peer networks. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '06, page 308–317, New York, NY, USA, 2006. Association for Computing Machinery.

[MS12] Carol A. Meyers and Andreas S. Schulz. The complexity of welfare maximization in congestion games. *Networks*, 59(2):252–260, March 2012.

[MS14] Konstantin Makarychev and Maxim Sviridenko. Solving optimization problems with diseconomies of scale via decoupling. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 571–580, Washington, DC, USA, 2014. IEEE Computer Society.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing.* Cambridge University Press, 2005.

[MV78] Hervé. Moulin and Jean. P. Vial. Strategically zero-sum games: the class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory*, 7(3-4):201–221, September 1978.

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory.* Cambridge University Press, USA, 2007.

[NW07] Moni Naor and Udi Wieder. Novel architectures for p2p applications: The continuous-discrete approach. *ACM Transactions on Algorithms*, 3(3):34–es, August 2007.

[OPS04] James B. Orlin, Abraham P. Punnen, and Andreas S. Schulz. Approximate local search in combinatorial optimization. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 587–596, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[Par17] Kun Il Park. *Fundamentals of Probability and Stochastic Processes with Applications to Communications.* Springer Publishing Company, Incorporated, 1st edition, 2017.

[PCFM20] Dario Paccagnan, Rahul Chandan, Bryce L. Ferguson, and Jason R. Marden. Incentivizing efficient use of shared infrastructure: Optimal tolls in congestion games. *CoRR*, abs/1911.09806, 2020.

[Pet94] Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4(2):133–157, April 1994.

[Pin08] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Springer Publishing Company, Incorporated, 3rd edition, 2008.

[PNS16] Georgios Piliouras, Evdokia Nikolova, and Jeff S. Shamma. Risk sensitivity of price of anarchy under uncertainty. *ACM Transactions on Economics and Computation*, 5:5:1–5:27, 2016.

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[Ros73a] Robert W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, Dec 1973.

[Ros73b] Robert. W. Rosenthal. The network equilibrium problem in integers. *Networks*, 3(1):53–59, 1973.

[Rou05] Tim Roughgarden. *Selfish Routing and the Price of Anarchy.* The MIT Press, 2005.

[Rou14] Tim Roughgarden. Barriers to near-optimal equilibria. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 71–80, Washington, DC, USA, 2014. IEEE Computer Society.

[Rou15] Tim Roughgarden. Intrinsic robustness of the price of anarchy. *Journal of the ACM*, 62(5):32, 2015.

[RS11a] Andrea W. Richa and Christian Scheideler. Self-stabilizing debruijn networks. In *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS, page 416–430, 2011.

[RS11b] Tim Roughgarden and Florian Schoppmann. Local smoothness and the price of anarchy in atomic splittable congestion games. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 255–267, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics.

[RST21a] Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir. Scheduling games with machine-dependent priority lists. *Theoretical Computer Science*, 855:90–103, 2021.

[RST21b] Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir. Scheduling with machine-dependent priority lists. *Unpublished Manuscript*, 2021.

[RT00] Tim Roughgarden and Eva Tardos. How bad is selfish routing? In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 93–102, Nov 2000.

[Rud09] Matthew R Rudary. *On predictive linear Gaussian models.* University of Michigan, ProQuest Dissertations Publishing, 2009.

[RVS20] Vipin Ravindran Vijayalakshmi and Alexander Skopalik. Improving approximate pure nash equilibria in congestion games. In Xujin Chen, Nikolai Gravin, Martin Hoefer, and Ruta Mehta, editors, *Web and Internet Economics*, pages 280–294, Cham, 2020. Springer International Publishing.

[Sch00] Christian Scheideler. *Probabilistic Methods for Coordination Problems.* PhD thesis, Paderborn University, 2000.

[Sch01] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings First International Conference on Peer-to-Peer Computing*, pages 101–102, 2001.

[Sch05] Christian Scheideler. How to spread adversarial nodes? rotate! In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 704–713, New York, NY, USA, 2005. Association for Computing Machinery.

[Sku01] Martin Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48(2):206–242, March 2001.

[Smi56] Wayne E Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

[SMK$^+$01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Computer Communication Review*, 31(4):149–160, August 2001.

[SR06] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, page 189–202, New York, NY, USA, 2006. Association for Computing Machinery.

[SR20] Alexander Skopalik and Vipin Ravindran Vijayalakshmi. Improving approximate pure nash equilibria in congestion games. *CoRR*, abs/2007.15520, 2020.

[STR19] Marc Schröder, Tami Tamir, and Vipin Ravindran Vijayalakshmi. Scheduling games with machine-dependent priority lists. *CoRR*, abs/1909.10199, 2019.

[STRV19] Marc Schröder, Tami Tamir, and Vipin Ravindran Vijayalakshmi. Scheduling games with machine-dependent priority lists. In Ioannis Caragiannis, Vahab Mirrokni, and Evdokia Nikolova, editors, *Web and Internet Economics*, pages 286–300, Cham, 2019. Springer International Publishing.

[SV08] Alexander Skopalik and Berthold Vöcking. Inapproximability of pure nash equilibria. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 355–364, New York, NY, USA, 2008. ACM.

[Swa12] Chaitanya Swamy. The effectiveness of stackelberg strategies and tolls for network congestion games. *ACM Transactions on Algorithms*, 8(4):36:1–36:19, October 2012.

[U.S64] U.S. Dept of Commerce. Traffic assignment manual for application with a large, high speed computer. 1964.

[Vij17] Vipin Ravindran Vijayalakshmi. *Bounding the Inefficiency of Equilibria in Congestion Games under Taxation.* Universität Paderborn, 2017.

[Vöc07] Berthold Vöcking. *Algorithmic Game Theory*, chapter 20: Selfish Load Balancing. Cambridge University Press, 2007.

[Waj17] David Wajc. Negative association-definition, properties, and applications. 2017.

[Wor86] Mike Worboys. The travelling salesman problem (A guided tour of combinatorial optimisation), edited by Eugene L. Lawler, Jan K. Lenstra, Alexander H. G. Rinnooy Kan and David B. Shmoys isbn 0-471-90413-9 (wiley). *The Mathematical Gazette*, 70(454):327–328, 1986.

Declaration of Authorship

I, Vipin Ravindran Vijayalakshmi

declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I do solemnly swear that:

1. This work was done wholly or mainly while in candidature for the doctoral degree at this faculty and university;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others or myself, this is always clearly attributed;
4. Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations;
5. I have acknowledged all major sources of assistance;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published before as:
   (a) Vipin Ravindran Vijayalakshmi, Marc Schröder, Tami Tamir. Scheduling games with machine-dependent priority lists. In: *Theoretical Computer Science* 855 (2021), p. 90-103.
   (b) Vipin Ravindran Vijayalakshmi, Alexander Skopalik. Improving Approximate Pure Nash Equilibria in Congestion Games. In: *Web and Internet Economics - 16th International Conference*, WINE 2020, Beijing, China, December 7-11, 2020, Proceedings.
   (c) Marc Schröder, Tami Tamir, Vipin Ravindran Vijayalakshmi. Scheduling Games with Machine-Dependent Priority Lists. In: *Web and Internet Economics - 15th International Conference*, WINE 2019, New York, NY, USA, December 10-12, 2019, Proceedings.
   (d) Thorsten Götte, Vipin Ravindran Vijayalakshmi, Christian Scheideler. Always be Two Steps Ahead of Your Enemy. In: *2019 IEEE International Parallel and Distributed Processing Symposium*, IPDPS 2019, Rio de Janeiro, Brazil, May 20-24, 2019.

(e) Alexander Skopalik, Vipin Ravindran Vijayalakshmi. Improving approximate pure Nash equilibria in congestion games. https://arxiv.org/abs/2007.15520 (2020).

(f) Marc Schröder and Tami Tamir and Vipin Ravindran Vijayalakshmi. Scheduling Games with Machine-Dependent Priority Lists. http://arxiv.org/abs/1909.10199 (2019).

(g) Thorsten Götte, Vipin Ravindran Vijayalakshmi, Christian Scheideler. Always be Two Steps Ahead of Your Enemy. http://arxiv.org/abs/1810.07077 (2018).

Date

Signature