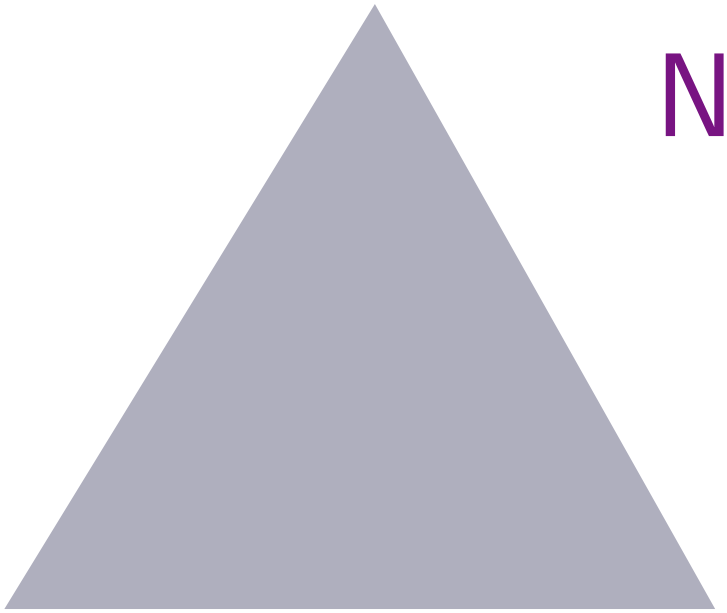
ANTJE BJELDE

ALGORITHM AND MECHANISM DESIGN FOR CONGESTED NETWORKS



Algorithm and Mechanism Design for Congested Networks

vorgelegt von

Antje Bjelde, M.Sc.
geb. in Eisenach

von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss

Vorsitzender:	Prof. Dr. Jochen Blath
Gutachter:	Prof. Dr. Max Klimm
	Dr. Felix Fischer
	Prof. Dr. Martin Skutella

Tag der wissenschaftlichen Aussprache: 10. April 2018

Berlin 2018

Acknowledgements

Writing this thesis would not have been possible without the invaluable support from friends, family and colleagues for which I want to express my deep gratitude here.

First and foremost, I would like to thank my supervisor and *Doktorvater* Max Klimm, who not only made this work possible by offering me a position and continuously supporting my work, but with whom I also enjoyed many a nice cup of coffee and lively discussions - he did not neglect to show me the fun part of academia. Many ideas began their path to proofs while working together on paper and whiteboard in his office.

Further, I am thankful to my second supervisor Felix Fischer for support on my continuing way from student to independent researcher and for his help in broadening my mathematical horizon. Moreover, I wish to thank Martin Skutella for taking the third assessment of this thesis.

Financially, I wish to thank the Matheon and the Einstein Center for Mathematics. In addition, finishing the thesis would not have been possible without the support from the Caroline von Humboldt completion grant for which I am very grateful. Further, the Berlin Mathematical School has enabled me to visit conferences with travel grants and also provides an excellent environment for young mathematicians for which I want to express my thanks. I also thank SIGARCH for the student travel grant to attend SPAA 2017.

I am especially thankful for having been able to spend the last three years in the mathematically stimulating and personally warm and welcoming atmosphere of the COGA group; both the old "West" at TU as the new "East" at HU. Sportivity in the west, superior coffee and lunch in the east, retreats in the wild, game and karaoke nights all over Berlin, conferences, research stays, and numerous fruitful discussions all over the world were highlights. I would in particular like to thank enthusiastic coauthors, witty office mates, and those who also appreciate rainbow unicorns. Special thanks to Jan, Julie, Miri, Philipp and Wiebke, who, in one way or another, helped the final version of this thesis to emerge. I have learned a lot from all of you.

Last, but not least, I would like to thank my friends and family for their perpetual support and encouragement, for helping to juggle private and work tasks, and for distracting me at just the right times. Finally, my heartfelt thanks goes to J., companion of my years, who takes the mathematical terms of $50 : 50$, equality and $(x^2 + y^2 - 1)^3 - x^2y^3 = 0$ without fuss to our everyday life.

Antje Bjelde

Contents

Introduction

Alice: *Would you tell me, please, which way I ought to go from here?*
Cat: *That depends a good deal on where you want to get to.*

Alice in Wonderland by Lewis Carroll [Car65]

Alice's question, innocently asked as it is, is not as trivial as it seems. Choosing which way to go can have a variety of consequences beyond just where one ends up. For instance, when caught in slow moving congested traffic, one might wonder whether one ought to have gone another way or even used another mode of transport. Such a situation is often negative for personal reasons. One could be late to an appointment, have stressful interactions with other traffic participants, suffer a general reduction of quality of life, or simply waste of time and energy. Such situations are also bad for society as a whole, since they negatively impact economic growth, cultural development and can have devastating effects on environment.

In fact, transport has contributed about 15% of carbon dioxide (CO_2) and 31% of ozone (O_3) of the total man-made addition of Greenhouse Gas (GHG) since preindustrial times [FBM+08]. And even now, transport is responsible for a quarter of Greenhouse Gas emissions in the European Union. Even more alarming, it is the only major economic sector in Europe where GHG emissions are not lower than 1990 levels, but are an alarming 20% higher [Eur16]. The European Commission for Mobility and Transport therefore envisions a 60% cut in transport emissions by the middle of the century [Eur11] in order to keep further devastating effects in control. GHG are a main reason for the long term effects of global warming, which include, among many others, worldwide reduction of biodiversity, harmful changes in ecosystems, and danger to subsistence of people (see, e. g. , Root et al. [RTPH+03], Cox et al. [CBJ+00]).

In sharp contrast to the goal of reducing GHG emissions, the German Federal Transport Infrastructure Plan expects an increase of 30% in total transport until 2030 in Germany. Amongst this, there is an expected increase in passenger transport by 12.2% and of freight transport by 38%. Interestingly, out of the 264.5 billion budget, 69% are used for maintenance [Fed16]. It is therefore indispensable to make sound transport planning decisions not only for environmental, but also for financial reasons. This thesis aims at providing some approaches for planning and managing

such infrastructures.

In order to reduce GHG emissions created by transport, the following measures come to mind: to use alternative fuels, to improve vehicle standards, to reduce overall demand and to improve operations management, which means better vehicle routing. There already is a lot of engineering work to build vehicles which work with a different energy supply or utilize available supplies more efficiently as GHG emissions are proportional to fuel consumption [Dep15].

In this work, we want to tackle two other main possibilities to reduce GHG emissions: reduce overall amount of transportation actors and provide better vehicle routing management. Both will result in less congested traffic and thus quicker completion time for transport and less overall use of fuel.

In addition, dealing with those questions provides a framework to identify improving possibilities for a wide range of infrastructure systems and networks which are influenced by congestion. With growing urban areas, deliberate infrastructure planning is now more important than ever. This includes the electric grid, telecommunication networks, but also of course public roads, and public transport and logistic networks. Reducing demand and providing intelligent routing leads to a decrease in financial, personal, and environmental burden. A better structured electric grid works more efficiently, thus losing less energy on the way to the network user. Faster telecommunications networks are increasingly important in todays multimedia society. A well thought-out public transport can lead to less private transportation and thus a decline in GHG emissions as well as less wastage of material. Efficient working logistic networks are important for economic growth and ultimately globalization.

One problem that we identify on our way is that those infrastructure systems are used by a large number of independent units. Indeed, they may even prefer to reach their personal goals instead of optimizing the overall network flow. Nash in 1950 [Nas50] and Wardrop in 1952 [War52] have laid groundwork for analyzing such systems of independent decision makers. Nash and Wardrop equilibria emerged from their works. In a nutshell, an equilibrium is a state in which no user can reach a better outcome for herself, e. g. a shorter travel time, when only she is changing her strategy or usage pattern, but all other users stay as they are and continue their strategy. This type of competition is studied within the field of noncooperative game theory. Unfortunately, equilibria, especially in complex network structures, are in general neither easy nor fast to determine and implement. But in order to create good infrastructure networks, it is important to understand the conditions under which good equilibria emerge.

In this context, we are particularly interested in the concept of congestion games. A network congestion game consists of a network with cost functions on the edges and (weighted) users with start and destination points within that network. A strategy of a user is to choose a path which connects these points. The goal is to reduce the cost of the strategy which is the sum of the costs of the edges it contains. The cost of each edge depends on the amount of users choosing it. An important quantity in

this context is the Price of Anarchy, which describes the ratio between a best possible and a worst possible equilibrium. This structure can be adapted to analyze how to route users in a sensible way such that the overall network usage is optimized. It can in some sense be considered as a congestion game for environmentally friendly, or altruistic, users. As it may take a lot of computational power and time to find an optimal solution, it is sensible to look for good enough solutions. That means we want to design algorithms which have a provable performance guarantee and which can thus be shown to give results within a certain range of an optimal solution even in the worst case. The first chapter of this thesis considers this setting.

Another easily identifiable problem in infrastructure systems is that users might want to manipulate data which they hand to an algorithm that then determines how a network is used. This happens if users try to influence the algorithm in order to achieve their personal goals, e. g. their preferred route or their favorite carpool. As such selfish behavior ignores the overall network usage, it might result in poor outcome. We are interested in incentive compatibility concepts where no user gains an advantage for herself by not acting according to her true preferences. The goal is thus to design a mechanism where strategic users can interact such that the resulting outcome reaches a certain goal. These types of problems have been studied in the field of mechanism design. This relatively new field has gained a lot of interest in the past decades. In fact, in 2007, Leonid Hurwicz, Roger B. Myerson and Eric S. Maskin won the highly prestigious Nobel prize in economics, "for having laid the foundations of mechanism design theory" [Nob14].

In particular, we are interested in strategyproof direct-revelation mechanisms. In a direct-revelation mechanism, each agent is asked to report their private information exactly once. Based on this, and possibly some randomization, the mechanism chooses an outcome. Strategyproofness ensures that the optimal strategy for each agent is to reveal her true private information and hence prevents her from manipulating her input. In chapter two, we thus deal with strategically thinking users in networks.

Finally, when looking at congested networks, we realize that the problems which we encounter are not static, but that we are working in a constantly changing environment. Users join in, leave, or change destinations without reporting a long time beforehand. We do not know all of this information before the algorithm runs and have thus to be able to produce output without complete information and react to new input along the way. The study of such algorithms has been formalized in the field of online algorithms, whose systematic study started with Sleator and Tarjan in 1985 [ST85].

Particular interesting for us is competitive analysis, named as such by Karlin, Manasse, Rudolph and Sleator in 1988 [KMRS88]. Broadly speaking, an online algorithm who receives information over time is compared to its offline counterpart, which has access to all information when it starts. The competitiveness of an online algorithm rises the closer its output is to the offline output of its corresponding offline algorithm. The third chapter of this thesis considers routing problems in an online setting.

To recapitulate, with this work we are tackling the problem of high congestion in networks from a mathematical point of view using methods from Operations Research and Algorithmic Game Theory. Congestion leads to a large number of problems, e. g. social, economical, and environmental. By providing approaches to reduce high congestion, we expect to improve the planning and maintaining of urban infrastructures, e. g. lower GHG emissions, have more efficient electric grids, provide faster telecommunications networks and more smoothly running logistic networks.

We identify two main possibilities to achieve those goals. First, we have to route users in a sensible way. This means on the one hand routing them in such a way that it is good for the overall network flow, and on the other hand also such that deviation from a proposed route will not benefit any user. Second, we reduce the overall demand for transportation by having users cooperate in an efficient and just way.

The problems that we encounter determine the methods and specific fields that we use. We expect independent users. They are likely to manipulate the algorithm. Also, they may appear over time. Out of the large field of intersection between mathematics, computer science, and economics, we thus decided to concentrate on approximation algorithms, mechanism design and online algorithms.

Thesis Outline

This thesis begins by giving some specific definitions and mathematical background for easier understanding of each of the main chapters. The preliminaries cover basic concepts of approximation algorithms, direct revelation mechanisms, strategyproofness, request answer games and competitive analysis.

The main part of the thesis consists of three chapters, each devoted to explore one possibility to reduce congestion in networks. The following paragraphs give an overview of content and highlights of our respective contribution. In order to make each chapter as self-contained and convenient for the reader as possible, the formal problem definition, related work and result summary for each problem can be found in the introductory section of each part. Also, in each chapter a technical discussion is provided and open problems laid out.

We close the thesis with a short conclusion with a focus on the practical applications of our results and outlook.

Chapter 1: Rerouting in Congested Networks

In the first main chapter of this thesis, Chapter 1, we provide a possibility to reroute users and thus reduce congestion by means of a local search algorithm in a network. In this network model, congestion grows as more users are using one path. Using the local search algorithm, users are routed such that the global network cost is reduced.

The mathematical framework for this problem is provided by general resource allocation problems where a set of commodities jointly uses a set of resources with a diseconomy of scale. The set of feasible allocations for each commodity is a commodity specific set of subsets of resources. Each commodity can have a different weight, or all weights are one which is then called unweighted. The cost of each resource is determined by a polynomial function with maximum degree d that maps the total weight put on a resource to its cost. We are interested in reducing the overall cost. For this model, we can imagine commodities to be users in a network and resources to be paths.

Main Contributions (Chapter 1) We propose a local search algorithm that switches one commodity to another set of resources if the total cost of the solution is decreased and continues until such switch is no longer possible. The algorithm is then analyzed. For this purpose, we are interested in the locality gap, which is the difference between local optimal soluation calculated by the algorithm and a global optimal solution.

Analyzing the local search algorithm, we show that the locality gap for unweighted resource allocation problems is of order $[\mathcal{O}(d/\log d)]^{d+1}$, for which we provide a corresponding lower bound. In addition, we give concrete values for the locality gap for small d. The linear case $d = 1$, which corresponds to at most quadratic total costs, can be evaluated to have a locality gap of at most three, which is tight. We also evaluate the general approximation guarantee for this case and give a lower bound of 1.02 which rises to 1.04 if the Unique Games Conjecture holds true.

Chapter 2: Non-Monetary Access Control Mechanisms

In Chapter 2, we provide a framework to reduce the total number of active participants in a network. This is achieved by letting possible users choose who may use the network or certain paths by giving nominations between them. As users are liable to influence the outcome by nominating strategically, we are interested in mechanisms where such a behavior does not grant an advantage to the individual user.

From a mathematical point of view, the problem we are looking at can be formulated in terms of a direct revelation mechanism without payment which is strategyproof, or dominant-strategies incentive-compatible (DSIC). This specific problem is known as impartial selection. We are given a set of agents. Each agent may nominate a number of other agents of her choosing. Our goal is to select k agents out of our set of agents with a high number of nominations. Each selection mechanism shall maintain impartiality, that means that the nominations which a user gives do not have any influence on her probability of being selected.

Main Contributions (Chapter 2) We consider the case of selecting two agents for which we investigate deterministic and randomized mechanisms and give upper and lower bounds. Both the case that exactly two agents have to be selected as well as the case that sometimes less than two agents can be selected are considered. Interestingly, the latter one leads to an improvement on the expected number

of nominations. We introduce the bidirectional permutation mechanism. It gives a best possible result of achieving at least $1/2$ of the total maximum number of nominations for deterministic selecting up to two agents. On the other side, it has been shown [AFPT11] that no deterministic exact mechanism can guarantee any positive approximation factor. The bidirectional permutation mechanism in the randomized variant achieves $2/3$ of the total maximum number of nominations for up to two agents in expectation. For this case, we can further show that no mechanism can do better than $3/4$ and give a best possible mechanism for the case of selecting two out of three agents. On the other hand, randomly selecting exactly two agents, we introduce the 2-partition mechanism with permutation, which achieves $7/12$ of maximum total number of nominations, while we establish an upper bound of $2/3$.

Further, we consider the case of selecting a general number of k agents. We generalize our algorithms to the k-partition mechanism with permutation for selecting exactly k agents and to the bidirectional k-permutation mechanism for selecting up to k agents. In addition, we introduce the exact dollar partition with permutation mechanism. As well as providing upper bounds for randomized and deterministic mechanism for exactly and up to k agents, by introducing our mechanisms we give lower bounds for these problems. Further, the exact dollar partition with permutation mechanism builds upon a general allocation subroutine, which gives a quick, simple solution to the problem of apportionment. This is the problem of fairly allocating representatives in proportion to group sizes and is interesting in its own right.

Chapter 3: Online Planning for Carpools

In the third main chapter, Chapter 3, we pick up the idea of reducing the total number of participants in the network. In addition, we integrate the approach to route users in a sensible way. We present an algorithm to plan online carpooling. Carpooling or ride-sharing is the practice of two or more people sharing journeys in the same, usually private, vehicle.

Mathematically, we are looking at the online dial-a-ride problem in a metric setting. As a special case, we first consider dial-a-ride on the line. Users arrive in an online manner in some place on the real line. They request to be transported to another place on the line. A server of given capacity picks the users up and delivers them, with or without temporary interruption, to their destination. The server starts at the origin and the problem can be closed, meaning the server has to return to the origin, or open. Our goal is to minimize the makespan, i. e. the time which the server needs until the last request is fulfilled.

Main Contributions (Chapter 3) We provide an algorithm for open and closed preemptive online dial-a-ride on the line. To analyze this algorithm, we compare it against its offline counterpart, which knows from the beginning which requests will appear at what point in time. We show that our algorithm is 2.41-competitive. Further, we provide a lower bound of 1.75 for the non-preemptive closed case. In addition, we show that a slightly modified version of the algorithm holds the 2.41-competitiveness in metric space for uncapacitated servers.

To understand the structure of optimal solutions better, we also consider the offline problem for dial-a-ride instances which have the same start and destination and thus only have to be visited. We prove that even in this case, we can construct instances such that the server has to make direction changes on the line arbitrarily often and give a dynamic program which computes the minimum completion time for this problem in quadratic time. Further, we show that in contrast to that the non-preemptive closed and open offline dial-a-ride on the line with capacity one is NP-complete.

Preliminaries

In this chapter, we review the main concepts which we need in this work and lay the foundation for the thesis. Though we assume that the reader is familiar with basic concepts and notions of mathematical optimization and algorithmic game theory, we recapitulate important definitions and explain major terms for ease of reading and understanding this thesis. We also point out important textbooks and standard literature introducing the corresponding topics.

Each of the following sections is devoted to one of the three main chapters of this thesis. First, we discuss approximation algorithms. In the second part, we cover direct-revelation mechanisms and talk about strategyproofness. Finally, the third section will deal with online algorithms in the sense of request-answer games and explain competitive analysis.

Rerouting in Congested Networks

We solve the problems that are considered in this thesis with algorithms and mechanisms. Our ideal algorithm solves a problem optimally, for every instance, and in a timely manner. The latter one is often compromised, e. g. due to NP-hardness. If we want to keep the premise of being able to solve every instance and also want to do it in a quick and efficient way, more precisely in polynomial time, then we restrain on our objective of optimality. This means we trade solution quality for a speed up of the algorithm. The study of approximation algorithms is based on this premise. In order to have a handle on how bad a solution can turn out, we introduce the notion of an approximation algorithm.

Definition (α-Approximation Algorithm) *An* α-approximation algorithm *with* $\alpha > 1$ *for a minimization problem and* $0 < \alpha < 1$ *for a maximization problem is an algorithm whose running time* $R(I)$ *is upper bounded by a polynomial function of the size of the input* $|I|$ *of the algorithm, i. e.* $R(I) \in \mathcal{O}(|I|^k)$ *for fixed* $k \in \mathbb{N}$*, and which for all instances of the problem outputs a solution whose value* $S(I)$ *is guaranteed to be within a factor of* α *of the value of the optimal solution* $O(I)$*, that means* $S(I) \leq \alpha \cdot O(I)$ *for a minimization problem and* $S(I) \geq \alpha \cdot O(I)$ *for a maximization problem. The value* α *is referred to as* approximation ratio.

For a more thorough introduction, please refer for instance to the books by Williamson and Shmoys [WS11], Hochbaum [Hoc97] or Vazirani [Vaz01].

While the first chapter does not explicitly deal with congestion games, they are nonetheless a basic concept related to the problem definition. They were first introduced by Rosenthal in 1973 [Ros73] and are thoroughly covered in the book Algorithmic Game Theory in the chapter Routing Games by Roughgarden [Rou07].

Non-Monetary Access Control Mechanisms

When trying to design a way to restrict access to a finite good where different users have to choose among themselves who will gain access, it is important to take into account possible manipulative user behavior. The restriction or selection mechanism should function well even when we assume strategic selfish behavior of each participant. To be more specific, we want each participant to reveal their true private information who should be granted access and not lie in order to gain an advantage. The field of mechanism design deals with this engineering approach to game theory. Because it starts with defining the rules of the game rather then looking at strategies for users, it is sometimes called reverse game theory. Here, a mechanism denotes a type of game in which users send a message based on which an outcome is chosen.

We are given a set of agents $I = \{1, \dots, n\}$. Each of the agents i has private information θ_i, which we also call the type of agent i, and which may, e. g. , determine the preferences of an agent over different outcomes of the game. Denote the space of all possible types of agent i by Θ_i. The space of all possible outcomes is denoted by Ω. The outcome may be, for example, that a certain agent receives a valued item. Further, each agent has a utility function $u_i \colon \Theta_i \times \Omega \to \mathbb{R}$, where $u_i(\theta_i, \omega)$ is the agent's utility given outcome ω when she has type θ_i.

As a simple example, imagine a group of people that has to select one representative amongst them, e. g. a political party that chooses someone to work in a national committee, or a group of authors which decides who should give the talk at a conference. The type of a person can be, for example: "I nominate Mx. Fisher and Mx. Smith.", "I think everyone, but my neighbor, would be a good representative.", or "I do not want to nominate anyone.". The outcome is that one person is selected. A utility function is now defined for each possible outcome by the revealed type of the person. For example, a utility could be high if a person is selected herself, and there could be a negative impact if a person is selected which was nominated but does not match the persons true preferences.

The goal of mechanism design in this thesis is to chose a smart outcome selection function. It gives the outcome for a given set of revealed private information in such a way that the agents have a strong incentive, meaning they expect high utility, to reveal their true preferences in the decision making process.

In general, a direct revelation mechanism m is a function which maps a space of types Θ to a set of outcomes Ω, thus $m \colon \Theta \to \Omega$. We use the utility function in order to model that one agent does not need to reveal her true private information. Instead, an agent may choose to misreport her type to the algorithm. However, while reporting a manipulated type may pose an advantage for some outcomes, it may also negatively impact an agents utility for other outcomes.

Definition (Deterministic Direct Revelation Mechanism) *We are given a set of agents $I = \{1, \dots, n\}$ with a set of types Θ_i for each agent i, a set of outcomes Ω and utility functions for each agent $u_i \colon \Theta_i \times \Omega \to \mathbb{R}$. A* deterministic direct revelation mechanism *is defined by an outcome selection function $m \colon \Theta_1 \times \cdots \times \Theta_n \to \Omega$.*

In other words, each agent reports her type to the mechanism exactly once, which then deterministically reveals the outcome.

As an example, consider a group of three people $\{a, b, c\}$ who want to select a representative amongst them. Each person chooses between the two others, e. g. a has possible types b or c, or $\Theta_a = \{b, c\}$. An outcome selection function could now be, described plainly: if one person is chosen by both others, e. g. $(\theta_a, \theta_b, \theta_c) = (b, c, b)$, this person is selected, so $m(b, c, b) = b$; in every other case, e. g. $(\theta_a, \theta_b, \theta_c) = (c, a, b)$, person a is selected, so $m(c, a, b) = a$.

If the outcome is not defined deterministically, but by a probability distribution over a set of outcomes, we call this a randomized direct revelation mechanism.

Definition (Randomized Direct Revelation Mechanism) *We are given a set of agents $I = \{1, \dots, n\}$ with a set of types Θ_i for each agent i, a set of outcomes Ω and utility functions $u_i \colon \Theta_i \times \Omega \to \mathbb{R}$. A* randomized direct revelation mechanism *is defined by a probability distribution over outcome selection functions $m \colon \Theta_1 \times \cdots \times \Theta_n \to X_\Omega$, where the set of probability distributions over Ω is denoted by X_Ω.*

For the example above with agents $\{a, b, c\}$ who select one of the other agents as their type, an outcome selection function could be to select an agent who is chosen by both other agents, and if that is not the case, each a, b or c is selected with probability $1/3$.

In broader terms, in a direct revelation mechanism holds that the set of strategies of an agent is equivalent to her set of types. A strategy, informally speaking, is a set of decision rules which describes for every possible situation which action an agent will select. Here, an agent only provides input to the mechanism once, i. e. by revealing a type.

For this work it is useful to understand the notion of strategyproofness. In a nutshell, a mechanism is strategyproof if no agent has any incentive to manipulate, i. e. not reveal her true preferences, but instead misreports her type.

Definition (Strategyproofness) *A direct revelation mechanism is* strategyproof *(or dominant strategies incentive compatible) if for each agent holds that for all possible reported sets of types of the other agents, her utility of revealing her true type is at least as good as for disreporting her type. Formally,*

$$u_i(\theta_i, s(\theta_i, \theta_{-i})) \geq u_i(\theta_i, s(\theta_i', \theta_{-i})) \qquad \forall \quad \theta_{-i} \in \Theta_{-i} \quad \textit{and} \quad \theta_i' \in \Theta_i$$

where $\Theta_{-i} = \Theta_1 \times \cdots \times \Theta_{i-1} \times \Theta_{i+1} \times \cdots \times \Theta_n$.

That means, the utility function for an agents true private information is at least as high as for any other possible information she can reveal given any set of informations reported by the other agents. Thus, revealing her private information or preferences truthfully is a weakly dominant strategy for each agent.

Looking at the example above, consider a mechanism which has an outcome selection function that always choses agent a, no matter which types are reported by the agents. This mechanism is strategyproof, as the utility of each agent does not change if they report a different type; in fact, their reported types do not influence their utility at all as it does not have any impact on the outcome.

In contrast, a typical example for a non strategyproof mechanism for cases where an agents utility only depends on whether she is selected herself is plurality voting, where each voter votes exactly once and the candidate with most received votes is selected. Here, e. g. voting for the strongest opponent may influence one's own chance of being selected.

For a further introduction to mechanisms and strategyproofness, please refer for example to Parkes' text [Par04] and the book by editors Nisan, Roughgarden, Tardos and Vazirani [NRTV07].

Online Planning for Carpools

In real world applications, detailed data is often not known from the beginning, but is revealed sequentially. An algorithm that deals with this type of problems, possibly adjusting its behavior as time passes, is called an *online algorithm*.

A framework to model most online algorithm problems was given in 1994 by Ben-David et al. [BBK+94] by introducing the notion of request-answer games. Here, an adversial player makes a series of requests which are subsequently answered by the online algorithm. A function then determines the cost for any sequence of requests and answers. Formally, it can be described as follows.

Definition (Request Answer Game) *A* request-answer game *is a triple* $(\mathcal{R}, \mathcal{A}, \mathcal{C})$ *consisting of a set of requests* $\mathcal{R}$*, a sequence of answer sets* $\mathcal{A} = A_1, A_2, \ldots$ *and a sequence of cost functions* $\mathcal{C} = C_1, C_2, \ldots$ *with*

$$C_i \colon \mathcal{R}^i \times A_1 \times \cdots \times A_i \to \mathbb{R}^+ \cup \{\infty\}$$

for $i \in \mathbb{N}$.

For many problems, it is natural to assume that the answer sets $A_1, A_2, \ldots$ are all equal.

An online algorithm gets a sequence of requests $(r_1, \ldots, r_n)$ for $n \in \mathbb{N}$ as input and is required to choose a sequence of answers $(a_1, \ldots, a_n)$ as output. The goal is to select the sequence of answers in such a way that the cost $C_n(r_1, \ldots, r_n, a_1, \ldots, a_n)$ is minimized. It is important to note that even if the requests are revealed sequentially, the cost depends not only on the current answer but on all previous answers. Thus, the decision at each point influences our cost in the future. Formally, we define a deterministic online algorithm in the following way.

Definition (Deterministic Online Algorithm) *A* deterministic online algorithm ALG *for a request answer game* $(\mathcal{R}, \mathcal{A}, \mathcal{C})$ *is a sequence of functions* $f_1, f_2, \ldots$ *with* $f_i \colon \mathcal{R}^i \to A_i$ *for* $i \in \mathbb{N}$*. For a given sequence of requests* $r = (r_1, r_2, \ldots, r_n)$*, the* output *of the algorithm*

is given by

$$ALG(r) = \big(f_1(r_1), f_2(r_1, r_2), \ldots, f_n(r_1, \ldots, r_n)\big)$$

and the cost *incurred by the algorithm is given by*

$$|ALG(r)| = C_n\big(r, ALG(r)\big).$$

Clearly, while the cost depends upon all requests and answers, our answers at each point only depend on the requests issued until this point. We have to make decisions based on partial information.

How do we assess the quality of an online algorithm? One answer for this question lies within the field of competitive analysis. Here, we compare the performance of an online algorithm against its "offline" counterpart which knows all requests from the beginning.

Definition (Optimal Offline Cost) *An* optimal offline cost *for a given request answer game $(\mathcal{R}, \mathcal{A}, \mathcal{C})$ and request sequence r of length n is the cost $|OPT(r)|$ for a best possible selection of answers for this request sequence, that is*

$$|OPT(r)| = \min_{a \in A_1 \times \cdots \times A_n} C_n\big(r, a\big).$$

The optimal offline cost, which translates into a best possible answer that an algorithm which has complete information can give, provides us with a way to measure how well the online algorithm performs. The quotient between this best possible answer and the output of an online algorithm is called competitive ratio.

Definition (Competitive Ratio) *A deterministic online algorithm ALG for a request-answer game $(\mathcal{R}, \mathcal{A}, \mathcal{C})$ is called* deterministic γ-competitive *if for any request sequence r we have that*

$$|ALG(r)| \leq \gamma \cdot |OPT(r)|.$$

The factor γ is called the (deterministic) competitive ratio *of ALG.*

Note that the competitive ratio of an online algorithm can be viewed as the approximation ratio achieved by that algorithm.

For a more thorough introduction to online algorithms, please refer for example to the seminal article by Sleator and Tarjan [ST85], the books by Borodin and El-Yaniv [BEY98] or by editors Fiat and Woeginger [FW98].

Chapter 1

Rerouting in Congested Networks

We study general resource allocation problems with a diseconomy of scale. In such problems, we are given a finite set of commodities that request certain resources. The cost of each resource grows superlinearly with the demand for it, and our goal is to minimize the total cost of the resources. In large systems with limited coordination it is desirable to solve such problems in a distributed manner. To this end it is natural to consider local dynamics where in each step a single commodity switches its allocated resources whenever the new solution after the switch has smaller total cost over all commodities. These dynamics converge to a local optimal solution and we are interested in quantifying the *locality gap*, i. e. , the worst case ratio of the cost of a local optimal solution and a global optimal solution.
In this chapter we derive tight bounds on the locality gap for unweighted commodities with cost functions that are polynomials with non-negative coefficients and maximal degree d. Our performance guarantee asymptotically matches the approximation guarantee of the currently best known centralized algorithm due to Makarychev and Sviridenko [MS14]. In contrast to their algorithm which is based on the randomized rounding of the solution of a convex programming relaxation, our algorithm is deterministic, combinatorial, and requires only local knowledge of the commodities.

Bibliographic Information: The results in this chapter are based on joint work with Max Klimm and Daniel Schmand, published at the 29th Symposium on Parallelism in Algorithms and Architectures (SPAA) 2017 [BKS17].

The last decade has seen a growth in the prevalence and use of personal navigation devices. They provide a quick and easy to use solution for finding one's way in unknown areas but also the possibly quickest way home in congested traffic. In fact, with them being widespread and employed by a large number of people, they open up a whole new way of traffic regulation. For instance, rather than having clashing shortest paths, one can target reduction of overall traffic jams by coordinating the routes of different people by means of personal navigation devices. In this chapter we aim to provide insights into simple procedures which help to reduce congestion in networks using individual rerouting.

In the first section 1.1 we give an introduction into the practical roots of our problem and explain how to get from the application to the formal mathematical problem. We also describe related work and put our results into the context of former findings. In Section 1.2 we then present the local search approach and introduce smoothness which is used to analyze the approximation guarantees of local optimal solutions. Building on that, in Section 1.3 we provide bounds on the locality gap for the case of unweighted resource allocation problems with polynomial cost functions

with positive coefficients and maximum degree d. We further show that even for linear costs, unweighted resource allocation problems cannot be approximated by a factor better than 1.02 unless $\mathsf{P} = \mathsf{NP}$. Our approximation algorithm can also be considered for the case of weighted resource allocation problems which will be covered in Section 1.4. To close this chapter we discuss our findings and open problems in Section 1.5.

1.1 Background

Take for example different people wanting to cross a busy city with different start and end points. Each of them might have a navigation system which, based on current predictions for traffic, proposes a best possible route. Now, it is widely assumed that congestion in streets can be modeled using a delay function which is a polynomial of degree four, this dates back to 1964. The US Bureau of Public Roads assumes in their "traffic assignment manual for application with a large, high speed computer" for the relationship between traveltime and volume that

$$T = T_0\left(1 + 0.15\left(\frac{\text{Assigned volume}}{\text{Practical capacity}}\right)^4\right],$$

where T is the traveltime in the congested network and T_0 is traveltime in an empty network [US 64]. Here, assigned volume is the amount of traffic which is routed via a certain road and practical capacity is its maximal capacity. It is thus reasonable to consider polynomial delay functions.

The computing of which route to take happens locally for each driver now, which is why we are interested in a local search algorithm. In addition, instead of trying to optimize the route for each individual driver, it is reasonable to assume that the navigation systems actually communicate locally at times and try to keep the overall flow in the city as smooth as possible. Further, to find out how good our result is, we are interested in the ratio between this local optimum and network best possible global solution, as we want to keep it small to route as environmentally friendly and sustainable as possible. Keeping these requirements in mind, we formulate the following mathematical problem.

1.1.1 Formulating the Mathematical Problem

We study general resource allocation problems where a set of commodities jointly uses a set of resources with a diseconomy of scale. The set of feasible allocations for each commodity is a commodity-specific set of subsets of resources that is either listed explicitly or given implicitly by a certain combinatorial structure. The goal is to find a single allocation for each commodity such that the total cost is minimized. We assume that costs are separable among resources and that the cost of a resource is given by a non-decreasing and convex resource-specific cost function. We consider mostly the unweighted resource allocation problem, but some of our results extend to the weighted version. In the unweighted case, the cost of a resource is a function

of the total number of commodities using the resource. In this case, the weight of each commodity is simply assumed to be one, while in the weighted case the weight of each commodity can differ and the cost of a resource is a function of the total weight.

Minimization problems of this kind appear in different contexts. In the area of energy efficient algorithms, resources model computing devices that can run at different speeds. As the speed is increased, the energy consumption increases superlinearly. Frequently, it is assumed that energy consumption can be modeled by a function

$$C_r(x) = k_r x^{q_r} = x k_r x^{q_r - 1}$$

of speed x where $k_r > 0$ and $q_r \in (1, 3]$ are device-specific parameters, see, e. g. , Albers [Alb10] for a reference. In the area of traffic networks, resources correspond to roads in a street network. The time needed to traverse a road increases with the total traffic on the road. Popular models are the previously mentioned travel time functions put forward by the US Bureau of Public Roads (BPR) which are of the form

$$c_r(x) = k_r(1 + 0.15(x/z_r)^4)$$

where $k_r > 0$ and $z_r > 0$ are road-specific parameters and x is the load of the resource, see the report by the BPR [US 64]. When minimizing the total congestion cost, we are interested in minimizing the average travel time of a unit of demand which is given by $C_r(x) = x c_r(x)$.

As a running example, let us consider the special case where the resources correspond to the edges of a graph $G = (V, E)$ and each commodity is specified by a triple $(u_i, v_i, w_i) \in V \times V \times \mathbb{R}_{>0}$. The goal is to route w_i units of demand of commodity i along a single (and simple) path from u_i to v_i in the network. Thus, a feasible solution to the problem corresponds to an unsplittable flow without capacity constraints, and we want to minimize a convex cost function of the flow. This fundamental problem appears under different names in the literature such as the Minimum Power Routing problem (e. g. , Andrews et al. [AAZZ12]) and Welfare Maximization in Congestion Games (e. g. , Meyers and Schulz [MS12]).

Currently, the approximation algorithm with the best performance guarantee for this problem is due to Makarychev and Sviridenko [MS14]. They propose a convex programming relaxation of the problem and show that for monomials with degree d, the integrality gap is equal to the $(d+1)$-st Bell number B_{d+1}. Using a randomized rounding technique, this yields a randomized algorithm with approximation guarantee $B_{d+1} + \epsilon$ for any $\epsilon > 0$ which can be shown to be of order $[\mathcal{O}(d/\log d))]^{d+1}$, see [BT10]. However, the convex programming approach of Makarychev and Sviridenko has the disadvantage that it requires a large amount of central coordination as the convex program has to be solved by a single central authority and the routing decisions of the commodities are based on the solution of the linear programming relaxation. This may be infeasible to implement in large systems without a central authority that is able to collect the data and solve the program. In fact, when optimizing large decentralized networks such as the Internet with respect to the energy consumption or the total delay, such a central authority is usually absent.

For the optimization of such decentralized systems, it is natural to consider improvement dynamics where the system starts in an arbitrary state $\mathbf{s}$ and at each step

one commodity chooses another path if it decreases her share of the cost, i. e. , commodity i switches from path P to path Q if

$$\sum_{r \in P \setminus Q} c_r(x_r(\mathbf{s})) > \sum_{r \in Q \setminus P} c_r(x_r(\mathbf{s}) + w_i),$$

where $x_r(\mathbf{s})$ denotes the load of resource r in state $\mathbf{s}$. The stable points of these dynamics are the Nash equilibria of the underlying congestion game. There is a vast amount of literature quantifying the price of anarchy of congestion games, i. e. , determining the worst case ratio of the total cost in a Nash equilibrium and the total cost of an optimal solution. Most relevant to our work, Aland et al. [ADG+11] showed that the price of anarchy for both weighted and unweighted congestion games with polynomial travel time functions c_r with maximal degree d (that correspond to polynomial total cost functions C_r with maximum degree $d+1$) is of order $[\mathcal{O}(d/\log d)]^{d+1}$. Unfortunately, this result does not yield an efficient distributed approximation algorithm since for weighted commodities, the improvement dynamics may cycle (cf. Harks et al. [HKM11]) and Nash equilibria may even be fully absent (cf. Harks and Klimm [HK12]). For unweighted commodities, the improvement dynamics are guaranteed to converge by a potential function argument due to Rosenthal [Ros73], but the convergence may take a number of steps that is exponential in the size of the underlying network, see Ackermann et al. [ARV08]. These issues make these dynamics unfavorable in practice.

In order to obtain polynomial convergence for the unweighted case, Awerbuch et al. [AAE+08] studied an approximate version of the improvement dynamics where each commodity only switches if the potential function drops by a factor of $1+\delta$. They showed that after a polynomial sequence of δ-best replies, a solution is reached which is an $(5/2 + \mathcal{O}(\delta))$-approximation to the minimal congestion cost in the case of linear cost functions. For polynomial cost functions with maximum degree d and positive coefficients their approach yields a $(d^{d-o(1)} + \mathcal{O}(\delta))$-approximation.

The technique of Awerbuch et al. however, relies on the existence of a potential function and thus cannot be applied to resource allocation problems with weighted commodities and non-linear cost functions.

To overcome this issue and give an alternative approach also for unweighted commodities, we study a different improvement dynamics where the system again starts in an arbitrary state $\mathbf{s}$, but all commodities take the impact of their path choices on the other commodities into account. More formally, we require that commodity i switches from path P to path Q if this switch decreases the overall cost of the solution, i. e. ,

$$\sum_{r \in P \setminus Q} x_r(\mathbf{s}) c_r\big(x_r(\mathbf{s})\big) - \big(x_r(\mathbf{s}) - w_i\big) c_r\big(x_r(\mathbf{s}) - w_i\big) > \sum_{r \in Q \setminus P} \big(x_r(\mathbf{s}) + w_i\big) c_r\big(x_r(\mathbf{s}) + w_i\big) - x_r(\mathbf{s}) c_r\big(x_r(\mathbf{s})\big).$$

This approach has the advantage that the total cost of the current solution is monotonically decreasing in each step, which implies that the dynamics reach a local optimum after a finite number of steps. In contrast to the long history of papers quantifying the efficiency of Nash equilibria, much less is known regarding the efficiency of local optimal solutions. This is the main issue addressed in this chapter.

Notation and Formal Problem Definition We study local improvement dynamics for general resource allocation problems with convex costs. In particular, we are interested in quantifying the locality gap, i. e. , the worst-case ratio between the cost of a local optimal solution and the cost of a global optimal solution. We assume that costs are of the form $C(x) = xp(x)$ where $p(x)$ is a polynomial with non-negative coefficients and maximal degree $d \in \mathbb{N}$, i. e. $p(x) = \sum_{j \in J} \alpha_j x^j$ with $J \subset [0, d]$, $\alpha_j \geq 0$ for all $j \in J$. Note that this is the standard form for the total cost in the congestion game literature and will be used throughout this chapter. The speed scaling literature usually writes costs as $C(x) = x^q$ for some $q > 1$. Clearly, both forms are equivalent up to a shift in the exponent.

We consider a finite set of commodities $N = \{1, \dots, n\}$ and a finite set of resources R. For each commodity $i \in N$, we are given (explicitly or implicitly) a set $S_i \subseteq 2^R$ of feasible solutions. Further, we are given a weight $x_i \in \mathbb{R}_{\geq 0}$ for each commodity. The cost of each resource $r \in R$ is determined by a non-negative function $c_r \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that maps the total weight put on a resource to its cost. Given a solution $\mathbf{s} = (s_1, \dots, s_n)$ with $s_i \in S_i$, the cost for commodity i is defined as

$$C_i(\mathbf{s}) := x_i \sum_{r \in s_i} c_r(x_r(\mathbf{s})),$$

where

$$x_r(\mathbf{s}) := \sum_{j \in N \colon r \in s_j} x_j$$

is the total weight put on resource r under solution $\mathbf{s}$. We are interested in finding solutions $\mathbf{s}$ that minimize the total cost of the commodities, i. e. , that minimize

$$C(\mathbf{s}) := \sum_{i=1}^{n} C_i(\mathbf{s}) = \sum_{r \in R} x_r(\mathbf{s}) c_r(x_r(\mathbf{s})).$$

We denote the space of all feasible solutions by $\mathbf{S} = S_1 \times \cdots \times S_n$ and an instance of a resource allocation problem by $I = (\mathbf{S}, (x_i)_{i \in N}, (c_r)_{r \in R})$. A resource allocation problem is called *unweighted* if $x_i = 1$ for all $i \in N$ and *weighted*, otherwise. For a solution $\mathbf{s} \in \mathbf{S}$ and a commodity $i \in N$, we write $\mathbf{s} = (s_i, \mathbf{s}_{-i})$ implying that $s_i \in S_i$ and $\mathbf{s}_{-i} \in S_1 \times \cdots \times S_{i-1} \times S_{i+1} \times \cdots \times S_n$. We also introduce the notation

$$C_{-i}(\mathbf{s}) := \sum_{j \in N \setminus \{i\}} C_j(\mathbf{s}).$$

We call a solution *local optimal*, if its cost cannot be decreased by reallocating a single commodity while keeping all other commodities at their current allocations.

Definition 1.1 (Local Optimum) *A solution* $\mathbf{s}$ *is called a* local optimum, *if*

$$C(\mathbf{s}) \leq C(s_i', \mathbf{s}_{-i})$$

for all $i \in N$ *and* $s_i' \in S_i$.

We are interested in the cost guarantees achieved by local optimal solutions. These guarantees are captured by the following notion of a locality gap.

Definition 1.2 (Locality Gap) *For an instance $I = (\mathbf{S}, (x_i)_{i\in N}, (c_r)_{r\in R})$ of a resource allocation problem, the locality gap is defined as*

$$\max\left\{\frac{C(\mathbf{s}')}{C(\mathbf{s})} : \mathbf{s}, \mathbf{s}' \in \mathbf{S} \text{ and } \mathbf{s}' \text{ is a local optimum}\right\}.$$

As a running example, it is useful to consider the natural case where the set of resources R corresponds to the set of edges E of a graph $G = (V, E)$. Each commodity is specified by a source node u_i and a node target v_i. The set S_i of feasible solutions of commodity i corresponds to the set of simple (u_i, v_i)-paths in G. We note, however, that all our results continue to hold in a more general setting where the set of feasible solutions $S_i \subset 2^R$ is arbitrary. We only need to make the minimal assumption that each commodity can efficiently optimize over its set S_i as long as the vector of feasible solution $\mathbf{s}_{-i}$ of the other commodities is fixed. In fact, we only need to require that this optimization over S_i can be done efficiently within arbitrary precision.

Assumption 1.3

1. *For every constant $\alpha \geq 1$ and for every commodity $i \in N$, there is a polynomial algorithm (oracle) $\mathfrak{O}_{i,\alpha} : \mathbf{S}_{-i} \to S_i$ that, given a partial solution $\mathbf{s}_{-i} \in \mathbf{S}_{-i}$ as input, computes a feasible solution $s'_i \in S_i$ with*

$$C(s'_i, \mathbf{s}_{-i}) \leq \alpha \min_{s_i \in S_i} C(s_i, \mathbf{s}_{-i}).$$

2. *A feasible solution $\mathbf{s} \in \mathbf{S}$ can be computed in polynomial time.*

Clearly, when the sets S_i corresponds to the set of paths in a network, this assumption is satisfied as shortest paths can be computed efficiently (see, i. e. , [DP84] for an overview).

1.1.2 Related Work and Previous Results

In addition to the works mentioned in the last section which are especially relevant to establish our approach, there are some other important results in this area and additional related previous work. Meyers and Schulz [MS12] study the problem of minimizing the total cost for general cost functions. They establish inapproximability results both for convex non-decreasing and non-increasing cost functions. As a contrast, they prove that for single-commodity networks, the problem can be solved in polynomial time by a reduction to minimum cost flows. More recently, Roughgarden [Rou14] studied the approximability of minimizing the total cost for unweighted resource allocation problems with polynomial cost functions and non-negative coefficients and maximal degree d. He showed that this problem cannot be approximated by a factor of $(\beta d)^{d/2}$ for some $\beta > 0$. He also observed that this non-approximability result gives an alternative proof for the fact that the price of anarchy is at least of order $d^{d/2}$ unless NP $=$ coNP as otherwise an equilibrium could serve as a certificate for the absence of a solution with a certain congestion cost.

Centralized approximation algorithms for the class of problems studied in this chapter were first presented by Andrews et al. [AAZZ12]. They observed that the

standard relaxation of the problem (where each commodity may split its demands) has an integrality gap of $\Omega(n^d)$. For the case of unweighted commodities, they use a clever transformation of the cost functions to obtain a constant integrality gap. Makarychev and Srividenko [MS14] gave a different convex programming relaxation similar to a configuration LP. They show that their relaxation achieves an integrality gap equal to the $(d+1)$-st Bell number. By applying a randomized rounding technique, for any $\delta > 0$, this integrality gap yields a randomized approximation algorithm with approximation guarantee $B_{d+1} + \delta$.

The local search approach pursued in this chapter has its roots in the work of Johnson et al. [JPY88] who introduced the complexity class PLS and showed that many natural local search problems such as finding a local optimal TSP tour are complete for this class. Fabrikant et al. [FPT04] showed that computing an equilibrium of a congestion game is PLS-complete. This result was strengthened by Ackermann et al. [ARV08] who showed the same result for games with linear costs, and by Skopalik and Vöcking [SV08] who showed that even the computation of an approximate equilibrium (for arbitrary costs) is PLS-complete. These negative results give additional justification for our local search paradigm that takes into account the global cost function instead on the private cost shares of the commodities.

The problem of computing an equilibrium is related to the minimization of the total congestion as equilibria typically are good approximations on the total congestion cost. Awerbuch et al. [AAE13] and Christodoulou and Koutsoupias [CK05] showed that the price of anarchy, i. e. , the worst case ratio of the cost of an equilibrium and the optimal cost, is $5/2$ for unweighted congestion games with linear costs. The latter authors also showed that for games with polynomial costs with maximum degree d the price of anarchy is of order $d^{d-o(1)}$. Aland et al. [ADG+11] provided tight results for the price of anarchy of both unweighted and weighted congestion games by solving for a given set of cost functions $\mathcal{C}$ an optimization problem of the form

$$\min_{\lambda \in \mathbb{R}, \mu \in (0,1)} \left\{ \frac{\lambda}{1-\mu} \mid c(x+y) \leq \lambda y c(y) + \mu x c(x) \text{ for all } x, y \in \mathbb{N}, c \in \mathcal{C} \right\}.$$

When $\mathcal{C}$ is the set of polynomials with non-negative coefficients and maximal degree d this gives a price of anarchy of order $(d/\log d)^{d+1}$. For the suboptimal choice of $\mu = 1/2$, the non-tight bound on the price of anarchy of $d^{d-o(1)}$ due to Christodoulou and Koutsoupias is obtained. Awerbuch et al. [AAE+08] showed that a polynomial sequence of approximate best replies reaches a solution with almost price of anarchy guarantees. Their approach requires $\mu \leq 1/2$ so that only the suboptimal guarantee of order $d^{d-o(1)}$ is obtained.

The approximation guarantees for local optima of the cost function are strongly related to altruistic versions of congestion games where players care not only for their own private costs but also for the total cost of a solution. Caragiannis et al. [CKK+10] consider unweighted congestion games with linear costs where for a $\xi \in [0,1]$ players strive to minimize $1-\xi$ times the cost of their own commodity plus ξ times the total cost of all other commodities. They obtain bounds on the price of anarchy for the resulting games depending on ξ which evaluate to three for $\xi = 1/2$. Chen et al. [CDKKS11] prove the same result using the smoothness framework due to Roughgarden [Rou09] thus establishing that this bound also holds for mixed and coarse correlated equilibria. The results in this chapter generalize

	Unweighted		Weighted	
Max. Degree d	This Work	[AAE+08]	[BKS17]	[AAE+08]
1	3	2.5	5.829	2.618
2	13	10	56.948	–
3	61	47	780.279	–
4	391	269	13,755.3	–
5	2,157	2,154	296,477	–
6	21,337	15,187	$7.55 \cdot 10^6$	–
7	154,725	169,247	$2.22 \cdot 10^8$	–
8	1,622,791	1,451,906	$7.40 \cdot 10^9$	–
9	16,880,931	20,241,038	$2.76 \cdot 10^{11}$	–
10	139,627,951	202,153,442	$1.13 \cdot 10^{13}$	–
d	$\mathcal{O}((\frac{2d}{\log d})^{d+1})$	$\mathcal{O}(d^{d-o(1)})$	$\mathcal{O}((\frac{d+1}{\log 2})^{d+1})$	–

Table 1.1: Approximation guarantees for unweighted and weighted resource allocation problems with polynomial costs and maximum degree d. All approximation guarantees can be obtained up to an additive term ϵ for any $\epsilon > 0$. Blue values are best known results, others are grey. The previous algorithm by Awerbuch et al. [AAE+08] does not apply to weighted problems with non-linear costs.

these results for the special case $\xi = 1/2$ to general polynomial cost functions and in our work [BKS17] also to weighted players.

The problem of computing global optima has been considered for further variants of congestion games. In a paper of Blumrosen and Dobzinski [BD07] the maximization versions of singleton congestion games with player-specific utilities introduced by Milchtaich [Mil96] have been analysed. They give a 18-approximation for the special case of linear utilities. De Keijzer and Schäfer [dS12] considered a class of maximization games where players prefer to share resources with other players and study the computation of social optima. Most recently, Harks et al. [HOV16] study polymatroid congestion games introduced by Harks et al. [HKP14] and give a H_r-approximation for the problem of minimizing the total congestion cost where H_r is the r-th harmonic number and r is the sum of the ranks of the player's polymatroids.

Finally, we note that the resource allocation problems considered in this chapter can be seen as the unsplittable counterpart of fractional resource sharing problems studied, e. g. , by Grigoriadis and Khachiyan [GK94], Jansen and Zhang [JZ08], Garg and Könemann [GK07] and Müller et al. [MRV11].

1.1.3 An Overview of Our Results

An overview of our results in comparison to the previous known work can be found in Table 1.1. We present a local search algorithm for general resource allocation problems with diseconomies of scale.

By analyzing the locality gap of the local optimal solutions, we provide the first non-trivial analysis of a local search algorithm that is guaranteed to converge for the unweighted and also the weighted case of the problem. By considering approximate improvement dynamics where commodities switch to another path only when the total cost is decreased by at least a factor of $1 + \delta$, where $\delta > 0$ is arbitrary, the dynamics are guaranteed to converge to an approximate local optimal solution in a

polynomial number of steps.

For unweighted resource allocation problems, we show that the locality gap is of order $[\mathcal{O}(d/\log d)]^{d+1}$. We also give a corresponding lower bound on the locality gap of $[\Omega(d/\log d)]^{d+1}$. For concrete values of d we can evaluate the locality gap explicitly. For the case $d = 1$, which corresponds to (at most) linear costs or total quadratic costs, the locality gap is three and this is tight. We further prove for this case a general inapproximability result of factor 1.02 which raises to 1.04 if the Unique Games Conjecture holds true. For the case $d = 2$, the locality gap is at most thirteen.

For unweighted problems with polynomial cost, our asymptotic approximation guarantee of $[\mathcal{O}(d/\log d)]^{d+1}$ poses a slight improvement over the asymptotic behavior of the algorithm by Awerbuch et al. [AAE+08] whose asymptotic behavior is of order $d^{d-o(1)}$. Our calculation of the concrete approximation guarantees for maximum degree up to ten suggests that our approximation is in fact better than the one by Awerbuch et al. for all $d \geq 9$. The approximation guarantee of $[\mathcal{O}(d/\log d)]^{d+1}$ asymptotically matches the currently best known guarantee of Makarychech and Srividenko [MS14]. In contrast to them, we show that by sacrificing the additional factor of $(1+\delta)$ our algorithm is deterministic, can be implemented in a distributed fashion, and does not rely on the centralized solution of a linear program and the randomized rounding of its solution.

In our work [BKS17], we further found that for weighted resource allocation problems the locality gap is $1/(\sqrt[d+1]{2}-1)^{d+1} \in [\mathcal{O}((d+1)/\log 2)]^{d+1}$ which is tight for all values of d. More specifically, for $d = 1$ the locality gap is $3+2\sqrt{2} \approx 5.829$ and for $d = 2$ the locality gap is $15\sqrt[3]{2}+12\sqrt[3]{4}+19 \approx 56.948$. For weighted resource allocation problems this yields the first deterministic algorithm, the first combinatorial algorithm, and the first distributed algorithm with non-trivial approximation guarantee.

1.2 A Local Search Based Approximation Algorithm

In this section, we develop a general approximation algorithm that is based on local search. Further, we provide a technique for the analysis of approximatively socially optimal strategy profiles for the general class of $(\lambda, \mu, 1)$-smooth games. Using that, we analyze the approximation guarantees of local optima computed by our local search algorithm for unweighted resource allocation problems.

1.2.1 Description of the Algorithm

Our approximation algorithm works as follows. The local search starts in an arbitrary solution $\mathbf{s} \in \mathbf{S}$. In each iteration, the oracles $\mathfrak{O}_{i,\alpha}$ are called to find a commodity which has an alternative solution $s_i' \in S_i$ such that the total cost is improved by a factor of at least $1+\delta$, i. e.

$$(1+\delta)C(s_i', \mathbf{s}_{-i}) \leq C(\mathbf{s})$$

Algorithm 1.1: Local Search Algorithm for Approximating Resource Allocation Problems

Input: An arbitrary solution $s \in S$ for the resource allocation problem
Output: A local optimal solution $s \in S$

```
repeat
    call oracle 𝔒_{i,α} to find i, t_i ∈ S_i with (1 + δ)C(t_i, s_{-i}) ≤ C(s)
    if such t_i exist then
        s ← {t_i, s_{-i}}
    else
        return s
```

for some sufficiently small $\delta > 0$. When no such commodity is found, the algorithm stops and returns the solution.

For a more formal definition, please refer to the pseudocode provided in Algorithm 1.1.

1.2.2 Introducing Smoothness for a General Framework

Orlin et al. [OPS04] showed in 2004 that every local search problem admits a polynomial-time approximation scheme (PTAS) in the sense that a $(1+\delta)$-approximate local optimal solution can be computed in polynomial time via approximate local improvements steps. We will be interested in determining how well local optimal solutions approximate the global minimum. To this end, we use the following notion of smoothness that was introduced by Chen et al. [CDKKS11] and used to analyze the price of anarchy of altruistic versions of unweighted congestion games with linear costs. It builds upon a similar notion of (λ, μ)-smoothness due to Roughgarden [Rou09].

Definition 1.4 (Smoothness) *The resource allocation problem $I = (\mathbf{S}, (x_i)_{i\in N}, (c_r)_{r\in R})$ is $(\lambda, \mu, 1)$-smooth if for any two solutions $\mathbf{s}, \mathbf{s}' \in \mathbf{S}$ we have*

$$\sum_{i=1}^{n} \Big(C(s_i', \mathbf{s}_{-i}) - C_{-i}(\mathbf{s}) \Big) \leq \lambda C(\mathbf{s}') + \mu C(\mathbf{s}).$$

We proceed to show that for every $(\lambda, \mu, 1)$-smooth resource allocation problem the locality gap is bounded by $\lambda/(1-\mu)$. Moreover, by applying a general framework by Orlin et al. [OPS04], it admits a polynomial, distributed, and deterministic $(\lambda/(1-\mu)+\epsilon)$-approximation algorithm for any $\epsilon > 0$.

Lemma 1.5 *Let $I = (\mathbf{S}, (x_i)_{i\in N}, (c_r)_{r\in R})$ be a resource allocation problem that is $(\lambda, \mu, 1)$-smooth for some $\lambda > 0$ and $\mu \in [0, 1)$. Then, the following holds.*

1. *The locality gap of I is at most $\frac{\lambda}{1-\mu}$.*
2. *For any $\epsilon > 0$, there is a $\left(\frac{\lambda}{1-\mu} + \epsilon\right)$-approximation algorithm based on local search that needs only a polynomial number of iterations.*

Proof. We start by proving 2. By Orlin et al. [OPS04, Theorem 2.3], for any $\delta > 0$, local search computes an approximate local optimal solution $\mathbf{s} \in \mathbf{S}$ with

$$\frac{C(\mathbf{s}) - C(s_i', \mathbf{s}_{-i})}{C(s_i', \mathbf{s}_{-i})} \leq \delta$$

for all commodities i and $s_i \in S_i$ in time polynomial in the problem size and $1/\delta$. We calculate

$$\begin{aligned}
\lambda C(\mathbf{s}') + \mu C(\mathbf{s}) &\geq \sum_{i=1}^{n} \Big(C(s_i', \mathbf{s}_{-i}) - C(\mathbf{s}) + C_i(\mathbf{s}) \Big) \\
&\geq \sum_{i=1}^{n} \Big(\frac{C(\mathbf{s})}{1+\delta} - C(\mathbf{s}) + C_i(\mathbf{s}) \Big) = -\frac{\delta n C(\mathbf{s})}{1+\delta} + C(\mathbf{s}) \\
&= \left(1 - \frac{\delta n}{1+\delta}\right) C(\mathbf{s}),
\end{aligned}$$

which implies

$$\left(1 - \mu - \frac{\delta n}{1+\delta}\right) C(\mathbf{s}) \leq \lambda C(\mathbf{s}').$$

For $\delta < \frac{1-\mu}{n-1+\mu}$ we have that

$$1 - \mu - \frac{\delta n}{1+\delta} > 0$$

and we obtain

$$\begin{aligned}
\frac{C(\mathbf{s})}{C(\mathbf{s}')} &\leq \frac{\lambda}{1 - \mu - \frac{\delta n}{1+\delta}} \\
&\leq \frac{\lambda}{1-\mu} + \frac{\delta n}{1+\delta} \cdot \frac{\lambda}{(1 - \mu - \frac{\delta n}{1+\delta})^2},
\end{aligned}$$

where we used that the function $x \mapsto \lambda/(1-\mu-x)$ is convex on its domain with derivative $\lambda/(1-\mu-x)^2$. Finally, let

$$\delta := \min\{ \frac{\epsilon(1-\mu)^2}{8\lambda n}, \frac{1-\mu}{2(n-1+\mu)}, 1\}.$$

We have

$$\begin{aligned}
\frac{\delta n \lambda}{(1+\delta)(1 - \mu - \frac{\delta n}{1+\delta})^2} &= \frac{\delta(1+\delta) n \lambda}{(1 - \mu - \delta(n-1+\mu))^2} \\
&= \frac{\delta n \lambda}{(1 - \mu - \delta(n-1+\mu))^2} + \frac{\delta^2 n \lambda}{(1 - \mu - \delta(n-1+\mu))^2} \\
&\leq \frac{2\delta n \lambda}{(1 - \mu - \delta(n-1+\mu))^2},
\end{aligned}$$

where we used that $\delta \leq 1$.

Since $\delta \leq \frac{1-\mu}{2(n-1+\mu)}$, we further have

$$1 - \mu - \delta(n - 1 + \mu) \geq \frac{1-\mu}{2}$$

and, thus,

$$\frac{\delta n \lambda}{(1+\delta)(1-\mu-\frac{\delta n}{1+\delta})^2} \leq \frac{8\delta n \lambda}{(1-\mu)^2} \leq \epsilon,$$

where for the final inequality, we used $\delta \leq \frac{\epsilon(1-\mu)^2}{8\lambda n}$.

We conclude that

$$C(\mathbf{s}) \leq C(\mathbf{s}') \leq \frac{\lambda}{1-\mu} + \epsilon$$

for all feasible solutions $\mathbf{s}' \in \mathbf{S}$, as claimed.

The claim for 1 follows from the above calculations for $\delta = 0$. □

1.2.3 Analyzing the Locality Gap

In this section, we analyze the approximation guarantees of local optima computed by local search using Algorithm 1.1 for unweighted resource allocation problems.

We start by showing some result for unweighted resource allocation problems with linear cost functions. The following proposition has been proven before by Chen et al. [CDKKS11]. In this work we show an alternative proof using the notion of $(\lambda, \mu, 1)$-smoothness to get familiar with the concept.

Proposition 1.6 *For every unweighted resource allocation problem with linear cost functions, i. e. polynomial cost functions with maximal degree $d = 1$, and with non-negative coefficients the following hold:*

1. *The locality gap is three.*
2. *For any $\epsilon > 0$ there is a polynomial time $3 + \epsilon$-approximation algorithm for the minimization of the total costs.*

Proof. By using Lemma 1.5, it suffices to show that an unweighted resource allocation problem with linear cost functions is $(3, 0, 1)$-smooth, i. e. we have to show that

$$\lambda C(s') + \mu C(s) \geq \sum_{i=1}^{n} C_i(s'_i, s_{-i}) + C_{-i}(s'_i, s_{-i}) - C_{-i}(s)$$

which means

$$3C(s') \geq \sum_{i=1}^{n} C_i(s'_i, s_{-i}) + C_{-i}(s'_i, s_{-i}) - C_{-i}(s).$$

We show that this inequality holds for all resources. Fix a resource r. Without loss of generality we can assume that all cost functions are of the form $f_r(x) = a_r x$ or $f_r(x) = b_r$, as resources can be split in a linear and a constant part.

First, focus on the case that the cost function is equal to $f_r(x) = b_r$. In this case, the sum $\sum_{i=1}^{n} C_i\left(s_i', s_{-i}\right)$ is upper bounded by y, where y is the number of commodities using resource r in s', and the rest is equal to zero.

Now, focus on the case that $f_r(x) = a_r x$. The first part of the sum is upper bounded by $y(x+1)$, where x is the number of commodities requesting resource r in s and y is the number of commodities using r in s'. The other parts of the sum calculates the difference of cost for all other commodities, given that commodity i changes from s_i to s_i'. This is upper bounded by $xy - x(x-1)$. So we need to show that

$$y(x+1) + xy - x(x-1) \leq 3y^2,$$

for all x and y that are integral and non-negative. We show this in the following Lemma 1.7, which completes the proof. □

Lemma 1.7 *For all x and y that are integral and non-negative holds that*

$$y(x+1) + xy - x(x-1) \leq 3y^2.$$

Proof. First, assume that $y = 1$. We calculate

$$\begin{aligned} y(x+1) + xy - x(x-1) &= -x^2 + 3x + 1 = -\left((x-1.5)^2\right) + 3.25 \\ &\leq 3. \end{aligned}$$

Second, assume that $y > x$ and $y \neq 1$. For this case, it suffices to show that

$$2y^2 + y + y \leq 3y^2,$$

which is equivalent to $2y \leq y^2$, which is clearly fulfilled.

Third, assume that $y \leq x$. We show this case by induction over x. For $x = 0$ we have $y \leq 3y^2$, which is clearly true. Now, suppose that

$$2xy + y + x \leq 3y^2 + x^2$$

holds true. We conclude that

$$\begin{aligned} 2(x+1)y + y + (x+1) &= 2xy + y + x + 2y + 1 \leq 3y^2 + x^2 + 2y + 1 \\ &\leq 3y^2 + (x+1)^2, \end{aligned}$$

which finishes the proof. □

We now continue with resource allocation problems with more general cost functions. In order to do so, we first prove the following lemma which gives a sufficient condition for an unweighted resource allocation problem to be $(\lambda, \mu, 1)$-smooth.

Lemma 1.8 *Let $\mathcal{C}$ be a set of convex cost functions, $\lambda > 0$ and $\mu \in [0, 1)$ such that*

$$\begin{aligned} yc(x+1) + xy\left[c(x+1) - c(x)\right] + x(x-1)\left[c(x-1) - c(x)\right] \\ \leq \lambda c(y)y + \mu c(x)x \end{aligned} \quad (1.1)$$

for all $x, y \in \mathbb{N}$ and $c \in \mathcal{C}$. Then, any unweighted resource allocation problem with cost functions in $\mathcal{C}$ is $(\lambda, \mu, 1)$-smooth.

Proof. Let I be an unweighted resource allocation problem as in the statement of the lemma and let $\mathbf{s}, \mathbf{s}'$ be two arbitrary solutions of I. Using the fact that c_r is monotonically increasing and convex and the notation

$$x_r = c_r(\mathbf{s}), y_r = c_r(\mathbf{s}'), \text{and } z_r = |\{i \in N \colon r \in s_i \cap s_i'\}|,$$

we have

$$\begin{aligned}
\sum_{i \in N} \Big(C(s_i', \mathbf{s}_{-i}) - C_{-i}(\mathbf{s}) \Big) &= \sum_{i \in N} \Big(C_i(s_i', \mathbf{s}_{-i}) + C_{-i}(s_i', \mathbf{s}_{-i}) - C_{-i}(\mathbf{s}) \Big) \\
&\leq \sum_{r \in R} y_r c_r(x_r + 1) + \sum_{i \in N} \Big(C_{-i}(s_i', \mathbf{s}_{-i}) - C_{-i}(\mathbf{s}) \Big) \\
&= \sum_{r \in R} y_r c_r(x_r + 1) + \sum_{r \in R} \Big((y_r - z_r) x_r [c_r(x_r + 1) - c_r(x_r)] \\
&\qquad + (x_r - z_r)(x_r - 1)[c_r(x_r - 1) - c_r(x_r)] \Big) \\
&\leq \sum_{r \in R} \Big(y_r c_r(x_r + 1) + y_r x_r \left[c(x_r + 1) - c_r(x_r) \right] \\
&\qquad - x_r(x_r - 1) \left[c_r(x_r) - c_r(x_r - 1) \right] \Big),
\end{aligned}$$

which completes the proof. □

Given some unweighted resource allocation problem with cost functions in some class $\mathcal{C}$, it remains to find appropriate λ and μ such that inequality (1.1) holds for all $c \in \mathcal{C}$. In order to do so, we set $\mu = 0$ and define

$$g_c(x, y) := \frac{1}{y c(y)} \Big(y c(x + 1) + xy \left[c(x + 1) - c(x) \right] + (x - 1) x \left[c(x - 1) - c(x) \right] \Big). \quad (1.2)$$

Setting

$$\lambda = \sup_{x, y \in \mathbb{N}, c \in \mathcal{C}} \{ g_c(x, y) \}$$

yields a $\lambda + \epsilon$ approximation algorithm for any $\epsilon > 0$ for any unweighted resource allocation problem with cost functions in $\mathcal{C}$. However, determining $\max_{x,y \in \mathbb{N}} g_c(x, y)$ even for a single cost function c is very challenging. Instead, we will work toward solving the relaxation $\max_{x \geq 1, y \geq 1} g_c(x, y)$. For this as a first step, we show that when cost function c is a monomial of type $c(x) = x^d$ for some $d > 0$, then the maximum $\max_{x \geq 1, y \geq 1} g_c(x, y)$ is attained when either $x = 1$, or $y = 1$, or both.

Lemma 1.9 *Let $c(x) = x^d$ for some $d > 0$. Then, the maximum $\max_{x \geq 1, y \geq 1} g_c(x, y)$ is attained at the boundary when either $x = 1$ or $y = 1$ (or both).*

Proof. For $c(x) = x^d$, we have

$$g_c(x, y) = \frac{1}{y^{d+1}} \Big(y(x + 1)^d + xy[(x + 1)^d - x^d] - (x - 1)x[x^d - (x - 1)^d] \Big).$$

Elementary calculus shows that

$$\frac{\partial g_c(ax,ay)}{\partial a} = \frac{1}{(ay)^{d+1}}\Big(xd(ax-1)^d - yd(ax+1)^d \\ - axy[(ax)^d - (ax+1)^d] - ax^2[(ax)^d - (ax-1)^d]\Big).$$

We obtain in particular that

$$\left.\frac{\partial g_c(ax,ay)}{\partial a}\right|_{a=1} = \frac{1}{y^{d+1}}\Big(xd(x-1)^d - yd(x+1)^d \\ - x^{d+2} - yx^{d+1} + xy(x+1)^d + x^2(x-1)^d\Big).$$

To prove the claimed result, it suffices to show that

$$A := \frac{\partial g_c(ax,ay)}{\partial a}\cdot y^{d+1} \leq 0$$

for all $x > 0$, $y > 0$. We calculate

$$A = d(x-1)^d x - d(x+1)^d y - x^2[x^d - (x-1)^d] + xy[(x+1)^d - x^d].$$

Using that the derivative of the function $z \mapsto z^d$ is dz^{d-1}, we can further obtain that

$$x^d - (x-1)^d \geq d(x-1)^{d-1}$$

as well as

$$(x+1)^d - x^d \leq d(x+1)^{d-1}.$$

This gives

$$\begin{aligned}\frac{A}{d} &\leq (x-1)^d x - (x+1)^d y - x^2(x-1)^{d-1} + xy(x+1)^{d-1} \\ &= (x-1)^{d-1}\big(x(x-1) - x^2\big) + (x+1)^{d-1}\big(xy - (x+1)y\big) \\ &= -x(x-1)^{d-1} - y(x+1)^{d-1}.\end{aligned}$$

Using that $x > 1$ and $y > 0$, this is negative and the claim follows. □

Building upon this lemma, we are now ready to state our main theorem for unweighted resource allocation problems.

Theorem 1.10 *For every unweighted resource allocation problem with polynomial cost functions with non-negative coefficients and maximal degree d the following hold for some $\alpha \in \mathcal{O}((2d/\log d)^{d+1})$:*

1. *The locality gap is α.*
2. *For any $\epsilon > 0$ there is a polynomial time $\alpha + \epsilon$-approximation algorithm for the minimization of the total costs.*

Proof. We start by showing the statement for monomials of the form $c(x) = x^k$ for some $k \leq d$. We are interested in determining

$$\max_{x \geq 1, y \geq 1} \frac{1}{y^{k+1}} \Big(y(x+1)^k + xy[(x+1)^k - x^k] - (x-1)x[x^k - (x-1)^k] \Big).$$

Using Lemma 1.9, the maximum is attained for either $x = 1$, or $y = 1$.

For $x = 1$, we obtain

$$g_c(1, y) = \frac{2^{k+1} - 1}{y^k},$$

which is obviously maximized for $y = 1$. We then have

$$2^{d+1} - 1 \in \mathcal{O}\left(\left(\frac{2d}{\log d} \right)^{d+1} \right)$$

finishing the proof for the case $x = 1$.

For $y = 1$, we have

$$\begin{aligned} g_c(x, 1) &= (x+1)^k - (x-1)x[x^k - (x-1)^k] + x[(x+1)^k - x^k] \\ &= [(x+1)^{k+1} - x^{k+1}] - (x-1)x[x^k - (x-1)^k]. \end{aligned}$$

Using

$$(x+1)^{k+1} - x^{k+1} \leq (k+1)(x+1)^k$$

and

$$x^k - (x-1)^k \geq k(x-1)^{k-1},$$

we obtain

$$g_c(x, 1) \leq (k+1)(x+1)^k - xk(x-1)^k \leq (k+1)(x+1)^k - k(x-1)^{k+1}.$$

Let

$$h(x) := (k+1)(x+1)^k - k(x-1)^{k+1}.$$

In the following, we will be interested in determining $\max_{x \geq 1} h(x)$. For $x = 1$, we have

$$h(1) = (k+1)2^k$$

and

$$h'(1) = k(k+1)2^{k-1}.$$

Since the negative term has higher order, we get that

$$\lim_{x \to \infty} h(x) = -\infty.$$

Thus, the maximum is attained for some $x \in (1, \infty)$. This implies that for the optimal x the following first-order conditions are satisfied:

$$(x+1)^{k-1} = (x-1)^k.$$

Let

$$B := \max\{(k+1)(x+1)^k - k(x-1)^{k+1} \colon x \geq 1\}.$$

We obtain the following equivalences:

$$\begin{aligned} B &= \max\{(k+1)(x+1)^k - k(x-1)^{k+1} \mid x \geq 1 \text{ and } (x+1)^{k-1} = (x-1)^k\} \\ &= \max\{(x+1)^{k-1}[(k+1)(x+1) - k(x-1)] \mid x \geq 1 \text{ and } (x+1)^{k-1} = (x-1)^k\} \\ &= \max\{(x+1)^{k-1}(x+2k+1) \mid x \geq 1 \text{ and } (x+1)^{k-1} = (x-1)^k\}. \end{aligned}$$

The equality constraint

$$(x+1)^{k-1} = (x-1)^k \tag{1.3}$$

can be reformulated using logarithmization as

$$(k-1)\log(x+1) = k\log(x-1)$$

or, equivalently,

$$k\big(\log(x+1) - \log(x-1)\big) = \log(x+1).$$

By the mean value theorem, we have

$$\log(x+1) - \log(x-1) = 2/\xi$$

for some $\xi \in (x-1, x+1)$ implying that any x solving equation (1.3) satisfies the inequalities

$$\frac{2}{x+1} \leq \frac{\log(x+1)}{k} \leq \frac{2}{x-1}.$$

This implies

$$B \leq \max\left\{(x+1)^{k-1}(x+2k+1) \;\middle|\; x \geq 1 \text{ and } \frac{2}{x+1} \leq \frac{\log(x+1)}{k} \leq \frac{2}{x-1}\right\},$$

since we made the domain of feasible x for the maximization only larger. The functions $x \mapsto 2/(x+1)$ and $x \mapsto 2/(x-1)$ are strictly decreasing and hence

$$\frac{2}{x+1} < \frac{2}{x-1}$$

for all $x \geq 0$. Moreover, $x \mapsto \log(x+1)/k$ is strictly increasing for all $k > 0$. This implies that

$$\left\{x \geq 0 \colon \frac{2}{x+1} \leq \frac{\log(x+1)}{k} \leq \frac{2}{x-1}\right\} = [\alpha, \omega],$$

where $\alpha > 0$ is the unique number with

$$\frac{2}{\alpha+1} = \frac{\log(\alpha+1)}{k}$$

and $\omega > 0$ is the unique number with

$$\frac{2}{\omega-1} = \frac{\log(\omega+1)}{k}.$$

Rearranging terms, we obtain

$$2k = (\alpha+1)\log(\alpha+1)$$
$$2k = (\omega-1)\log(\omega+1).$$

Recall that the inverse of the function $z \mapsto z\log(z)$ is $z \mapsto z/W(z)$ where W is the Lambert W-function, which is also referred to as product logarithm or omega function. This implies

$$\alpha = 2k/W(2k) - 1.$$

Using that

$$\frac{\log(x+1)}{k} > \frac{\log(x-1)}{k}$$

for all $x \geq 0$ and both functions are strictly increasing, we derive that ω is bounded from above by the unique number ω' satisfying

$$2k = (\omega'-1)\log(\omega'-1)$$

which implies

$$\omega' = 2k/W(2k) + 1.$$

We have established

$$B \leq \max\Big\{(x+1)^{k-1}(x+2k+1) \mid x \in \big[\max\{1,\alpha\},\omega'\big]\Big\}.$$

Using that $(x+1)^{k-1}(x+2k+1)$ is increasing in x, this gives

$$\begin{aligned} B &\leq (\omega'+1)^{k-1}(\omega'+2k+1) \\ &= \Big(\frac{2k}{W(2k)}+2\Big)^{k-1}\Big(\frac{2k}{W(2k)}+2k+2\Big). \end{aligned}$$

Finally, we use the asymptotics of the Lambert W-function which are due to Corless et al. [CGH+96] who showed

$$W(z) = \log(z) - \log\log z + \frac{\log\log z}{\log z} + \mathcal{O}\Big(\Big(\frac{\log\log z}{\log z}\Big)^2\Big)$$

and obtain

$$B \leq \Big(\frac{2k}{\log(2k)-\log\log(2k)}+2\Big)^{k-1}\Big(\frac{2k}{\log(2k)-\log\log(2k)}+2k+2\Big).$$

Since the right hand side is increasing in k, we conclude that there is an α-approximation algorithm for the unweighted resource allocation problem with monomials of type $c(x) = x^k$ for some $k \leq d$, where

$$\alpha = \Big(\frac{2d}{\log(2d) - \log\log(2d)} + 2\Big)^{d-1}\Big(\frac{2d}{\log(2d) - \log\log(2d)} + 2d + 2\Big).$$

Clearly,

$$\frac{2d}{\log(2d) - \log\log(2d)} + 2d + 2 \leq \Big(\frac{2d}{\log(2d) - \log\log(2d)}\Big)^2$$

for d large enough. This implies

$$\alpha \in \mathcal{O}\bigg(\Big(\frac{2d}{\log(2d) - \log\log(2d)}\Big)^{d+1}\bigg) \subseteq \mathcal{O}\big((2d/\log d)^{d+1}\big).$$

To show the result for arbitrary polynomials with non-negative coefficients, note that (1.1) is invariant under multiplication of c by a positive scalar $a > 0$. This implies that any resource allocation problem with monomials of the type $c(x) = ax^k$ with $k \leq d$ and $a > 0$ yields the desired approximation factor. Finally, note that any resource allocation problem with polynomial costs can be transformed into an equivalent resource allocation problem with scaled monomial costs by splitting up resources. This concludes the proof. □

For given concrete values of d, we can also directly solve the optimization problem $\max_{x,y\in\mathbb{N}} g_c(x,y)$ for $c(x) = x^d$. However, handling the integrality constraints in a satisfactory way turns out to be somewhat intricate as the following theorem provides.

Theorem 1.11 *For any $\epsilon > 0$ there is a polynomial time $(\alpha_d + \epsilon)$-approximation for every unweighted resource allocation problem with polynomial cost functions with maximal degree d and non-negative coefficients, where $\alpha_1 = 3$, $\alpha_2 = 13$, $\alpha_3 = 61$, $\alpha_4 = 391$, and $\alpha_5 = 2,157$.*

Proof. The approximation factor for $d = 1$ has been established in Proposition 1.6.

In the following we show how to optimize $g_c(x,y)$ (1.2) for small d providing better bounds on α. By the argumentation above from Theorem 1.10, we can assume that $c(x) = x^d$. We want to maximize $g_c(x,y)$ for $x, y \in \mathbb{N}$ and given $d \in \mathbb{N}$. Using Lemma 1.8 with $\mu = 0$, the maximal value of $g_c(x,y)$ upper bounds the approximation factor, denoted by α_d.

In a maximum of $g_c(x,y)$ we have the inequalities

$$g_c(x+1,y) - g_c(x,y) \leq 0, \tag{1.4}$$
$$g_c(x-1,y) - g_c(x,y) \leq 0, \tag{1.5}$$
$$g_c(x,y+1) - g_c(x,y) \leq 0,$$
$$g_c(x,y-1) - g_c(x,y) \leq 0.$$

For monomial costs with degree d, inequality (1.4) simplifies to

$$\frac{1}{y^{d+1}}\Big(y(x+2)^{d+1} - (2y+x)(x+1)^{d+1} + (2x-y)x^{d+1} + x(x-1)^{d+1}\Big) \leq 0.$$

Solving this, we get

$$y \leq x \frac{(x+1)^{d+1} - 2x^{d+1} + (x-1)^{d+1}}{(x+2)^{d+1} - 2(x+1)^{d+1} + x^{d+1}}. \tag{1.6}$$

Inequality (1.5) simplifies to

$$\frac{1}{y^{d+1}} \Big(y(x+1)^{d+1} - (x-1)x^{d+1} \\ - (y+2x-3)(x-1)^{d+1} + (x-1)(x-2)^{d+1} \Big) \leq 0.$$

Solving this, we get

$$y \geq \frac{(x-1)x^{d+1} - 2(x-1)^{d+2} + (x-1)(x-2)^{d+1}}{(x+1)^{d+1} - 2x^{d+1} + (x-1)^{d+1}}. \tag{1.7}$$

Thus, y is the unique number in $\mathbb{N}$ for which (1.6) and (1.7) hold.

Let $d = 2$. Then, by the calculation above,

$$y \in \left[\frac{x^2 - 2x + 1}{x}, \frac{x^2}{x+1} \right] = \left[x - 2 + \frac{1}{x}, x - 1 + \frac{1}{x+1} \right].$$

For $x \in \mathbb{N}$, this is equal to $y = x - 1$, as the latter summands in both ends of the interval are smaller than 1. Putting $y = x - 1$ into (1.2), we get

$$g(x, x-1) = 1 + \frac{-4x^2 + 5x - 1}{x^3 - 3x^2 + 3x - 1}$$

and using first order conditions, this is maximal for $x = 2$. Then, $g_c(2,1) = 13$ which establishes the approximation factor for $d = 2$.

Let $d = 3$. Then,

$$y \in \left[\frac{6x^3 - 18x^2 + 19x - 7}{6x^2 + 1}, \frac{6x^3 + x}{6x^2 + 12x + 7} \right] \\ = \left[x - 3 + \frac{18x - 4}{6x^2 + 1}, x - 2 + \frac{18x + 14}{6x^2 + 12x + 7} \right].$$

For $x \geq 3$, y is equal to $x - 2$ as the latter summands are smaller than one for $x \geq 3$. For $x = 1$ and $x = 2$, no natural number lies in these bounds. Putting $y = x - 2$ into $g_c(x, y)$ (1.2), we get that the maximum is attained for $x = 3$. Then $g_c(3,1) = 61$ which establishes the approximation factor for $d = 3$.

Let $d = 4$. Then,

$$y \in \left[x - 4 + \frac{12x^2 - 6x + 3}{2x^3 + x}, x - 3 + \frac{12x^2 + 18x + 9}{2x^3 + 6x^2 + 7x + 3} \right].$$

For $x \geq 4$, y is equal to $x - 3$. For $x = 1$ and $x = 2$, no natural number lies within these bounds. For $x = 3$, y is equal to one. Putting these results into $g_c(x, y)$ (1.2), we get that the maximum is attained for $x = 3$. Then $g_c(3,1) = 391$ which establishes the approximation factor for $d = 4$.

d	x	y	α_d	d	x	y	α_d	d	x	y	α_d
1	2	1	3	6	4	1	21337	11	6	1	2245436935
2	2	1	13	7	4	1	154725	12	6	1	25849065061
3	3	1	61	8	5	1	1622791	13	6	1	244659182023
4	3	1	391	9	5	1	16880931	14	7	1	5242736411263
5	4	1	2157	10	5	1	139627951	15	7	1	68592232075641

Table 1.2: Calculating approximation factors for unweighted resource allocation problem with polynomial cost functions with maximal degree d and non-negative coefficients.

Let $d = 5$. Then,

$$y \in \left[x - 5 + \frac{150x^3 - 120x^2 + 120x - 26}{15x^4 + 15x^2 + 1}, x - 4 + \frac{120x^3 + 330x^2 + 330x - 124}{15x^4 + 60x^3 + 105x^2 + 90x + 31} \right].$$

For $x \geq 10$, y is equal to $x - 4$. For $x \in \{1, 2, 3, 7, 8, 9\}$, no natural number lies within these bounds. For $x \in \{4, 5, 6\}$, y is equal to $x - 3$. Putting these results into $g_c(x, y)$(1.2), we get that (1.2) reaches its maximum at $x = 4$ and $y = 1$. Then we get $g_c(4, 1) = 2157$ which establishes the approximation factor for $d = 5$.

With similar calculations for the values $d = 6, \ldots, 10$, we derived the values in the second column of Table 1.2. Intriguingly, for $d \geq 3$, we attain maximal values in $g_c(x, y)$ for

$$(x, y) = \left(\left\lfloor \frac{d+1}{3} \right\rfloor + 2,\ 1 \right),$$

as suggested by Table 1.2. □

1.3 Lower Bounds

We provide and analyze a local search approximation algorithm in the last section and therewith give an upper bound on the locality gap and on the approximation factor of unweighted resource allocation problems. We will complement this work in this section and give a lower bound on the locality gap and on possible approximation guarantees.

1.3.1 Lower Bound on the Locality Gap

For the case of linear cost functions, it was shown by Chen et al. [CDKKS11] that the bound on the locality gap of three is tight. In this section, we provide a lower bound of $\left\lfloor \frac{d}{\log(d+2)} - 1 \right\rfloor^{d+1}$ on the locality gap for general polynomials with maximum degree d.

Theorem 1.12 *There is an unweighted resource allocation problem with monomial costs with degree d and locality gap $\left\lfloor \frac{d}{\log(d+2)} - 1 \right\rfloor^{d+1}$.*

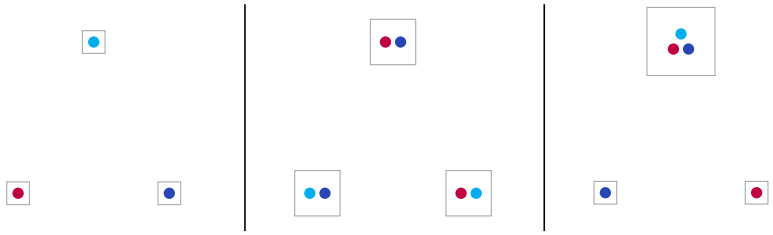


Figure 1.1: Example for an unweighted resource allocation problem. Each commodity is represented by one color. The cost for each resource, which are represented by grey squares, is x^2; depending on how many commodities are using the resources, the rectangles change accordingly in size to represent their change in cost. In the left picture, an optimal solution where each commodity i uses resource r_i is displayed, its cost is $1^2 \cdot 1 \cdot 3 = 3$. In the picture in the middle, a local optimal solution is displayed: each commodity uses the resources $R \setminus r_i$. The cost incurred in this solution is the cost of the resource times the number of commodities using that resource times the number of resources, thus $2^2 \cdot 2 \cdot 3 = 24$. This evaluates to a locality gap of $24/3 = 8 = \lfloor \frac{2}{\log(4)} - 1 \rfloor^3$. In the picture on the right, it is shown why this is indeed a local optimal solution: if only one commodity changes resources, the total cost is not reduced, as we can calculate: $3^2 \cdot 3 \cdot 1 + 1^2 \cdot 1 \cdot 2 = 29 \geq 24$.

Proof. For any given $d > 0$ and any $x \in \mathbb{Z}_{>0}$, we design the following resource allocation problem with x commodities and x resources. We are given a set of resources $R = \{r_1, \ldots, r_x\}$, where $c_r(y) = y^d$ for $r \in R$. We define the resource allocation problem in such a way that every commodity i can only be assigned to either r_i or its complement $R \setminus \{r_i\}$. Please refer also to Figure 1.1.
Clearly,

$$\mathbf{s}' = (\{r_1\}, \ldots, \{r_x\})$$

is a global optimal solution with

$$C(\mathbf{s}') = x.$$

We proceed by showing that

$$\mathbf{s} = (R \setminus \{r_1\} \times \ldots \times R \setminus \{r_x\})$$

is also a local optimal solution with

$$C(\mathbf{s}) = x(x-1)^{d+1}.$$

In order to do so, we show that assigning a commodity i to resource r_i instead of to $R \setminus \{r_i\}$ does not decrease the total cost. We have

$$C(\{r_i\}, \mathbf{s}_{-i}) - C(\mathbf{s}) = x^{d+1} + (x-1)(x-2)^{d+1} - x(x-1)^{d+1}.$$

We show that this difference is larger or equal to zero for all $x \leq \left\lfloor \frac{d}{\log(d+2)} \right\rfloor$. We obtain

$$\frac{\log(d+2)}{d} \leq \frac{1}{x} \leq \log(x) - \log(x-1)$$

which means

$$\log(d+2) + d\log(x-1) \leq d\log x$$

and thus

$$(x-1)^d(d+2) \leq x^d.$$

Note that

$$(x-1)^{d+1} - (x-2)^{d+1} \leq (d+1)(x-1)^d$$

as the function $y \mapsto y^{d+1}$ is convex with derivative $(d+1)y^d$. By using further that $(x-2)/x \leq 1$, this implies

$$\begin{aligned}\left[(x-1)^{d+1} - (x-2)^{d+1}\right] + \frac{1}{x}(x-2)^{d+1} &\leq (x-1)^d(d+1) + (x-1)^d \\ &= (x-1)^d(d+2).\end{aligned}$$

Putting both inequalities together and multiplying by x we get

$$x(x-1)^{d+1} - (x-1)(x-2)^{d+1} \leq x^{d+1}.$$

We can conclude that $\mathbf{s}$ is a local optimal solution, as no change of strategy by a single commodity leads to a smaller total cost.

Then we calculate the locality gap as

$$\begin{aligned}\frac{C(\mathbf{s})}{C(\mathbf{s}')} = \frac{x(x-1)^{d+1}}{x} &\leq (x-1)^{d+1} \\ &\leq \left(\left\lfloor \frac{d}{\log(d+2)} \right\rfloor - 1\right)^{d+1}\end{aligned}$$

which finishes the proof. □

1.3.2 APX-hardness for Linear Costs

Roughgarden [Rou14] showed the existence of a constant $\beta > 0$ such that there is no $(\beta d)^{d/2}$ approximation algorithm for unweighted resource allocation problems with cost functions from $\mathcal{C}^d$ unless P = NP. Since this result does not give a concrete value for a small d, we complement it by providing an inapproximability result for $d = 1$.

We show by a reduction from MaxCut that there is no approximation algorithm with a factor better than 1.02 for computing an optimal solution for unweighted resource allocation problems with linear cost functions, unless P = NP. This implies APX-hardness, meaning that there is no polynomial time approximation scheme or in other words, optimal solutions for this problem cannot be approximated up to a factor of $1+\epsilon$ for every $\epsilon > 0$.

We start by stating the maximum cut problem, stylised as MaxCut.

Definition 1.13 (MaxCut) *We are given an undirected graph $G = (V, E)$. Each subset $X \subseteq V$ of the vertices defines a* cut, *that is the division of the vertex set in two subsets X and $V \setminus X$. The* size of a cut $\delta(X)$ *is defined as the number of edges in that cut, that is $\delta(X) = |\{e = \{u, v\} \in E : u \in X, v \in V \setminus X\}|$. The* maximum cut problem, *also stylised as* MaxCut*-problem, is the problem to find a cut X^* of maximum size, that is, for each cut $X \subseteq V$ of G holds $\delta(X) \leq \delta(X^*)$.*

Trevisan et al. [TSSW00] showed that it is NP-hard to approximate MaxCut within factor $\frac{16}{17} + \epsilon$. Hastad [Hås01] was able to show that this result is in fact tight. We use this inapproximability factor to derive APX-hardness for unweighted resource allocation problems with linear costs. For the proof, we need a structural result about the size of maximum cuts.

Lemma 1.14 *Let X^* be a maximum cut of a graph $G = (V, E)$, and $\delta(X^*)$ be the number of edges in that cut. Then $\delta(X^*) \geq 1/2|E|$.*

Proof. We prove this statement by a probabilistic argument. Let X be an arbitrary cut of G, defined by placing each vertex $v \in V$ with probability $1/2$ in X, and probability $1/2$ in $V \setminus X$. Each edge is then with probability $1/2$ in the cut set defined by X, as the two incident vertices of an edge have probability $1/4$ of being both in X, $1/4$ of being both in $V \setminus X$, and with probability $1/2$ both vertices are in different sets und thus in the cut. We thus obtain

$$\mathbb{E}(\delta(X)) = \frac{1}{2}|E|.$$

Using the probabilistic method as introduced by Erdős [Erd47], we conclude: as X was chosen arbitrarily, there exists at least one cut X^* with size at least the expected value, that is $1/2|E|$. In particular, for a maximum cut X^* holds that

$$\delta(X^*) \geq \frac{1}{2}|E|.$$

□

We are now able to state the main theorem of this section.

Theorem 1.15 *There is no polynomial time α-approximation algorithm for unweighted resource allocation problems with linear cost functions for any $\alpha < 52/51$, unless $P = NP$.*

Proof. Consider the following reduction of MaxCut. Given an instance $G = (V, E)$ of MaxCut, construct the following resource allocation problem. For each $v \in V$ we introduce a commodity. For each edge $\{v, w\}$ we introduce two resources $r_{\{v,w\}1}$ and $r_{\{v,w\}2}$. Remark that $r_{\{v,w\}i} = r_{\{w,v\}i}$ for $i \in \{1, 2\}$. The commodity corresponding to a node v has exactly two feasible solutions. She can choose between a solution S_{v1} consisting of resources

$$\left\{ r_{\{v,n\}1} | n \in \mathcal{N}(v) \right\}$$

and solution S_{v2} consisting of resources

$$\left\{ r_{\{v,n\}2} | n \in \mathcal{N}(v) \right\}.$$

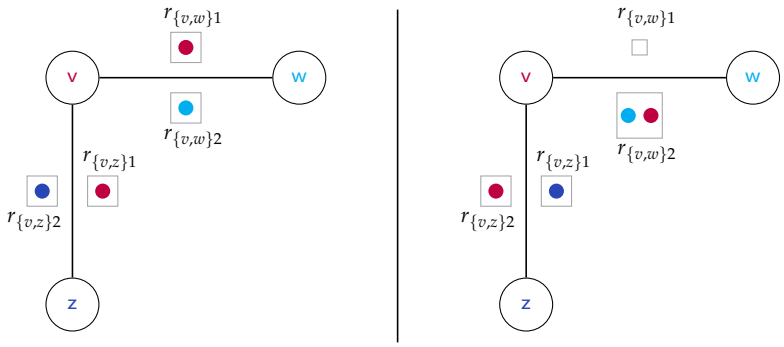


Figure 1.2: Schematic for constructing a resource allocation problem for every instance of MAXCUT. The graph from the MAXCUT-instance is underlying light grey. Each edge corresponds to two resources. Each commodity has two possible strategies: choosing all resources on their incident edges with index one (purple on the left, blue on the right) or choosing all resources on their incident edges with index two (light blue and blue on the left, purple and light blue on the right). The incurred cost is $4 \cdot 1 = 4$ on the left and $2 \cdot 1 + 2 \cdot 2 = 6$ on the right.

Let the cost of each resource correspond to the number of commodities using that resource, i. e. , $c_r = x$. Please refer also to Figure 1.2.

The idea is that the overall solution gets expensive, if two neighbors v and w pick solutions S_{v1} and S_{w1} or solutions S_{v2} and S_{w2}. In this case, they both use resources $r_{\{v,w\}1} = r_{\{w,v\}1}$ or $r_{\{v,w\}2} = r_{\{w,v\}2}$. The overall solution becomes cheap if they pick different solutions and thus do not share a resource. For a given cut $X \subseteq V$ in G, let $\delta(X)$ be the number of edges in the cut, i. e.

$$\delta(X) = |\{(u,v)|u \in X, v \notin X\}|.$$

Note that each cut $X \in V$ has a one to one correspondence to all commodities corresponding to vertices $v \in X$ picking S_{v1} and all other commodities corresponding to $w \in V \setminus X$ picking S_{w2}. So, for a given cut X we can calculate the cost of the corresponding solution to the resource allocation problem as follows:

$$C(S) = \sum_{v \in V} \delta_v + 2\left(|E| - \delta(X)\right) = 4|E| - 2\delta(X),$$

since each edge is responsible for a cost of at least two if it is in the cut, and each non-cut edge is responsible for an additional cost of two, paying $2 \cdot 2$ instead of $2 \cdot 1$.

Now suppose there is a polynomial time α-approximation algorithm for the resource allocation problem. We can conclude that we can calculate a solution S (and corresponding cut X) with the following cost:

$$C(S) \leq \alpha C(S^*),$$

and plugging the cost in we get

$$4|E| - 2\delta(X) \leq \alpha\left(4|E| - 2\delta(X^*)\right)$$

which can be transformed to

$$2|E|(1-\alpha) \leq \delta(X) - \alpha\delta(X^*)$$

and finally

$$2|E|(1-\alpha)+\alpha\delta(X^*) \leq \delta(X).$$

It is worth noting that $(1-\alpha)$ is negative as the resource allocation problem is a minimization problem. Now we use the fact from Lemma 1.14 that

$$\frac{1}{2}|E| \leq \delta(X^*)$$

to get

$$\delta(X^*)\,(4-3\alpha) \leq \delta(X)$$

which is equivalent to

$$4-3\alpha \leq \frac{\delta(X)}{\delta(X^*)}.$$

Now suppose that $\alpha < 52/51$. We obtain

$$4-3\cdot\frac{52}{51} < \frac{\delta(X)}{\delta(X^*)}$$

and finally

$$\frac{16}{17} < \frac{\delta(X)}{\delta(X^*)}.$$

But this is a contradiction, as it was shown by Håstad [Hås01] that MAXCUT is not polynomial-time-approximable beyond a factor of $16/17$, unless P = NP. We conclude that there is no α-approximation with $\alpha < 52/51 \approx 1.02$ for the unweighted resource allocation problem with linear costs. □

In 2002, Khot made an interesting conjecture in his seminal paper "On the power of unique 2-prover 1-round games", labeled the *Unique Games Conjecture* [Kho02], which gives some insights into approximability of a number of problems. Using this result, we can formulate an even better lower bound for our problem. We start by citing the conjecture.

Conjecture 1.16 (Unique Games Conjecture UGC [Kho02]) *Given a constant $k \in \mathbb{N}$, a directed graph $G = (V, E)$, and a collection of permutations $\pi_e \colon [k] \to [k]$. Each π_e imposes a constraint on a map $L \colon V \to [k]$ in the following sense. The constraint for edge (u, v) is fulfilled, if and only if $\pi_e(L(v)) = L(u)$.*

Given sufficiently small $\epsilon > 0$ and $\delta > 0$, there is a constant k such that it is NP-hard to decide whether there is a map L that fulfills a $(1-\delta)$ fraction of all constraints or all maps do not fulfill more than an ϵ fraction of the constraints.

Assuming the unique games conjecture [Kho02], there is no approximation algorithm for MAXCUT with a factor better than 0.878 as shown by Khot et al. [KKMO07]. We can thus derive an even tighter bound for unweighted resource allocation problems with linear costs.

Theorem 1.17 *Assuming the unique games conjecture [Kho02], there is no polynomial time α-approximation algorithm for unweighted resource allocation problems with linear cost functions for any $\alpha < 1.04$, unless $P = NP$.*

Proof. As proven by Khot et al. [KKMO07], we know that, if the unique games conjecture holds true, MAXCUT is not polynomial-time-approximable beyond a factor of 0.878. Suppose there is a polynomial time α-approximation algorithm for the resource allocation problem. Now suppose that $\alpha < 1.04$. Using the proof to Theorem 1.15, we obtain from

$$4 - 3\alpha \leq \frac{\delta(X)}{\delta(X^*)}$$

that

$$4 - 3 \cdot 1.04 < \frac{\delta(X)}{\delta(X^*)}$$

which is equivalent to

$$0.88 < \frac{\delta(X)}{\delta(X^*)}.$$

But this is a contradiction, as it was shown by Khot et al. [KKMO07] that, supposing the Unique Games Conjecture, MAXCUT is not polynomial-time-approximable beyond a factor of 0.878, unless P = NP. We conclude that, if the UGC holds true, there is no α-approximation with $\alpha < 1.04$ for the unweighted resource allocation problem with linear costs. □

1.4 Further Results for Weighted Problems

Using the same algorithm and techniques as presented before, we can also obtain results for the weighted variant of the problem, which we want to state here.

Analyzing the local search algorithm presented in Algorithm 1.1 for resource allocation problems with weighted commodities, we arrive at the following conclusion.

Theorem 1.18 ([BKS17]) *For every weighted resource allocation problem with polynomial costs with non-negative coefficients and maximal degree d, the following hold for*

$$\alpha = \frac{1}{(\sqrt[d+1]{2} - 1)^{d+1}} \in \mathcal{O}\left(\left(\frac{d+1}{\log 2}\right)^{d+1}\right):$$

1. *The locality gap is at most α.*
2. *For any $\epsilon > 0$, there is a polynomial time $\alpha + \epsilon$-approximation algorithm for the minimization of the total costs.*

Further, we can design a resource allocation problem that gives a lower bound on the locality gap.

Theorem 1.19 ([BKS17]) *There is a resource allocation problem with monomial costs with degree d and locality gap arbitrarily close to*

$$\frac{1}{\left(\sqrt[d+1]{2}-1\right)^{d+1}} \in \mathcal{O}\left(\left(\frac{d+1}{\log 2}\right)^{d+1}\right).$$

Thus, our algorithm yields the first deterministic, the first combinatorial, and the first distributed algorithm with non-trivial approximation guarantee.

1.5 Discussion and Open Problems

In this chapter, we have dealt with the problem of rerouting users through a congested network with the goal to reduce the overall network transportation cost. We formulated it in more general terms as a resource allocation problem with diseconomies of scale. This approach provides background to improve routing strategies and can therefore help to reduce congestion, which subsequently leads, e. g. , to shorter traversing times in networks such as streets in a city, thus reducing environmental impact and personal stress.

Our idea was to present and analyse a simple algorithm which can be implemented in a distributed manner and works for general resource allocation problems. It computes a local optimum in the sense that no change of strategy of a single player improves the overall network cost by a factor of at least $(1+\delta)$. Its strength thus lays in its simplicity, which makes it easy to explain and versatile. The algorithm is also deterministic and combinatorial and can be implemented in a distributed manner.

In addition to presenting this local search algorithm, we provided a framework to analyse the locality gap of resource allocation problems using (λ, μ)-smoothness. Using the framework, this leads to an $(\alpha+\epsilon)$-approximation and locality gap of α with $\alpha \in \mathcal{O}((2d/\log d)^{d+1})$ for the case of unweighted commodities and general polynoms with maximum degree d and also to specific values for small d. This seems particularly interesting, as popular models for travel time functions in travel networks are of degree four [US 64]. However, calculating concrete values for smaller d turns out to be rather involved, so there may be a different approach which leads to further results in this case. This appears especially interesting with regard to the work of Awerbuch et al. [AAE+08], as our results suggest that our algorithm performs better for values of $d \geq 9$.

Additionally, our general framework using smoothness can be applied to resource allocation problems with different and possibly more intricate cost functions. This leaves space for further work on this direction.

Moreover, the question remains open whether a more refined smoothness argument may lead to new approximation guarantees.

To complement the upper bound on the locality gap, we constructed an instance which leads to a lower bound on the locality gap of $\left\lfloor \frac{d}{\log(d+2)} - 1 \right\rfloor^{d+1}$. Clearly, closing the gap remains a main goal of further work. With regards to the problems that appear in practice, this seems desirable even just for the special cases of small d.

In this context, we gave a lower bound on α-approximation algorithms for the

case of linear costs, or $d = 1$, showing that none such exists for $\alpha < 1.02$. This complements Roughgarden [Rou09], who did not give concrete values for small d. Intriguingly, if the Unique Games Conjecture by Khot [Kho02] holds true, our lower bound can even be tightened up to $\alpha < 1.04$. As many other recent results do, this points to the importance of settling the open conjecture.

Keeping the question of rerouting in traffic networks in mind, also the question of how to solve the task in a distributed or more parallel manner warrants attention. Our local search algorithm relies on an oracle which names commodities which should switch to a different set of resources. It can be modified to work in a more parallel way. For instance, consider the following algorithm: each commodity calculates whether there is a different set of resources for itself which will lead to a lower total cost if switched to. If there is, a switch is executed with a certain probability p and all other commodities are updated to the new situation. Clearly, if two commodities switch at the same time, this might lead, in fact, to higher total costs. Therefore, as the switching occurs over time, the probability that two or more commodities switch resources at the same time should be reasonably low. Preliminary work suggests that the switching probability may lie between $1/n$ and $1/n^2$, where n is the total number of commodities.

Chapter 2

Non-monetary Access Control Mechanisms

We study mechanisms that select members of a set of agents based on nominations by other members, and that are impartial in the sense that agents cannot influence their own chance of selection. Prior work has shown that deterministic mechanisms for selecting any fixed number k of agents are severely limited and cannot extract a constant fraction of the nominations of the k most highly nominated agents. We prove here that this impossibility result can be circumvented by allowing the mechanism to sometimes, but not always select fewer than k agents. This added flexibility also improves the performance of randomized mechanisms, for which we show a separation between mechanisms that make exactly two or up to two choices, and give upper and lower bounds for mechanisms allowed more than two choices.

Bibliographic Information: The results in this chapter are based on joint work with Felix Fischer and Max Klimm, published at the 11th Conference on Web and Internet Economics (WINE) 2015 [BFK15] and in the ACM Transactions on Economics and Computation [BFK17].

Since the early 2000s, the concept of smart cities has gained momentum and public interest in it has grown. The words Smart City in this context are used to address technology based changes and innovations in urban areas. Giffinger et al. [GFK+07] name as characteristics of a smart city in 2007 smart economy, people, governance, mobility, environment and living. As evident by that, reducing congestion and increasing mobility are almost universal issues for cities to address (see, e. g. , [Sor16]) and are, e. g. , tackled in the Horizon 2020 work programme 2018–2020 [Eur17] by the European Commission as well as in the Smart City Challenge by the U.S. Department of Transportation [US 16]. A smart city needs effective transport management to reduce congestion. It can build on new technologies like algorithms for big data, but also on other quick and more effective tools than hitherto used to coordinate efforts.

While rerouting users reduces the amount of traffic that happens on particular roads, we can achieve a similar effect by reducing the total number of users in a network. In order to still transport what is needed, this means that users have to work together, which is, as established above, also key for a smart city. Indeed, it has been shown that co-operating between transport companies leads to a decrease in transport emissions of 30% [POP13]. Providing frameworks for cooperation is thus

a promising path to take towards reducing the environmental and societal impact of transport networks. This chapter aims at providing tools to make fair decisions, e. g. in such coordination processes.

We start with the real world roots of our problem in Section 2.1. They lead us to a formal problem definition, for which we present related work. In the main part of this chapter, we first give deterministic and randomized mechanisms for choosing two agents out of a group in Section 2.2, one based on partition and one on permutation. We then adapt those mechanisms to choose more than two agents in Section 2.3 and analyze them consecutively. In addition, we present another approach based on dollar partition. To complement the lower bounds that all of those algorithms provide, we calculate some upper bounds in Section 2.4. Finally, in Section 2.5 we discuss our findings and point out open problems.

2.1 Background

As an example, consider a telecommunication channel with fixed capacity that a number of users plans to access. An access mechanism asks nominations from the users about which of the other users should be given access to the channel. It is natural to assume that the main interest of each user is to be given access, so a selection mechanism must take into account that users may misreport their opinion about who they think is eligible for access as long as they can increase their own chances of being given access. It is a natural question what percentage of the nominations a mechanism must lose in order to be strategyproof , i. e. , in order to prevent misreporting of that kind. Clearly, this problem also appears in other network contexts, for instance, in choosing which person shall drive a carpool. Looking at the requirements, we can formulate the following mathematical problem.

2.1.1 Formulating the Mathematical Problem

We consider the setting of impartial selection which was first studied by Alon et al. [AFPT11] and by Holzman and Moulin [HM13]. The goal in this setting is to select members of a set of agents based on nominations cast by other members of the set, under the assumption that any agent will reveal her true opinion about other agents as long as she cannot influence her own chance of selection. The assumption of impartiality seems justified, and is routinely made, in many situations where a strong correlation exists between expertise and self-interest, like the selection of representatives from within a group and the use of peer review in the allocation of funding and scientific or academic credit.

Formally, the impartial selection problem can be modeled by a directed graph with n vertices, one for each agent, in which edges correspond to nominations. A selection mechanism then chooses, possibly using randomization, a set of vertices for any given graph, and it is impartial if the chances of a particular vertex to be chosen do not depend on its outgoing edges. As impartiality may prevent us from simply selecting the vertices with maximum indegree corresponding to the most highly nominated agents, it is natural to instead approximate this objective. For

an integer k, a selection mechanism is called a *k-selection mechanism* if it selects at most k vertices of any input graph. We call a k-selection mechanism *exact* if it always selects exactly k agents. A k-selection mechanism is called *α-optimal*, for $\alpha \leq 1$, if for any input graph the sum of indegrees of the selected vertices is at least α times the sum of the k largest indegrees.

In prior work, a striking separation was shown between mechanisms that do not use randomness and those that do. On one hand, no deterministic exact α-optimal mechanism exists for selecting any fixed number of agents and any $\alpha > 0$ [AFPT11]. On the other, a mechanism that aligns the agents along a random permutation from left to right and selects a single agent with a maximum number of nominations from its left achieves a bound of $\alpha = 1/2$ [FK15]. This bound is in fact best possible subject to impartiality [AFPT11].

Notation and Formal Problem Definition For $n \in \mathbb{N}$, let

$$\mathcal{G}_n = \big\{(N, E) : N = \{1, \dots, n\}, E \subseteq \{(i, j) \in N \times N : i \neq j\}\big\}$$

be the set of simple directed graphs with n vertices. Further, let $\mathcal{G} = \bigcup_{n \in \mathbb{N}} \mathcal{G}_n$. For $G = (N, E) \in \mathcal{G}$ and $S, X \subseteq N$ let

$$\delta_S^-(X, G) = |\{(j, i) \in E : G = (N, E), j \in S, i \in X\}|$$

denote the sum of indegrees of vertices in X from vertices in S. We use $\delta^-(X, G)$ as a shorthand for $\delta_N^-(X, G)$ and denote the maximal sum of indegrees of a set of k vertices by

$$\Delta_k(G) = \max_{X \subseteq N, |X| = k} \delta^-(X, G).$$

When $X = \{i\}$ for a single vertex i, we write $\delta_S^-(i, G)$ instead of $\delta_S^-(\{i\}, G)$. Most of the time, the graph G will be clear from context. We then write $\delta_S^-(X)$ instead of $\delta_S^-(X, G)$, $\delta^-(X)$ instead of $\delta^-(X, G)$, and Δ_k instead of $\Delta_k(G)$.

For $n, k \in N$, let $\mathcal{X}_n = \big\{X : X \subseteq \{1, \dots, n\}\big\}$ be the set of subsets of the first n natural numbers and let $\mathcal{X}_{n,k} = \{X \in \mathcal{X}_n : |X| = k\}$ be the subset of these sets with cardinality k. A *k-selection mechanism* for $\mathcal{G}$ is then given by a family of functions

$$f \colon \mathcal{G}_n \to [0, 1]^{\bigcup_{\ell=0}^{k} \mathcal{X}_{n,\ell}}$$

that maps each graph to a probability distribution on subsets of at most k of its vertices. In a slight abuse of notation, we use f to refer to both the mechanism and individual functions from the family.

We call mechanism f *deterministic* if $f(G) \in \{0, 1\}^{\bigcup_{\ell=0}^{k} \mathcal{X}_{n,\ell}}$, i. e. if $f(G)$ puts all probability mass on a single set for all $G \in \mathcal{G}$; and we call it *exact* if $\big(f(G)\big)_X = 0$ for every $n \in \mathbb{N}$, $G \in \mathcal{G}_n$, and $X \in \mathcal{X}_n$ with $|X| < k$, i. e. if the mechanism never selects a set X of vertices with strictly less than k vertices.

Mechanism f is *impartial* on $\mathcal{G}' \subseteq \mathcal{G}$ if on this set of graphs the probability of selecting vertex i does not depend on its outgoing edges, i. e. if for every pair of graphs $G = (N, E)$ and $G' = (N, E')$ in $\mathcal{G}'$ and every $i \in N$,

$$\sum_{X \in \mathcal{X}_n, i \in X} (f(G))_X = \sum_{X \in \mathcal{X}_n, i \in X} (f(G'))_X$$

whenever $E \setminus (\{i\} \times N) = E' \setminus (\{i\} \times N)$. It is *universally impartial* if it is a convex combination of deterministic impartial mechanisms, i. e. if there exist deterministic impartial mechanisms $f_1, \dots, f_m$ and $a_1, \dots, a_m \geq 0$ such that $\sum_{i=1}^{m} a_i = 1$ and for all $G \in \mathcal{G}'$, $f(G) = \sum_{i=1}^{m} a_i f_i(G)$. Note that while impartiality requires the outgoing edges of a vertex i to have no influence at all on whether i is selected or not, they may influence both the number and the identities of other vertices selected. All mechanisms we consider are impartial or even universally impartial on $\mathcal{G}$, and we simply refer to such mechanisms as impartial or universally impartial mechanisms.

Finally, a k-selection mechanism f is *α-optimal* on $\mathcal{G}' \subseteq \mathcal{G}$, for $\alpha \leq 1$, if for every graph in $\mathcal{G}'$ the expected sum of indegrees of the vertices selected by f differs from the maximum sum of indegrees for any k-subset of the vertices by a factor of at most α, i. e. if

$$\inf_{\substack{G \in \mathcal{G}' \\ \Delta_k(G) > 0}} \frac{\mathbb{E}_{X \sim f(G)}[\delta^-(X, G)]}{\Delta_k(G)} \geq \alpha.$$

We call a mechanism α-optimal if it is α-optimal on $\mathcal{G}$.

For randomized mechanisms, and as far as impartiality and α-optimality are concerned, we can restrict attention to mechanisms that are *symmetric*, i. e. invariant with respect to renaming of the vertices (see [FK15]). It may further be convenient to view a k-selection mechanism as assigning probabilities to vertices rather than sets of vertices, with the former summing to at most k or exactly k for each graph. By the Birkhoff-von Neumann theorem [Bir46], the two views are equivalent in the following way.

Lemma 2.1 *Let $n \in \mathbb{N}$, $p \in [0,1]^n$, and $m = \sum_{i=1}^{n} p_i$. Then there exists a random variable Y with values in $[0,1]^{\mathcal{X}_{n,\lfloor m \rfloor} \cup \mathcal{X}_{n,\lceil m \rceil}}$ such that for all $i \in \{1, \dots, n\}$ holds*

$$\sum_{X \in \mathcal{X}_n, i \in X} \mathbb{P}[Y = X] = p_i.$$

Proof. First consider the case where m is an integer, and let $M = \{1, \dots, m\}$ and let $\bar{M} = \{m+1, \dots, n\}$. Since $\sum_{i=1}^{n} p_i = m$, there exists $Q \in [0,1]^{n \times n}$ such that

$$\sum_{j \in M \cup \bar{M}} q_{ij} = 1 \text{ and } \sum_{j \in M} q_{ij} = p_i,$$

for all $i \in \{1, \dots, n\}$. Further, for all $j \in M \cup \bar{M}$, we have $\sum_{i=1}^{n} q_{ij} = 1$. Thus Q is doubly stochastic, and by the Birkhoff-von Neumann theorem can be written as a convex combination of permutation matrices. For each individual permutation matrix R there then exists a set $X \in \mathcal{X}_{n,m}$ such that $r_{ij} = 1$ for some $j \in M$ if and only if $i \in X$, which shows the claim.

When m is an arbitrary number, we can write p as a convex combination of two vectors p' and p'' such that $\sum_{i=1}^{n} p'_i = \lfloor m \rfloor$ and $\sum_{i=1}^{n} p''_i = \lceil m \rceil$. The claim then follows by applying the above reasoning independently to p' and p''. □

2.1.2 Related Work and Previous Results

The theory of impartial decision making was first considered by de Clippel et al. [dCMT08], for the case of a divisible resource to be shared among a set of agents.

The difference between divisible and indivisible resources disappears for randomized mechanisms, but the mechanisms of de Clippel et al. [dCMT08] allow for fractional nominations and do not have any obvious consequences for our setting. Impartial selection is a rather fundamental problem in social choice theory, with applications ranging from the selection of committees to academic peer review. The problem we consider here was first studied by Holzman and Moulin [HM13] and Alon et al. [AFPT11]. Even the case of selecting one agent was considered to be very challenging. It was conjectured by Alon et al. [AFPT11] that there is a mechanism that selects in expectation an agent that receives half the maximum number of nominations any user receives. This conjecture was answered to the positive by Fischer and Klimm [FK15]. They were also able to show that when the most popular agent receives many nominations, even a fraction of $3/4$ of the maximum number of nominations to an agent can be extracted by a mechanism. Further, the articles of Holzman and Moulin [HM13] and of Fischer and Klimm [FK15] provide a good introduction to the history of the problem and early literature.

When agents are interested purely in their own selection, the problem can be viewed as an example of mechanism design without money, an agenda put forward by Procaccia and Tennenholtz [PT13]. In peer review the need for impartiality is only one of a number of issues along with information elicitation and incentivization of effort, and a natural approach would be to combine our mechanisms with mechanisms seeking to achieve the other goals (see, e. g. , [WP12, WBKP13]). Other authors have taken a more holistic view of peer review and peer selection and have aimed for more practical and more heuristic mechanisms (see, e. g. , [KLMP15, ALM+16]). Tamura and Ohseto [TO14] were the first to consider impartial mechanisms selecting more than one agent and showed that these can circumvent some of the impossibility results of Holzman and Moulin [HM13]. Later, Tamura [Tam16] gave an axiomatic characterization of the mechanisms. A characterization of symmetric randomized selection mechanisms for the special case where each agent nominates exactly one other agent was given by Mackenzie [Mac15]. Inspiration for relaxing the requirement to always select the same number of agents comes from the power of multiple choices in load balancing, where even two choices can lead to dramatically lower average load, as shown by Mitzenmacher et al. [MRS01]. The related concept of resource augmentation, first used by Sleator and Tarjan [ST85], is a common technique in the analysis of online algorithms and has also been applied to a problem in mechanism design, e. g. by Caragiannis et al. [CFRF+16]. Mackenzie [Mac17] recently studied the relationship between impartiality, exactness, and randomization for various mechanisms used over the centuries in electing the pope.

2.1.3 An Overview of Our Results

We show in this chapter that a relaxation of exactness is, in addition to randomization, another remedy to the strong impossibility result concerning exact deterministic k-selection mechanisms in that it enables the design of α-optimal mechanisms for constant α. Specifically, for $k = 2$, we introduce the bidirectional permutation mechanism. Running it on a fixed permutation and selecting an agent for each direction of that permutation is $1/2$-optimal, which is best possible. Flexibility in the

k	Randomized Exact	Randomized At Most
1	$\frac{1}{2}$	$\frac{1}{2}$
2	$\left[\frac{7}{12}, \frac{2}{3}\right]$	$\left[\frac{2}{3}, \frac{3}{4}\right]$
⋮		
k	$\left[\frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right), \frac{k+1}{k+2}\right]$	$\left[\frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right), \frac{k+1}{k+2}\right]$
⋮		
$n-2$	$\left[\frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right), \frac{7k^3+5k^2-6k+12}{7k^3+13k^2-2k}\right]$	$\left[\frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right), \frac{k+1}{k+2}\right]$
$n-1$	$\left[\frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right), \frac{k}{k+1}\right]$	$\left[\frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right), \frac{2k}{2k+1}\right]$

Table 2.1: [Lower, Upper] Bounds on α for α-optimal randomized impartial selection of at most or exactly k agents out of n.

k	Deterministic Exact	Deterministic At Most
1	0	0
2	0	$\frac{1}{2}$
k	0	$\left[\frac{1}{k}, \frac{k-1}{k}\right]$

Table 2.2: [Lower, Upper] Bounds on α for α-optimal deterministic impartial selection of at most or exactly k agents. Deterministic exact mechanisms cannot be α-optimal for any $\alpha > 0$.

exact number of selected agents is beneficial also in the realm of randomized impartial mechanisms: given a set of three agents, for example, a $3/4$-optimal mechanism exists selecting two agents or fewer, whereas the best mechanism selecting exactly two agents is only $2/3$-optimal. For 2-selection from an arbitrary number of agents, we give a randomized exact $7/12$-optimal mechanism and a randomized $2/3$-optimal mechanism that is not exact. Finally we provide upper and lower bounds on the performance of mechanisms allowed to make more than two choices. A summary of our current state of knowledge about randomized mechanisms is shown in Table 2.1 and about deterministic mechanisms in Table 2.2.

2.2 Mechanisms for Selecting Two Agents

In this section, we want to explore impartial mechanisms that select up to two agents. Both deterministic and randomized mechanism are studied and we prove how lessening the requirement to select exactly two agents proves to be beneficial in both cases.

2.2.1 Deterministic Mechanisms for Two Agents

Focusing on the exact case, Alon et al. [AFPT11] showed that deterministic k-selection mechanisms cannot be α-optimal for any $k \in \{1, \ldots, n-1\}$ and $\alpha > 0$. This result is a rather simple observation for $k = 1$, but quite surprising when $k > 1$. For $(n-1)$-selection in particular, any deterministic mechanism that is both exact

Algorithm 2.1: The Bidirectional Permutation Mechanism, using Extraction Mechanism Ξ_π

Input: Graph $G = (N, E)$
Output: Set $\{i_1, i_2\} \subseteq N$ of at most two vertices
Let $\pi = (1, \ldots, n)$
Set $i_1 := \Xi_\pi(G)$ ▷ select vertex based on forward edges
Set $i_2 := \Xi_{\bar{\pi}}(G)$ ▷ select vertex based on backward edges
return $\{i_1, i_2\}$

Algorithm 2.2: The Extraction Mechanism Ξ_π

Input: Graph $G = (N, E)$, permutation $(\pi_1, \ldots, \pi_n)$ of N
Output: Vertex $i \in N$
Set $i := \pi_1$, $d := 0$ ▷ candidate vertex and its indegree from its left
for $j = 2, \ldots, n$ **do**
  if $\delta^-_{\pi_{<\pi_j} \setminus \{i\}}(\pi_j) \geq d$ **then** ▷ compare current considered vertex and candidate
    Set $i := \pi_j$, $d := \delta^-_{\pi_{<\pi_j}}(\pi_j)$ ▷ current vertex becomes new candidate
return i

and impartial must sometimes exclude precisely the unique vertex with positive indegree. While it is not difficult to convince ourselves that a relaxation of exactness is not helpful in the case of 1-selection, we will exhibit momentarily a deterministic impartial mechanism that for any graph selects either one or two vertices whose overall indegree is at least the largest indegree of any vertex in the graph.

To explain our mechanism we need some additional notation. Let $N = \{1, \ldots, n\}$. For a graph $G = (N, E)$ and a permutation $\pi = (\pi_1, \ldots, \pi_n)$ of N, denote by

$$E_\pi = \big\{(u, v) \in E : \pi_i = u, \pi_j = v \text{ for some } i, j \text{ with } 1 \leq i < j \leq n\big\}$$

the set of *forward edges* of G with respect to π. Denote by $\bar{\pi}$ the permutation obtained by reading π backwards, such that $\bar{\pi}_i = \pi_{n+1-i}$ for $i = 1, \ldots, n$. Finally, for a permutation π and $j \in \{1, \ldots, n\}$, let

$$\pi_{<j} = \{\pi_1, \pi_2, \ldots, j\} \setminus \{j\}$$

denote the set of vertices in the prefix of π up to but not including j.

The first mechanism we consider, which we call the bidirectional permutation mechanism, considers the vertices one by one according to a fixed permutation π and in each step compares the current vertex π_j to a single candidate vertex π_ℓ with $\ell < j$. In determining the indegree of the candidate vertex π_ℓ it takes into account the outgoing edges of vertices $\pi_1, \ldots, \pi_{\ell-1}$. For the indegree of the current vertex π_j it takes into account the outgoing edges of vertices $\pi_1, \ldots, \pi_{j-1}$, with the exception of π_ℓ. If the latter is greater than or equal to the former, π_j becomes the new candidate, and the candidate after the final step is the first vertex selected by the mechanism. The same procedure is then applied with permutation $\bar{\pi}$ to find a second vertex. A formal description of the bidirectional permutation mechanism is given as Algorithm 2.1. It is formulated in terms of Algorithm 2.2, which we

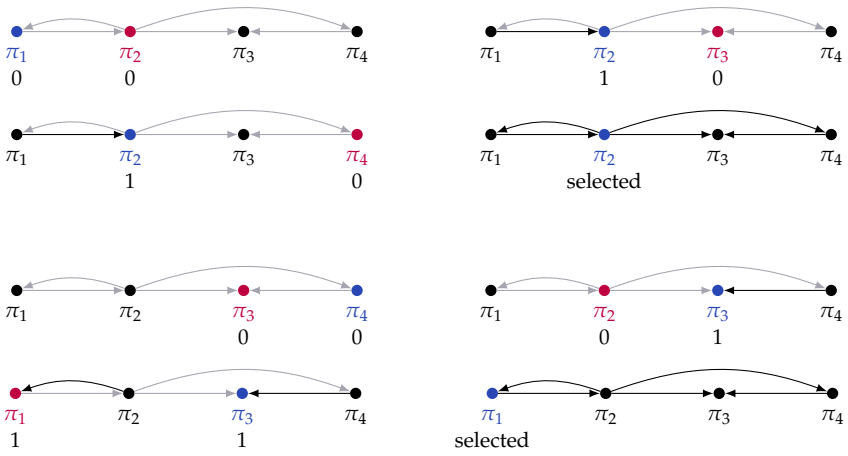


Figure 2.1: Example for the bidirectional permutation mechanism 2.1, first going forward direction selecting vertex π_2 as shown in the first four graphs, then backward direction selecting vertex π_1. The mechanism considers the vertices one by one, comparing the currently considered vertex, depicted in red, to the current candidate, depicted in blue. Edges that are in that stage considered by the mechanism are black, others grey; current indegrees, given the considered edges, are written below the vertices.

call the extraction mechanism and which is identical to a mechanism of Fischer and Klimm [FK15] except for its use of a given permutation rather than a random one. For an example, please refer to Figure 2.1.

It is worth noting that the bidirectional permutation mechanism may select only one vertex, namely if the same vertex is chosen for both directions of the permutation. This happens for example later in the graph of Figure 2.2.

To see that the bidirectional permutation mechanism is impartial, we first note that this is true for a single run of the extraction mechanism. Indeed, the outcome of the latter is influenced by the outgoing edges of any given vertex only when that vertex can no longer be selected.

Lemma 2.2 *The extraction mechanism is impartial.*

Proof. For a contradiction, assume that the algorithm is not impartial, i. e. , there are a set N of agents, a distinguished agent $i^* \in N$ and two nomination graphs $G = (N, E)$ and $G' = (N, E')$ with

$$E \setminus \Big(\bigcup_{j \in N \setminus \{i\}} (i, j) \Big) = E' \setminus \Big(\bigcup_{j \in N \setminus \{i\}} (i, j) \Big)$$

such that i^* is selected for G but not for G'. In the following, we denote by $\delta_\pi(i)$ and $\delta'_\pi(i)$ the number of incoming forward edges of agent i in G and G', respectively. Let $j \in \arg\max_{i \in N} \delta'_\pi(i)$ be arbitrary. The values $\delta_\pi(i)$ and $\delta'_\pi(i)$ are equal for all agents that appear in the permutation not after i^*. Thus, if i^* is a candidate in G, it is also candidate in G'. For the agents that appear after i^* in π, the values $\delta_\pi(i)$ and $\delta'_\pi(i)$ differ by at most one; but as i^* is candidate in G and in G', the values are in fact equal. This contradicts the existence of an agent becoming candidate after i^* in G, but not in G'. Thus i^* is selected in G', which is a contradiction. □

We now note that the composition of impartial selection mechanisms stays impartial.

Lemma 2.3 *Let $f_1, \ldots, f_k$ be impartial 1-selection mechanisms. Then the mechanism which selects all the vertices selected by at least one of the mechanisms $f_1, \ldots, f_k$ is an impartial k-selection mechanism.*

Proof. By impartiality of f_ℓ, for $\ell = 1, \ldots, k$, the outgoing edges of a vertex do not influence whether this vertex is selected by f_ℓ. This holds for any ℓ and any vertex, so it also holds for the mechanism that selects the vertices selected by at least one of the mechanisms. □

Impartiality of the bidirectional permutation mechanism, which is the union of two impartial 1-selection mechanisms, now follows with Lemma 2.2 and Lemma 2.3 because the union of the results of two impartial 1-selection mechanisms yields an impartial 2-selection mechanism.

Corollary 2.4 *The bidirectional permutation mechanism is impartial.*

We proceed to show that the bidirectional permutation mechanism is $1/2$-optimal, starting from the observation that the vertex selected by Ξ_π has a maximum number of incoming forward edges with respect to π.

Lemma 2.5 *If $i = \Xi_\pi(G)$, then*

$$\delta^-_{\pi_{<i}}(i, G) = \max_{j=1,\ldots,n} \{\delta^-_{\pi_{<j}}(j, G)\}.$$

Proof. Let

$$d^* = \max_{j=1,\ldots,n} \{\delta^-_{\pi_{<j}}(j)\},$$

and i^* an arbitrary vertex with $\delta^-_{\pi_{<i^*}}(i^*) = d^*$. When i^* is considered by the mechanism, so are at least $d^* - 1$ of its incoming forward edges, as one of the incoming forward edges may originate from the current candidate i, i. e.

$$\delta^-_{\pi_{<i^*}\setminus\{i\}}(i^*) \in \{d^* - 1, d^*\}.$$

If

$$\delta^-_{\pi_{<i^*}\setminus\{i\}}(i^*) = d^*, \text{ or both } \delta^-_{\pi_{<i^*}\setminus\{i\}}(i^*) = d^* - 1 \text{ and } \delta^-_{\pi_{<i}}(i) \leq d^* - 1,$$

then i^* becomes the new candidate. Since the indegree d of the current candidate will be updated to d^*, any other vertex that possibly becomes a candidate after i^* has d^* incoming forward edges as well, establishing the claim for this case.

If, on the other hand, both

$$\delta^-_{\pi_{<i^*}\setminus\{i\}}(i^*) = d^* - 1 \text{ and } \delta^-_{\pi_{<i}}(i) = d^*,$$

then i remains the candidate. As $d = d^*$ any further candidate has d^* incoming forward edges as in the first case. □

Figure 2.2: Graphs for which the bidirectional permutation mechanism returns only one vertex (left side, π_2 is selected twice) and is only $1/2$-optimal (right side, π_1 and π_2 are selected).

Theorem 2.6 *The bidirectional permutation mechanism is impartial and $1/2$-optimal.*

Proof. Impartiality follows directly from Corollary 2.4.

Now consider a graph $G = (N, E)$, a vertex i^* with $\delta^-(i^*) = \Delta_1$, let $i_1 = \Xi_\pi(G)$ and $i_2 = \Xi_{\bar{\pi}}(G)$. By Lemma 2.5,

$$\delta^-_{\pi_{<i_1}}(i_1) \geq \delta^-_{\pi_{<i^*}}(i^*) \text{ and } \delta^-_{\bar{\pi}_{<i_2}}(i_2) \geq \delta^-_{\bar{\pi}_{<i^*}}(i^*),$$

regardless of whether $i_1 \neq i_2$ or $i_1 = i_2$. Thus

$$\begin{aligned}\delta^-(\{i_1, i_2\}) &\geq \delta^-_{\pi_{<i_1}}(i_1) + \delta^-_{\bar{\pi}_{<i_2}}(i_2) \\ &\geq \delta^-_{\pi_{<i^*}}(i^*) + \delta^-_{\bar{\pi}_{<i^*}}(i^*) = \delta^-(i^*) = \Delta_1 \geq \frac{1}{2}\Delta_2,\end{aligned}$$

as claimed. □

To see that the analysis is tight, consider the graph on the right side in Figure 2.2. For this graph, the mechanism selects vertices π_2 and π_1 with an overall indegree of one, while the maximum overall indegree of a set of two vertices is two. We see later, in Theorem 2.26, that the bound of $1/2$ is in fact best possible.

There are two ways to interpret the result in this section. Since the largest indegree is at least half of the sum of the two largest indegrees, relaxing exactness allows us to circumvent the strong lower bound of Alon et al. [AFPT11] when $k = 2$. Alternatively, for $k = 1$, the tradeoff between impartiality and quality of the outcome disappears if one is allowed to sometimes but not always select an additional vertex. This kind of resource augmentation result, comparing an optimal algorithm to one from a restricted class that is given additional resources, is commonly used in the analysis of online algorithms and has recently also been applied to truthful mechanisms for facility assignment [CFRF+16]. Curiously, the bound of $1/2$ is best possible for exact randomized mechanisms selecting a single vertex as we will see in the next section, thus, randomization can be perfectly substituted by the ability to sometimes select an additional vertex.

2.2.2 Randomized Mechanisms for Two Agents

In light of the results of the previous section, it is natural to ask whether a relaxation of exactness enables better bounds also for randomized mechanisms. We answer this question in the affirmative and give the first nontrivial bounds for both exact and inexact 2-selection mechanisms, as well as an example that shows a strict separation between the two classes.

We begin by considering an exact mechanism, which we call the 2-partition mechanism with permutation. The mechanism randomly partitions the set of vertices into two sets A_1 and A_2 such that $\mathbb{P}[i \in A_1] = \mathbb{P}[i \in A_2] = 1/2$ for all $i \in N$, $A_1 \cup A_2 = N$,

Algorithm 2.3: The 2-Partition Mechanism with Permutation

Input: Graph $G = (N, E)$ with $n \geq 2$
Output: Vertices $i_1, i_2 \in N$

Assign each $i \in N$ to A_1 or A_2 independently and uniformly at random
Choose a permutation $(\pi_1, \ldots, \pi_n)$ of N uniformly at random
for $j = 1, 2$ **do**
    Set $i_j := \Xi_{\pi, A_j}(G)$ ▷ select one vertex from each of the two sets
▷ if one set is empty, select 2nd vertex from other set
if $A_2 = \emptyset$ **then** choose i_2 uniformly at random from $A_1 \setminus \{i_1\}$
if $A_1 = \emptyset$ **then** choose i_1 uniformly at random from $A_2 \setminus \{i_2\}$
return $\{i_1, i_2\}$

Algorithm 2.4: The Extraction Mechanism $\Xi_{\pi,A}$ Restricted to a Set $A \subseteq N$

Input: Graph $G = (N, E)$, permutation $(\pi_1, \ldots, \pi_n)$ of N, set $A \subseteq N$
Output: Vertex $i \in N$

Set $i := \pi_1$, $d := 0$ ▷ candidate vertex and indegree from its left
for $j = 2, \ldots, n$ **do**
    Set $S := (N \setminus A) \cup (\pi_{<\pi_j} \setminus \{i\})$ ▷ vertices whose nominations are considered
    if $\pi_j \in A$ **and** $\delta_S^-(\pi_j) \geq d$ **then** ▷ compare current vertex and candidate
        Set $i := \pi_j$, $d := \delta_{S \cup \{i\}}^-(\pi_j)$ ▷ current vertex becomes new candidate
return i

and $A_1 \cap A_2 = \emptyset$. It then selects one vertex from each of the sets by applying the extraction mechanism with a random permutation, while also taking into account incoming edges from the respective other set. Algorithm 2.3 is a formal description of the mechanism. It builds on the restricted extraction mechanism $\Xi_{\pi,A}$ of Algorithm 2.4, which works similar to the extraction mechanism considered earlier, except that only vertices $i \in A$ may ever become a candidate. Thus, at any time the number of incoming forward edges of a vertex in A is determined, with all edges originating in vertices in $N \subseteq A$ being counted as forward edges. Note that for the special case that $A = N$, we obtain $\Xi_{\pi,N} = \Xi_\pi$. Using the same line of reasoning as for the original unrestricted extraction mechanism, it is straightforward to show the following.

Lemma 2.7 *The restricted extraction mechanism is impartial.*

Proof. When A is empty, the mechanism selects the same vertex for any graph and therefore is impartial. Otherwise the first vertex in A to appear in π becomes a candidate and only vertices from A are considered thereafter, so the mechanisms selects a vertex from A. Moreover, the mechanism only takes into account outgoing edges of vertices that can no longer be selected, either because they are not in A or because they have already been considered and are not currently the candidate. This directly implies impartiality. □

Taking Lemma 2.3 and Lemma 2.7 together, impartiality of the 2-partition mechanism with permutation is immediately implied.

Corollary 2.8 *The 2-partition mechanism with permutation is impartial.*

To prove how well the algorithm works and what approximation ratio it achieves, we further need the following lemma.

Lemma 2.9 *If* $i = \Xi_{\pi,A}(G)$, *then*

$$\delta^-_{(N\setminus A)\cup\pi_{<i}}(i,G) \geq \max_{j\in A}\{\delta^-_{(N\setminus A)\cup\pi_{<j}}(j,G)\}.$$

Proof. The statement is trivial in the case that A is empty. Otherwise, when there is at least one vertex in A, consider

$$d^* = \max_{j\in A}\{\delta^-_{(N\setminus A)\cup\pi_{<j}}(j,G)\}$$

and i^* such that $\delta^-_{(N\setminus A)\cup\pi_{<i^*}}(i^*) = d^*$. Analogously to the proof of Lemma 2.5, we consider the iteration in which the mechanism decides whether i^* should become the new candidate. Let i be the candidate at the begin of that iteration. If we have

$$\delta^-_{(N\setminus A)\cup\pi_{<i^*}\setminus\{i\}}(i^*) = d^*$$

or both

$$\delta^-_{(N\setminus A)\cup\pi_{<i^*}\setminus\{i\}}(i^*) = d^* - 1 \text{ and } \delta^-_{(N\setminus A)\cup\pi_{<i}}(i) \leq d^* - 1,$$

then i^* becomes the new candidate and consequently d is updated to d^*.

If, on the other hand, both

$$\delta^-_{(N\setminus A)\cup\pi_{<i^*}\setminus\{i\}}(i^*) = d^* - 1 \text{ and } \delta^-_{(N\setminus A)\cup\pi_{<i}}(i) = d^*,$$

then i stays the candidate and d remains equal to d^*.

In both cases we have $d = d^*$, which implies that any future candidate j has at least d^* incoming edges from $(N \setminus A) \cup \pi_{<j}$. □

Using these observations, we obtain our result for the 2-partition mechanism with permutation.

Theorem 2.10 *The 2-partition mechanism with permutation is impartial and* 7/12*-optimal.*

Proof. Impartiality follows immediately by using Corollary 2.8. Now consider a graph $G = (N, E)$, two distinct vertices $i_1^*, i_2^* \in N$ with

$$\delta^-(i_1^*) + \delta^-(i_2^*) = \Delta_2,$$

and let i_1 and i_2 be the two vertices selected by the mechanism from sets A_1 and A_2, respectively. We distinguish two cases, depending on whether i_1^* and i_2^* are in the same set or different sets of the partition (A_1, A_2).

First assume that i_1^* and i_2^* are in different sets, and without loss of generality let $i_1^* \in A_1$ and $i_2^* \in A_2$. In the permutation π used by the mechanism and chosen

uniformly at random, an arbitrary vertex $i \in N \setminus \{i_1^*, i_2^*\}$ appears before or after each of i_1^* or i_2^* with equal probability, so

$$\begin{aligned}\mathbb{P}\left[i \in A_1 \cap \pi_{<i_1^*}\right] &= \mathbb{P}\left[i \in A_1 \cap \bar{\pi}_{<i_1^*}\right] \\ &= \mathbb{P}\left[i \in A_2 \cap \pi_{<i_2^*}\right] = \mathbb{P}\left[i \in A_2 \cap \bar{\pi}_{<i_2^*}\right] = \frac{1}{4}.\end{aligned}$$

When i_1^* is considered by the mechanism, so are any incoming edges from vertices in A_2 and any incoming edges from vertices in A_1 that appear in π before i_1^*. Thus, by Lemma 2.9,

$$\begin{aligned}\mathbb{E}\left[\delta^-(i_1)\right] &\geq \mathbb{E}\left[\delta^-_{A_2 \cup \pi_{<i_1^*}}(i_1^*)\right] \\ &= \textstyle\sum_{i \in N} \mathbb{P}\left[i \in A_2 \cup (A_1 \cap \pi_{<i_1^*})\right] \cdot \chi[(i, i_1^*) \in E],\end{aligned}$$

where χ denotes the indicator function on Boolean expressions, i. e. $\chi[\phi] = 1$ if expression ϕ holds and $\chi[\phi] = 0$ otherwise. By taking i_2^* out of the sum and using that i_1^* and i_2^* are in different sets of the partition and thus $\mathbb{P}[i_2^* \in A_2] = 1$, we obtain

$$\begin{aligned}\mathbb{E}\left[\delta^-(i_1)\right] &\geq \textstyle\sum_{i \in N \setminus \{i_2^*\}} \Big(\mathbb{P}[i \in A_2 \cup (A_1 \cap \pi_{<i_1^*})] \cdot \chi[(i, i_1^*) \in E]\Big) \\ &\qquad + \mathbb{P}[i_2^* \in A_2 \cup (A_1 \cap \pi_{<i_1^*})] \cdot \chi[(i_2^*, i_1^*) \in E] \\ &= \textstyle\sum_{i \in N \setminus \{i_2^*\}} \Big((1 - \mathbb{P}[i \in (A_1 \cap \bar{\pi}_{<i_1^*})]) \cdot \chi[(i, i_1^*) \in E]\Big) + \chi[(i_2^*, i_1^*) \in E] \\ &= \textstyle\sum_{i \in N \setminus \{i_2^*\}} \Big(1 - \frac{1}{4}\Big)\chi[(i, i_1^*) \in E] + \chi[(i_2^*, i_1^*) \in E] \\ &\geq \frac{3}{4} \sum_{i \in N} \chi[(i, i_1^*) \in E] \\ &= \frac{3}{4}\delta^-(i_1^*).\end{aligned}$$

As the same line of reasoning applies to i_2^*, we have

$$\mathbb{E}\left[\delta^-(i_2)\right] \geq \frac{3}{4}\delta^-(i_2^*)$$

and conclude for this case that

$$\mathbb{E}\left[\frac{\delta^-(i_1, i_2)}{\Delta_2}\right] \geq \frac{\frac{3}{4}\delta^-(i_1^*) + \frac{3}{4}\delta^-(i_2^*)}{\delta^-(i_1^*) + \delta^-(i_2^*)} \geq \frac{3}{4}.$$

Now assume that i_1^* and i_2^* are in the same set of the partition, and without loss of generality that $i_1^*, i_2^* \in A_1$ and $\delta^-(i_1^*) \geq \delta^-(i_2^*)$. In the permutation π used by the mechanism and chosen uniformly at random, an arbitrary vertex $i \in N \setminus \{i_1^*, i_2^*\}$ appears before, between, or after i_1^* and i_2^* with probability $1/3$ each, so

$$\begin{aligned}\mathbb{P}\left[i \in A_2\right] &= \frac{1}{2} \quad \text{and} \\ \mathbb{P}\left[i \in A_1 \cap \pi_{<i_1^*} \cap \pi_{<i_2^*}\right] &= \mathbb{P}\left[i \in A_1 \cap ((\pi_{<i_1^*} \cap \bar{\pi}_{<i_2^*}) \cup (\bar{\pi}_{<i_1^*} \cap \pi_{<i_2^*}))\right] \\ &= \mathbb{P}\left[i \in A_1 \cap \bar{\pi}_{<i_1^*} \cap \bar{\pi}_{<i_2^*}\right] = \frac{1}{6}.\end{aligned}$$

If $i_1^* \in \bar{\pi}_{<i_2^*}$, a possible edge from i_2^* to i_1^* would be considered by the mechanism, and by Lemma 2.9,

$$\begin{aligned}\mathbb{E}\left[\delta^-(i_1)\right] &\geq \mathbb{E}\left[\delta^-_{A_2 \cup \pi_{<i_1^*}}(i_1^*)\right] \\ &\geq \sum_{i \in N \setminus \{i_2^*\}} \Big(\mathbb{P}[i \in A_2 \cup (A_1 \cap \pi_{<i_1^*})] \cdot \chi[(i, i_1^*) \in E] \Big) \\ &\qquad + \mathbb{P}[i_2^* \in A_2 \cup (A_1 \cap \pi_{<i_1^*})] \cdot \chi[(i_2^*, i_1^*) \in E] \\ &= \sum_{i \in N} \Big((1 - \mathbb{P}[i \in (A_1 \cap \bar{\pi}_{<i_1^*})]) \cdot \chi[(i, i_1^*) \in E] \Big) \\ &\qquad + \mathbb{P}[i_2^* \in (A_1 \cap \pi_{<i_1^*})] \cdot \chi[(i_2^*, i_1^*) \in E].\end{aligned}$$

We have

$$\mathbb{P}\left[i \in (A_1 \cap \bar{\pi}_{<i_1^*})\right] = \mathbb{P}\left[i \in (A_1 \cap \bar{\pi}_{<i_1^*} \cap \bar{\pi}_{<i_2^*})\right] = \frac{1}{6}$$

since we assumed that $i_1^* \in \bar{\pi}_{<i_2^*}$ and obtain

$$\mathbb{E}\left[\delta^-(i_1)\right] \geq \frac{5}{6} \sum_{i \in N} \chi[(i, i_1^*) \in E] = \frac{5}{6} \delta^-(i_1^*).$$

Analogously, if $i_2^* \in \bar{\pi}_{<i_1^*}$,

$$\mathbb{E}\left[\delta^-(i_1))\right] \geq \mathbb{E}\left[\delta^-_{A_2 \cup \pi_{<i_2^*}}(i_2^*)\right] = \frac{5}{6} \delta^-(i_2^*).$$

As each of the two events takes places with probability $1/2$, we conclude for this case that

$$\mathbb{E}\left[\frac{\delta^-(i_1 \cup i_2)}{\Delta_2}\right] \geq \frac{\frac{1}{2}\left(\frac{5}{6}\delta^-(i_1^*) + \frac{5}{6}\delta^-(i_2^*)\right)}{\delta^-(i_1^*) + \delta^-(i_2^*)} = \frac{5}{12}.$$

Averaging over both cases we finally obtain

$$\alpha \geq \frac{1}{2}\left(\frac{3}{4} + \frac{5}{12}\right) = \frac{7}{12},$$

as claimed. □

The 2-partition mechanism with permutation improves on the best deterministic mechanism for 2-selection, and it is natural to ask whether it can be improved upon further by a randomized 2-selection mechanism that is not exact. The answer to this question is not obvious: while the ability to select fewer vertices may make impartiality easier to achieve, actually selecting fewer vertices runs counter to the objective of selecting vertices with a large sum of indegrees. Indeed, in the case of 1-selection, no separation exists between exact and inexact mechanisms. In contrast, for 2-selection, an obvious approach turns out to be effective: taking the best deterministic mechanism, which uses both directions of a fixed permutation, and invoking it for a random permutation. The resulting mechanism, which we call the randomized bidirectional permutation mechanism, is shown as Algorithm 2.5.

Algorithm 2.5: The Randomized Bidirectional Permutation Mechanism

Input: Graph $G = (N, E)$
Output: Set $\{i_1, i_2\} \subseteq N$ of at most two vertices
1 Choose a permutation $(\pi_1, \ldots, \pi_n)$ of N uniformly at random
2 Invoke Algorithm 2.1, the bidirectional permutation mechanism, for G and π

Theorem 2.11 *The randomized bidirectional permutation mechanism is impartial and* 2/3*-optimal.*

Proof. The proof of Theorem 2.6 shows impartiality for any permutation that does not depend on the input to the mechanism, including one that is chosen uniformly at random.

Now consider a graph $G = (N, E)$, two distinct vertices $i_1^*, i_2^* \in N$ with

$$\delta^-(i_1^*) + \delta^-(i_2^*) = \Delta_2,$$

and let $i_1 = \Xi_\pi(G)$ and $i_2 = \Xi_{\bar{\pi}}(G)$ for the permutation π used by the mechanism. As the mechanism is invariant under reversing the permutation, assume without loss of generality that i_1^* appears before i_2^* in π, i. e. that $i_1^* \in \pi_{<i_2^*}$. As π was chosen uniformly at random, an arbitrary vertex $i \in N \setminus \{i_1^*, i_2^*\}$ appears before, between, or after i_1^* and i_2^* with probability 1/3 each. By applying Lemma 2.5 to both i_1 and i_2,

$$\begin{aligned}\mathbb{E}\left[\frac{\delta^-(\{i_1, i_2\})}{\Delta_2}\right] &\geq \mathbb{E}\left[\frac{\delta^-_{\pi_{<i_1}}(i_1) + \delta^-_{\bar{\pi}_{<i_2}}(i_2)}{\Delta_2}\right] \\ &\geq \mathbb{E}\left[\frac{\max\left\{\delta^-_{\pi_{<i_1^*}}(i_1^*), \delta^-_{\pi_{<i_2^*}}(i_2^*)\right\} + \max\left\{\delta^-_{\bar{\pi}_{<i_1^*}}(i_1^*), \delta^-_{\bar{\pi}_{<i_2^*}}(i_2^*)\right\}}{\Delta_2}\right]. \qquad (2.1)\end{aligned}$$

Recall that possibly $i_1 = i_2$, and note that the bound is correct in this case as well since we account in the first call of the extraction mechanism only for the forward edges of i, and in the second call of the extraction mechanism only for the backward edges of i. To bound the right-hand side of equation (2.1), we use the assumption that $i_1^* \in \pi_{<i_2^*}$ and observe that

$$\begin{aligned}\mathbb{E}\left[\max\left\{\delta^-_{\pi_{<i_1^*}}(i_1^*), \delta^-_{\pi_{<i_2^*}}(i_2^*)\right\}\right] \geq \mathbb{E}\left[\delta^-_{\pi_{<i_2^*}}(i_2^*)\right] &= \sum_{i \in N} \mathbb{P}\left[i \in \pi_{<i_2^*}\right] \cdot \chi\left[(i, i_2^*) \in E\right] \\ &= \sum_{i \in N} (1 - \mathbb{P}\left[i \in \bar{\pi}_{<i_2^*}\right]) \cdot \chi\left[(i, i_2^*) \in E\right] \\ &= \frac{2}{3} \sum_{i \in N} \chi\left[(i, i_2^*) \in E\right] \\ &= \frac{2}{3}\, \delta^-(i_2^*).\end{aligned}$$

Note that this bound only gets better when $(i_1^*, i_2^*) \in E$, as $\mathbb{P}[i_1^* \in \pi_{<i_2^*}] = 1$ by assumption . An analogous argument for the other direction yields

$$\mathbb{E}\left[\max\left\{\delta^-_{\bar{\pi}_{<i_1^*}}(i_1^*), \delta^-_{\bar{\pi}_{<i_2^*}}(i_2^*)\right\}\right] \geq \frac{2}{3}\, \delta^-(i_1^*),$$

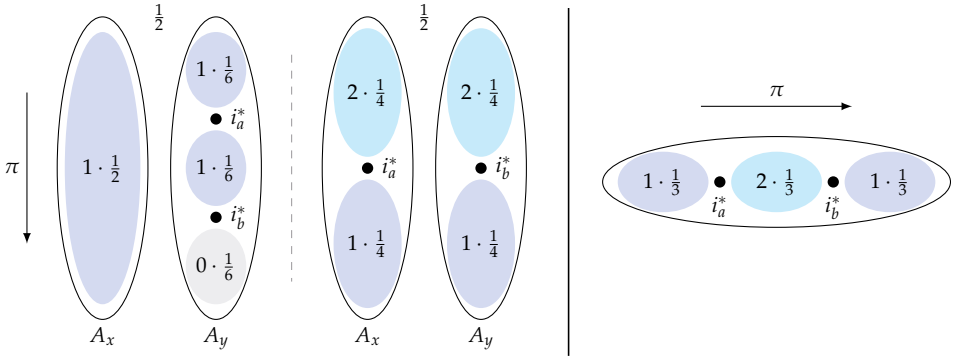


Figure 2.3: Schematic for the proof to Theorem 2.12. The two pictures on the left are for the 2-partition mechanism with permutation. The case that both vertices which receive a nomination are in the same partition is depicted on the left, the case that both vertices are in different partitions is depicted in the middle, each appearing with probability $1/2$. The picture in the right is for the randomized bidirectional permutation mechanism. Different positions for the vertex casting nominations are marked with colored ellipses which contain the probability that the nominating vertex is in that position. A position that leads to no nominated vertex being selected is colored light grey, one nominated vertex being selected dark blue, and both nominated vertices being selected is colored light blue.

and by plugging both bounds into equation (2.1) we obtain

$$\mathbb{E}\left[\frac{\delta^-(i_1)+\delta^-(i_2)}{\Delta_2}\right] \geq \frac{\frac{2}{3}\delta^-(i_1^*)+\frac{2}{3}\delta^-(i_2^*)}{\delta^-(\{i_1^*,i_2^*\})} \geq \frac{2}{3},$$

as claimed. □

It is not hard to see that our analysis of the 2-partition mechanism with permutation and the bidirectional permutation mechanism is tight.

Theorem 2.12 *The 2-partition mechanism with permutation is at most $7/12$-optimal. The randomized bidirectional permutation mechanism is at most $2/3$-optimal.*

Proof. Consider a graph with a large number of vertices and only two edges (i, i_1^*) and (i, i_2^*), and observe that the maximum overall indegree of any set of two vertices is two. Please refer also to Figure 2.3.

The 2-partition mechanism with permutation independently and uniformly at random assigns each of i_1^*, i_2^*, and i to one of two sets, such that in particular i_1^* and i_2^* are in the same set with probability $1/2$ and in different sets with probability $1/2$.

If i_1^* and i_2^* are in the same set, the mechanism selects at most one of them. If i is in the respective other set, i. e. with probability $1/2$, this happens with probability one. If i is in the same set, it happens only with probability $2/3$, namely when i appears before either i_1^* and i_2^* in a permutation π chosen uniformly at random.

If i_1^* and i_2^* are in different sets, one of them is selected with probability one, the other only when i appears before it in π, which happens with probability $1/2$. In summary we thus expect

$$\alpha \leq \left(\frac{1}{2}\left(\frac{1}{2}+\frac{1}{2}\cdot\frac{2}{3}\right)+\frac{1}{2}\left(1+\frac{1}{2}\right)\right)/2 = \frac{7}{12}.$$

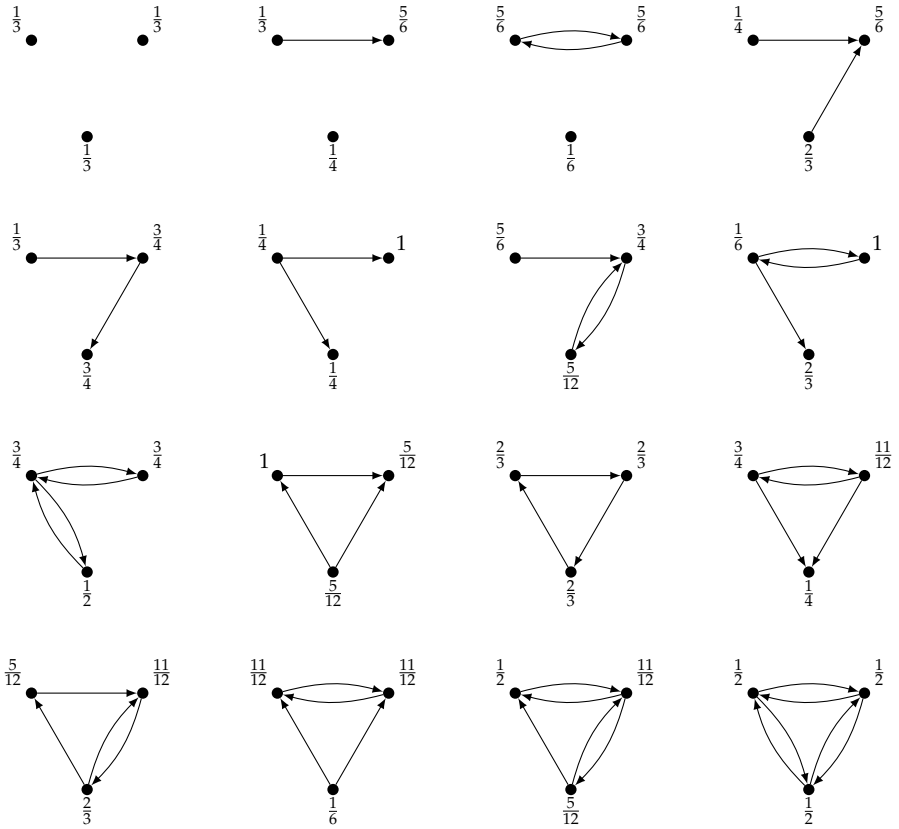


Figure 2.4: A 3/4-optimal impartial mechanism for $n = 3$ and $k = 2$ which is given explicitly by the selection probabilities for all sixteen voting graphs. The bound of 3/4 is best possible by Theorem 2.28.

The randomized bidirectional permutation mechanism selects one of i_1^* and i_2^* with probability one, the other only when i appears between i_1^* and i_2^* in a permutation π chosen uniformly at random, which happens with probability $1/3$. Thus we conclude that

$$\alpha \leq \left(\frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 1\right)/2 = \frac{2}{3}.$$

□

As special cases of Theorem 2.28 and Theorem 2.29 in Section 2.4, we will respectively obtain upper bounds of 3/4 and 2/3 for 2-selection mechanisms without and with exactness. These bounds suggest that neither the randomized bidirectional permutation mechanism nor the 2-partition mechanism with permutation is the best mechanism within its class. Indeed, Figure 2.4 shows a 3/4-optimal impartial mechanism selecting at most two of three vertices, which certifies that the randomized bidirectional permutation mechanism is not the best and that relaxing exactness is strictly beneficial.

The mechanism of Figure 2.4 can be obtained as the solution of an optimization problem to maximize the expected overall indegree of the vertices selected, subject

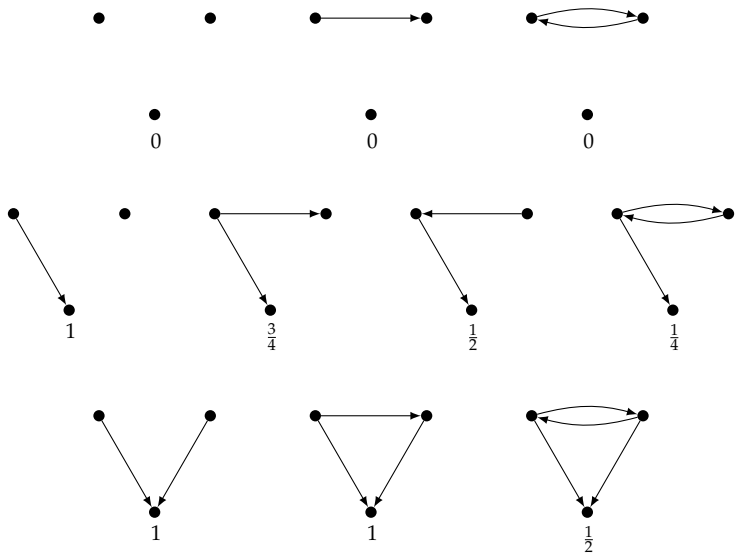


Figure 2.5: A 3/4-optimal impartial mechanism for $n = 3$ and $k = 2$. The graphs give the selection probability of the respective agent at the bottom, regardless of her outgoing nominations.

Algorithm 2.6: The 3/4-optimal Impartial Mechanism for Selecting Up to Two out of Three Agents as depicted in Figure 2.5

Input: Graph $G = (N, E)$ with $|N| = 3$
Output: Set $\{i_1, i_2\} \subseteq N$ of at most two vertices

for $i \in N$ **do** ▷ calculate selection probability for each vertex
  $p_i := \delta_G^-(i)$ ▷ indegree
  $p_i \leftarrow p_i - |J|/4$ for $J := \{j \in N : (j,i) \in E, \delta_G^+(j) = 2\}$ ▷ set of agents which casts nominations to i and a third agent
  $p_i \leftarrow p_i - |H|/2$ for $H := \{h \in N : (h,i) \in E, \delta_{G\setminus i}^-(j) = 1\}$ ▷ set of agents which casts nominations to i and receives nominations from a third agent
  if $p_i > 1$ **then** $p_i \leftarrow 1$
return up to two vertices $\subseteq N$ according to selection probabilities p_i

to impartiality. This still leaves some degrees of freedom. Here, we have chosen the probabilities in such a way that two agents have different probabilities to be selected iff the nominations that the respective other agents give are different.

Given impartiality, the probability of a given agent to be selected can not change if her own nominations change. Thus, instead of giving the selection probabilities for all voting graphs, it is sufficient to look at all nomination graphs an agent can be part of without her casting nominations. For three agents, there are eleven such nomination graphs. Figure 2.5 gives another impartial mechanism for chosing two out of three candidates. It also is 3/4-optimal.

Interestingly, the mechanism of Figure 2.5 can also be described in a relatively compact form as given in Algorithm 2.6. Remark that the final step, returning up to two vertices given selection probabilities which sum up to less than two, is possible due to Lemma 2.1. Indeed, it can easily be achieved with, i. e. , the Allocation

Algorithm 2.7: The k-Partition Mechanism with Permutation

Input: Graph $G = (N, E)$ with $n \geq 2$
Output: Vertices $i_1, \ldots, i_k \in N$

Assign each $i \in N$ independently and uniformly at random to one of k sets $A_1, \ldots, A_k$
Choose a permutation $(\pi_1, \ldots, \pi_n)$ of N uniformly at random
for $j = 1, \ldots, k$ **do**
    if $A_j \neq \emptyset$ **then**
        $i_j := \Xi_{\pi, A_j}(G)$ ▷ select one vertex from each set using extraction mechanism
for $j = 1, \ldots, k$ **do**
    if $A_j = \emptyset$ **then**
        Choose i_j uniformly at random from $N \setminus \{i_1, \ldots, i_k\}$
return $\{i_1, \ldots, i_k\}$ **return** $\{i_1, \ldots, i_k\}$

Subroutine, which we will later describe in Algorithm 2.11.

The mechanism of Figure 2.4s and Algorithm 2.6s lack of universal impartiality illustrates one of the main obstacles that prevent us from obtaining tighter bounds and generalize our results to the selection of more than two vertices. Here, a mechanism is called universally impartial if it is a convex combination of deterministic impartial mechanisms. Mechanisms that are impartial but not universally impartial are notoriously difficult to analyze and sometimes exhibit rather peculiar behavior. The last two rows of the rightmost column of Figure 2.4 for example show a decrease in the probability of selecting two of the vertices as their indegrees go up, and this is both necessary for $3/4$-optimality and difficult to justify.

2.3 Selecting More Than Two Agents

The central component of our best inexact mechanisms, its use of one or both of the directions of a random permutation, does not generalize in any obvious way to the selection of additional vertices. It seems to be harder in general to maintain impartiality when choosing more agents. In the following chapter, we give an overview over generalizations of the partition and permutation mechanism as well as introduce the exact dollar partition with permutation mechanism and analyze their quality.

2.3.1 Partition Mechanisms for k Agents

In this section, we give a natural generalization of the 2-partition mechanism with permutation. The algorithm uses a partition into k sets and then selects a vertex from each set using the extraction mechanism restricted to a subset. This algorithm achieves a lower bound for randomized algorithms. As it selects exactly k agents, the lower bounds holds for both the exact case as well as the relaxed condition of selecting up to k agents. The general mechanism is described in Algorithm 2.7. Its impartiality is easy to see, and we use an argument similar to that in the proof of

Lemma 2.9 to obtain a performance guarantee that approaches $1 - 1/e$ as k grows.

Theorem 2.13 *The k-partition mechanism with permutation is impartial and α-optimal for $\alpha = \frac{k}{k+1}\left(1 - \left(\frac{k-1}{k}\right)^{k+1}\right)$.*

Proof. Impartiality follows directly from Lemma 2.3 and Lemma 2.7 as it is a composition of impartial mechanisms.

Now consider a graph $G = (N, E)$, and a set I^* of vertices with $|I^*| = k$ and also $\sum_{i \in I^*} \delta^-(i) = \Delta_k$, and denote the vertex selected by the mechanism from A_j by i_j, for $j = 1, \dots, k$. For a fixed set A_j with $A_j \cap I^* \neq \emptyset$, let $i^* \in A_j \cap I^*$ be such that $i \in \pi_{<i^*}$ for all $i \in A_j \cap I^* \setminus \{i^*\}$. Then,

$$\begin{aligned}\delta^-(i_j) &\geq \delta^-_{N \setminus A_j}(i^*) + \delta^-_{A_j \cap \pi_{<i^*}}(i^*) \\ &= \delta^-_{N \setminus (A_j \cup I^*)}(i^*) + \delta^-_{(A_j \cap \pi_{<i^*}) \setminus I^*}(i^*) + \delta^-_{I^*}(i^*),\end{aligned}$$

where the inequality holds by a similar argument as in the proof of Lemma 2.9 and the equality because we have chosen i^* to be the vertex in $A_j \cap I^*$ that appears last in π.

In the permutation π used by the mechanism and chosen uniformly at random, a given vertex appears with probability $|A_j \cap I^*| / (|A_j \cap I^*| + 1)$ after i^*, so

$$\begin{aligned}&\mathbb{E}\left[\delta^-_{(A_j \cap \pi_{<i^*}) \setminus I^*}(i^*) \,\middle|\, |A_j \cap I^*| = l\right] \\ &\quad= \sum_{i \in N \setminus I^*} \mathbb{P}\left[i \in A_j \,\middle|\, |A_j \cap I^*| = l\right] \cdot \mathbb{P}\left[i \in \pi_{<i^*} \,\middle|\, |A_j \cap I^*| = l\right] \cdot \chi\left[(i, i^*) \in E\right] \\ &\quad= \frac{1}{k} \cdot \frac{l}{l+1}\Big(\mathbb{E}\left[\delta^-(i^*) - \delta^-_{I^*}(i^*)\right]\Big),\end{aligned}$$

where we have used that

$$\mathbb{P}\left[i \in A_j \,\middle|\, |A_j \cap I^*| = l\right] = \mathbb{P}\left[i \in A_j\right] = \frac{1}{k}$$

for $i \in N \setminus I^*$. Similarly,

$$\begin{aligned}\mathbb{E}\left[\delta^-_{N \setminus (A_j \cup I^*)}(i^*) \,\middle|\, |A_j \cap I^*| = l\right] &= \sum_{i \in N \setminus I^*} \mathbb{P}\left[i \in N \setminus A_j \,\middle|\, |A_j \cap I^*| = l\right] \cdot \chi\left[(i, i^*) \in E\right] \\ &= \frac{k-1}{k}\Big(\mathbb{E}\left[\delta^-(i^*) - \delta^-_{I^*}(i^*)\right]\Big).\end{aligned}$$

Thus, for a vertex chosen by the algorithm we expect indegree

$$\begin{aligned}\mathbb{E}\left[\delta^-(i_j) \,\middle|\, |A_j \cap I^*| = l\right] &\geq \left(\frac{k-1}{k} + \frac{l}{k(l+1)}\right) \cdot \mathbb{E}\left[\delta^-(i^*) - \delta^-_{I^*}(i^*)\right] + \mathbb{E}\left[\delta^-_{I^*}(i^*)\right] \\ &\geq \left(\frac{k-1}{k} + \frac{l}{k(l+1)}\right) \frac{\Delta_k}{k},\end{aligned}$$

and by linearity of expectation and using the law of total expectiation,

$$\begin{aligned}
\mathbb{E}\left[\frac{\sum_{j=1}^{k}\delta^{-}(i_j)}{\Delta_k}\right] &= \frac{1}{\Delta_k}\sum_{j=1}^{k}\mathbb{E}\left[\delta^{-}(i_j)\right] \\
&= \frac{1}{\Delta_k}\sum_{j=1}^{k}\sum_{l=1}^{k}\mathbb{E}\left[\delta^{-}(i_j)\,\middle|\,|I^*\cap A_j|=l\right]\cdot\mathbb{P}\left[|I^*\cap A_j|=l\right] \\
&\geq \frac{k}{\Delta_k}\sum_{l=1}^{k}\left(\frac{k-1}{k}+\frac{l}{k(l+1)}\right)\frac{\Delta_k}{k}\cdot\binom{k}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l} \\
&= \frac{k-1}{k}\sum_{l=1}^{k}\binom{k}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l} \\
&\qquad + \sum_{l=0}^{k}\frac{l}{k(l+1)}\binom{k}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l} \\
&= \frac{k-1}{k}\left(1-\left(1-\frac{1}{k}\right)^{k}\right) \\
&\qquad + \sum_{l=0}^{k}\frac{l}{k(l+1)}\binom{k}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l}. \qquad (2.2)
\end{aligned}$$

We can further calculate

$$\begin{aligned}
&\sum_{l=0}^{k}\frac{l}{k(l+1)}\binom{k}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l} \\
&\quad= \sum_{l=0}^{k}\frac{1}{k}\left(1-\frac{1}{l+1}\right)\binom{k}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l} \\
&\quad= \frac{1}{k}-\frac{1}{k}\sum_{l=0}^{k}\frac{1}{l+1}\cdot\frac{k!}{l!(k-l)!}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k-l} \\
&\quad= \frac{1}{k}-\frac{1}{k}\sum_{l=0}^{k}\frac{1}{k+1}\binom{k+1}{l+1}k\left(\frac{1}{k}\right)^{l+1}\left(1-\frac{1}{k}\right)^{(k+1)-(l+1)} \\
&\quad= \frac{1}{k}-\frac{1}{k+1}\sum_{l=1}^{k+1}\binom{k+1}{l}\left(\frac{1}{k}\right)^{l}\left(1-\frac{1}{k}\right)^{k+1-l} \\
&\quad= \frac{1}{k}-\frac{1}{k+1}\left(1-\left(1-\frac{1}{k}\right)^{k+1}\right)
\end{aligned}$$

and simplify the equation (2.2) to conclude that

$$\begin{aligned}
\mathbb{E}\left[\frac{\sum_{j=1}^{k}\delta^{-}(i_j)}{\Delta_k}\right] &\geq \left(1-\frac{1}{k}\right)\left(1-\left(1-\frac{1}{k}\right)^{k}\right)+\frac{1}{k}-\frac{1}{k+1}\left(1-\left(1-\frac{1}{k}\right)^{k+1}\right) \\
&= \frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right),
\end{aligned}$$

as claimed. □

It is again not hard to see that this analysis is tight.

Theorem 2.14 *If the k-partition mechanism with permutation is α-optimal, then we have that* $\alpha \leq \frac{k}{k+1}\left(1-\left(\frac{k-1}{k}\right)^{k+1}\right)$.

Proof. Consider a graph with a large number of vertices and k edges $(i, i_1^*), \ldots, (i, i_k^*)$ only. Let $I^* = \{i_1^*, \ldots, i_k^*\}$. When partitioning the vertices into the sets $A_1, \ldots, A_k$, it is without loss of generality to assume that $i \in A_1$. For each $j \in \{2, \ldots, k\}$, a vertex with indegree one is selected from A_j if and only if $I^* \cap A_j \neq \emptyset$. This happens with probability $1 - \left(\frac{k-1}{k}\right)^k$, so by linearity of expectation the expected sum of indegrees of the vertices selected from $A_2 \cup \cdots \cup A_k$ is

$$\mathbb{E}\left[\sum_{j=2}^{k} \delta^-(i_j)\right] = (k-1)\left(1-\left(1-\frac{1}{k}\right)^k\right).$$

From A_1, a vertex with indegree one is selected if $A_1 \cap \bar{\pi}_{<i} \cap I^* \neq \emptyset$, and this condition is in fact necessary with probability going to one as the number of vertices with indegree zero goes to infinity.

For any $l \in \{0, \ldots, k\}$ we have

$$\mathbb{P}[|A_1 \cap I^*| = l] = \binom{k}{l}\frac{1}{k}^l\left(1-\frac{1}{k}\right)^{k-l}$$

and

$$\mathbb{P}\left[A_1 \cap I^* \cap \bar{\pi}_{<i} \neq \emptyset \,\middle|\, |A_1 \cap I^*| = l\right] = \frac{l}{l+1}.$$

The probability of selecting a vertex with indegree one from $A_1 \cap I^*$ thus goes to

$$\sum_{l=0}^{k} \frac{l}{l+1}\binom{k}{l}\left(\frac{1}{k}\right)^l\left(1-\frac{1}{k}\right)^{k-l}$$

as the number of vertices goes to infinity.

The maximum overall indegree of any set of k vertices in the graph is k, so

$$\alpha \leq \frac{k-1}{k}\left(1-\left(1-\frac{1}{k}\right)^k\right) + \sum_{l=0}^{k} \frac{l}{k(l+1)}\binom{k}{l}\left(\frac{1}{k}\right)^l\left(1-\frac{1}{k}\right)^{k-l}.$$

This expression is equal to the lower bound in equation (2.2), and we conclude that the analysis in the proof of Theorem 2.13 is tight. □

2.3.2 Permutation Mechanisms for k Agents

Our understanding of deterministic mechanisms for the selection of more than two vertices is particularly limited. However, using techniques which we established beforehand, we can obtain a bound of an expected fraction of $1/k$ of the total best possible number of nominations using a permutation mechanism, as stated in the following theorem.

Figure 2.6: An example which proves that repeating the permutation mechanism destroys impartiality. On the left graph, π_4 is selected in the first round and π_3 in the second; if π_2 changes her nomination to π_4 however, as shown in the right graph, then π_3 is selected in the first round and π_2 in the second, which means that π_2 can influence her own probability to win by nominating strategically, a contradiction to impartiality.

Theorem 2.15 *The bidirectional permutation mechanism for selecting up to two agents is $1/k$-optimal for deterministic non exact selection of k agents.*

Proof. We observe that the selection of only two instead of k vertices reduces the guarantee by a factor of at most $2/k$. Using the bound provided by Theorem 2.6, we calculate $2/k \cdot 1/2 = 1/k$, which proofs the desired bound. □

Unfortunately, there is no natural generalization of the bidirectional permutation mechanism to select up to k agents. Many possible approaches that come to mind turn out to not be impartial. This seems particularly true for mechanisms that choose k agents by applying k times a selection of one agent while taking into account previous selections. For instance, applying the permutation algorithm two times in the same direction and not allowing the agent chosen in the first round to be candidate again is not impartial, as the simple example in Figure 2.6 shows.

We can, however, modify the bidirectional permutation algorithm such that each direction produces not one, but $k/2$ agents. Here again, we only look at nominations from agents that cannot be selected anymore, but instead keep $k/2$ agents as candidates. New agents enter the set of candidates if the number of nominations we see is at least as high as the number of nominations of our current worst candidate, which then leaves the set. If two or more candidates have the same lowest number of nominations, the one earlier in the permutation is considered worse. This still reduces the number of nominations that can be seen, as we can not see the nominations that the candidates cast each other. As k grows, our approximation guarantee gets worse. A formal description of the Algorithm, which we call bidirectional permutation algorithm for k agents, can be found in Algorithm 2.8. It uses as subroutine the h-extraction mechanism, described in Algorithm 2.9.

Similarly to the bidirectional permutation mechanism, we note the following.

Lemma 2.16 *The bidirectional k-permutation mechanism is impartial.*

Proof. We divide for each vertex her outgoing edges into two subsets: those going forward in the permutation and those going backward. First, we consider the $\lceil \frac{k}{2} \rceil$ vertices selected by forward direction. Only the edges going forward in the permutation can be considered in that case. The outgoing edges of a vertex are only taken into account once this vertex has been considered by the algorithm and is not a candidate anymore, thus can not be selected. While a vertex has not been considered or is still a candidate, her edges are not considered by the algorithm. Hence, for

Algorithm 2.8: The Bidirectional Permutation Mechanism for k Agents, using h-Extraction Mechanism Ξ^h_π

Input: Graph $G = (N, E)$, integer $k \leq |N|$
Output: Set $\{i_1, i_2, \ldots, i_k\} \subseteq N$ of at most two vertices

Let $\pi = (1, \ldots, n)$
Set $h := \lceil k/2 \rceil$
Set $\{i_1, \cdots, i_h\} := \Xi^h_\pi(G)$ ▷ select vertices based on forward edges
Set $\{i_{h+1}, \cdots, i_k\} := \Xi^{k-h}_{\bar{\pi}}(G)$ ▷ select vertices based on backward edges
return $\{i_1, i_2, \ldots, i_k\}$

Algorithm 2.9: The h-Extraction Mechanism Ξ^h_π

Input: Graph $G = (N, E)$, permutation $(\pi_1, \ldots, \pi_n)$ of N
Output: Set of vertices $\{i_1, \ldots, i_h\} \in N$

Set $i_1 := \pi_1, \ldots, i_h := \pi_h$ ▷ candidate vertices
Set $d_1 := 0, \ldots d_h = 0, \pi(i_1) = 1, \ldots, \pi(i_h) = h$ ▷ candidate indegrees and positions
Set $\ell := 1$ ▷ current worst candidate
for $j = h+1, \ldots, n$ **do**
    if $\delta^-_{\pi_{<\pi_j} \setminus \{i_1, \cdots, i_h\}}(\pi_j) \geq d_\ell$ **then** ▷ compare worst candidate and current vertex
        $d_\ell := \delta^-_{\pi_{<\pi_j} \setminus (\{i_1, \ldots, i_h\} \setminus i_\ell)}(\pi_j)$, $i_\ell := \pi_j$, $\pi(i_\ell) = j$ ▷ current vertex is candidate
        Set $W := \{w \in \{1, \ldots, h\} \mid d_w \leq d_r \quad \forall r \in \{1, \ldots, h\}\}$,
        let $\ell \in W$ s.t. $\pi(i_\ell) < \pi(i_v) \quad \forall v \in W \setminus \ell$ ▷ update current worst candidate
return $\{i_1, \ldots, i_h\}$

each vertex, her outgoing edges do not influence whether this vertex is selected. The same holds for backward direction. □

We now proceed to show an approximation guarantee for the bidirectional k-permutation mechanism, starting from the observation that the vertices selected by algorithm Ξ^h_π have a high number of incoming forward edges with respect to π.

Lemma 2.17 *If* $\{i_1, \ldots, i_h\} = \Xi^h_\pi(G)$, *then*

$$\sum_{i \in \{i_1, \ldots, i_h\}} \delta^-_{\pi_{<i}}(i, G) \geq \max_{J \subseteq N, |J| = h} \left\{ \sum_{j \in J} \delta^-_{\pi_{<j}}(j, G) \right\} - h^2 + h.$$

Proof. Let

$$D^* = \max_{J \subseteq N, |J| = h} \left\{ \sum_{j \in J} \delta^-_{\pi_{<j}}(j, G) \right\},$$

and let i^* be an arbitrary vertex with indegree $d^* = \delta^-_{\pi_{<i^*}}(i^*)$ in such a set of vertices D^* with maximum degrees. When i^* is considered by the algorithm, so are at least $d^* - h$ of its incoming forward edges, h of which may originate from the current candidates $(i_1, \ldots, i_h)$, i. e.

$$\delta^-_{\pi_{<i^*} \setminus \{i_1, \ldots, i_h\}}(i^*) \in \{d^* - h, \ldots, d^*\}.$$

If

$$\delta^-_{\pi_{<i^*}\setminus\{i_1,\dots,i_h\}}(i^*) < \delta^-_{\pi_{<i}}(i) \text{ for all } i \in \{i_1,\dots,i_h\}$$

then i^* is not selected for the pool of candidates in the algorithm, but also the smallest indegree d at the moment when i^* is considered by the algorithm is at least $d^* - h + 1$. If, on the other hand, both

$$\delta^-_{\pi_{<i^*}\setminus\{i_1,\dots,i_h\}}(i^*) = d^* - h + \ell \text{ and } \exists\, i \in (i_1,\dots,i_h) \text{ s.t. } \delta^-_{\pi_{<i}}(i) \leq d^* - h + \ell,$$

for some $\ell \in \{1,\dots,h\}$, then i^* becomes a new candidate. For the special case that

$$\delta^-_{\pi_{<i^*}\setminus\{i_1,\dots,i_h\}}(i^*) = d^* - h \text{ and } \exists\, i \in (i_1,\dots,i_h) \text{ s.t. } \delta^-_{\pi_{<i}}(i) \leq d^* - h,$$

also i^* enters the pool of candidates and makes a vertex, say i_d for some $d \in \{1,\dots,h\}$, leave the pool of candidates. But as $\delta^-_{\pi_{<i^*}\setminus\{i\}}(i^*) = d^* - h$, each of the vertices in $\{i_1,\dots,i_h\}$ nominates i^*, and thus

$$\delta^-_{\pi_{<i^*}\setminus\left(\{i_1,\dots,i_h\}\setminus i_d\right)}(i^*) = d^* - h + 1.$$

In both cases when i^* enters the candidate pool, any vertex that possibly becomes a candidate after i^* is considered by the algorithm and makes possibly i^* leave the candidate pool has at least $d^* - h + 1$ incoming forward edges as well. In all cases, at any point in the mechanism after i^* is considered, there is always at least one vertex in the candidate pool with at least $d^* - h + 1$ incoming vertices and thus loosing at most $h - 1$ incoming nominations for vertex i^*. As this argument can be made for each of the h candidate vertices, we reach a total indegree of

$$\max_{J\subseteq N, |J|=h} \left\{ \sum_{j\in J} \delta^-_{\pi_{<j}}(j, G) \right\} - h \cdot (h-1),$$

establishing the claim for this case. □

Theorem 2.18 *The bidirectional k-permutation mechanism is impartial and achieves in expectation at least*

$$\begin{cases} \frac{\Delta_k}{2} - \frac{k^2}{2} + k & \text{if } k \text{ even} \\ \frac{k-1}{2k}\,\Delta_k - \frac{k^2-k+1}{2} & \text{if } k \text{ odd} \end{cases}$$

nominations.

Proof. Impartiality follows directly from Lemma 2.16.

Let k be even. Now consider a graph $G = (N, E)$, a set of vertices I^* with

$$\sum_{i^*\in I^*} \delta^-(i^*) = \Delta_{k/2},$$

and let $I_1 = \Xi^{k/2}_{\pi}(G)$ and $I_2 = \Xi^{k/2}_{\bar{\pi}}(G)$. By Lemma 2.17,

$$\sum_{i_z\in I_z} \delta^-_{\pi_{<i_z}}(i_z) \geq \sum_{i^*\in I^*} \delta^-_{\pi_{<i^*}}(i^*) - \frac{k^2}{4} + \frac{k}{2}$$

for $z \in \{1,2\}$, regardless of whether any elements are both in I_1 and I_2. Thus, by linearity of expectation,

$$\begin{aligned}\mathbb{E}\left[\sum_{i\in I_1\cup I_2}\delta^-(i)\right] &\geq \mathbb{E}\left[\sum_{i_1\in I_1}\delta^-_{\pi_{<i_1}}(i_1)+\sum_{i_2\in I_2}\delta^-_{\bar{\pi}_{<i_2}}(i_2)\right]\\ &\geq \sum_{i^*\in I^*}\mathbb{E}\left[\delta^-_{\pi_{<i^*}}(i^*)\right]-\frac{k^2}{4}+\frac{k}{2}+\sum_{i^*\in I^*}\mathbb{E}\left[\delta^-_{\bar{\pi}_{<i^*}}(i^*)\right]-\frac{k^2}{4}+\frac{k}{2}\\ &= \sum_{i^*\in I^*}\mathbb{E}\left[\delta^-(i^*)\right]-\frac{k^2}{2}+k=\Delta_{k/2}-\frac{k^2}{2}+k\geq\frac{1}{2}\Delta_k-\frac{k^2}{2}+k,\end{aligned}$$

as claimed.

Let k be odd. Now consider a graph $G=(N,E)$, a set of vertices I^* with

$$\sum_{i^*\in I^*}\delta^-(i^*)=\Delta_{\frac{k-1}{2}},$$

a set of vertices I^{**} with

$$\sum_{i^*\in I^{**}}\delta^-(i^*)=\Delta_{\frac{k+1}{2}}$$

and $I^*\subset I^{**}$, and let $I_1=\Xi_{\pi}^{\frac{k+1}{2}}(G)$ and $I_2=\Xi_{\bar{\pi}}^{\frac{k-1}{2}}(G)$. By Lemma 2.17, we have

$$\sum_{i_1\in I_1}\delta^-_{\pi_{<i_1}}(i_1)\geq\sum_{i^*\in I^{**}}\delta^-_{\pi_{<i^*}}(i^*)-\frac{(k+1)^2}{4}+\frac{k+1}{2}$$

and similarly we get

$$\sum_{i_2\in I_2}\delta^-_{\bar{\pi}_{<i_2}}(i_2)\geq\sum_{i^*\in I^*}\delta^-_{\bar{\pi}_{<i^*}}(i^*)-\frac{(k-1)^2}{4}+\frac{k-1}{2},$$

regardless of whether any elements are both in I_1 and I_2. Using further linearity of expectation, we get

$$\begin{aligned}\mathbb{E}\left[\sum_{i\in I_1\cup I_2}\delta^-(i)\right] &\geq \mathbb{E}\left[\sum_{i_1\in I_1}\delta^-_{\pi_{<i_1}}(i_1)+\sum_{i_2\in I_2}\delta^-_{\bar{\pi}_{<i_2}}(i_2)\right]\\ &\geq \sum_{i^*\in I^{**}}\mathbb{E}\left[\delta^-_{\pi_{<i^*}}(i^*)\right]-\frac{(k+1)^2}{4}+\frac{k+1}{2}\\ &\qquad+\sum_{i^*\in I^*}\mathbb{E}\left[\delta^-_{\bar{\pi}_{<i^*}}(i^*)\right]-\frac{(k-1)^2}{4}+\frac{k-1}{2}\\ &= \sum_{i^*\in I^*}\mathbb{E}\left[\delta^-(i^*)\right]-\frac{k^2-k+1}{2}=\Delta_{\frac{k-1}{2}}-\frac{k^2-k+1}{2}\\ &\geq \frac{k-1}{2k}\Delta_k-\frac{k^2-k+1}{2},\end{aligned}$$

as claimed. □

Algorithm 2.10: The Exact Dollar Partition with Permutation Mechanism, using Allocation Subroutine $\xi_{t_1,\dots,t_k}$ and Extraction Mechanism $\Xi^h_{\pi,A}$

Input: Graph $G = (N, E)$, integer $k \leq |N|$
Output: Set $\{i_1, i_2, \dots, i_k\} \subseteq N$ of up to k vertices

Assign each $i \in N$ independently and uniformly at random to one of k sets $A_1, \dots, A_k$
Choose a permutation $(\pi_1, \dots, \pi_n)$ of N uniformly at random
Set $R := \emptyset$
for $i = 1, \dots, k$ **do**
    if $\delta^-_{A_i}(N \setminus A_i) \equiv 0$ **then**
        Let $R \leftarrow R \cup i$ ▷ sets that do not give outside nominations
for $j = 1, \dots, k$ **do**
    Set $t_j := \left(\sum_{i \in \{1,\dots,k\} \setminus (j \cup R)} \frac{\delta^-_{A_i}(A_j)}{\sum_{\ell \in \{1,\dots,k\} \setminus i} \delta^-_{A_i}(A_\ell)} \right) + \frac{|R|}{k-1}$ ▷ received dollar shares
    if $j \in R$ **then**
        Let $t_j \leftarrow t_j - \frac{1}{k-1}$ ▷ corrective term for sets without outside nominations
Set $(s_1, \dots, s_k) := \xi_{t_1,\dots,t_k}$ ▷ improved subroutine to get a discrete allocation according to a probability distribution with expected value $(t_1, \dots, t_k)$
Set $I := \emptyset$ ▷ set of selected agents
for $j = 1, \dots, k$ **do**
    Let $I \leftarrow I \cup \Xi^{s_j}_{\pi,A_j}(G)$ ▷ select s_j agents from A_j using extraction mechanism
return $\{i_1, i_2, \dots, i_k\}$

2.3.3 Exact Dollar Partition with Permutation

In addition to generalizations of the permutation and partition mechanism, we present a new mechanism to chose k agents. It is strongly based on Exact Dollar Partition by Aziz et al. [ALM+16]. They were themselves inspired by a strategyproof mechanism for dividing a continuous resource called *Dividing a Dollar* by de Clippel et al. [dCMT08].

One of the main weaknesses of the k-partition Algorithm 2.7 is that even if many agents with a high number of nominations are in one partition, only one of them can be selected. Disadvantageous cases for large k and only k agents with nominations can easily be constructed as the probability of two or more agents who receive nominations being in one partition and thus losing nominations grows with larger k. Exact Dollar Partition with Permutation attempts to solve this problem by first using the nominations to decide how many agents shall be selected from each partition.

The mechanism of Aziz et al. [ALM+16] uses a randomized apportionment subroutine that they state is interesting in its own right. Apportionment is the allocation of representatives or resources in proportion to group sizes or demands. It can be considered as a rather fundamental problem as it appears in a variety of contexts. The goal of the subroutine is to round fractional group sizes to integers in a fair way. However, the subroutine proposed by Aziz et al. [ALM+16] is fairly intricate. One of our main contributions is to propose a routine which we believe to be simpler to achieve apportionment.

Exact Dollar Partition with Permutation, described in Algorithm 2.10, works as

Algorithm 2.11: Allocation Subroutine $\xi_{t_1,\dots,t_k}$ to Map a Rational Allocation to Probability Distribution Over Discrete Allocations

Input: Rational allocation $(t_1, \dots, t_k)$
Output: Discrete allocation $(s_1, \dots, s_k)$ selected according to a distribution that has as expected value $(t_1, \dots, t_k)$

Set $s^1 := (0, \dots, 0)$ ▷ s: discrete allocation
Set $a := 1$ ▷ a: current position in list of discrete allocations
Set $z^1 := 0, z^2 := 1$ ▷ z^a: probability starting point of allocation s^a respectively probability end point of allocation s^{a+1} to calculate allocation probability
Set $\ell := 1$ ▷ ℓ: total number of discrete allocations
for $i = 1, \dots, k$ **do** ▷ distribute rational allocation parts t_i one by one
    if $t_i \geq 1$ **then** ▷ distribute integral amount of t_i to each discrete allocation
        for $j = 1, \dots, \ell$ **do**
            Let $s_i^j \leftarrow s_i^j + \lfloor t_i \rfloor$
        Let $t_i \leftarrow t_i - \lfloor t_i \rfloor$
    while $t_i > 0$ **do** ▷ distribute non-integral amount of t_i
        if $a \equiv \ell + 1$ **then** ▷ if end of list is reached, continue again from start
            Let $a \leftarrow 1$
        else if $t_i \geq z^{a+1} - z^a$ **then** ▷ enough t_i left to continue
            Let $s_i^a \leftarrow s_i^a + 1, t_i \leftarrow t_i - z^{a+1} + z^a, a \leftarrow a + 1$ ▷ distribute part of t_i
        else ▷ t_i ends before next probability starting point
            Set $s^{\ell+1} := s^\ell, z^{\ell+1} \leftarrow z^\ell, z^{\ell+2} := 1$ ▷ add new discrete allocation
            for $m = \ell, \dots, a + 1$ **do**
                Let $s^m \leftarrow s^{m-1}, z^m \leftarrow z^{m-1}$ ▷ s^a and s^{a+1} are equal, the discrete allocations afterwards move one index up
            Let $\ell \leftarrow \ell + 1, s_i^a \leftarrow s_i^a + 1, z^{a+1} \leftarrow z^a + t_i, a \leftarrow a + 1$, **break** ▷ update values
$x \sim \text{unif}[0, 1)$, let f such that $z^f \leq x$ and $z^{f+1} > x$
return $(s_1^f, \dots, s_k^f)$

follows. The mechanism randomly partitions the set of vertices into k sets $A_1, \dots, A_k$ such that each partition size differs by at most one. It is then decided how many agents are selected from each individual partition. In order to do that, each partition "divides a dollar" proportionally to the number of nominations they give to the other partitions. That is, each partition B has a number of nominations $v_B(A)$ that they give to another partition A and these numbers are normalized such that

$$\sum_{A \in \{A_1,\dots,A_k\} \setminus B} v_B(A) = 1.$$

If one partition B does not contain vertices which give any nominations to vertices outside of B, their dollar is shared uniformly amongst the other partitions such that we have $v_B(A) = 1/k-1$ for all $A \in \{A_1, \dots, A_k\} \setminus B$. This leads to each partition A having a total share allocated to them by the other partitions of

$$t_A := \sum_{B \in \{A_1,\dots,A_k\} \setminus A} v_B(A).$$

If all t_A are integer, then we already have a discrete allocation s of how many agents

Algorithm 2.12: The h-Extraction Mechanism $\Xi^h_{\pi,A}$ Restricted to a Set $A \subseteq N$

Input: Graph $G = (N, E)$, permutation $(\pi_1, \ldots, \pi_n)$ of N, set $A \subseteq N$
Output: Set of vertices $\{i_1, \ldots, i_h\} \in N$

if $|A| \leq h$ **then**
  return A ▷ A is smaller than the set of agents to be selected from it
Set $m := 1, \ell := 1, j := 1$ ▷ m: worst candidate, ℓ: number of candidates
while $\ell < h$ **do** ▷ find first h candidates in A
  if $\pi_j \in A$ **then**
    Set $i_\ell := \pi_j, d_\ell := \delta^-_{N\setminus A}(\pi_j)$ ▷ new candidate and her indegree
    if $d_\ell < d_m$ **then**
      $m \leftarrow \ell$ ▷ update current worst candidate
    Let $\ell \leftarrow \ell + 1$
  Let $j \leftarrow j + 1$
while $j \leq |N|$ **do**
  if $\pi_j \in A$ **and** $\delta^-_{(N\setminus A)\cup\pi_{<\pi_j}\setminus\{i_1,\cdots,i_h\}}(\pi_j) \geq d_m$ **then** ▷ compare indegree of current worst candidate and current vertex
    Let $d_m \leftarrow \delta^-_{(N\setminus A)\cup\pi_{<\pi_j}\setminus(\{i_1,\ldots,i_h\}\setminus i_m)}(\pi_j), i_m \leftarrow \pi_j$ ▷ current vertex becomes new candidate
    Set $W := \{w \in \{1,\ldots,h\} \mid d_w \leq d_r \quad \forall r \in \{1,\ldots,h\}\}$,
    $m \in W$ s.t. $i_m < i_v \quad \forall v \in W \setminus m$ ▷ find new worst candidate
  Let $j \leftarrow j + 1$
return $\{i_1, \ldots, i_h\}$

will be selected from each partition, namely $s_A := t_A$. Otherwise, we use the improved allocation subroutine to compute a probability distribution over discrete allocations such that the expected value of this distribution is t_A for each partition A and receive a discrete allocation $(s_{A_1}, \ldots, s_{A_k})$ according to that distribution. We proceed to select s_A agents from each partition A using the h-extraction mechanism restricted to set A. It basically works the same as the k-partition algorithm if only one agent shall be selected from a partition. If more than one agent shall be selected from a partition, then the mechanism uses the k-permutation algorithm within that partition while also taking into consideration nominations from outside.

Algorithm 2.10 gives a formal description of the mechanism. It builds upon Allocation Subroutine $\xi_{A_1,\ldots,A_k}(G)$ which is described in Algorithm 2.11 and the h-Extraction Mechanism restricted to set A which is described in Algorithm 2.12.

In order to prove impartiality, first we must show that the allocation subroutine indeed achieves to return a discrete allocation according to a probability distribution that has as expected value the rational allocations that each partition receives. The idea behind the allocation subroutine is explained in Figure 2.7. The rational allocations are looked at one by one. Each t_A corresponds to a line piece with length t_A which we can imagine of drawing one following another. While drawing, we start at position zero and each new piece starts where the last ended. When a line piece crosses position one, we stop, and continue drawing at position zero next to the former line. Each discrete allocation corresponds to an interval somewhere between

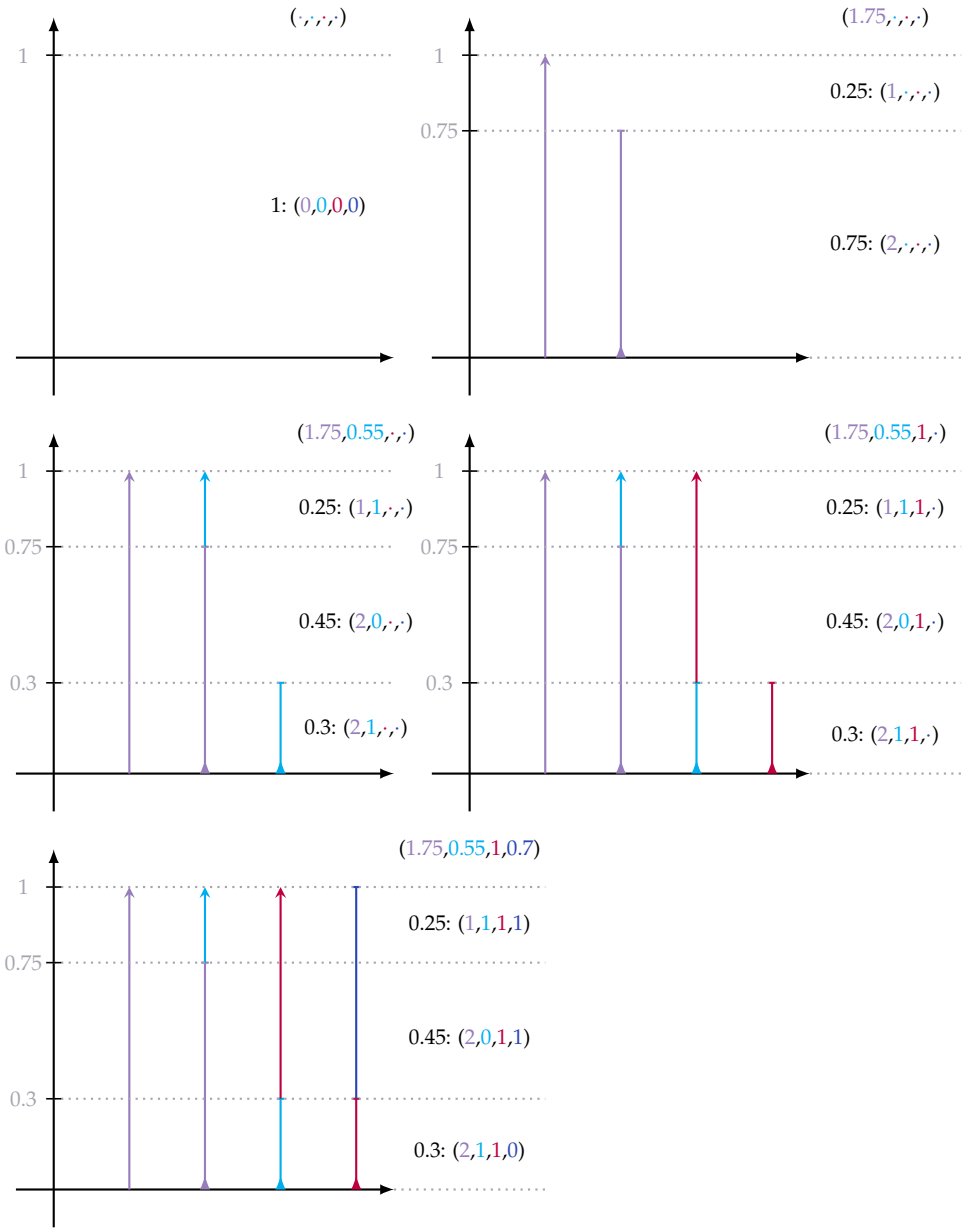


Figure 2.7: Example for the Allocation Subroutine in Algorithm 2.11. Given a rational allocation (1.75,0.55,1,0.7), the subroutine calculates a probability distribution over discrete allocations. We start with one discrete allocation s^1 =(0,0,0,0) on interval $[0,1)$. After adding t_1 =1.75, we have two discrete allocations: s^1 =(2,0,0,0) with $z^1 = 0$ on interval $[0, 0.75)$ and s^2 =(1,0,0,0) with $z^2 = 0.75$ on interval $[0.75, 1)$; $l = 2$ here and $z^3 = 1$. After adding t_2 =0.55, we get three discrete allocations: s^1 =(2,1,0,0) on $[0, 0.3)$, s^2 =(2,0,0,0) on $[0.3, 0.75)$ and s^3 =(1,1,0,0) on $[0.75, 1)$. Similarly, adding t_3 =1, we continue to have three discrete allocations: s^1 =(2,1,1,0) on $[0, 0.3)$, s^2 =(2,0,1,0) on $[0.3, 0.75)$ and s^3 =(1,1,1,0) on $[0.75, 1)$. After adding t_4 =0.7, we have our final three discrete allocations: s^1 =(2,1,1,0) with $z^1 = 0$ on interval $[0, 0.3)$ and thus with probability 0.3, s^2 =(2,0,1,1) with $z^2 = 0.3$ on interval $[0.3, 0.75)$ and hence probability 0.45, and s^3 =(1,1,1,1) with $z^3 = 0.75$ on interval $[0.75, 1)$ with probability 0.25.

zero and one. We start the process with an empty single discrete allocation which corresponds to the complete interval between zero and one. When a new rational allocation part t_A is added, each existing discrete allocation s^i adds the integral part of t_A to place s^i_A and we continue with the rational part of t_A. Now, either the end point of the line corresponding to t_A lies on the start- or endpoint of an existing discrete allocation; then, each allocation s^i which the line t_A spans adds one to s^i_A. Otherwise, the discrete allocation s^j which corresponds to the interval where t_A ends, is split in two discrete allocations,both with the same entries as s^j, but new intervals. The first one starts where s^j started and its interval ends at the point where t_A ends, the second one starts at the endpoint of t_A and ends where s^j ended. Now again, all allocations s_i which t_A spans up to that point add one to s^i_A. When the process is finished, the probability of each discrete allocation to be selected corresponds to the length of their interval.

Theorem 2.19 *The allocation subroutine $\xi_{t_1,\dots,t_k}$ returns one integer allocation $(s_1,\dots,s_k)$ of at most $k+1$ different integer allocations with*

$$s_i \in \{\lfloor t_i \rfloor, \lceil t_i \rceil\} \quad \forall\, i \in \{1,\dots,k\}.$$

The expected allocation of each partition i is its share t_i for each $i = 1,\dots,k$, that is

$$\mathbb{E}\big[(s_1,\dots,s_k)\big] = (t_1,\dots,t_k).$$

Further,

$$\sum_{i=1,\dots,k} s_i \in \left\{ \left\lfloor \sum_{i=1,\dots,k} t_i \right\rfloor, \left\lceil \sum_{i=1,\dots,k} t_i \right\rceil \right\}, \tag{2.3}$$

which is, in particular, equal to k when the subroutine is evoked by the exact dollar partition with permutation mechanism.

Proof. The allocation subroutine, which runs in polynomial time to the size of the input $(t_1,\dots,t_k)$, computes up to $\ell \leq k+1$ different discrete allocations s^a, as it starts with one discrete allocation and in each of the k iterations at most one new allocation is introduced.

We now proof by induction that at any time after t_i was considered by the algorithm, we have

$$\sum_{a=1}^{\ell} \big(z^{a+1} - z^a\big) s^a_i = t_i. \tag{2.4}$$

We consider time i when t_i is processed by the algorithm. At the begin of the iteration, we have $s^a_i = 0$ which changes to $s^a_i = \lfloor t_i \rfloor$ for all $a = 1,\dots,\ell$ after line six in the algorithm, thus

$$\sum_{a=1}^{\ell} \big(z^{a+1} - z^a\big) s^a_i = \sum_{a=1}^{\ell} \big(z^{a+1} - z^a\big) \lfloor t_i \rfloor = \big(z^{\ell+1} - z^1\big) \lfloor t_i \rfloor = \lfloor t_i \rfloor.$$

Remark that $t_i - \lfloor t_i \rfloor < 1$ and thus after the first step only the non integer amount of t_i has to be distributed. In the while loop starting at line seven in the algorithm,

each time that one s_i^a with discrete allocation start point z^a and discrete allocation end point z^{a+1} for some $a = 1, \dots, \ell$ is considered and thus increased by one, we decrease t_i by $(z^{a+1} - z^a) \cdot 1$. As the while loop considers the discrete allocations one after another, $t_i - \lfloor t_i \rfloor < 1$, and $z^{\ell+1} - z^1 = 1$, this also means that each s_i^a is at most once increased by one during that stage, thus

$$s_i^a \in \{\lfloor t_i \rfloor, \lceil t_i \rceil\} \quad \forall\, a \in \{1, \dots, \ell\}. \tag{2.5}$$

Let A be the set of discrete allocations where this increase happens. Then at the end of iteration i we have

$$\begin{aligned}
\sum_{a=1}^{\ell} \big(z^{a+1} - z^a\big) s_i^a &= \sum_{a \in A} \big(z^{a+1} - z^a\big) s_i^a + \sum_{a \in \{1,\dots,\ell\} \setminus A} \big(z^{a+1} - z^a\big) s_i^a \\
&= \sum_{a \in A} \big(z^{a+1} - z^a\big) (\lfloor t_i \rfloor + 1) + \sum_{a \in \{1,\dots,\ell\} \setminus A} \big(z^{a+1} - z^a\big) \lfloor t_i \rfloor \\
&= \sum_{a \in A} \big(z^{a+1} - z^a\big) \cdot 1 + \sum_{a \in \{1,\dots,\ell\}} \big(z^{a+1} - z^a\big) \lfloor t_i \rfloor \\
&= (t_i - \lfloor t_i \rfloor) + 1 \cdot \lfloor t_i \rfloor = t_i,
\end{aligned}$$

as claimed. Now consider time $i + 1$. If no new allocation is added, the statement stays true as s_i^a is not altered by the algorithm for any $a = 1, \dots, \ell$. Otherwise, at most one new allocation enters the list of discrete allocation, suppose by splitting allocation s^{a^*}. Then we have $s_i^{a^*+1} = s_i^{a^*}$, and for the new number of discrete allocations $\ell(i+1)$ after iteration $i + 1$ we get

$$\begin{aligned}
\sum_{a=1}^{\ell(i+1)} \big(z^{a+1} - z^a\big) s_i^a &= \sum_{a=1}^{a^*-1} \big(z^{a+1} - z^a\big) s_i^a + \sum_{a=a^*+2}^{\ell(i+1)} \big(z^{a+1} - z^a\big) s_i^a \\
&\qquad + \big(z^{a^*+1} - z^{a^*}\big) s_i^{a^*} + \big(z^{a^*+2} - z^{a^*+1}\big) s_i^{a^*+1} \\
&= \sum_{a=1}^{a^*-1} \big(z^{a+1} - z^a\big) s_i^a + \big(z^{a^*+2} - z^{a^*}\big) s_i^{a^*} + \sum_{a=a^*+2}^{\ell(i+1)} \big(z^{a+1} - z^a\big) s_i^a \\
&= t_i,
\end{aligned}$$

as there is a one to one correspondence of equal summands from this iteration and the iteration before.

As each discrete allocation s^a is given as output by the algorithm with a probability of $z^{a+1} - z^a$, we can now calculate

$$\begin{aligned}
\mathbb{E}\big[(s_1, \dots, s_k)\big] &= \sum_{a=1}^{\ell} \mathbb{P}(s^a \text{ is selected}) \cdot (s_1^a, \dots, s_k^a) \\
&= \sum_{a=1}^{\ell} (z^{a+1} - z^a) \cdot (s_1^a, \dots, s_k^a) \\
&= \left(\sum_{a=1}^{\ell} (z^{a+1} - z^a) \cdot s_1^a, \dots, \sum_{a=1}^{\ell} (z^{a+1} - z^a) \cdot s_k^a \right) \\
&= (t_1, \dots, t_k),
\end{aligned}$$

where the last step holds due to equation (2.4).

Further, as each time that a new discrete allocation a^* is added after t_i was considered by the algorithm we have that the new value $s_i^{a^*}$ is equal to s_i^a for some other allocation a, and using term (2.5), we get

$$s_i \in \{\lfloor t_i \rfloor, \lceil t_i \rceil\}.$$

It remains to show that expression (2.3) is true. We prove this by showing the stronger statement

$$\sum_{i=1}^{j} s_i^a = \begin{cases} \left\lceil \sum\limits_{i=1}^{j} t_i \right\rceil & \text{if } z^a < \sum\limits_{i=1}^{j} t_i - \left\lfloor \sum\limits_{i=1}^{j} t_i \right\rfloor \\ \left\lfloor \sum\limits_{i=1}^{j} t_i \right\rfloor & \text{otherwise} \end{cases}$$

for all $j = 1, \ldots, k$ and $a = 1, \ldots, \ell$ by induction over the length of the sum j. The statement is clear for $j = 1$, as $s_1^a = \lfloor t_i \rfloor$ for all a with $z^a \geq t_i - \lfloor t_i \rfloor$, and $s_1^a = \lceil t_i \rceil$ otherwise, as the discrete allocations corresponding to the latter ones are considered after line seven in the algorithm. Now suppose the statement is true for j and consider $j + 1$. Let a^* be such that

$$z^{a^*} = \sum_{i=1}^{j} t_i - \left\lfloor \sum_{i=1}^{j} t_i \right\rfloor$$

and observe that this is the value which a has in the beginning of the iteration considering t_{j+1} in the allocation subroutine. We distinguish two cases. If

$$z^{a^*} + t_{j+1} - \lfloor t_{j+1} \rfloor \leq 1,$$

then

$$\left\lfloor \sum_{i=1}^{j+1} t_i \right\rfloor = \left\lfloor \sum_{i=1}^{j} t_i \right\rfloor + \lfloor t_{j+1} \rfloor$$

and the statement follows as the value of a after the iteration considering t_{j+1} in the allocation subroutine has finished is some a^{**} such that

$$z^{a^{**}} = z^{a^*} + t_{j+1} - \lfloor t_{j+1} \rfloor$$

and thus we have

$$s_j^a = \begin{cases} \lceil t_{j+1} \rceil & \text{if } z^{a^*} \leq z^a < z^{a^{**}} \\ \lfloor t_{j+1} \rfloor & \text{otherwise,} \end{cases}$$

which leads to the desired result. In the second case, we have

$$\left\lfloor \sum_{i=1}^{j+1} t_i \right\rfloor = \left\lfloor \sum_{i=1}^{j} t_i \right\rfloor + \lfloor t_{j+1} \rfloor + 1$$

and, similarly, the statement follows as the value of a after the iteration considering the rational share t_{j+1} in the allocation subroutine has finished is a^{**} such that

$$z^{a^{**}} = z^{a^*} + t_{j+1} - \lfloor t_{j+1} \rfloor - 1$$

and we have

$$s_j^a = \begin{cases} \lceil t_{j+1} \rceil & \text{if } z^{a^*} \leq z^a \text{ or } z^a < z^{a^{**}} \\ \lfloor t_{j+1} \rfloor & \text{otherwise,} \end{cases}$$

which finishes the proof. □

Remark 2.20 Remark that this proof, and indeed the allocation subroutine, also works in the case that

$$\sum_{i=1,\dots,k} t_i \neq k.$$

Similarly to Aziz et al. [ALM+16], we remark that the allocation subroutine $\xi_{t_1,\dots,t_k}$ is simple with an intuitive graphical explanation, can be computed in linear time, is in expectation equal to the rational allocation shares, returns shares s_i for each partition that are either $\lfloor t_i \rfloor$ or $\lceil t_i \rceil$, computes only at most k discrete allocations and relies on a single round of randomization, i. e. one random number. We therefore believe it is an improvement to the allocation subroutine proposed by Aziz et al. [ALM+16] and, as they state, of further interest. In order to see further interest outside of this work, consider the problem where we are given a number of jobs and a larger number of people, such that a subset of the people shall be selected which will be assigned to the jobs. Further, the people are divided into groups and the assignment of the jobs should happen in a fair way concerning the representation of groups, that means proportional to group size. This problem appears amongst others in the German Bundestag with its multi party system, the European Parliament with its many different member countries of different size as well as the US congress with different states, and in many other cases where committees are selected. It is not only studied in fields immediately related to this thesis such as operations research, computer science and economics, but also in political science (see, e. g. , [Bir76, You94, Puk14]).

Further, it is of relevance to the impartial mechanisms presented in this work which are described explicitly by giving selection probabilities for all voting graphs.

Remark 2.21 Given a set of agents with selection probabilities, the allocation subroutine returns a set of agents which meet in expectation these selection probabilites.

Having established that the subroutines deliver the desired result, we now observe the following for Exact Dollar Partition with Permutation (EDPP).

Lemma 2.22 *Exact Dollar Partition with Permutation is impartial.*

Proof. As Exact Dollar Partition is strategyproof (impartial) by Aziz et al. [ALM+16], impartiality of Exact Dollar Partition with Permutation follows directly with Theorem 2.19, Lemma 2.16 and Theorem 2.13. To see that, observe that no agent can influence how many agents are chosen from her partition, as this part of the mechanism

works the same way as Exact Dollar Partition [ALM+16] and our different allocation subroutine does not change the expected number of agents chosen from each partition by Theorem 2.19. For each agent, her nominations count either only for agents in other partitions or after she can not be selected in her partition anymore, parallel to Lemma 2.16 and Theorem 2.13. □

EDPP is designed in order to avoid a particular problem for k-partition when many agents with a high number of nominations end up being in the same partition. In contrast to that, in the case of one agent with a very high number of nominations, several agents without nominations in the same partition and several agents with a medium high number of nominations in other partition, EDPP might end up having to select most agents from the partition with only one agent with a very high number of nominations and thus missing all agents with medium number of nominations. It is therefore difficult to establish an approximation factor. However, the following theorem serves to establish how well Exact Dollar Partition with Permutation works in comparison with other mechanisms.

Theorem 2.23 *Exact Dollar Partition with Permutation is impartial and in expectation at least as good as Exact Dollar Partition by Aziz et al. [ALM+16].*

Proof. Impartiality follows with Lemma 2.22. Now consider an instance of k-selection and divide the agents in k partitions according to Exact Dollar Partition. In expectation, Exact Dollar Partition (EDP) and Exact Dollar Partition with Permutation (EDPP) choose the same number of agents in each partition. It suffices to show that for any number ℓ of agents selected in any one partition the number of nominations they receive is as least as high in EDPP as in EDP. Let $\{i_1, \dots, i_\ell\}$ be the agents selected in partition A_j by EDP with

$$d^{\mathrm{EDP}} = \sum_{i \in \{i_1, \dots, i_\ell\}} \delta^-_{V \setminus A_j}(i).$$

By Theorem 2.19, we can assume that the number of agents selected in partition A_j in EDP and in EDPP is equal. We now consider the vertices selected by EDPP. When a vertex $i_h \in \{i_1, \dots, i_\ell\}$ is considered by the algorithm, its indegree is at least

$$\delta^-_{(V \setminus A_j) \cup \left(\pi_{<i_h} \setminus \left\{i_1^{(h)}, \dots, i_\ell^{(h)}\right\}\right)}(i_h) \geq \delta^-_{V \setminus A_j}(i_h),$$

where $\left\{i_1^{(h)}, \dots, i_\ell^{(h)}\right\}$ are the candidate vertices when i_h is considered. Similarly to the proof of Theorem 2.18, any vertex that enters the set of candidates after i_h is considered and makes i_h leave the set of candidates has at least as many incoming nominations. Thus,

$$\begin{aligned} d^{\mathrm{EDPP}} &\geq \sum_{h=1,\dots,\ell} \delta^-_{(V \setminus A_j) \cup \left(\pi_{<i_h} \setminus \left\{i_1^{(h)}, \dots, i_\ell^{(h)}\right\}\right)}(i_h) \\ &\geq \sum_{h=1,\dots,\ell} \delta^-_{V \setminus A_j}(i_h) = d^{\mathrm{EDP}}, \end{aligned}$$

which finishes the proof. □

We can therefore extrapolate that conclusions drawn for EDP by comparing it to other mechanisms hold for EDPP as well.

Aziz et al. [ALM+16] compare EDP in computational experiments with Vanilla, Partition, Dollar Raffle, Dollar Partition Raffle and Credible Subset. *Vanilla* selects the k agents that receive the highest number of nominations. It is not impartial. *Partition* divides agents in k partitions and choses in each partition one agent that receives most nominations from outside that partition. *Dollar Raffle* works by having each agent divide a dollar amongst all other agents. The results are scaled to define a probability distribution over the agents. Repeatedly, one agent is selected according to that distribution, until k agents are selected. *Dollar Partition Raffle* divides agents in partitions and uses dollar shares to define a probability distribution over the partitions. Using this distribution, it draws repeatedly partitions and selects the best not yet selected agent from that partition according to nominations from outside that partition until k agents are selected. Both Dollar Raffle and Dollar Partition Raffle are not impartial. *Credible Subset* first defines two groups: T is the set of agents who receive the k highest nominations, and P is the set of agents who would be in the set T if they did not give any nominations. Then with probability $\frac{k+|P|}{k+m}$, the mechanism selects k agents uniformly at random from $T \cup P$, and otherwise it selects none. Credible Subset is impartial.

They implemented these mechanisms with Python and PrefLib [MW13] and tested them using data similar to the data put forward in a pilot by the National Science Foundation [Nat15], which was used to experiment on new peer review processes. It has 131 proposals, each reviewer giving $m = 7$ reviews, and has a global acceptance rate of roughly 20 per cent. Creating the data, they assumed that there is a reference order amongst all proposals. Agents are acting according to that ranking with a certain error which is described by a dispersion parameter. Each agent ranks exactly m other agents from the other clusters and assigns Borda scores, that is m minus rank of that agent, to them. This model corresponds to weighted or multiple nominations and as each agent casts $\frac{m^2-m}{2}$ nominations, it is very regular. The experimental results for each mechanism are then compared to Vanilla and the Ground Truth, which is the unchanged reference order.

Aziz et al. [ALM+16] draw the following conclusions. First, as Credible Subset often returns no agents, it is difficult to compare. When the number of reviews goes up, the number of times that Credible Subset returns zero agents goes up as well. When it returns agents though, it performs about as well as Vanilla. EDP strictly outperforms the other dollar based mechanisms and thus we expect that also EDPP outperforms those. Further, EDP even outperforms partition, selecting on average between 0.5% and 5% more top agents. It also has between 3% and 25% lower standard deviation and selects at least as many top performing agents. In addition, increasing the number of selected agents k increases the advantage that EDP has over Partition, which may be attributed to the higher probability of having more than one top agent per partition. Compared to Vanilla, we expect a loss in performance of about 7% when using the impartial mechanism EDP. This is about the same difference as EDP shows to Partition, the next best impartial mechanism. Those results are, by Theorem 2.23, transferable to EDPP.

To conclude, we expect EDPP to select more often top agents, with better worst

case performance and lower variance than the so far best known impartial mechanisms, if given a broadly regular input graph. This is a reasonable assumption as many voting schemes are based on every voter giving a predetermined number of votes and may make EDPP favorable in practice.

2.4 Upper Bounds on Approximation Factors

We conclude this chapter by giving upper bounds on the performance of impartial k-selection mechanisms for any value of k, and for both deterministic mechanisms and randomized mechanisms with and without exactness.

2.4.1 Upper Bounds for Deterministic Mechanisms

By giving in the following a complete characterization of deterministic mechanisms for a total of three agents, we achieve an upper bound for this special case. The characterization is also interesting in its own right. Holzman and Moulin [HM13] proved that each deterministic impartial mechanism which selects exactly one agent out of a set of three agents is either constant, i. e. , it selects the same agent for each input graph, or it is such that the nominations of one fixed agent decide whom of the other two agents is selected. It is straightforward to derive a similar result for any deterministic impartial mechanism that selects $k \in \{0, \dots, 3\}$ agents out of a set of three agents.

Theorem 2.24 *For three agents, all deterministic impartial mechanisms which select exactly $k \in \{0, \dots, 3\}$ agents are either constant or of one of the following forms:*

1. *$k = 1$ and there is an agent i whose nominations decide whom of the other two agents is selected.*
2. *$k = 2$ and there is an agent i who is selected and whose nominations decide whom of the other two agents is selected as well.*

Proof. Clearly, every mechanism that selects zero agents and every mechanism that selects three agents is constant. The first non-trivial case, $k = 1$, is proven by Holzman and Moulin [HM13] in a more general setting. For $k = 2$, note that exactly one of the three agents is not selected. Reversing the roles of "being selected" and "not being selected" does not alter the definition of impartiality. Thus, the characterization of Holzman an Moulin [HM13] applies for this case which implies the result. □

Using this characterization, we proceed to derive the following upper bounds for impartial selection amongst three agents.

Theorem 2.25 *There is no deterministic 1-selection mechanism that is α-optimal on $\mathcal{G}_3$ for any $\alpha > 0$. Further, for any deterministic 2-selection mechanism which is α-optimal on $\mathcal{G}_3$, we have $\alpha \leq \frac{1}{2}$.*

Proof. Consider an arbitrary 1-selection mechanism. By using Theorem 2.24, we see that there is an agent i who is not selected, but who decides whom of the other agents

are selected. There are always instances where only agent i gets nominations and on these instances, zero of a possible positive number of nominations are selected.

Further, consider an arbitrary 2-selection mechanism. Again, by Theorem 2.24, we see that there is an agent i who is always selected and who decides which other agent is selected. As there are always instances where i does not get any nominations and the other agents get an equal number of nominations possibly larger than zero, at most $1/2$ of the total nominations in those cases are guaranteed to be achieved, namely by selecting only one of the remaining two agents. □

The proof technique used here can, in a certain sense, be generalized and lead to the following bounds. Most interestingly, they imply that the bidirectional permutation mechanism is the best deterministic mechanism for $k = 2$.

Theorem 2.26 *Consider a deterministic k-selection mechanism that is α-optimal on $\mathcal{G}_n$, where $k < n$. Then $\alpha \leq \frac{k-1}{k}$.*

Proof. We construct the following instance of the k-selection problem. We consider a graph $G = (V, E)$ with n vertices, where $k+1$ vertices are arranged in a directed cycle and the remaining vertices do not have any outgoing edges, that is

$$V = \{1, \dots, n\} \text{ and } E = \{(i, i+1) : i = 1, \dots, k\} \cup \{(k+1, 1)\}.$$

Denote by F the set of vertices selected from G by an arbitrary deterministic k-selection mechanism. Observe that there exists a vertex $i \in \{1, \dots, k+1\} \setminus F$ in the directed cycle which is not selected. Let

$$G' = (V, E \setminus (\{i\} \times V))$$

be the graph where i does not cast any nominations, which means that there are only k nominations in total on a directed path. Observe that by impartiality, the mechanism does not select i from G', as only i changed its nominations and was not selected earlier. The mechanism thus selects at most $k-1$ out of the k vertices with positive indegree in G' and, consequently, it cannot be more than $(k-1)/k$-optimal. □

Using the fact that $(2-1)/2 = 1/2$ and Theorem 2.6, we obtain the following corollary.

Corollary 2.27 *The bidirectional permutation mechanism is the best deterministic mechanism for $k = 2$.*

2.4.2 Upper Bounds for Randomized Mechanisms

The next result applies to randomized mechanisms without the requirement of exactness and shows that the mechanism of Figure 2.4 or Algorithm 2.6 for the case when $k = 2$ and $n = 3$ is best possible.

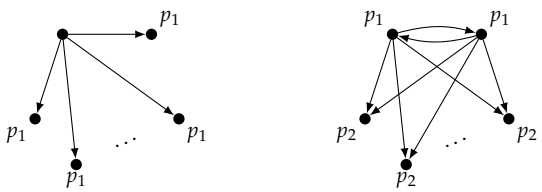


Figure 2.8: Impartial probability assignment for two graphs with n vertices

Theorem 2.28 *Consider a k-selection mechanism that is impartial and α-optimal on $\mathcal{G}_n$, where $k < n$. Then*

$$\alpha \leq \begin{cases} \frac{1}{2} & \text{if } k = 1 \\ \frac{3}{4} & \text{if } k = 2 \\ \frac{2k}{2k+1} & \text{if } 3 \leq k = n-1 \\ \frac{k+1}{k+2} & \text{otherwise.} \end{cases}$$

Proof. First assume that $k \leq n-2$, and consider for this case the two graphs on n vertices where $k+2$ of the vertices have edges as in Figure 2.8 on the left and the remaining vertices do not have any incoming or outgoing edges. It is without loss of generality to assume that randomized k-selection mechanisms are symmetric in the sense that they assign equal probabilities to vertices that are indistinguishable up to their index. Further, it is easily verified that any symmetric impartial mechanism must assign probabilities as shown in Figure 2.8. In the graph on the left, the mechanism chooses a set of vertices with expected overall indegree $(k+1)p_1$, while the maximum overall indegree for a set of k vertices is k, so

$$\alpha \leq \frac{(k+1)p_1}{k}.$$

In the graph on the right, the mechanism chooses a set of vertices with expected overall indegree $2p_1 + 2kp_2$, while the maximum overall indegree for a set of k vertices is $2k$. Thus

$$\alpha \leq \frac{2p_1 + 2kp_2}{2k} = \frac{p_1}{k} + p_2 \leq \frac{p_1}{k} + \left(1 - \frac{2p_1}{k}\right) = 1 - \frac{p_1}{k},$$

where the second inequality holds because

$$2p_1 + kp_2 \leq k$$

by the graph on the right. In summary,

$$\alpha \leq \min\left\{\frac{(k+1)p_1}{k}, 1 - \frac{p_1}{k}\right\} \leq \frac{k+1}{k+2},$$

where the second inequality holds because the minimum takes its maximum value when the two terms are equal.

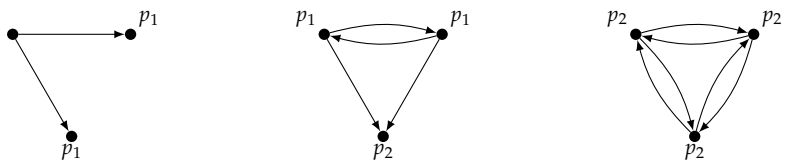


Figure 2.9: Impartial probability assignment for three graphs with $n = 3$

Now assume that $k = n - 1$ and consider the two graphs on n vertices and edges as shown in Figure 2.8. Any symmetric impartial mechanism must again assign probabilities as shown. In the graph on the left, the mechanism chooses a set of vertices with expected overall indegree kp_1, while the maximum overall indegree for a set of k vertices is k, so

$$\alpha \leq p_1.$$

In the graph on the right, the mechanism chooses a set of vertices with expected overall indegree $2p_1 + 2(k-1)p_2$, while the maximum overall indegree for a set of k vertices is $2(k-1)+1$. Thus

$$\alpha \leq \frac{2p_1 + 2(k-1)p_2}{2(k-1)+1} \leq \frac{2p_1 + 2(k-1)(\frac{k}{k-1} - \frac{2p_1}{k-1})}{2k-1} = \frac{2k-2p_1}{2k-1},$$

where the second inequality holds because

$$2p_1 + (k-1)p_2 \leq k$$

by the graph on the right. In summary,

$$\alpha \leq \min\left\{p_1, \frac{2k-2p_1}{2k-1}\right\} \leq \frac{2k}{2k+1},$$

where the second inequality holds because the minimum takes its maximum value when the two terms are equal.

In the special case where $k = 2$, an additional graph can be used to obtain a stronger bound. For this, consider situations where three vertices have outgoing edges as in Figure 2.9 and the remaining $n-3$ vertices do not have any outgoing edges. Note that the first two graphs are the same as those in Figure 2.8 when $k = 2$. It is again easily verified that any impartial mechanism must assign probabilities as shown. Thus

$$\alpha \leq \min\left\{\frac{2p_1}{2}, \frac{6p_2}{4}\right\} \leq \{p_1, 3-3p_1\} \leq \frac{3}{4},$$

where the first inequality holds by the first and third graph, the second inequality because

$$2p_1 + p_2 \leq 2$$

by the second graph, and the third inequality because the minimum takes its maximum value when the two terms are equal.

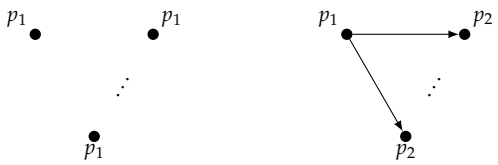


Figure 2.10: Impartial probability assignment for two graphs with n vertices

The bound for $k = 1$ is easily obtained by considering the special case of the graphs in Figure 2.8 where two vertices have outgoing edges as shown and the others do not have any outgoing edges. Then

$$\alpha \leq p_1 \leq \frac{1}{2},$$

where the inequalities hold respectively by the first and second graph. □

2.4.3 Upper Bounds for Exact Randomized Mechanisms

Our last result concerns randomized mechanisms which are exact. Together with the mechanism of Figure 2.4, it shows a strict separation between randomized mechanisms with and without exactness. Our result does not preclude improvements over the 2-partition mechanism with permutation when $n > 3$. A comparison with Theorem 2.28 further suggests that the influence of the exactness constraint may be limited to cases where almost all vertices are selected.

Theorem 2.29 *Consider a k-selection mechanism that is exact, impartial, and α-optimal on $\mathcal{G}_n$, where $k < n$. Then*

$$\alpha \leq \begin{cases} \frac{1}{2} & \text{if } k = 1 \\ \frac{k}{k+1} & \text{if } 2 \leq k = n-1 \\ \frac{5}{7} & \text{if } 2 = k = n-2 \\ \frac{7k^3+5k^2-6k+12}{7k^3+13k^2-2k} & \text{if } 3 \leq k = n-2 \\ \frac{k+1}{k+2} & \text{otherwise.} \end{cases}$$

Proof. First assume $2 \leq k = n-1$, and consider the two graphs with $n = k+1$ vertices shown in Figure 2.10. By impartiality, the probability of selecting the vertex at the top left must be equal for both graphs. Any symmetric mechanism assigns equal probabilities to all vertices in the left graph and equal probabilities to all vertices with indegree one in the right graph. Denoting the former probability by p_1 and the latter by p_2, exactness implies that

$$(k+1)p_1 = p_1 + kp_2 = k,$$

so

$$p_1 = \frac{k}{k+1} \text{ and } p_2 = \frac{k - k/(k+1)}{k} = \frac{k}{k+1},$$

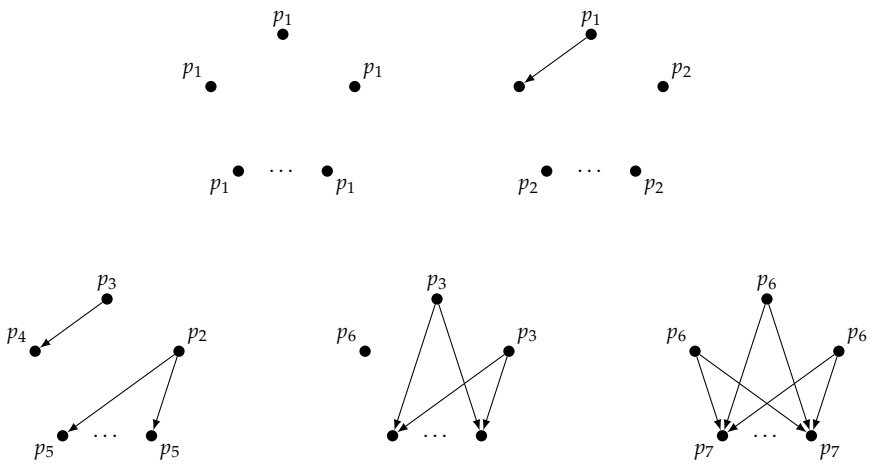


Figure 2.11: Impartial probability assignment for five graphs with n vertices

and thus

$$\alpha \leq \frac{kp_2}{k} = \frac{k}{k+1}.$$

Now assume that $3 \leq k = n-2$, and consider the five graphs with five vertices shown in Figure 2.11. Using similar arguments as above it is easily established that any symmetric impartial mechanism must assign probabilities as shown. By the first and second graph,

$$p_1 = k/(k+2) \text{ and } p_1 + kp_2 \geq k-1,$$

and thus

$$p_2 \geq \frac{k-1}{k} - \frac{p_1}{k} = \frac{k-1}{k} - \frac{1}{k+2} = \frac{(k-1)(k+2)-k}{k(k+2)}. \tag{2.6}$$

By the third graph,

$$p_2 + p_3 + p_4 + (k-1)p_5 = k$$

and thus

$$\begin{aligned} \frac{1}{k}p_4 + \frac{k-1}{k}p_5 &= 1 - \frac{1}{k}p_2 - \frac{1}{k}p_3 \\ &\leq 1 - \frac{(k-1)(k+2)-k}{k^2(k+2)} - \frac{1}{k}p_3 = \frac{k^3+k^2+2}{k^2(k+2)} - \frac{1}{k}p_3 \end{aligned} \tag{2.7}$$

where the inequality holds by equation (2.6). By the fourth and fifth graph,

$$2p_3 + p_6 \geq 1 \text{ and } 3p_6 + (k-1)p_7 = k,$$

and thus

$$p_7 = \frac{k}{k-1} - \frac{3}{k-1}p_6 \leq \frac{k}{k-1} - \frac{3}{k-1}(1-2p_3) = \frac{1}{k-1}(6p_3 + k - 3). \tag{2.8}$$

Finally, by the third and fifth graph,

$$\begin{aligned}\alpha &\leq \min\left\{\frac{1}{k}p_4 + \frac{k-1}{k}p_5, p_7\right\} \\ &\leq \min\left\{\frac{k^3+k^2+2}{k^2(k+2)} - \frac{1}{k}p_3, \frac{1}{k-1}(6p_3+k-3)\right\},\end{aligned} \tag{2.9}$$

where the second inequality holds by (2.7) and (2.8). The minimum takes its maximum value when the two terms are equal, i. e. when

$$\begin{aligned}\frac{k^3+k^2+2}{k^2(k+2)} - \frac{1}{k}p_3 &= \frac{1}{k-1}(6p_3+k-3), \\ \frac{6}{k-1}p_3 + \frac{1}{k}p_3 &= \frac{k^3+k^2+2}{k^2(k+2)} - \frac{k-3}{k-1}, \\ \frac{7k-1}{k(k-1)}p_3 &= \frac{(k^3+k^2+2)(k-1) - k^2(k+2)(k-3)}{k^2(k+2)(k-1)}.\end{aligned}$$

Thus

$$\begin{aligned}p_3 &= \frac{(k^3+k^2+2)(k-1) - k^2(k+2)(k-3)}{k(k+2)(7k-1)} \\ &= \frac{k^4+k^3+2k-k^3-k^2-2-k^4+k^3+6k^2}{k(k+2)(7k-1)} \\ &= \frac{k^3+5k^2+2k-2}{k(k+2)(7k-1)},\end{aligned}$$

and by plugging this into (2.9),

$$\begin{aligned}\alpha &\leq \frac{k^3+k^2+2}{k^2(k+2)} - \frac{k^3+5k^2+2k-2}{k^2(k+2)(7k-1)} \\ &= \frac{7k^4+7k^3+14k-k^3-k^2-2-k^3-5k^2-2k+2}{k^2(k+2)(7k-1)} \\ &= \frac{7k^3+5k^2-6k+12}{k(k+2)(7k-1)}.\end{aligned}$$

In the special case where $2 = k = n-2$, only a single vertex is chosen with probability p_5 in the third graph of Figure 2.11, so by symmetry $p_2 = p_3$ and $p_4 = p_5$. Thus

$$p_7 = 2 - 3p_6 \leq 2 - 3(1-2p_3) = 6p_3 - 1 = (6-6p_4) - 1 = 5 - 6p_4,$$

and

$$\alpha \leq \min\{p_4, p_7\} \leq \min\{p_4, 5-6p_4\} \leq \frac{5}{7},$$

where the last inequality holds because the minimum again takes its maximum value when the two terms are equal.

The bounds for the remaining two cases, of $1/2$ if $k = 1$ and for all not yet discussed cases of $(k+1)/(k+2)$, follow directly from Theorem 2.28. □

2.5 Discussion and Open Problems

This chapter was dedicated to the introduction, description and subsequent analysis of mechanisms designed to fairly select representatives of a group. By doing so, we propose a framework to reduce the total number of actors on streets as, e. g. , impartial mechanisms facilitate collaboration between companies. It also serves as a tool to settle disagreements when the number of actors wanting to use, e. g. , a certain pathway is higher than its capacity.

The mechanisms we designed are impartial in the sense that the probability of an agent to be selected does not depend on her outgoing nominations. With the exception of mechanisms that are asymptotically optimal, when many agents are selected (see [AFPT11]) or when agents receive many nominations (see [BNV14]), only very little was previously known about the impartial selection of more than one agent. Our understanding of 2-selection is now much better, with some room for improvement in the case of randomized mechanisms. We provide the bidirectional permutation mechanism to select up to two agents, which is $1/2$-optimal in the deterministic case and $2/3$-optimal in the randomized case, as well as the randomized 2-partition mechanism with permutation to select exactly two agents, which is $7/12$-optimal. The analysis for these mechanisms is tight. As no deterministic exact mechanism can obtain any positive optimization factor, this illustrates that the relaxation of exactness can benefit both deterministic and randomized mechanisms, and that randomization can be beneficial independently of exactness.

About k-selection for $k > 2$, in particular about deterministic mechanisms for this task, we still know relatively little. This lack of understanding is witnessed by the fact that the optimal deterministic mechanism selecting up to two agents, one for each direction of a permutation, does not generalize in any obvious way to the selection of more than two agents. Nevertheless, we propose a variant of the randomized bidirectional permutation mechanism to select up to k agents for which we can give a lower bound on the number of nominations of the selected agents which is dependent on k and Δ_k. More importantly, we propose a generalized variant of the partition mechanism with permutation which is $\frac{k}{k+1}\left(1 - \left(\frac{k-1}{k}\right)^{k+1}\right)$-optimal and selects exactly k agents. Our analysis is tight. We further got insights into mechanisms for mostly regular nomination graphs by introducing the Exact Dollar Partition with Permutation mechanism, which also provides a simple subroutine to solve the problem of apportionment.

Looking at the upper bounds which we established, it is apparent that while substantial progress has been made, there is still space for improvement. We may hope for stronger techniques to bound the power of randomized mechanism that are universally impartial in the sense that they can be obtained as a convex combination of deterministic impartial mechanisms, and to design mechanisms that are impartial but not universally impartial. We saw in Section 2.2.2 by looking at optimal mechanisms to select two out of three agents that optimal mechanisms in the latter category may exhibit rather unintuitive nonmonotonicity properties, which in turn complicates their analysis. Meanwhile, the existence of a near-optimal mechanism in the limit of many selected agents suggests that the upper rather than the lower bounds may be correct. However, it is not clear whether the relaxation of exact-

ness and usage of randomization independently of exactness can be beneficial for all values of n and k.

Further lines of research might focus on new techniques to improve the k-partition mechanism and decide how many agents shall be selected from each partition. Also further generalizations of the bidirectional permutation mechanism could lead to new insights. Another technique to consider in order to gain insights to find new impartial mechanisms can be extracted from Figure 2.5 and Algorithm 2.6, where structural results for the selection probabilities for each agent, regardless of their outgoing nominations and only considering the structure of the remaining graph (possibly only of agents which nominate the agent for which we are currently calculating the selection probability), are given. An additional direction to investigate is as follows. Only those agents M_0, who receive a maximum number of nominations m, and those agents M_1, who receive $m-1$ nominations and nominate every agent in M, and iteratively all agents who receive $m-z$ nominations and nominate every agent in $\bigcup_{i=1,\dots,z-1} M_i$ get assigned positive probability to be selected. Here, no agent can change with their nominations whether they are part of this set or not. However, it is not evident how the probability to be selected amongst those agents shall be distributed while also maintaining impartiality.

Chapter 3

Online Planning for Carpools

We consider the online DIAL-A-RIDE problem on the line. Requests appear online over time on the real line or in Euclidean space and need to be transported by a server to a specified destination. The server is originally located at the origin.
Both the uncapacitated and the capacitated version, where only a certain number $c \in \mathbb{N}$ of requests can be transported at the same time, as well as the preemptive, where dropping a request if while serving it and finishing it later, and the non-preemptive variant are considered.
We introduce an preemptive algorithm with improved competitive ratio for the capacitated setting which can also be adapted to work in Euclidean space, if no capacity restrictions are given. We improve the lower bound for non-preemptive online DIAL-A-RIDE on the line and generalize known and give new complexity results for the underlying offline problem.

Bibliographic Information: The results in this chapter are based on joint work with Yann Disser, Jan Hackfeld, Christoph Hansknecht, Marten Lipmann, Julie Meißner, Kevin Schewior, Miriam Schlöter and Leen Stougie which have been published in the Proceedings of the Symposium on Discrete Algorithms (SODA) 2017 [BDH+17].

In the 1970s, carpooling became prominent in the USA, partly due to raised gas and energy prices. This enthusiasm, however, did not last. After a steady decline during the 1980 and 1990s, we have recently seen a renewed hope and growth in carpooling. This is attributed to the positive effect of widely spread internet and mobile phones, which enable users to easily search for riders or passengers. The positive effects of carpooling, both personal such as reduction of stress and cost, as well as the environmental and societal effects, such as reduction of greenhouse gas emissions and higher sustainability, are undeniable. An increase in the use of carpools is thus an important step for nowadays societies. For many travellers, easy access, time sensitive travel and especially flexibility are key. In this chapter, we take those needs into account and work towards designing carpooling solutions that help reducing congestion.

First, we give an introduction to the problem in Section 3.1 and explain how researching it can help to reduce congestion in networks. We continue with an overview of related work and known results, then we clarify the notation and give a formal problem definition. In Section 3.2, we provide a competitive online algorithm for the preemptive open online DIAL-A-RIDE on the line and analyze it. The algorithm as well as the analysis generalizes with the same competitive ratio to the Euclidean space. We also give a lower bound on the competitive ratio for the non-

preemptive closed case. Further, in Section 3.3 we consider the offline version of the problem and provide some lower bounds for approximability of DIAL-A-RIDE and give an upper bound by way of a simple approximation algorithm. In addition, in Section 3.4 we take a look at the closely related problem TSP on the line, the traveling salesperson problem, which can be interpreted as DIAL-A-RIDE with High Five requests, that is requests which only have to be visited, but not transported. We achieve lower bound results for this which also hold for the more general DIAL-A-RIDE. To sum it up, we discuss our results in Section 3.5.

3.1 Background

Delivery problems and vehicle routing are widely studied problems in Operations Research and also Computer Science. This type of problems may occur in a large number of different settings, e. g. , robotics and the transportation of goods and passengers.

In order to reduce congestion in networks, one viable approach is to reduce the number of agents in that network, for example the number of cars on the streets. If we want, at the same time, to keep transporting the same amount of people, we might have to reassign people to different vehicles. This is also known as carpooling or ride sharing. Obviously, picking up, transporting and dropping off a larger amount of people may result on more complicated driving patterns for the remaining vehicles. In order to reduce the strain on the environment, we should thus find smart ways for driving those vehicles.

As an example, consider workers in Silicon Valley, many of whom living in San Francisco and commuting to work in Mountain View. Organizing a carpool, they need to request a ride to Mountain View, be picked up from home and as the last step transported down to Mountain View. The picking up may also happen from designated pick-up places with simple structure, for instance on Market Street, which is central, a straight line through the center, and easy to reach. As the trip out of San Francisco to Mountain View can be assumed to always start from the same place, the question is how to smartly organize the pick up. Doing that, we have to keep in mind that people give their requests in an online fashion, meaning they are not always known by the start of the day due to changes in daily routines. The total time for picking up all workers should be as short as possible, so that the final trip to Mountain View, the longest leg of the daily commute, can be started as early as possible.

In this scenario, it thus makes sense to reduce the makespan, i. e. the total time a vehicle is driving as this correlates with the smallest environmental impact. At the same time, people wanting to take a vehicle to be transported to different places might not be known from the start, but might appear over time which largely complicates matters. In positive contrast, we may assume this problem to happen in a metric environment. There might be special cases where every passenger has to go to the same place such as their working place, and thus the main problem is just to find a smart way to pick everyone up, or people have to be transported to different places. Or one might want to look at the offline setting, and all people tell us their

pick-up times before as they happen at the same time each day, because daily routines do not change as much. Taking all these requirements into consideration, we can formulate the following mathematical problem.

3.1.1 Formulating the Mathematical Problem

In the online dial-a-ride problem (referred to as DIAL-A-RIDE) on the line, we consider a server initially located at the origin of the real line that has to serve requests that appear over time. The server has unit speed and serves requests (in any order) by moving to the position of the corresponding request at some time after its release. Each request specifies a source and destination that need to be visited by the server in this order. If the capacity of the server is finite, it limits the number of requests that can be transported simultaneously. Our objective in online DIAL-A-RIDE on the line is to minimize the makespan, i. e. , the time until all requests have been served.

As we see, the requests arrive over time and an online algorithms has to decide about the movement of the server while only knowing the requests that are available already. In particular, the number of requests n can be arbitrary and is unknown in advance. A standard performance measure for an online algorithm in this setting is the competitive ratio, i. e. the best bound on the ratio of the makespan of the online algorithm to the makespan of the optimal offline solution that holds for any instance.

In the *closed* variant of the problem, the server needs to return to the origin after serving all requests, while the *open* variant has no such requirement. Moreover, we distinguish servers with a bounded capacity $c \geq 1$, i. e. a server that can only transport at most c objects at the same time, and servers with unbounded capacity. For bounded capacity, we distinguish between the *preemptive* version, where a server can drop unfinished requests off at any time at the server's current position and pick them up again later to finish the request, and the *non-preemptive* version, where a request that has been started to be served has to be finished before the server can pick up new requests.

The online DIAL-A-RIDE problem on the line arises, e. g. , when controlling industrial or personal elevators, vertical or horizontal delivery systems or robotic arms for depositing material.

A special case of online DIAL-A-RIDE is the online Traveling Salesperson Problem (referred to as TSP), where source and destination of each request are equal. The main application we see is for collecting people from one area to later go to the same working place. Other examples include the collection of objects from mass storage shelves or robotic arms with restricted degrees of freedom for welding and screwing.

In order to get a better grasp on this problem, it is helpful to clarify differences to related problems. Online DIAL-A-RIDE is a natural online problem similar to the classical k-server problem (see [MMS90]). In the latter, source and destination are equal and the order in which requests need to be served is prescribed, and the problem thus becomes trivial on the line for $k = 1$ server. In contrast, online DIAL-A-RIDE on the line is a non-trivial problem that arises in 1-dimensional collection and delivery problems.

Related are also the online DIAL-A-RIDE and online TSP where the objectives are different. For example, instead of minimizing the makespan, i. e. the time until all requests are served, one can minimize the average flow time, i. e. the average time after each request is served. Further, one could minimize the maximum flow time, i. e. the maximum time after a request is served or minimize the weighted sum of completion times. In contrast, we believe that minimizing the makespan is a good objective for measuring the system efficiency.

While both online DIAL-A-RIDE and online TSP on the line are among the most natural online problems and have been studied extensively over the last two decades [AKR00, AFL+94, AFL+95, AFL+01, BKdPS01, FS01, JW08, Kru01, KdPPS03, KLL+02], no satisfactory (tight) analysis was known for either problem in terms of competitive ratios. We narrow the gaps for online DIAL-A-RIDE on the line by giving improved bounds. In addition to online results, we study the computational complexity of the underlying offline problems. Though we will call them the offline DIAL-A-RIDE, we actually mean the version with release times.

For ease of reference, we summarize notation and problem definition in the following paragraph.

Notation and Formal Problem Definition We consider a server that moves along the real line with (at most) unit speed. We let p_t denote the position of the server at time $t \geq 0$ and assume (without loss of generality) that $p_0 = 0$. With this notation, the speed limitation of the server can equivalently be expressed as $|p_t - p_{t'}| \leq |t - t'|$ for all $t, t' \geq 0$. A series of requests $\sigma_1, \ldots, \sigma_n$ arrives over time with $\sigma_i = (a_i, b_i; t_i)$, where $t_i \geq 0$ denotes the release time of the request and $a_i, b_i \in \mathbb{R}$ denote its source and target position, respectively. For TSP, we have $a_i = b_i$ and write $\sigma_i = (a_i; t_i)$. If not stated otherwise, we assume $t_1 \leq t_2 \leq \ldots \leq t_n$. Moreover, we assume without loss of generality that $t_i \geq |a_i|$ holds, as the server can not reach σ_i before time $|a_i|$ and it only helps the algorithm to know a request earlier. We further use the notation $p^{\mathrm{U}} := \max_{i=1,\ldots,n}\{a_i, b_i, 0\}$ to denote the upmost point that needs to be visited by the server, and similarly $p^{\mathrm{D}} := \min_{i=1,\ldots,n}\{a_i, b_i, 0\}$ for the downmost point. Here and throughout we refer to the negative direction of the real line as *down* and the positive direction as *up*.

In both DIAL-A-RIDE and TSP on the line, all requests need to be *served*. For DIAL-A-RIDE, the server may collect request σ_i at time $t \geq t_i$ if $p_t = a_i$. In the *preemptive* DIAL-A-RIDE problem, the server can drop off any request it is carrying at its current location at any time. If request σ_i is dropped off at point p at time t, we consider it to be modified to the new request $(p, b_i; t)$. In the *non-preemptive* DIAL-A-RIDE problem, the server may only drop off a request at its target location. We consider a request served if it is ever dropped off at its target location. For TSP, we consider a request served if $p_t = a_i$ for some time $t \geq t_i$.

If the server has finite capacity $c \geq 1$, it can carry at most c requests at any time. We assume that no time is needed for picking up and dropping off requests, so that the server can pick up and drop off any number of requests at the same time, as long as its capacity is not exceeded.

We refer to a valid trajectory of the server together with the description of when

it picks up and drops requests as a tour T. If the tour ends at $p_0 = 0$, we call it *closed*, otherwise it is *open*. We denote the makespan, which is the time when all requests are served, of the tour T by $|T|$. The objective in the open (closed) version of both DIAL-A-RIDE and TSP is to find an open (closed) tour T that serves all requests and minimizes $|T|$.

In the *offline* setting, we assume all requests to be known from the start. We let T^{OPT} denote an optimal offline tour. In the *online* setting, we assume that request σ_i is revealed at its release time t_i, at which point the tour of the server until time t_i must already have been fixed irrevocably. We measure the quality of an online algorithm via its competitive ratio, i. e. , the maximum over all sequences of requests of the ratio between the makespan of the tour it produces and the makespan of the optimal tour $|T^{\mathrm{OPT}}|$. It is worth mentioning that if the total number of requests were known from the beginning, the server could just wait in the origin until the last request arrives and then start to follow an optimal tour. As an optimal tour is at least as long as the time which passes until the last request appears, this approach would result in a tour at most twice as long as any optimal offline tour. We thus assume the total number of requests n to be unknown.

In order to describe the trajectory of the server, we use the notation "move(a)" for the tour that moves the server from its current position to the point $a \in \mathbb{R}$ with unit speed and the notation "waituntil(s)" for the tour that keeps the server stationary until time s. We use the operator $\oplus$ to concatenate tours. For example, if T_0 is a tour of the server that ends at time t_0 at position p_{t_0}, then $T_0 \oplus \mathrm{move}(a)$ describes the tour that ends at time $t_0 + |a - p_{t_0}|$, is identical to the tour T_0 until time t_0 and in addition satisfies $p_t = p_{t_0} + (a - p_{t_0})(t - t_0)$ for $t_0 \leq t \leq t_0 + |a - p_{t_0}|$. Similarly, the tour $T_0 \oplus \mathrm{waituntil}(s)$ ends at time $\max\{t_0, s\}$, is identical to the tour T_0 until time t_0 and that satisfies $p_{t_0} = p_t$ for all $s \in [t_0, s]$. For TSP on the line, we do not explicitly specify when a request is served, but we assume that the server serves a request whenever possible, i. e. , whenever the server passes the location of a request that is already released and not yet served.

Note that any instance on the real line can be approximated with arbitrary precision (e. g. , in terms of makespan, competitive ratio) by an instance with a finite integral number of points. This can be done by scaling and rounding the requests to make them integral and assuming that the server never moves above p^{U} or below p^{D} and thus not outside of so called *extreme requests*, meaning those who have start or destination equal to p^{U} or p^{D}.

3.1.2 Related Work and Previous Results

There is a multitude of variants of the problem. In the general single server DIAL-A-RIDE problem, a single server moves in a metric space with at most unit speed and has to transport objects from different sources to different destinations. Further, the requests for the objects to be transported can have release dates or deadlines, the server may be allowed to drop objects at intermediate locations (preemption) and the server can have a limited capacity, i. e. a bound on the number of objects that can be transported at the same time. Moreover, one can require the server to return to the starting position after serving all requests (closed variant) or make no such

requirement (open variant).

For the the closed online DIAL-A-RIDE problem without preemption, Feuerstein and Stougie [FS01] show a lower bound of two for the competitive ratio in metric spaces, and present an algorithm with a best-possible competitive ratio of two for the case that the server has infinite capacity. Ascheuer et al. [AKR00] analyze different algorithms for the same setting and present a 2-competitive algorithm for any finite capacity $c \geq 1$. Krumke [Kru01] gives an upper bound of 3.41 for the open non-preemptive variant.

For minimizing the sum of completion times instead of the makespan, Feuerstein and Stougie [FS01] further show a lower bound of three for a server with unit capacity and a lower bound of 2.41 independent of the capacity. Moreover, they provide a 15-competitive algorithm for the real line and unlimited capacity. For the same objective function, Krumke et al. [KdPPS03] present an algorithm with a competitive ratio of 5.83 for a server with unit capacity in an arbitrary metric space.

A single server dial-a-ride problem with coinciding source and destination and makespan objective is the (metric) *traveling salesperson problem*. For the online TSP problem in general metric spaces, Ausiello et al. [AFL+01] show a lower bound of two on the competitive ratio for the open version and a 1.64 lower bound for the closed version, both bounds being achieved on the real line. For the open online TSP, they present a 2.5-competitive algorithm, and for the closed version they give a 2-competitive algorithm. Jaillet and Wagner [JW08] give 2-competitive algorithms for the closed version that can additionally deal with precedence constraints or multiple servers. Blom et al. [BKdPS01] consider the closed online TSP problem on the non-negative part of the real line and present a best possible algorithm with competitive ratio 1.5. They also study a "fair" setting where the optimum does not travel outside the convex hull of the known requests, and they derive an algorithm for the real half-line with a better competitive ratio of 1.28 for this setting. Krumke et al. [KLL+02] show that there cannot be a competitive algorithm for open online TSP with the objective of minimizing the maximum flow time instead of minimizing the makespan. For the real line they define a fair setting and give a competitive algorithm for this setting.

Ausiello et al. [AFL+01] give a competitive ratio of $(9 + \sqrt{17})/8 \approx 1.64$ for closed variant of online TSP on the line and provide a 1.75-competitive algorithm. Further, for the open variant of online TSP on the line, they [AFL+01] provide a lower bound on the competitive ratio of two and give a 2.33-competitive algorithm.

If we replace the objective function in the traveling salesperson problem with the weighted sum of completion times, then the problem is known as the *traveling repairperson problem*. Feuerstein and Stougie [FS01] show a lower bound of 5.83 on the best-possible competitive ratio for this problem and provide a 9-competitive algorithm for the real line. Krumke et al. [KdPPS03] give a best-possible online algorithm with competitive ratio 5.83 for general metric spaces.

The offline version of the TSP and DIAL-A-RIDE problem is a well-studied NP-hard problem (e. g. , see [LLRKS85]).

For DIAL-A-RIDE with capacity $c = 2$ without preemption, Guan [Gua98] proves NP-hardness by a reduction to a special variant of the 3SAT problem. Interestingly

this proof does not employ any release dates. In the reduction this satisfiability problem is transformed to an instance of the closed offline DIAL-A-RIDE problem on the line with capacity $c = 2$. We derive that the open offline DIAL-A-RIDE problem on the line with capacity $c = 2$ is also NP-hard.

Proposition 3.1 *The open* DIAL-A-RIDE *problem on the line with capacity* 2 *is NP-hard.*

Proof. Guan [Gua98] shows in his reduction to a variant 3SAT that the clause set is satisfiable if and only if there is a closed schedule to the DIAL-A-RIDE problem on the line, in which the server is loaded to full capacity at any time and each request is transported on the direct path from their source to their destination. This tour is a lower bound on the length of any feasible DIAL-A-RIDE schedule independent of its final destination. Hence, the clause set is satisfiable if and only if there is an open schedule to the DIAL-A-RIDE problem, which the server is loaded to full capacity at any time and each request is transported on the direct path from their source to their destination. □

As this is a special case of metric DIAL-A-RIDE, this of course generalizes to the open offline DIAL-A-RIDE problem in metric spaces.

In [KLL+02] Krumke presents a 3-approximation algorithm for the DIAL-A-RIDE problem on the line with capacity $c > 1$, without release dates, and with the starting point on one of the end points of the line. The offline DIAL-A-RIDE problem on the line with unbounded capacity is open. It is known that the problem is easy without release dates [dPLS+04]. It becomes hard if each request comes with a deadline in addition to its release time [Tsi92].

There are many offline variants of the DIAL-A-RIDE problem, differing in capacities, the underlying metric space, release times and deadlines, open versus closed tours, and in whether preemption is allowed (e. g. , see [dPLS+04]). The special case without release times and unit capacity is known as the *stacker crane problem*. Attalah and Kosaraju [AK88] present a polynomial algorithm for the closed, non-preemptive stacker crane problem on the real line. Frederickson and Guan [FG93] show that this problem is NP-complete on trees.

Guan [Gua98] shows that the DIAL-A-RIDE problem without release times remains easy on the line with capacities larger than one if preemption is allowed, and that it remains hard on trees. Finally, Charikar and Raghavachari [CR98] give a $\mathcal{O}(\sqrt{c}\log n \log\log n)$-approximation for the closed DIAL-A-RIDE problem in metric spaces with n points and without preemption. In the same paper they claim a 2-approximation for the problem on the line, however this result seems to be incorrect (personal communication).

Afrati et al. [ACP+86] show that the offline traveling repairperson problem is NP-hard in general, but can be solved in time $\mathcal{O}(n^2)$ for the real line and unweighted sum of completion times objective.

Further, the version where more than one server is used to serve requests has been studied. Optimizing the makespan, Gørtz, Nagarajan and Ravi [GNR15] present an $\mathcal{O}(log^3 n)$-approximation algorithm for preemptive multi-vehilce DIAL-A-RIDE and an $\mathcal{O}(logt)$-approximation for the case without capacity constraints. Here, t is the number of distinct servers.

3.1.3 An Overview of Our Results

In the following, we outline our results and put them in context of former work.

We provide a simple preemptive 2.41-competitive algorithm for the online DIAL-A-RIDE problem on the line and in Euclidean space, which improves a (non-preemptive) $2+\sqrt{2} \approx 3.41$-competitive algorithm by Krumke [Kru01]. For the closed DIAL-A-RIDE variant, the lower bound of 1.64 by Ausiello et al. [AFL+01] was improved for one server with unit capacity without preemption to 1.71 by Ascheuer et al. [AKR00]. We improve this bound further to 1.75 for any finite capacity $c \geq 1$. In addition, we give lower bounds for preemptive and non-preemptive online DIAL-A-RIDE on the line, in particular a lower bound of 2.04 which is the first bound greater than two for the open variant of the problem.

Regarding offline TSP on the line with release times, Psaraftis et al. [PSMK90] showed a dynamic program that solves the open variant in quadratic time. We refute their claim that all optimal closed tours have a very simple structure with a counterexample which generalizes to DIAL-A-RIDE, and we adapt their algorithm to find an optimal closed tour in quadratic time. For the non-preemptive offline DIAL-A-RIDE problem on the line, results have previously been obtained for the closed variant without release times. For capacity $c = 1$, Gilmore and Gomory [GG64] and Atallah and Kosaraju [AK88] gave polynomial time algorithms, and Guan [Gua98] proved hardness for the case $c = 2$. We show that both the open and closed variant of the problem are NP-hard for *any* capacity $c \geq 2$, even without release times. Additionally, we show that the case with release times and any restricted capacity $c \geq 1$ is NP-hard.

3.2 Algorithms for Online Dial-A-Ride

In this section we give a $(1+\sqrt{2})$-competitive algorithm for preemptive online DIAL-A-RIDE on the line. The algorithm works in both the capacitated and the uncapacitated case and also in both closed and open version. The analysis of the algorithm can also be adapted to work for the uncapacitated case not only on the line, but in a metric space with Euclidean distance. To complement this upper bound, we provide a lower bound of 1.75 on any competitive ratio for the non-preemptive closed DIAL-A-RIDE on the line.

3.2.1 Online Dial-A-Ride on the Line

Let T_S^{OPT} denote the optimal tour over a set of requests S, where the server starts at position $p_0 = 0$ in the origin, and let R_t denote the set of released but not yet delivered requests at a time t.

Our algorithm works as follows: The server stays at position p_0 until the first request arrives. With every new arriving request, the server stops its current tour and returns to p_0, or stays at p_0 if it is already at the origin. The server starts following the tour $T_{R_t}^{\text{OPT}}$ at time $\sqrt{2} \cdot |T_{R_t}^{\text{OPT}}|$. By "unload" we denote the operation of unloading

Algorithm 3.1: Online Algorithm for the Preemptive DIAL-A-RIDE Problem with Fixed Capacity $c \geq 0$

This function is called upon receiving a new request
Input: New request σ, current position p_t, unserved requests R_t
Output: Tour starting at p_t and serving all requests in R_t
return unload $\oplus$ move(p_0) ▷ drop requests off and move to origin
$\oplus$ waituntil$(\sqrt{2} \cdot |T^{\text{OPT}}_{R_t}|) \oplus T^{\text{OPT}}_{R_t}$ ▷ serve unserved requests

all the requests which the server is currently carrying at the current position. That means, each such request $(a, b; t')$ is changed to $(p_t, b; t')$. A formal description of the algorithm can be found in Algorithm 3.1.

In order to prove that the algorithm returns the desired result, we most importantly confirm that the server can return to the origin in time whenever a new request is released. To show that, we first need a few useful properties which are summarized in the following lemma.

Lemma 3.2 *Without loss of generality, the following holds:*

1. *If T^{OPT} visits position $p \neq p_0$, then there is a request with source or destination q such that*
$$\frac{q - p_0}{p - p_0} \geq 1.$$
2. *Let R_i^0 be the requests in R_i with their original sources and destinations (before being moved). Then,*
$$|T^{\text{OPT}}_{R_i}| \leq |T^{\text{OPT}}_{R_i^0}|. \tag{3.1}$$

Proof. Let p^{D} and p^{U} be the minimum and maximum values, respectively, among all source and destinations of requests and p_0. We can replace all subtours of T^{OPT} beyond $[p^{\text{D}}, p^{\text{U}}]$ by waiting at p^{D} or p^{U}, without increasing the overall length of the tour as no requests will be picked up or dropped off during such a subtour. Hence, the first property holds.

For the second property, it is sufficient to show that our algorithm never moves requests away from their destination, since then $T^{\text{OPT}}_{R_i^0}$ is a valid tour also for R_i, and hence inequality (3.1) holds. To see this, consider T^{OPT}_S for an arbitrary set of requests S. Consider the path that a request $\sigma = (a, b; t) \in S$ is taking in T^{OPT}_S and replace all subpaths that increase σ's distance to b by unloading the request σ and leaving it at its current position, and having it picked up again at the next pass of the server which moves in direction of σ's destination. This does not increase the length of the tour, and we obtain that, without loss of generality as this argument holds for every request in S, we can assume that T^{OPT}_S never moves requests away from their destination. Since our algorithm consists of repeated application of tours of the form $T^{\text{OPT}}_{R_i}$, the same holds for our algorithm. □

In the following, we denote the i-th request by $\sigma_i = (a_i, b_i; t_i)$ and slightly abuse notation to let $R_i := R_{t_i}$.

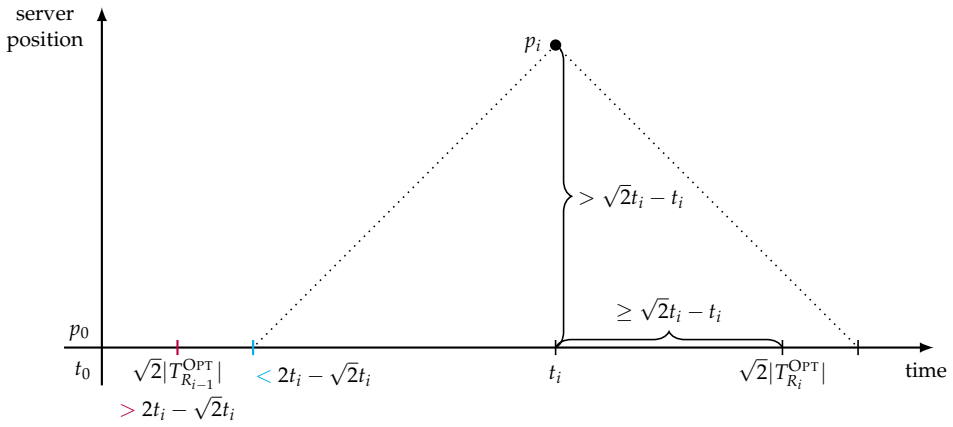


Figure 3.1: Sketch of the contradiction for the proof of Theorem 3.3

Theorem 3.3 *Algorithm 3.1 is* $(1+\sqrt{2}) \approx 2.41$*-competitive for the preemptive open and closed online* DIAL-A-RIDE *problem on the line with capacity* $c \geq 1$.

Proof. Let $|T^{\text{OPT}}|$ be the length of an optimal tour and $|T^{\text{ALG}}|$ be the length of the tour produced by our Algorithm 3.1. Assume that the server can always return to p_0 before time $\sqrt{2} \cdot |T^{\text{OPT}}_{R_i}|$ upon receiving a new request σ_i. Then, after the last request σ_n is received, the server stays in p_0 until time $\sqrt{2} \cdot |T^{\text{OPT}}_{R_n}|$ and then finishes all requests within time $|T^{\text{OPT}}_{R_n}|$. Thus we have in total

$$|T^{\text{ALG}}| \leq \sqrt{2} \cdot |T^{\text{OPT}}_{R_n}| + |T^{\text{OPT}}_{R_n}|.$$

Using the second property of Lemma 3.2, we have

$$|T^{\text{OPT}}_{R_n}| \leq |T^{\text{OPT}}_{R^0_n}| \leq |T^{\text{OPT}}|,$$

and the algorithm is $(1+\sqrt{2})$-competitive, as claimed.

It remains to show that the server can always return to p_0 before time $\sqrt{2} \cdot |T^{\text{OPT}}_{R_i}|$ when receiving a new request σ_i at time t_i. We prove this by induction over the number i of released requests. Clearly, the statement is true if $t_i = 0$ or $i = 1$, as the server does not leave p_0 before time t_i. Now consider the request σ_i with $i > 1, t_i > 0$. Please refer also to Figure 3.1. As an optimal tour $T^{\text{OPT}}_{R_i}$ over the set of requests R_i cannot finish before all requests are released, we have

$$|T^{\text{OPT}}_{R_i}| \geq t_i.$$

For the sake of contradiction, suppose that the server cannot return to p_0 before time

$$\sqrt{2} \cdot |T^{\text{OPT}}_{R_i}| \geq \sqrt{2} \cdot t_i.$$

This implies that the distance from p_i to p_0 is at least

$$|p_i - p_0| > \sqrt{2} \cdot t_i - t_i. \tag{3.2}$$

In order to reach position p_i at time t_i, the server cannot have been at p_0 after time

$$t_i - (\sqrt{2} \cdot t_i - t_i) = 2t_i - \sqrt{2} \cdot t_i.$$

Algorithm 3.2: Online Algorithm for Preemptive Uncapacitated Open or Closed DIAL-A-RIDE.

This function is called upon receiving a new request
Input: New request σ, current position p_t, unserved requests R_t
Output: Tour starting at p_t and serving all requests in R_t
return move$(p_0) \oplus$ waituntil$(\sqrt{2} \cdot |T^{\text{OPT}}_{R_t}|) \oplus T^{\text{OPT}}_{R_t}$

Consider the last request σ_{i-1} that was released before σ_i at time t_{i-1}. The tour that the server was following until time t_i was thus the tour serving R_{i-1}. By induction, the server was at p_0 at time $\sqrt{2} \cdot |T^{\text{OPT}}_{R_{i-1}}|$, and from before we know that it cannot have been at p_0 after time $2t_i - \sqrt{2} \cdot t_i$. We thus have

$$\sqrt{2}|T^{\text{OPT}}_{R_{i-1}}| \leq 2t_i - \sqrt{2}t_i, \text{ or } |T^{\text{OPT}}_{R_{i-1}}| \leq \sqrt{2} \cdot t_i - t_i. \tag{3.3}$$

Using the first property of Lemma 3.2, there must have been a request in R_{i-1} with source or destination at distance at least $|p_i - p_0|$ from p_0. This means the optimal tour $T^{\text{OPT}}_{R_{i-1}}$, which starts at p_0, must have length at least $|p_i - p_0|$ to serve this request. With inequality (3.2), we thus have

$$|T^{\text{OPT}}_{R_{i-1}}| \geq |p_i - p_0| > \sqrt{2} \cdot t_i - t_i, \tag{3.4}$$

a contradiction to inequality (3.3). □

Note that Algorithm 3.1 and its analysis in Theorem 3.3 does not rely on the restricted capacity of the server. Thus, the same result holds for the uncapacitated version of the problem. Note that for this case we do not need to specify whether the problem is preemptive or non-preemptive, as the server has no capacity constraints and thus dropping a request off before having finished to serve it does not pose an advantage.

Corollary 3.4 *Algorithm 3.1 is* $(1 + \sqrt{2}) \approx 2.41$*-competitive for the open and closed online* DIAL-A-RIDE *problem on the line without capacity constraints.*

3.2.2 Online Dial-A-Ride in the Euclidean Space

In this subsection, we prove that a slightly modified version of Algorithm 3.1 works in the uncapacitated case, even if the server does not move on a line, but in the Euclidean space M with Euclidean metric, that means

$$||p - q|| = \sqrt{(p - q) \cdot (p - q)},$$

where $\cdot$ denotes the dot product between two vectors. The server moves again with unit speed, that is $||p_t - p_{t'}|| \leq ||t - t'||$ for all $t, t' \geq 0$. A series of requests $\sigma_1, \ldots, \sigma_n$ arrives over time with $\sigma_i = (a_i, b_i; t_i)$, where $t_i \geq 0$ denotes the release time of the request and $a_i, b_i \in M$ denote its source and destination position, respectively. The other preconditions remain the same.

The algorithm differs from Algorithm 3.1 in that currently loaded requests are not unloaded when a new request appears. A formal description can be found in Algorithm 3.2.

We denote by $\sigma^E(t) = \left(a^E(t), b^E(t); t^E(t)\right)$ the most extreme request at time t, that is the unserved request which has either start or destination position farthest from the origin p_0. In particular, that means

$$||a^E(t)|| \geq ||a^i(t)||, ||a^E(t)|| \geq ||b^i(t)|| \text{ or } ||b^E(t)|| \geq ||a^i(t)||, ||b^E(t)|| \geq ||b^i(t)||$$

for all unserved requests σ^i at time t.

The following lemma is analogous to Lemma 3.2.

Lemma 3.5 *Without loss of generality, the following holds for Algorithm 3.2:*

1. *If T^{OPT} visits position $p \neq p_0$ at time t, then*

$$\frac{\max\left\{||a^E(t) - p_0||, ||b^E(t) - p_0||\right\}}{||p - p_0||} \geq 1.$$

2. *Let R_i^0 be the requests in R_i with their original sources and destinations (before being moved). Then,*

$$|T_{R_i}^{\text{OPT}}| \leq |T_{R_i^0}^{\text{OPT}}|.$$

Proof. As T^{OPT} is optimal, we can assume that the server always moves along a shortest trajectory between two endpoints, i. e. , sources or destinations, of unserved requests. This is without loss of generality as all other parts of the tour can be replaced, without increase of the overall length of the tour, by waiting at an endpoint before taking the direct connection between endpoints. Let p_t^u and p_t^d be those endpoints at time t and let without loss of generality

$$||p_t^d - p_0|| \geq ||p_t^u - p_0||.$$

Consider the server's position p_t at time t. Since we are operating in the Euclidean spance, we can express p_t as a linear combination of p_t^u and p_t^d. We obtain

$$\begin{aligned}
||p_t - p_0|| &= ||xp_t^u + (1-x)p_t^d - p_0|| = ||xp_t^u - xp_0 + (1-x)p_t^d - (1-x)p_0|| \\
&\leq x||p_t^u - p_0|| + (1-x)||p_t^d - p_0|| \\
&\leq x||p_t^d - p_0|| + (1-x)||p_t^d - p_0|| = ||p_t^d - p_0|| \\
&\leq \max\left\{||a^E(t) - p_0||, ||b^E(t) - p_0||\right\},
\end{aligned}$$

which concludes the first part of the proof.

For the second property, observe that, as the capacity of the server is unbounded, each request that has been picked up at their source but not delivered to their destination, may be carried by the server until its destination is reached. Thus, the tour $T_{R_i^0}^{\text{OPT}}$ is also valid for R_i, and hence the second property is fulfilled. □

Theorem 3.6 *Algorithm 3.2 is $(1+\sqrt{2}) \approx 2.41$-competitive for the uncapacitated open or closed online* DIAL-A-RIDE *problem in the Euclidean space with Euclidean metric.*

Proof. This proof is identical to the proof of Theorem 3.3 if we replace Lemma 3.2 by Lemma 3.5 and $|\cdot|$ by $||\cdot||$ throughout. Note that non-preemptiveness is not needed in this case, as the server has infinite capacity and hence can keep all requests which it has started to serve until it finishes serving them. □

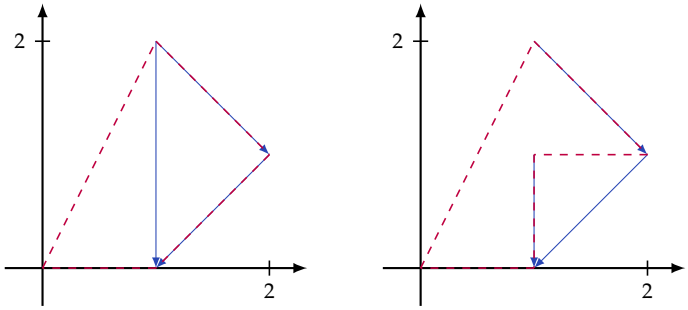


Figure 3.2: Sketch of an instance for Remark 3.7, which proves that it is important for Algorithm 3.2 to have a server without capacity constraints. Requests are in blue and optimal tours in red. On the right, even though one request is actually shorter, the optimal tour is longer.

We conclude this section with three remarks concerning the prerequisites for Algorithm 3.2 to work and its generalizations.

Remark 3.7 Note that Algorithm 3.1 relies on the observation in Lemma 3.2 that we can drop off requests anywhere closer to their destinations without increasing the length of an optimum tour. For Euclidean space this is no longer true, as illustrated by the following example in $\mathbb{R}^2$. Please refer also to Figure 3.2. Let

$$\sigma_1 = \big((1,2),(1,0);0\big), \quad \sigma_2 = \big((0,0),(1,2);t\big), \quad \sigma_3 = \big((1,2),(2,1);t\big),$$
$$\sigma_4 = \big((2,1),(1,0);t\big), \quad \sigma_5 = \big((1,0),(0,0);t\big)$$

for some $t \geq 0$. Clearly, waiting until time t and then following the trajectory

$$(0,0) \to (1,2) \to (2,1) \to (1,0) \to (0,0)$$

yields an optimum (open or closed) tour of length $t + \sqrt{5} + 2\sqrt{2} + 1 \approx t + 6.06$ for the original set of requests. Now assume that we started to serve σ_1 at some point before t and dropped it of at $(1,1)$, i. e. , along the shortest connection between $(1,2)$ and $(1,0)$, thus modifying σ_1 to

$$\sigma_1' = ((1,1),(1,0);0).$$

This increases the length of an optimum tour starting at time t, e. g. ,

$$(0,0) \to (1,2) \to (2,1) \to (1,1) \to (1,0) \to (0,0)$$

to $t + \sqrt{5} + \sqrt{2} + 3 \approx t + 6.65$. Algorithm 3.2 avoids this issue by never dropping off requests before reaching their destinations.

Remark 3.8 Algorithm 3.2 does indeed work in general metric spaces. To see this, remark that when we run Algorithm 3.2, the server always runs an optimal tour for a given set of requests. Thus, when the server is at a point p_i, the optimal tour for this given set of requests which the server is following currently is at least of length $||p_i - p_0||$ and hence the first part of inequality (3.4) holds in all metric spaces. This also serves as alternative proof for Theorem 3.6.

Remark 3.9 It is possible to adapt Algorithm 3.2 also for the capacitated case or the non-preemptive case, for both the open and the closed version, both on the line and in metric spaces. In these cases, the server finishes its loaded requests when it receives a new request before returning to the origin; this costs at most a time of $|T_{R_t}^{\text{OPT}}|$. The server then starts a new tour at time $(1+\sqrt{2}) \cdot |T_{R_t}^{\text{OPT}}|$. In total, this approach reaches a competitive ratio of $2+\sqrt{2} \approx 3.41$ which matches the previous upper bound of Krumke [Kru01].

3.2.3 A Lower Bound on the Competitive Ratio

In this subsection, we provide a lower bound for non-preemptive closed DIAL-A-RIDE on the line that improves the lower bound of 1.70 by Ascheuer et al. [AKR00].

Theorem 3.10 *No algorithm for the non-preemptive closed* DIAL-A-RIDE *problem on the line with fixed capacity $c \geq 1$ has competitive ratio lower than $\rho = 1.75$.*

Proof. Consider any ρ-competitive online algorithm ALG. We define an instance with three types of requests. There is one simple request $\sigma_0 = (0, 0; 1)$, c requests of type $\sigma_1^i = (t, 0; t)$ and c requests of type $\sigma_2^i = (-t, 0; t)$, where $i = 1, \ldots, c$ and $t \geq 1$ is the time when ALG serves σ_0. Clearly, an optimal tour which knows all requests from the beginning can first serve requests σ_1^i, then immediately σ_0 and last σ_2^i. This tour takes a time of $4t$. Now look at an online algorithm ALG. First, assume that ALG serves the c requests $\sigma_1^1, \ldots, \sigma_1^c$ not together in one tour from t to 0. Then ALG can pick up any σ_1^i at the earliest at time $2t$ and have them delivered at $3t$. But ALG has to return to t to pick up the remaining requests σ_1^i it did not bring to 0 in the first trip which takes another $2t$ of time and also has to get requests σ_2^i. In total, we have $|T^{\text{OPT}}| = 4t$ and $|T^{\text{ALG}}| \geq 7t$ in this case and hence obtain

$$\frac{|T^{\text{ALG}}|}{|T^{\text{OPT}}|} \geq \frac{7}{4}. \tag{3.5}$$

An analogous argument shows that ALG has to take the c requests $\sigma_2^1, \ldots, \sigma_2^c$ together to the origin or we have again inequality (3.5) fulfilled.

Thus, from now on we assume that ALG picks up all c requests $\sigma_j^1, \ldots, \sigma_j^c$ together before going to the origin for $j = 1, 2$. Let $t' \geq 2t$ be the first time when ALG picks up all c requests σ_1^i or all c requests σ_2^i. We then have $|T^{\text{OPT}}| = 4t$ and $|T^{\text{ALG}}| \geq t' + 3t$.

If $t' \geq 4t$, then

$$\rho \geq \frac{|T^{\text{ALG}}|}{|T^{\text{OPT}}|} \geq \frac{7}{4}$$

as claimed.

Otherwise, without loss of generality, we assume that the c requests σ_1^i are picked up before the c requests σ_2^i. Furthermore, we introduce the additionally high five request $\sigma_3 = (t, t; t' + 1/7)$. This new request appears while ALG is serving the c requests that have to be transported from t to the origin. As we have capacity constraint c and do not allow preemption, those c requests have to be finished serving. Only after they are finished and the server is at the origin, the new high five request and requests σ_2^i can be served which takes time $2t$ each, thus $|T^{\text{ALG}}| \geq t' + 5t$. Please

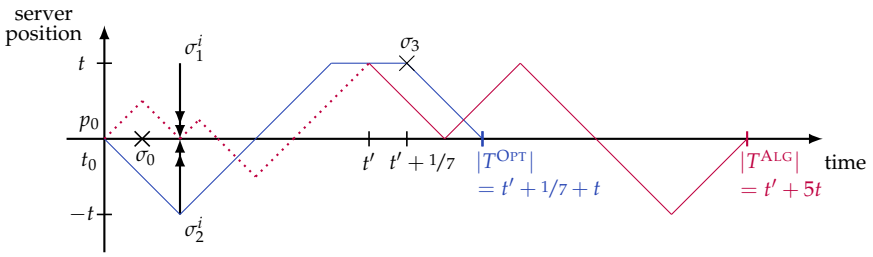


Figure 3.3: Construction of a DIAL-A-RIDE instance for Theorem 3.10. Note that the bold arrows with a double tip represents the c identical requests σ_1^i and σ_2^i. A possible tour for the algorithm is depicted in red, the respective optimal tour for those requests in blue.

refer also to Figure 3.3. We now differentiate between two cases. If $t' \leq 3t - 1/7$, we have $|T^{\text{OPT}}| = 4t$ as the new request appears at time $3t$ and can immediately be served by T^{OPT}. By using $t' \geq 2t$, we get

$$\rho \geq \frac{|T^{\text{ALG}}|}{|T^{\text{OPT}}|} \geq \frac{t' + 5t}{4t} \geq \frac{7}{4},$$

as claimed. Now if $t' > 3t - 1/7$, then $|T^{\text{OPT}}| = t' + 1/7 + t$ as, after first serving σ_2^i and the new request, T^{OPT} still has to return to the origin and by doing so finish serving the c requests σ_1^i. But then

$$\rho \geq \frac{t' + 5t}{t' + 1/7 + t}.$$

This is monotonically decreasing in t' for $t, t' \geq 1$. Since we have $t' < 4t$ from above, we get

$$\rho > \frac{9}{5 + 1/7} = 1.75,$$

which concludes the proof. □

3.3 Some Results for the Offline Setting

As a number of online algorithms do, our Algorithm 3.1 relies on an offline algorithm to run. It is thus important consider the computational difficulty of this subproblem.

In the following chapter, we look at some results for the hardness of the offline problems. Remark that though we do call them offline, we indeed consider the version with release times. Those release times are known from the beginning.

We will first look at hardness results for DIAL-A-RIDE on the line. Further, we add results for TSP on the line which generalize to DIAL-A-RIDE, as TSP is a special case of this. We finish this chapter by giving a simple approximation algorithm for offline DIAL-A-RIDE on the line.

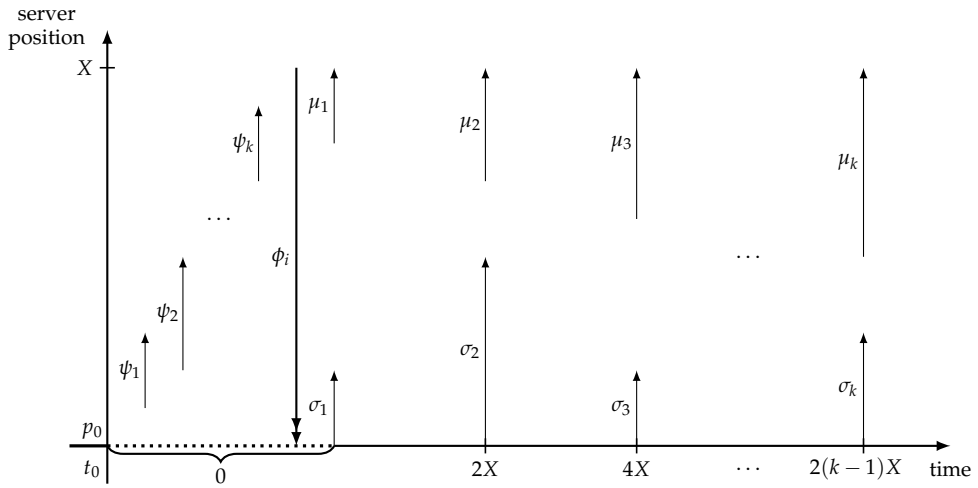


Figure 3.4: Construction of a DIAL-A-RIDE instance for Theorem 3.12. Note that each displayed request actually stands for c identical requests. The bold arrow with a double tip even represents the $c \cdot k$ identical requests ϕ_i.

3.3.1 Hardness of Offline Dial-a-Ride

For the non-preemptive offline DIAL-A-RIDE problem on the line we show that the open and closed variant with release times are NP-hard. We reduce from the Circular Arc Coloring problem (see, e. g. , [GJ79]), which is also used in a reduction for minimizing the sum of completion times of DIAL-A-RIDE on the line with capacity $c = 1$ (see [dPLS+04]).

We start by stating the Circular Arc Coloring decision problem.

Definition 3.11 (Circular Arc Coloring) *Let $\mathcal{I}$ be a family of intervals on a circle, and let $k \in \mathbb{Z}_{>0}$ be a fixed parameter. The decision problem* Circular Arc Coloring *is to decide if a coloring of all intervals $I \in \mathcal{I}$ with k colors exists, such that no two intervals of the same color overlap.*

Using this, we can now prove a hardness result for offline DIAL-A-RIDE.

Theorem 3.12 *The non-preemptive closed offline* DIAL-A-RIDE *problem on the line with fixed capacity $c \geq 1$ is NP-complete.*

Proof. The problem is in NP, as we can decide whether or not a server tour completes before a fixed deadline in polynomial time. To proof NP-completeness, we give a reduction from an NP-complete problem. Let an instance of the circular arc coloring problem be given by a circle with circumference X, a set $\mathcal{I}$ of intervals $I = [\ell, r)$ on the circle with $\ell, r \in [0, X)$, and coloring number k. Without loss of generality we can assume that there are exactly k intervals $I_1^0, ..., I_k^0$ overlapping zero. If there are fewer, we can add intervals $[0, \varepsilon)$ to the instance for sufficiently small ε. If there are more, the instance is trivial as each coloring with only k colors will color at least two of the intervals which overlap zero with the same color by pigeonhole principle. We define a set of requests to construct a DIAL-A-RIDE problem. The construction of requests with their start and end position is also displayed in Figure 3.4. For each

interval $I_j^0 = [\ell_j, r_j), 1 \leq j \leq k$, we create $2 \cdot c$ requests in the following way. We introduce c requests

$$\sigma_j = (a_j, b_j; t_j) = (0, r_j; 2(j-1)X)$$

and c requests

$$\mu_j = (\ell_j, X; 2(j-1)X).$$

For any other interval $I = [\ell, r)$ define c requests

$$\psi = (\ell, r; 0)$$

and furthermore define $c \cdot k$ requests

$$\phi_i = (0, X; 0)$$

for $i = \{1, ..., k\}$. The intuition is that we cut the circle open at point zero and hang it up, such that each interval not crossing zero is equal to c requests appearing at time zero, requests which are going from their lower part of their interval to their upper part. Moreover, intervals crossing zero which are "cut open" appear later and as two times c requests, the first c covering the intervals before, the others the one after zero.

Any feasible server tour for this instance with start position in the origin travels at least k times from the origin at zero to X and back to serve the $c \cdot k$ requests ϕ_i. Hence, any feasible tour has length at least $2kX$. A tour of exactly that length is closed and partitions the requests into k sets, each served during one trip from the origin to X and back. Observe that requests σ_k and μ_k must be served on the last trip because they are released at time $2(k-1)X$. Then requests σ_{k-1} and μ_{k-1} must be served on the previous trip. They cannot be served before because of their release time and they cannot be served on the last trip as the server has capacity c and serves requests σ_k whose interval overlaps with that of σ_{k-1}, and also serves requests μ_k whose interval overlaps with that of μ_{k-1}. Iteratively we deduce that requests σ_j and μ_j, for $j \leq k$, must be served on the same trip.

This shows, that if we use the k-partitioning induced by the tour to color the intervals on the circle, meaning each server trip between zero and k corresponds to one color, we get a feasible circular arc coloring.

Conversely, we can transform any circular arc coloring into a feasible server tour of length $2kX$ by scheduling the requests corresponding to one color on the same trip from the origin to X. □

This result, apart from the case $c = 1$, is also subsumed in the next theorem, which can be achieved using similar techniques.

Theorem 3.13 ([BDH+17]) *The non-preemptive closed offline* DIAL-A-RIDE *problem on the line with capacity $c \geq 2$ is NP-complete, even when all release times are zero.*

Putting these two theorems together, we achieve the following NP-completeness result.

Theorem 3.14 *The non-preemptive open and closed offline* DIAL-A-RIDE *problem on the line are NP-complete. For capacity $c \geq 2$ fixed this even holds when all release times are zero.*

Proof. Theorems 3.12 and 3.13 show the statement for the closed problem variant. It remains to show the open case. In the proof of Theorem 3.12, we give a problem for which we show that deciding whether there is a closed tour of length $2kX$ is hard. Consider this problem again and suppose that the server does not have to return to the origin after having served the last request. But also for this open problem variant, any feasible tour has at least length $2kX$ as a feasible server tour starts at position zero and travels at least k times from X to zero to serve the k requests ϕ_i. This shows that any tour attaining this bound is a closed tour.

The second part of the theorem can be shown using the proof of Theorem 3.13 where closed tours of full capacity are considered (refer to [BDH+17]). Again, as all tours are using their full capacity to and from the origin, there cannot be a smaller open tour and thus all minimal open tours finish at the origin, hence are in fact closed. □

Remark 3.15 It is particularly interesting that in the hardness reduction the server's start position is at one of the two extreme positions occurring in the tour. For the same problem but allowing preemption, Karp showed that it is solvable in polynomial time [Kar72], while the hardness of the problem with an arbitrary start position is not known.

Remark 3.16 Non-preemptive offline DIAL-A-RIDE on the line with capacity restraints is known to be easy without release dates (see [dPLS+04]) and becomes hard if each request comes with a deadline in addition to its release time (see [Tsi92]). Thus the only complexity question remaining open is the case with unbounded capacity.

3.3.2 Hardness of Offline TSP on the Line

In this subscetion, we give some results for the Traveling Salesperson Problem on the line with release times. As those can be viewed as "high five" requests, each request σ_i is given by release time t_i and target position a_i.

Psaraftis et al. [PSMK90] show that open offline TSP on the line with release dates can be solved in quadratic time. For the closed variant they claim that the optimal tour has the following structure [PSMK90, pp. 215–216], first moving to one extreme end, then to the other and back to the origin:

$$\begin{aligned} & \text{waituntil}() \oplus \text{move}(p^{\mathrm{U}}) \oplus \text{move}(p^{\mathrm{D}}) \oplus \text{move}(p_0) \\ \textbf{or} \quad & \text{waituntil}() \oplus \text{move}(p^{\mathrm{D}}) \oplus \text{move}(p^{\mathrm{U}}) \oplus \text{move}(p_0). \end{aligned}$$

Here the waiting time at the origin is chosen maximally such that all requests are still served. We contradict this claim by showing that an optimal server tour may need to turn around arbitrarily many times.

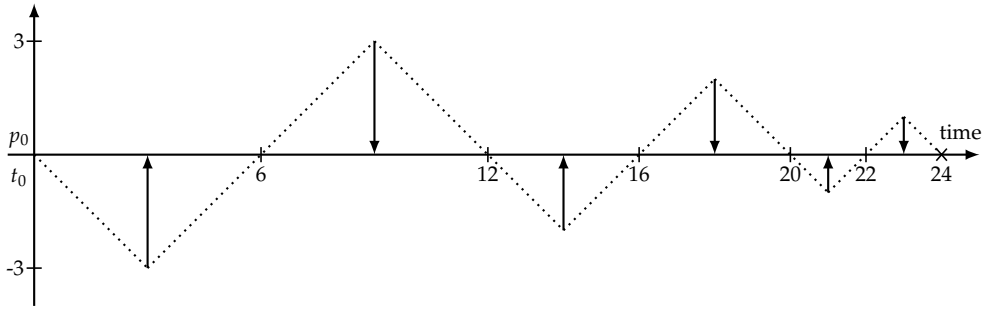


Figure 3.5: Sketch of an instance for Theorem 3.17

Theorem 3.17 *For every $k \in \mathbb{N}$, there is an instance of closed* TSP *on the line such that any optimal solution turns around at least $2k$ times.*

Proof. We analyze the following instance consisting of $2k+1$ requests

$$\sigma_i = (a_i; t_i) \text{ with } a_i = 0, 1, -1, 2, -2, 3, -3, \ldots, k, -k$$
$$\text{and } t_i = M, M-1, M-3, M-6, M-10, M-15, \ldots, k,$$

for $M := 2k(k+1)$. An instance for $k = 3$ is depicted in Figure 3.5.

We show this instance has a unique optimal server tour with $2k$ turnarounds. Observe first, that the difference between two consecutive release times $t_i - t_{i-1}$ is exactly the travel time of the server between the two requested positions $|a_i - a_{i-1}|$. Now consider the server tour T serving each of the requests exactly at its release time. It is obviously feasible and also optimal, as it ends at time $M = 2k(k+1)$, which is the release time of the request at position zero. By construction, the server alternatively serves requests down and up of position zero and thus turns around $2k$ times.

Now assume that there is another optimal server tour T'. Then T' must serve the request at position zero at time M, as this is its release time as well as the makespan of T'. As we observed

$$t_i - t_{i-1} = |a_i - a_{i-1}|,$$

the tour T' also serves the previous request exactly at its release time. Iteratively the same must hold for all other positions. This shows any request is served exactly at its release time and hence, the tour is exactly the one described as T above. □

As TSP on the line is a special case of DIAL-A-RIDE on the line, this result of course generalizes.

In the following, we show a dynamic program that solves closed offline TSP on the line in quadratic time. It is inspired by the one of Psaraftis et al. [PSMK90] for the open variant and a dynamic program of Tsitsiklis [Tsi92] for the same problem with deadlines instead of release times.

To describe the dynamic for the closed offline TSP problem on the line, we make the following non-restrictive assumptions on the requests:

Algorithm 3.3: Dynamic Program for Closed Offline TSP on the Line

Input: Set of requests $\sigma_i = (a_i; t_i)$ with $t_i \geq |a_i|$ for all $-d \leq i \leq u$ and $a_i < a_{i+1}$ for all $-d \leq i < u$

Output: Minimum completion time $C_{\max}$ of a tour

Let $C^+_{-d-1,u} \leftarrow t_u$ ▷ minimum time needed to serve upmost request

Let $C^-_{-d,u+1} \leftarrow t_{-d}$ ▷ minimum time needed to serve downmost request

for $i = -d, ..., u$ **do**

  Let $C^-_{i,u+1} \leftarrow \max\{t_i, C^-_{i-1,u+1} + a_i - a_{i-1}\}$ ▷ serving requests starting at the lowest one, one after another

for $j = u, .., d$ **do**

  Let $C^+_{-d-1,j} \leftarrow \max\{t_j, C^+_{d-1,j+1} + a_{j+1} - a_j\}$ ▷ serving requests starting at the upmost one, one after another

for $f = u + d, ..., 0$ **do**

  for $i = -d, .., u - f$ **do**

    Let $j \leftarrow i + f$

    Let $C^-_{i,j} \leftarrow \max\{t_i, \min\{C^+_{i-1,j} + a_j - a_i, C^-_{i-1,j} + a_i - a_{i-1}\}\}$

    Let $C^+_{i,j} \leftarrow \max\{t_j, \min\{C^+_{i,j+1} + a_{j+1} - a_j, C^-_{i,j+1} + a_j - a_i\}\}$ ▷ starting with outmost requests, find the points where the server has to turn to another direction to get the decreasing zig zag shape

return $C_{\max} = C^+_{0,0}$

- Each position is requested at most once, as several requests to the same position can all be fulfilled when the one with the largest release time is served.
- There is a request $\sigma_0 = (0; 0)$.
- The requests $\sigma_i = (a_i; t_i)$ are labeled by increasing position, i. e. $a_i > a_{i-1}$, and we choose the indices such that all requests for a position above the start position, $a_i > 0$, have an index $i > 0$ and symmetrically all requests to serve a position $a_i < 0$ have an index $i < 0$. Then we have a sequence of requested positions of the form:

$$a_{-d} < \cdots < a_{-1} < a_0 = 0 < a_1 < \cdots < a_u.$$

Our dynamic program, depicted in Algorithm 3.3 relies on the fact that an optimal server tour has a zig-zag shape with decreasing amplitude; we thus need to find the points when the server turns around to another direction. We compute for each index pair $-d - 1 \leq i < j \leq u + 1$ the completion time of two tours. We use $C^+_{i,j}$ for the best tour serving all requests σ_k with $k \leq i$ or $k \geq j$ and ending at position a_j. This means the tour serves all requests to position a_j and larger positions as well as all requests to position a_i and smaller positions. After that, all requests that still have to be served are between positions a_i and a_j, thus the server will not have to be outside of those bounds again. This yields the zig-zag shape with decreasing amplitude. The time $C^-_{i,j}$ is the completion time of the best tour that serves the same set of requests and ends at position a_i. Starting with the largest difference $j - i$ and iteratively decreasing it, we recursively compute $C^+_{i,j}$ and $C^-_{i,j}$.

We start by considering the two request sets $\{\sigma_{-d}\}$ and $\{\sigma_u\}$. By assumption, the release time t_i exceeds the travel time between start position zero and requested position a_i for each request, as any earlier release time would only help the algorithm. Thus the initial values are

$$C^+_{-d-1,u} = t_u \text{ and } C^-_{-d,u+1} = t_{-d}.$$

We then compute by recursion the completion time of the tours $C^+_{i,j}$ by computing $C^+_{-d-1,u}$ and $C^-_{-d,u+1}$ for the two request sets $\{\sigma_u\}$ and $\{\sigma_{-d}\}$. Then we recursively compute $C^+_{i,j}$ and $C^-_{i,j}$, starting with large difference $f = j - i$ and iteratively decreasing it. We output $C^+_{0,0}$ as the minimum completion time of a feasible server tour.

To prove the correctness of Algorithm 3.3 we use two lemmas. We first prove a structural result about feasible server tours and then show that the recursion we use in the dynamic program is correct. We assume without loss of generality that each request is served when its requested position is visited for the last time in the tour.

Lemma 3.18 *At any time in a feasible server tour ending at position p, the set of served requests is the union of two disjoint sets $S_1 = \{\sigma_d, ..., \sigma_i\}$ for $a_i \leq p$, and $S_2 = \{\sigma_j, ..., \sigma_u\}$ for $p \leq a_j$, both of which are contiguous.*

Proof. Assume there is a time t, at which the set of requests does not have the claimed structure. Then there is a request σ_1 served until time t and a request σ_2 served after t with either

$$p \leq a_1 < a_2 \text{ or } a_1 < a_2 \leq p.$$

Without loss of generality assume the first to be true. At time $t' > t$, when request σ_2 is served, the server is at position a_2. The server tour ends at position p and thus passes position a_1 at some time after t. This contradicts that request σ_1 has been served until time t, which was our assumption. □

Lemma 3.19 *We are given an instance of* TSP *on the line with requests $\sigma_d, ..., \sigma_u$ to positions $a_d < a_{d+1} < ... < a_u$ and completion times $C^+_{i,j+1}, C^-_{i,j+1}, C^+_{i-1,j}$ and $C^-_{i-1,j}$ for some indices $d \leq i \leq j \leq u$; then the minimal completion times $C^+_{i,j}$ and $C^-_{i,j}$ are given by the following recursion:*

$$C^+_{i,j} = \max\{t_j, \min\{C^+_{i,j+1} + a_{j+1} - a_j, C^-_{i,j+1} + a_j - a_i\}\},$$
$$C^-_{i,j} = \max\{t_i, \min\{C^+_{i-1,j} + a_j - a_i, C^-_{i-1,j} + a_i - a_{i-1}\}\}.$$

Proof. The completion time $C^+_{i,j}$ we give is feasible, as it can be achieved by executing the tour attaining $C^+_{i,j+1}$ or the tour attaining $C^-_{i,j+1}$ and then moving to position a_j, waiting there until the release time t_j has passed.

By Lemma 3.18, the tour serving requests $\sigma_d, ..., \sigma_i$ and $\sigma_j, ..., \sigma_u$, serves request σ_j last and request σ_{j+1} or σ_i immediately before. Thus, the completion time exceeds the minimum of $C^+_{i,j+1} + a_{j+1} - a_j$ and $C^-_{i,j+1} + a_j - a_i$ and hence

$$C^+_{i,j} \geq \min\{C^+_{i,j+1} + a_{j+1} - a_j, C^-_{i,j+1} + a_j - a_i\}.$$

Furthermore, we cannot serve request σ_j before its release time t_j, thus

$$C^+_{i,j} \geq t_j.$$

Symmetrically the recursion for $C^-_{i,j}$ is constructed. □

These two lemmas allow us to prove the correctness of Algorithm 3.3.

Theorem 3.20 *Algorithm 3.3 computes the minimum completion time of a server tour for offline TSP on the line in time $O(n^2)$.*

Proof. By Lemma 3.18, the optimal server tour has the structural property that at any fixed point in time, it has served the union of two disjoint contiguous sets. We start computing the best completion time of a tour for pairs of request sets containing one request each. Iteratively, we increase the number of contained requests $u - d - f$ by decreasing the parameter f. We end with $f = 0$ and compute the earliest completion time of tours serving the complete request set in this iteration. The recursion we use is correct by Lemma 3.19 and the computation order is feasible, as the recursion formula only contains request set pairs of strictly smaller size.

The algorithm uses quadratic time for the two nested loops and all other steps take linear or constant time. Hence the algorithm runs in time $O(n^2)$. □

We finish this subsection by a simple observation which explains the connection between our Algorithm 3.3 and the algorithm proposed by Psaraftis et al. [PSMK90].

Remark 3.21 The algorithm by Psaraftis et al. [PSMK90] for open TSP on the line can be obtained from Algorithm 3.3 by changing the computation order of the completion times and returning the minimum completion time of all possible end positions $\min_{-d \leq i \leq u} C^+_{i,i}$.

3.3.3 Approximating Offline Dial-A-Ride

This section is dedicated to the approximation of the offline problem. In the classification of closed dial-a-ride problems by Paepe et al. [dPLS+04], offline TSP on the line is the problem $1|s = t, d_j|\text{line}|C_{\max}$. De Paepe et al. [dPLS+04] claim Tsitsiklis [Tsi92] shows a polynomial algorithm, but this is for the open version. We solve the closed variant by giving a polynomial algorithm (Theorem 3.20) as well as a counterexample in Theorem 3.17 to the algorithm of Psaraftis et al. [PSMK90]. Our Theorem 3.14 shows that the problem $1, \text{cap}1|d_j|\text{line}|C_{\max}$ in the same classification scheme describing DIAL-A-RIDE on the line with capacity one is NP-hard. While this is implicitly claimed by De Paepe et al. [dPLS+04], no proof is given. For the generalization to arbitrary capacity but without release dates, described as $1||C_{\max}$, Guan [Gua98] showed hardness for capacity $c = 2$ and our new hardness proof in Theorem 3.12 handles any capacity $c \geq 2$.

We complement these hardness results by describing a simple approximation algorithm for the closed offline non-preemptive DIAL-A-RIDE problem. We make use of an algorithm by Atallah and Kosaraju [AK88] for the offline non-preemptive DIAL-A-RIDE problem when all release time are zero. Our algorithm first executes their algorithm. This yields a server tour which visits all requests, but may visit

Algorithm 3.4: Approximation Algorithm for the Closed Non-Preemptive Offline DIAL-A-RIDE Problem

Input: Set of requests $\sigma_i = (t_i, a_i, b_i)$ for $i = 1, \ldots, n$, server capacity $c \geq 1$
Output: Feasible server tour T with $|T| \leq (c+1) \cdot |T^{\text{OPT}}|$
1 Run the algorithm of Atallah and Kosaraju to compute the optimal tour S without release times
2 Let $t^{\max}$ be the largest release time
3 **return** the tour T which waits at the origin until time $t^{\max}$ and then executes the tour S

some before their release time. A new start time for this tour is then computed, such that no request is visited before its release time. Our algorithm is formally described in Algorithm 3.4.

Theorem 3.22 *Algorithm 3.4 computes a tour of length at most* $(c+1) \cdot |T^{\text{OPT}}|$ *for the closed non-preemptive* DIAL-A-RIDE *problem.*

Proof. The tour S is the optimal tour for a server of capacity one, when no request has a release time. Its length is at most $c \cdot |T^{\text{OPT}}|$ as repeating the optimal tour c times is a feasible tour with capacity one. The largest release time $t^{\max}$ is a lower bound on the length of the optimal tour, as it cannot finish before all requests are released. This means that the tour computed by Algorithm 3.4 has length

$$|T^{\text{ALG}}| \leq (c+1) \cdot |T^{\text{OPT}}|,$$

which finishes the proof. □

3.4 Further Results

In addition to the findings presented in this work, our joint work [BDH+17] has resulted in some further intriguing outcome for the Travelling Salesperson Problem.

Comparing our conclusions to previous work, for online TSP in the line, our results as presented in [BDH+17] are best-possible online algorithms for both the open and closed variant of online TSP on the line, as well as a new (tight) lower bound for the open variant. The new algorithm for the closed variant has a competitive ratio of $\mathcal{C} = (9 + \sqrt{17})/8 \approx 1.64$, where $\mathcal{C}$ is the nonnegative root of the polynomial $4x^2 - 9x + 4$, matching a lower bound of Ausiello et al. [AFL+01] and improving on their 1.75-competitive algorithm. Further, for the open version of the problem, we give a lower bound of $(\rho - \varepsilon)$ on the best-possible competitive ratio for any $\varepsilon > 0$, where $\rho \approx 2.04$ is the second-largest root (out of the four real roots) of $9\rho^4 - 18\rho^3 - 78\rho^2 + 210\rho - 107$. This is the first bound strictly greater than two. We provide an optimal online algorithm matching this bound and improving on the 2.33-competitive algorithm by Ausiello et al. [AFL+01]. Note that for this problem, a lower bound of two is obvious: At time one, we present a request either at minus one or at one, whichever is further away from the online server. The online tour has length at least two while the optimum tour has length one.

ONLINE	Closed		Open	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Online TSP on the Line				
new		**1.64** [BDH+17]	**2.04** [BDH+17]	**2.04** [BDH+17]
old	1.64 [AFL+95, AFL+01]	1.75 [AFL+95, AFL+01]	2 [AFL+94, AFL+01]	2.33 [AFL+94, AFL+01]
DIAL-A-RIDE on the Line				
preemp.	1.64 [AFL+95, AFL+01]	2 [AKR00]	**2.04** (Th. 3.23)	**2.41** (Th. 3.3)
non-preemp.	**1.75** (Th. 3.10)		**2.04** (Th. 3.23)	3.41 [Kru01]

Table 3.1: Overview of our results for online TSP on the line and online DIAL-A-RIDE on the line.

OFFLINE	Closed	Open
TSP on the Line	$\mathcal{O}(n^2)$ (Th. 3.20)	$\mathcal{O}(n^2)$ [PSMK90]
DIAL-A-RIDE on the Line		
non-preemptive	NP-hard (Th. 3.14)	NP-hard (Th. 3.14)

Table 3.2: Overview of our results for offline TSP and DIAL-A-RIDE on the line with release times.

The current state of the art for DIAL-A-RIDE and TSP is depicted in Table 3.1 for the online and in Table 3.2 for the offline version of the problem. Those results settle online TSP on the line from the perspective of competitive analysis and enable us to state a lower bound for open online DIAL-A-RIDE.

Theorem 3.23 *Let $\rho \approx 2.04$ be the second-largest root (out of the four real roots) of*

$$9\rho^4 - 18\rho^3 - 78\rho^2 + 210\rho - 107.$$

There is no $(\rho - \varepsilon)$-competitive algorithm for open preemptive and non-preemptive DIAL-A-RIDE *on the line for any $\varepsilon > 0$.*

Proof. As DIAL-A-RIDE is a more general version of TSP, this follows immediately from Theorem 4.1 in [BDH+17]. □

3.5 Discussion and Open Problems

In this chapter, we studied the mathematical formulation and analysis of some carpooling problems. By doing so, we are hoping to present an approach to reduce the total number of actors on streets while at the same time provide sensible routing, thus reducing congestion and the overall drain on material, time and environment.

Our main focus was on online DIAL-A-RIDE-problems. We presented a simple algorithm which is $(1+\sqrt{2})$-competitive for a number of cases: in the capacitated and uncapacitated, preemptive, open and closed DIAL-A-RIDE on the line. However, it does not immediately apply to any non-preemptive version of the problem. They seem to be much harder to handle as simple manipulations of an already started

server tour are not as easy. If a request that is started to being served has to be finished instead of just being shortened because of a readjustment of the tour, it seems more common to be led astray. This reflects in the best-known algorithm for non-preemptive online DIAL-A-RIDE being 3.41-competitive [Kru01], which is seconded by our results.

Understanding how good algorithms on the line work is important, as it can give us insight which we need for a more general metric setting. The latter is harder, as, e. g. , we can not expect the server to always stay within certain clear bounds and even within a convex structure. Thus the distances between two points can lead to problems. While it is conceivable that a situation where requests have to be served on a line arises, it is not in general the case. We thus modified the algorithm to also work in a general metric setting. For uncapacitated, open and closed, preemptive DIAL-A-RIDE, we present an algorithm which works in the Euclidean space. It relies on the fact that the server never moves too far away from the origin. As many problems tend to be able to be modelled within Euclidean space, this is a significant step. Further, it seems likely that the algorithm also works in additional spaces.

Trying to get more insight into how good online algorithms on the line work in general, we proved that no algorithm for non-preemptive closed DIAL-A-RIDE on the line with fixed capacity $c \geq 1$ can achieve competitive ratio of less than 1.75.

Closing the gap between lower and upper bound on the competitive ratio for each problem is an interesting research topic that remains open. Further directions to explore include expanding our knowledge on online DIAL-A-RIDE in metric spaces in general as well as investigating algorithms which work in a non-preemptive setting.

Our second focus was on understanding the structure of optimal algorithms for the underlying offline DIAL-A-RIDE problems where all requests with their release times are known from the beginning. We could show that the non-preemptive closed DIAL-A-RIDE on the line with capacity ≥ 1 is NP-complete and give a simple $c + 1$-approximation algorithm. This again points to why the non-preemptive case seems harder to tackle also in the online setting. The complexity of offline DIAL-A-RIDE on the line with unbounded capacity remains open.

In this context, it is intriguing to investigate the special case of DIAL-A-RIDE where the capacity does not play any role, that means, if all requests have a source equal to their destination and thus do not have to be transported, but only visited. This is known as TSP on the line with release times and we could prove that there is always an optimal tour which has a certain zig zag structure. Further, we gave an algorithm which computes the minimum completion time for this case in $O(n^2)$. It remains open to find a similar structural result for the more general DIAL-A-RIDE which may give us insight into the design of good competitive algorithms. As TSP is a widely studied and important model for a variety of fields, the result is also interesting in its own right.

An additional question to investigate is, to which other DIAL-A-RIDE instances the simple 2.41-competitive algorithm can be adapted. Clearly, it can be adapted to work in the general metric as well as, with some loss in the competitive ratio, in the capacitated and the non-preemptive version of the problem, see Remark 3.9. As the algorithm has a simple and versatile structure, we may find additional special cases for which it can be used.

Conclusion

Alice: *I don't much care where [I get to]...*
Cat: *Then it doesn't matter which way you go.*
Alice: *...so long as I get somewhere.*
Cat: *Oh, you're sure to do that, if you only walk long enough.*

Alice in Wonderland by Lewis Carroll [Car65]

While the truth in the Cheshire Cat's statement is undeniable, it can hardly be considered good advice about how to choose directions sensibly. Indeed, not only the final destination is important, but also the path and mode of transport one decides on to get there. Nowadays more than ever this also includes taking into consideration the decisions and behavior of other people.

One of the main goals of this thesis was to identify ways to reduce congestion in networks. An increasingly large number of actors in local and global networks and the use of new technology poses new challenges for network design and management, but also offers new possibilities, e. g. in terms of communication and collaboration between users. We hope to decrease the financial, personal, and environmental burden that congestion carries. More specifically, improved flow leads to reduced Greenhouse Gas emissions, higher sustainability, more efficient, faster, and well thought-out networks, as well as to less wastage of material and personal stress. Ultimately, this results in economic growth and improves globalization prospects.

We identified two main starting points to improve network flow. First, to provide better routing management, and second, to reduce the number of actors which are using the network. We encountered three problems which we concentrated on: users act independently in possibly huge and complex networks, they might manipulate data which they send to a regulation agency, and neither networks nor user decisions are static. As optimal solutions are therefore not always easy to find, we attacked these problems using different approximation methods.

Taking into consideration the complexity of large network structures in which many users act independently, it is reasonable to assume that not all instances of a problem can be improved in polynomial time in an optimal way. Therefore, we decided to use approximation algorithms to approach good routing solutions which reduce the overall network congestion, thus approximating the optimal solution due

to time sensitivity constraints. Further, we observed that the growth in prevalence and increasing use of personal navigation devices, which are able to communicate with another, leads to new ways of traffic regulation, for example when guiding drivers through cities. Considering this, we introduced a local search algorithm for a general resource allocation problem with diseconomies of scale, which can for example model rerouting users in networks. With urban population growing and an estimated 60% of humans living in urban areas by 2030 and every third person living in a city with at least half a million inhabitants [Uni16], this is a prevailing problem to consider regarding traffic and public transport, but also, e. g. , electric grids and telecommunication networks. The local search algorithm was analyzed for the case that the diseconomies of scale follow a polynomial pattern. As traffic congestion is widely assumed to follow a polynomial of degree four, this is a reasonable assumption and delivers practice oriented results. Our bound on the locality gap, which describes the difference between a global and local optimum, was of the same order as a previous result by Makarychev and Sviridenko [MS14], but, in contrast to theirs, our algorithm works in a distributed manner and is deterministic. This seems desirable considering the complexity of the network and fast changes within, and also the amount of personal navigation devices which may be used to calculate local optima. In contrast, while we gave both upper and lower bounds on the locality gap, the bounds did not match up. Considering that traffic congestion is often modelled by a polynomial of degree four, closing the gap appears important even just for small maximum degrees. Indeed however, it is possible that the local search algorithm works much better in practice for many problems, as the calculated locality gap describes a worst case scenario. Test runs on real world data are therefore a prospect of further work.

Some simplifications were made in order to describe this practical problem in a mathematical way. For instance, while we took capacity restraints into consideration, they have only been given implicitly by making routes prohibitively expensive for commodities in the resource allocation model. Another simplification was that we assumed users to act in a way which is optimal for the global network flow, or at least follow the guidance of their personal navigation devices without derivation. Concerning our results, ideally, we would want the task at hand to be solved in an even more distributed way to take advantage of these devices, as algorithms that run in a parallel manner on all devices while keeping communication up remain important.

In everyday life, users may not have the best overall network flow in mind, and thus may be susceptible to hand skewed data to a decision maker in order to gain an advantage. This problem is urgent when it comes to, e. g. , reducing the total number of actors in a network. In smart cities, collaboration is thus a key factor in order to reach main goals of creating effective transport management and increased mobility. Further, for many collaborative projects, it is important to choose a representative of a group for specific tasks. An example here could be choosing a representative to go to supraregional meetings, or selecting someone for usage of a telecommunication channel with a fixed capacity which many users want to access. In such a setting, users can give nominations between themselves. If we want to select a given number

of users in an impartial way, which means that no user can nominate strategically instead of truthfully to influence her own chance of being selected, previous work has shown that always selecting the users with the highest number of nominations is no option. We thus again want to approximate an optimal solution, here by sacrificing optimality in order to gain impartiality. We gained two major insights: first, relaxing exactness is beneficial. This means that sometimes but not always being able to select less than the desired number of users can actually pose an advantage for the mechanism and lead to better approximation factors. In fact, relaxing exactness is the only way to achieve any positive result with deterministic mechanisms. And second, using randomized instead of deterministic mechanisms additionally improves the approximation factor which we were able to gain. This might in practice also be of relevance as randomized mechanisms in a group of people tend to be perceived as more fair, irrelevant of actual fairness. Our results also suggest that our algorithms may well work better on real world data with roughly regular nomination structures, as observed by Aziz et al. [ALM+16]. Further, we developed as part of one of our mechanisms a subroutine which solves the problem of apportionment. Apportionment is the problem of distributing a number of objects, e. g. often seats in politics, in proportion to shares of a number of groups, e. g. political parties. This problem arises, e. g. , when assembling committees in the German Bundestag. Many apportionment rules rely on rounding and we introduce a mechanism which similarly always allocates shares that could be achieved by rounding proper shares up or down. But in addition, our algorithm picks one discrete distribution in such a way that the expected allocated number of seats for each group correspond to their (rational) proportion of total seats. We believe that this result can lead to fair solutions in many practical contexts where apportionment arises, whether, e. g. , political parties, or university committees are concerned.

One of the main issues when comparing our model to practice is that coalitions are not considered. While a user can not change her own probability of winning, she could form a treaty with another user and thus influence both their winning chances. This is studied within the field of cooperative game theory and leads to possibly completely different mechanisms. Moreover, looking at our results, it seems desirable, especially for large k, to find mechanisms with a better approximation ratio as this will make arguing for the more widespread usage of impartial mechanisms easier. Further work seems necessary for discovering more universally impartial mechanisms, as our results suggest that they may deliver higher outcomes, even though they seem to exhibit some unintuitive characteristics.

As becomes apparent when looking at traffic networks which are used by different and always new users at different times of the day, some problems in networks are not static, but have a lot of flexibility, with possibly sudden changes happening to the input, e. g. destinations. This makes finding optimal solutions very difficult. Thus, we once more decided to concentrate on approximating the optimal, here due to not knowing the future, and hence by means of online algorithms. We investigated online problems for carpools, as they are used by a number of different people which have different schedules and daily routines, e. g. commuters from San Francisco to Mountain View. Ideally, carpools provide flexibility, time sensitive travel

and easy access to users and are positive for personal reasons such as cost and stress factors, as well as societal and environmental reasons, especially concerning GHG emissions and sustainability. We formulated this problem in mathematical terms as online dial-a-ride where a server has to pick up requests and deliver them to their destination. Here, we try to reduce the total time until the server is finished. We developed a competitive algorithm which works on the line and in euclidean space and can thus be used for a variety of real world applications, as they often tend to be able to be modelled within (two- or three-dimensional) Euclidean space. Further, one main focus was dial-a-ride on the line, as this translates, e. g. , to the elevator problem, many other problems especially in industrial settings such as transport in fabrics, and carpools with designated pick up areas such as Market Street in San Francisco which is central and easy to reach from most places, to pick up people working in Mountain View. Our insights suggest that having to finish one request once it is picked up lead to online algorithms which give worse results than those for the case where instead, requests can be dropped off and picked up again at a later date. In an industrial setting, the second option, preemptiveness, is already a reasonable assumption. Due to better results, it seems desirable to favor it even for transporting people. Further, we could show that one can construct instances where a server has to turn around an arbitrary number of times for an optimal solution, even on the line, even if requests only have to be visited instead of being delivered. This leads to questions whether it might be better sometimes to not go for an optimal solution, which may finish earlier but uses a lot of electricity and resources doing that, or instead to go for a solution which is slightly slower but much more sustainable and economical. Using our insights, we constructed a dynamical program which solves offline dial-a-ride with high-five requests on the line in quadratic time. As such problems, known as travelling salesperson problems, appear often in applications, e. g. to deliver packages or pizza or pick up people with the same destination, the result is also interesting in its own right.

Our model does not take into consideration pick up or drop off times, which may lead to different results in practice. Also, while it is conceivable that a situation where requests have to be served on a line arises, it is not in general the case. Further, one of the limitations, as is often the case, lies in that one has to choose which goal one wants to concentrate on reaching. Here, we decided on reducing the makespan, i. e. the total time until all jobs are completed. Maybe, for the sake of reducing GHG emissions, one might choose to reduce the distance traveled. Another reasonable assumption would be that people actually give more information to the algorithm. Especially if it is for commuting purposes, people tend to arrive within a specified timeframe which may lead to stronger results.

To sum the results up, we have selected three problems to concentrate on reducing congestion in networks, each of them a different flavor of creating approximations to optimal solutions. One with better routing management, provided by a local search algorithm. One by reducing the total number of actors in the network, introducing impartial selection mechanisms. And one combining better routing management and reducing the number of actors by providing a framework for carpools with the online dial-a-ride problem.

One of the main issues we encountered was that the simplification and concentration on important points in the mathematical model does not always provide an adequate picture of reality. In practice, there are often many more requirements to consider. Another problem is the extent and magnitude of the problem. It is not evident which subproblems should be chosen that have significant influence on the overall network flow.

Especially considering the complexity of the task, it seems reasonable to look for solutions which approach optimality, as we do in this work, rather than looking for optimal solutions for a limited model. For our approach offers solutions which are less vulnerable to changes, which are foreseeable when transferring theoretical results into applications. Also, as our proved approximation guarantees are due to worst case scenarios, the simple and straightforward algorithms and mechanisms may perform better on real world data.

All three models suggested in the chapters of this thesis present a valid approach to reduce congestion in networks. The problems have been chosen such that the overall network flow will be reduced. Both better routing management by the local search algorithm in chapter one, and reducing the total number of actors with the impartial mechanism in chapter two, as well as a combination of those with the online dial-a-ride in chapter three have been addressed and thus main vantage points explored. Due to the complexity of the problem, concentration on selected subproblems was necessary, trying to keep problems simple, yet close to reality. Looking at more concrete problems, i. e. a certain city with fixed streets, telecommunications networks in one area, electric grid for one country, it seems probable that more specific tasks can be addressed as our approach was very general. While specific tasks may demand solutions that go beyond the general approach offered here, these results provide good starting points for further research in a wide area.

Bibliography

[AAE+08] Baruch Awerbuch, Yossi Azar, Amir Epstein, Vahab S. Mirrokni, and Alexander Skopalik, *Fast convergence to nearly optimal solutions in potential games*, Proc. 9th ACM Conf. Electron. Commerce, 2008, pp. 264–273. 16, 19, 20, 21, 40

[AAE13] Baruch Awerbuch, Yossi Azar, and Amir Epstein, *The price of routing unsplittable flow*, SIAM J. Comput. **42** (2013), no. 1, 160–177. 19

[AAZZ12] Matthew Andrews, Antonio F. Anta, Lisa Zhang, and Wenbo Zhao, *Routing for power minimization in the speed scaling model*, IEEE/ACM Trans. Netw. **20** (2012), 285–294. 15, 18

[ACP+86] Foto Afrati, Stavros Cosmadakis, Christos H. Papadimitriou, George Papageorgiou, and Nadia Papakostantinou, *The complexity of the travelling repairman problem*, Inform. Théor. Appl. **20** (1986), no. 1, 79–87. 93

[ADG+11] Sebastian Aland, Dominic Dumrauf, Martin Gairing, Burkhard Monien, and Florian Schoppmann, *Exact price of anarchy for polynomial congestion games*, SIAM J. Comput. **40** (2011), no. 5, 1211–1233. 16, 19

[AFL+94] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo, *Serving requests with on-line routing*, Proc. 4th Scandinavian Symp. and Workshop on Algorithm Theory, 1994, pp. 37–48. 90, 110

[AFL+95] _____, *Competitive algorithms for the on-line traveling salesman*, Proc. 4th Int. Workshop on Algorithms and Data Structures, 1995, pp. 206–217. 90, 110

[AFL+01] _____, *Algorithms for the on-line travelling salesman*, Algorithmica **29** (2001), no. 4, 560–581. 90, 92, 94, 109, 110

[AFPT11] Noga Alon, Felix A. Fischer, Ariel D. Procaccia, and Moshe Tennenholtz, *Sum of us: Strategyproof selection from the selectors*, Proc. 13th Conf. Theoret. Aspects Rationality and Knowledge, 2011, pp. 101–110. 6, 43, 44, 46, 47, 51, 85

[AK88] Mikhail J. Atallah and S. Rao Kosaraju, *Efficient solutions to some transportation problems with applications to minimizing robot arm travel*, SIAM J. Comput. **17** (1988), no. 5, 849–869. 93, 94, 108

[AKR00] Norbert Ascheuer, Sven O. Krumke, and Jörg Rambau, *Online Dial-a-Ride problems: Minimizing the completion time*, Proc. 17th Annu. Sympos. Theoret. Aspects Comput. Sci., 2000, pp. 639–650. 90, 92, 94, 100, 110

[Alb10] Susanne Albers, *Energy-efficient algorithms*, Commun. ACM **53** (2010), 86–96. 15

[ALM+16] Haris Aziz, Omer Lev, Nicholas Mattei, Jeffrey S. Rosenschein, and Toby Walsh, *Strategyproof peer selection: Mechanisms, analyses, and experiments*, Proc. 30th Nat. Conf. Artificial Intell., 2016, pp. 390–396. 46, 68, 75, 76, 77, 114

[ARV08] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking, *On the impact of combinatorial structure on congestion games*, J. ACM **55** (2008), no. 6, 1–22. 16, 19

[BBK+94] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson, *On the power of randomization in on-line algorithms*, Algorithmica **11** (1994), no. 1, 2–14. 11

[BD07] Liad Blumrosen and Shahar Dobzinski, *Welfare maximization in congestion games*, IEEE J. Sel. Areas Commun. **25** (2007), no. 6, 1224–1236. 20

[BDH+17] Antje Bjelde, Yann Disser, Jan Hackfeld, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Kevin Schewior, Miriam Schlöter, and Leen Stougie, *Tight bounds for online TSP on the line*, Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms, 2017, doi: 10.1137/1.9781611974782.63, pp. 994–1005. 87, 103, 104, 109, 110

[BEY98] Allan Borodin and Ran El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, New York, NY, USA, 1998. 12

[BFK15] Antje Bjelde, Felix A. Fischer, and Max Klimm, *Impartial selection and the power of up to two choices*, Proc. 11th Int. Conf. Web and Internet Econ., 2015, doi: 10.1007/978-3-662-48995-6-11, pp. 146–158. 42

[BFK17] ———, *Impartial selection and the power of up to two choices*, ACM Trans. Economics and Comput. **5** (2017), no. 4, 21:1–21:20, doi: 10.1145/3107922. 42

[Bir46] George D. Birkhoff, *Tres observaciones sobre el algebra lineal*, Rev. Fac. Cienc. Exact. Puras y Aplic. Univ. Nac. Tucumán Rev. Ser. A **5** (1946), 147–151. 45

[Bir76] Garrett Birkhoff, *House monotone apportionment schemes*, Proc. Nat. Acad. Sci. U.S.A. **73** (1976), no. 3, 684–686. 75

[BKdPS01] Michiel Blom, Sven O. Krumke, Willem de Paepe, and Leen Stougie, *The online TSP against fair adversaries*, INFORMS J. Comput. **13** (2001), no. 2, 138–148. 90, 92

[BKS17] Antje Bjelde, Max Klimm, and Daniel Schmand, *Brief announcement: Approximation algorithms for unsplittable resource allocation problems with diseconomies of scale*, Proc. 29th ACM Symp. Parallelism Algorithmics and Architecture, 2017, doi: 10.1145/3087556.3087597, pp. 227–229. 13, 20, 21, 39, 40

[BNV14] Nicolas Bousquet, Sergey Norin, and Adrian Vetta, *A near-optimal mechanism for impartial selection*, Proc. 10th Int. Conf. Web and Internet Econ., 2014, pp. 133–146. 85

[BT10] Daniel Berend and Tamir Tassa, *Improved bounds on Bell numbers and on moments of sums of random variables*, Probab. Math. Statist. **30** (2010), 185–205. 15

[Car65] Lewis Carroll, *Alice's Adventures in Wonderland*, Macmillan, New York, NY, USA, 1865. 1, 112

[CBJ+00] Peter Cox, Richard A. Betts, Chris Jones, Steven A. Spall, and Ian J. Totterdell, *Acceleration of global warming due to carbon-cycle feedbacks in a coupled model*, Nature **408** (2000), 184–187. 1

[CDKKS11] Po-An Chen, Bart De Keijzer, David Kempe, and Guido Schäfer, *The robust price of anarchy of altruistic games*, Proc. 7th Int. Conf. Web and Internet Econ., 2011, pp. 383–390. 19, 22, 24, 33

[CFRF+16] Ioannis Caragiannis, Aris Filos-Ratsikas, Soren K.S. Frederiksen, Kristoffer A. Hansen, and Zihan Tan, *Truthful facility assignment with resource augmentation: An exact analysis of serial dictatorship*, Proc. 12th Int. Conf. Web and Internet Econ., 2016, pp. 236–259. 46, 51

[CGH+96] Robert M. Corless, Gaston H. Gonnet, David E.G. Hare, David J. Jeffrey, and Donald E. Knuth, *On the Lambert W function*, Adv. Comput. Math. **5** (1996), no. 1, 329–359. 30

[CK05] George Christodoulou and Elias Koutsoupias, *The price of anarchy of finite congestion games*, Proc. 37th Annu. ACM Symp. Theory Computing, 2005, pp. 67–73. 19

[CKK+10] Ioannis Caragiannis, Christos Kaklamanis, Panagiotis Kanellopoulos, Maria Kyropoulou, and Evi Papaioannou, *The impact of altruism on the efficiency of atomic congestion games*, Proc. Int. Symp. Trustworthy Global Comput., 2010, pp. 172–188. 19

[CR98] Moses Charikar and Balaji Raghavachari, *The finite capacity dial-a-ride problem*, Proc. 39th Annu. IEEE Symp. Found. Comput. Sci., 1998, pp. 458–467. 93

[dCMT08] Geoffroy de Clippel, Herve Moulin, and Nicolaus Tideman, *Impartial division of a dollar*, J. Econom. Theory **139** (2008), no. 1, 176–191. 45, 46, 68

[Dep15] Department for Environment, Food and Rural Affairs, *Environmental reporting guidelines: including mandatory greenhouse gas emissions reporting guidance*, Tech. report, UK Government, London, UK, 2015. 2

[DP84] Narsingh Deo and Chi-Yin Pang, *Shortest-path algorithms: Taxonomy and annotation*, Networks **14** (1984), no. 2, 275–323. 18

[dPLS+04] Willem E. de Paepe, Jan Karel Lenstra, Jiri Sgall, René A. Sitters, and Leen Stougie, *Computer-aided complexity classification of Dial-a-Ride problems*, INFORMS J. Comput. **16** (2004), no. 2, 120–132. 93, 102, 104, 108

[dS12] Bart de Keijzer and Guido Schäfer, *Finding social optima in congestion games with positive externalities*, Proc. 20th Annu. European Symp. Algorithms, 2012, pp. 395–406. 20

[Erd47] Paul Erdős, *Some remarks on the theory of graphs*, Bull. Amer. Math. Soc. **53** (1947), no. 4, 292–294. 36

[Eur11] European Commission for Mobility and Transport, *White paper roadmap to a single european transport area: Towards a competitive and resource efficient transport system*, Tech. Report EUR-Lex - 52011DC0144 - EN, EUR-Lex, Brussels, Belgium, 2011. 1

[Eur16] European Environment Agency, *Transitions towards a more sustainable mobility system*, Tech. Report EEA Report No 34/2016, European Union, Copenhagen, Denmark, 2016. 1

[Eur17] European Commission, *EN Horizon 2020 work programme 2018-2020*, Tech. Report C(2017)7124, European Union, Brussels, Belgium, 2017. 42

[FBM+08] Jan Fuglestved, Terje Berntsen, Gunnar Myhre, Kristin Rypdal, and Ragnhild B. Skeie, *Climate forcing from the transport sectors*, Proc. Nat. Acad. Sci. U.S.A. **105** (2008), 454–458. 1

[Fed16] Federal Ministry of Transport and Digital Infrastructure, *The 2030 federal transport infrastructure plan*, Tech. report, Federal Ministry of Transport and Digital Infrastructure, Division Z 32, Berlin, Germany, 2016. 1

[FG93] Greg N. Frederickson and Dih Jiun Guan, *Nonpreemptive ensemble motion planning on a tree*, J. Algorithms **15** (1993), no. 1, 29–60. 93

[FK15] Felix A. Fischer and Max Klimm, *Optimal impartial selection*, SIAM J. Comput. **44** (2015), no. 5, 1263–1285. 44, 45, 46, 49

[FPT04] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar, *The complexity of pure Nash equilibria*, Proc. 36th Annu. ACM Symp. Theory Computing, 2004, pp. 604–612. 19

[FS01] Esteban Feuerstein and Leen Stougie, *On-line single-server Dial-a-Ride problems*, Theoret. Comput. Sci. **268** (2001), no. 1, 91–105. 90, 92

[FW98] Amos Fiat and Gerhard J. Woeginger (eds.), *Online Algorithms, the State of the Art*, Lect. Notes Comp. Sci., vol. 1442, Springer, Heidelberg, Germany, 1998. 12

[GFK+07] Rudolf Giffinger, Christian Fertner, Hans Kramar, Robert Kalasek, Nataša Milanović, and Evert Meijers, *Smart cities - ranking of european medium-sized cities*, Tech. report, Vienna University of Technology, 2007. 42

[GG64] Paul C. Gilmore and Ralph E. Gomory, *Sequencing a one state-variable machine: A solvable case of the traveling salesman problem*, Oper. Res. **12** (1964), no. 5, 655–679. 94

[GJ79] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, USA, 1979. 102

[GK94] Michael D. Grigoriadis and Leonid D. Khachiyan, *Fast approximation schemes for convex programs with many blocks and coupling constraints*, SIAM J. Optim. **4** (1994), 86–107. 20

[GK07] Naveen Garg and Jochen Könemann, *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*, SIAM J. Comput. **37** (2007), 630–652. 20

[GNR15] Inge Li Gørtz, Viswanath Nagarajan, and R. Ravi, *Minimum makespan multi-vehicle dial-a-ride*, ACM Trans. Algorithms **11** (2015), no. 3, 23:1–23:29. 93

[Gua98] Dih Jiun Guan, *Routing a vehicle of capacity greater than one*, Discrete Appl. Math. **81** (1998), no. 1-3, 41–57. 92, 93, 94, 108

[Hås01] Johan Håstad, *Some optimal inapproximability results*, J. ACM **48** (2001), no. 4, 798–859. 36, 38

[HK12] Tobias Harks and Max Klimm, *On the existence of pure Nash equilibria in weighted congestion games*, Math. Oper. Res. **37** (2012), no. 3, 419–436. 16

[HKM11] Tobias Harks, Max Klimm, and Rolf H. Möhring, *Characterizing the existence of potential functions in weighted congestion games*, Theory Comput. Syst. **49** (2011), no. 1, 46–70. 16

[HKP14] Tobias Harks, Max Klimm, and Britta Peis, *Resource competition on integral polymatroids*, Proc. 10th Int. Conf. Web and Internet Econ., 2014, pp. 189–202. 20

[HM13] Ron Holzman and Hervé Moulin, *Impartial nominations for a prize*, Econometrica **81** (2013), no. 1, 173–196. 43, 46, 78

[Hoc97] Dorit S. Hochbaum (ed.), *Approximation Algorithms for NP-hard Problems*, PWS Publishing Co., Boston, MA, USA, 1997. 8

[HOV16] Tobias Harks, Tim Oosterwijk, and Tjark Vredeveld, *A logarithmic approximation for polymatroid congestion games*, Oper. Res. Lett. **44** (2016), 712–717. 20

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis, *How easy is local search?*, J. Comput. System Sci. **37** (1988), 79–100. 19

[JW08] Patrick Jaillet and Michael R. Wagner, *Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses*, Oper. Res. **56** (2008), no. 3, 745–757. 90, 92

[JZ08] Klaus Jansen and Hu Zhang, *Approximation algorithms for general packing problems and their application to the multicast congestion problem*, Math. Program. **114** (2008), 183–206. 20

[Kar72] Richard M. Karp, *Two combinatorial problems associated with external sorting*, Combin. Algorithms, Courant Comp. Sci. Symp., 1972, pp. 17–29. 104

[KdPPS03] Sven O. Krumke, Willem E. de Paepe, Diana Poensgen, and Leen Stougie, *News from the online traveling repairman*, Theoret. Comput. Sci. **295** (2003), no. 1-3, 279–294. 90, 92

[Kho02] Subhash Khot, *On the power of unique 2-prover 1-round games*, Proc. 34th Annu. ACM Symp. Theory Computing, 2002, pp. 767–775. 38, 39, 41

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell, *Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?*, SIAM J. Comput. **37** (2007), no. 1, 319–357. 38, 39

[KLL+02] Sven O. Krumke, Luigi Laura, Maarten Lipmann, Alberto Marchetti-Spaccamela, Willem de Paepe, Diana Poensgen, and Leen Stougie, *Non-abusiveness helps: An O(1)-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem*, Proc. 5th Int. Workshop Approx. Algorithms Combin. Optim. Probl., 2002, pp. 200–214. 90, 92, 93

[KLMP15] David Kurokawa, Omer Lev, Jamie Morgenstern, and Ariel D. Procaccia, *Impartial peer review*, Proc. 24th Int. Joint. Conf. Artificial Intell., 2015, pp. 582–588. 46

[KMRS88] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator, *Competitive snoopy caching*, Algorithmica **3** (1988), no. 1, 79–119. 3

[Kru01] Sven O. Krumke, *Online Optimization Competitive Analysis and Beyond*, 2001, Habilitation thesis. 90, 92, 94, 100, 110, 111

[LLRKS85] Eugene L. Lawler, Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and David B. Shmoys (eds.), *The Traveling Salesman Problem; A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, UK, 1985. 92

[Mac15] Andrew Mackenzie, *Symmetry and impartial lotteries*, Games Econom. Behav. **94** (2015), 15–28. 46

[Mac17] ———, *A game of the throne of Saint Peter*, 2017, Working paper. 46

[Mil96] Igal Milchtaich, *Congestion games with player-specific payoff functions*, Games Econom. Behav. **13** (1996), no. 1, 111–124. 20

[MMS90] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator, *Competitive algorithms for server problems*, J. Algorithms **11** (1990), no. 2, 208–230. 89

[MRS01] Michael Mitzenmacher, Andrea W. Richa, and Ramesh Sitaraman, *The power of two random choices: A survey of techniques and results*, Handbook of Randomized Computing (S. Rajasekaran, P.M. Pardalos, J.H. Reif, and J. Rolim, eds.), vol. 1, Springer, 2001, pp. 255–312. 46

[MRV11] Dirk Müller, Klaus Radke, and Jens Vygen, *Faster min–max resource sharing in theory and practice*, Math. Prog. Comp. **3** (2011), 1–35. 20

[MS12] Carol A. Meyers and Andreas S. Schulz, *The complexity of welfare maximization in congestion games*, Networks **59** (2012), no. 2, 252–260. 15, 18

[MS14] Konstantin Makarychev and Maxim Sviridenko, *Solving optimization problems with diseconomies of scale via decoupling*, Proc. 55th Annu. IEEE Symp. Found. Comput. Sci., 2014, pp. 571–580. 13, 15, 19, 21, 113

[MW13] Nicholas Mattei and Toby Walsh, *Preflib: A library of preference data* HTTP://PREFLIB.ORG, Proc. 3rd Int. Conf. Algorithmic Decision Theory, Lect. Notes in Artificial Intell., Springer, 2013, pp. 259–270. 77

[Nas50] John F. Nash, *Equilibrium points in n-person games*, Proc. Nat. Acad. Sci. U.S.A. **36** (1950), 48–49. 2

[Nat15] National Science Foundation, *FY 2014 report on the NSFs merit review process*, Tech. Report NSB-2015-14, National Science Foundation, 2015. 77

[Nob14] Nobel Media AB, *The prize in economic sciences 2007 - press release*, 2014, http://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2007/press.html, accessed 2017-10-06. 3

[NRTV07] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, New York, NY, USA, 2007. 11

[OPS04] James B. Orlin, Abraham P. Punnen, and Andreas S. Schulz, *Approximate local search in combinatorial optimization*, SIAM J. Comput. **33** (2004), no. 5, 1201–1214. 22, 23

[Par04] David C. Parkes, *On learnable mechanism design*, Collectives and the Design of Complex Systems (Kagan Tumer and David Wolpert, eds.), Springer, 2004, pp. 107–131. 11

[POP13] Lorena Pradenas, Boris Oportus, and Víctor Parada, *Mitigation of greenhouse gas emissions in vehicle routing problems with backhauling*, Expert Systems with Appl. **40** (2013), no. 8, 2985 – 2991. 42

[PSMK90] Harilaos N. Psaraftis, Marius M. Solomon, Thomas L. Magnanti, and Tai-Up Kim, *Routing and scheduling on a shoreline with release times*, Management Sci. **36** (1990), no. 2, 212–223. 94, 104, 105, 108, 110

[PT13] Ariel D. Procaccia and Moshe Tennenholtz, *Approximate mechanism design without money*, Trans. Econ. and Comput. **1** (2013), no. 4, Article 18. 46

[Puk14] Friedrich Pukelsheim, *Proportional Representation: Apportionment Methods and Their Applications*, Springer, Switzerland, 2014. 75

[Ros73] Robert W. Rosenthal, *A class of games possessing pure-strategy Nash equilibria*, Int. J. Game Theory **2** (1973), no. 1, 65–67. 9, 16

[Rou07] Tim Roughgarden, *Routing games*, Algorithmic Game Theory (Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, eds.), Cambridge University Press, Cambridge, UK, 2007, pp. 461–486. 9

[Rou09] ———, *Intrinsic robustness of the price of anarchy*, Proc. 41st Annu. ACM Symp. Theory Computing, 2009, pp. 513–522. 19, 22, 41

[Rou14] ———, *Barriers to near-optimal equilibria*, Proc. 55th Annu. IEEE Symp. Found. Comput. Sci., 2014, pp. 71–80. 18, 35

[RTPH+03] Terry Root, Jeff T. Price, Kimberly Hall, Stephen H. Schneider, Cynthia Rosenzweig, and Alan Pounds, *Fingerprints of global warming on wild animals and plants*, Nature **421** (2003), 57–60. 1

[Sor16] Steffen Sorrell, *Worldwide smart sities: energy, transport and lighting 2016-2021*, Tech. report, Juniper Research, 2016. 42

[ST85] Daniel D. Sleator and Robert E. Tarjan, *Amortized efficiency of list update and paging rules*, Comm. ACM **28** (1985), no. 2, 202–208. 3, 12, 46

[SV08] Alexander Skopalik and Berthold Vöcking, *Inapproximability of pure Nash equilibria*, Proc. 40th Annu. ACM Symp. Theory Computing, 2008, pp. 355–364. 19

[Tam16] Shohei Tamura, *Characterizing minimal impartial rules for awarding prizes*, Games Econom. Behav. **95** (2016), 41–46. 46

[TO14] Shohei Tamura and Shinji Ohseto, *Impartial nomination correspondences*, Soc. Choice Welf. **43** (2014), no. 1, 47–54. 46

[Tsi92] John N. Tsitsiklis, *Special cases of traveling salesman and repairman problems with time windows*, Networks **22** (1992), no. 3, 263–282. 93, 104, 105, 108

[TSSW00] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson, *Gadgets, approximation, and linear programming*, Proc. 41th Annu. IEEE Symp. Found. Comput. Sci., 2000, pp. 617–626. 36

[Uni16] United Nations Department of Economic and Social Affairs, *Population division (2016) - the world's cities in 2016*, Tech. Report Data Booklet ST/ESA/SER.A/392, United Nations, 2016. 113

[US 64] US Bureau of Public Roads, *Traffic assignment manual for application with a large, high speed computer*, Tech. report, U.S. Department of Commerce, Urban Planning Division, For sale by the Superintendent of Documents, U.S. Govt. Print. Off., Washington D.C., USA, 1964. 14, 15, 40

[US 16] US Department of Transportation, *Smart Cities Challenge: lessons for building cities of the future*, Tech. report, US Government, 2016. 42

[Vaz01] Vijay V. Vazirani, *Approximation Algorithms*, Springer New York, Inc., New York, NY, USA, 2001. 8

[War52] John G. Wardrop, *Some theoretical aspects of road traffic research*, Proc. Inst. Civil Engineers **1** (1952), no. 3, 325–362. 2

[WBKP13] Jens Witkowski, Yoram Bachrach, Peter Key, and David C. Parkes, *Dwelling on the negative: Incentivizing effort in peer prediction*, Proc. 1st AAAI Conf. on Human Comput. and Crowdsourcing, 2013, pp. 190–197. 46

[WP12] Jens Witkowski and David C. Parkes, *Peer prediction without a common prior*, Proc. 13th Conf. Econ. Comp., 2012, pp. 964–981. 46

[WS11] David P. Williamson and David B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, Cambridge, UK, 2011. 8

[You94] H. Peyton Young, *Equity: In Theory and Practice*, Princeton University Press, Princeton, NJ, USA, 1994. 75