Fairness in Communication and Computer Network Design

Nilsson, Pål

2006

Fairness in communication and computer network design

Pål Nilsson

LUND
UNIVERSITY

Department of Communication Systems
Lund Institute of Technology

ISSN 1101-3931
ISRN LUTEDX/TETS–1080–SE+138P

Printed in Sweden
Tryckeriet i E-huset
Lund 2006

To my family

This thesis is submitted to Research Board FIME - Physics, Informatics, Mathematics and Electrical Engineering - at Lund Institute of Technology, Lund University in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering.

Contact information:
Pål Nilsson
Department of Communication Systems
Lund University
Box 118
SE-221 00 LUND
Sweden

Phone: +46 46 222 03 68
Fax: +46 46 14 58 23
e-mail: paln@telecom.lth.se

Abstract

In communication networks, fair sharing of resources is an important issue for one main reason. The growth of network capacity is in general not matching the rapid growth of traffic. Consequently, the resources consumed by each user have to be limited. This implies that users cannot always be assigned the end-to-end bandwidth they ask for. Instead, the limited network resources should be distributed to users in a way that assures fair end-to-end bandwidth assignment among them.

Obtaining fairness between network users and at the same time assuring efficient network utilization, is a source of non-trivial network optimization problems. Complicating factors are that each user has limited access to the (limited) network resources and that different users require and consume different amounts and types of resources.

In this thesis different types of optimization problems associated with fair resource sharing in communication networks are studied. Initially, the notions of max-min fairness, proportional fairness, α-fairness etc., are put in a formal framework of fair rational preference relations. A clear, unified definition of fairness is presented.

The theory is first applied to different types of allocation problems. Focus is put on convex and non-convex max-min fair traffic allocation problems, and a difference in difficulty between the two groups of problems is demonstrated.

The studies are continued by an investigation of proportionally fair dimensioning. Two different cases are studied – a simpler, when no resilience to failures is required, and a more complicated, assuming the possibility of link failures.

In the context of fair sharing of the resources of a communication network, this thesis presents several original theoretical findings as well as solution algorithms for the studied problems. The results are accompanied by numerical results, illustrating algorithm efficiency for virtually all of the studied problems.

Acknowledgements

My gratitude goes out to all the people who did, in any sense, contribute to the creation of this thesis. Thanks to my precious family – Rebecka, Alfred and Lovisa, and to Mum and Dad who did always support me. Many thanks to my supervisor Michal Pioro and to my colleague Eligijus Kubilinskas. I am also grateful to Ulf Ahlfors, Jens K Andersson, Mikael Andersson, Jianhua Cao, Mateusz Dzida, Zbigniew Dziong, Gabor Fodor, Torgny Holmberg, Johan M Karlsson, Ulf Körner, Christian Nyberg, Wlodzimierz Ogryczak, Henrik Persson, Lars Reneby and Michal Zagozdzon, who did all help me in different aspects during my Ph.D. studies. Long live

Contents

Chapter 1

Introduction

1.1 Background

The recent emergence of new, sophisticated Internet services, that lay claim to an increasing amount of communication resources, have made the demand on communication networks grow considerably. It has been observed that the new data-oriented services exhibit a much higher growth rate, both in terms of number of users and in terms of induced traffic per user, than traditional voice services.

Internet, being the network of communication networks, is known to be of a best-effort nature, where each service is completed to the level of quality that transmission media (links, switches, routers etc.) can provide. This implies that some applications (using a set of services) work well, while others fail totally (or partially). The most common problem, when a certain application fails, is lack of transmission bandwidth. If it were entirely known which services are going to make use of the network resources in terms of duration, location, and required capacity, the resources could of course be adapted accordingly. In practice, this is however very rarely the case. In fact, it is characteristic for the traffic of data networks that its distribution changes rapidly, both over short and long time periods, and is for this reason very difficult to predict. This statement can be argued more valid the smaller is the scale of the network being considered, as the more aggregation of sources, the more stable is the aggregated traffic. Nevertheless, lack of sufficiently accurate predictions sometimes make resource provisioning through overdimensioning the only tractable way of obtaining satisfactory network operation. The overdimensioning strategy consists in, without any deeper thought, just to add capacity sufficient to handle the most extreme traffic situations. It is clear that overdimensioning is economically questionable, since the strategy is intimately associated with poor resource utilization. On the other hand, if the network does not match the maximal traffic demand generated by the users, the network will be subject to overloads. Overloads have the negative effect of deteriorating the quality of service perceived by the users – the data transfer rates decrease because individual packets are delayed or even lost.

Besides ensuring high connection acceptance rate, one of the most important objectives for a network operator is to minimize the network overload. This is because successful network management relies on continuos provisioning of an acceptable quality of service to users. Instead of relying on the so-called best-effort communication, the guaranteed quality of service can be obtained by controlling the amount of traffic that enters the network. The control process, usually referred to as traffic admission control, decides how much traffic can be generated by each user and how many users can be simultaneously served. In order for the traffic admission control to reduce the aggregated stream of traffic that enters the network, a key mechanism is that of partial or entire denial of service. On the other hand, low probability

of service denial is another measure for the quality of service perceived by users. It is therefore reasonable to require that the traffic admission control imposes some fair sharing of network resources between its potential users. This observation holds true both for small and large-scale networks. For a small network a "user" may correspond to a single terminal, whereas in a large backbone network a "user" may be constituted by thousands of terminals connected to the backbone via a (smaller scaled) local network. The problem of determining how much traffic of each traffic stream should be admitted to the network, and how the admitted traffic should be routed so as to satisfy the requirements of high network utilization and fairness, is a challenging problem for the Internet solutions using admission control. This thesis addresses this type of problems.

1.2 Backbone networks

In the communications community, a *backbone network* (or simply a backbone), is a network that realizes the connections between local networks, as illustrated in Figure 1.1. From the backbone point of view, the local networks can be seen as edge networks, to which the subscribers (traffic sources) are connected. Consequently, the traffic can be regarded as generated in the local networks. Traffic from different users in a local network is concentrated and forwarded to more distant places by the backbone.

A similar process takes place at practically every level of the network hierarchy. For instance, the local networks are connected to a regional network, which in turn is connected to a national network, and so on. As the local networks induce the traffic, the traffic experienced by the backbone is an aggregated version of its local network counterparts. Considered over a sufficiently short time-period, the backbone network traffic is, compared to an access network, predictable and stable (and of course very large) in the sense of flow magnitudes. The hierarchical structure is present in traditional telecommunication networks, as well as in modern large-scale data communication networks. When connecting a large set of users, the hierarchical network topology arises because of geography, cost considerations, and different communities of interest.

Traditionally, a large portion of the traffic in a communication network is intended for local transmission. However, the pattern of traffic flows is in a changing process, as the cost of long-distance transmission is decreasing and the usage of distant servers increases, and accordingly, traffic load of the backbone is increasing. The reduced long-distance transmission costs are mainly due to the introduction of very efficient optical transmission techniques, along with the substantial improvement of the computer-controlled hardware, constituting network switches and routers.

To meet the need for transmission facilities that could handle optical

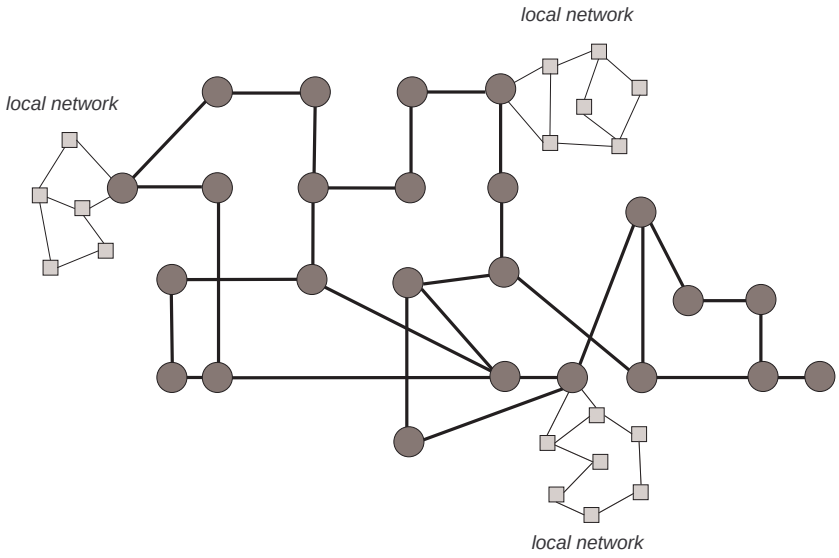


Figure 1.1: A backbone network interconnects the local networks (the depicted backbone is the swedish *GigaSunet*).

technology, the Synchronous Optical Network (SONET) standard was introduced by Bellcore in 1984 [22]. SONET is a standard for optical communication transport, providing framing as well as rate hierarchy and optical parameters. The SONET standard became adopted by American National Standards Institute (ANSI) in 1988 [22]. A slightly different version, the Synchronous Digital Hierarchy (SDH), was later adopted by the International Telecommunications Union - Telecommunication Standardisation Sector (ITU-T). Today, most high-speed digital backbone networks are SONET/SDH-based [7]. Both SONET and SDH define signal hierarchies, where the signals with the top rates are OC-192 (SONET) and STM-64 (SDH), both facilitating a data rate of 9953.28 Mb/s [13]. New techniques for optical transmission are however being rapidly developed. An example is the so-called Wave-Division Multiplexing (WDM) technology, which multiplies the transmission capacity of a SONET/SDH system by the number of wavelengths supported by the WDM system (there are for instance ITU standards defining 81 wavelengths [13]).

A backbone can be regarded as composed of a hierarchical set of distinguished resource layers. In today's networks it is common to run IP (Internet Protocol) over ATM (Asynchronous Transfer Mode), ATM over SDH, and

SDH over WDM. Apparently, to optimize operation of this type of multi-layer network, one has to take into account how the different layers interact. Clearly, what concerns the design of an entire backbone, no layer can be effectively treated as a stand-alone layer. However, the current trend is to simplify this architecture, in order to reduce network equipment and management costs, as well as network complexity. It is commonly foreseen that the service layers (the two packet layers, IP and ATM), will be integrated into one resource layer based on Multi-Path Label Switching (MPLS). This leads to a single packet layer control plane, instead of two. Furthermore, it is predicted that IP packets will most likely be transported directly over the WDM transport layer, enriched with a control plane (based on e.g. Generalized MPLS (GMPLS) and/or Automatically Switched Transport Network (ASTN)). Hence, the next generation backbone networks are most probable to be built as IP-over-WDM networks [3].

1.3 Backbone network traffic

Roughly, traffic generated in an edge network can be put in one of two categories, namely i) *elastic traffic*, which is a type of best-effort traffic that employs whatever resoruces it is assigned, and ii) *non-elastic traffic* which is traffic that is typically sensitive to assigned bandwidth and transmission delays [49]. The data communications community is well aware of the potential traffic property differences, as for instance the internet protocol (IPv6) features the so called flow-labelling function [8], which accomodates different Quality of Service (QoS) requirements. Examples of elastic traffic are classical Internet applications as web, file transfer protocol (FTP), and electronic mail. Real-time applications, as e.g., voice over IP (VoIP) and video-conferencing, are on the other hand typically non-elastic. Since the different types of traffic are generated in the edge networks, they are as indicated above, multiplexed before transmitted into the backbone network. Therefore, the aggregated backbone traffic flows consist of both elastic and non-elastic traffic, and an appropriate denomination would be *semi-elastic* traffic. This pushes forward the fact that the backbone traffic flows require a minimal bandwidth level (constituted by the sum of the required bandwidth for the non-elastic applications), but still are able consume bandwidth exceeding this level (due to its elastic portion).

1.4 The multicommodity flow model

A convenient way of representing a communication network is by the notion of a graph. A graph $\mathcal{G}$ is an object composed by a set of vertices $\mathcal{V}$, and a set of edges $\mathcal{E}$, compactly written as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Using the graph to model a network, we consider the network nodes as graph vertices, and the network

links, connecting the different nodes, as graph edges. In the sequel we will treat a given communication network as a graph with certain properties, and use the words links and nodes to denote its vertices and edges. Each link has a capacity that limits the commodity possible to transfer between its two associated nodes. In the general setting, a link facilitates transfer of various entities, i.e., it may carry several commodities simultaneously. This is the key feature of the so called *multicommodity flow model*, which will be exclusively used in this thesis. If a link enables the transfer of commodities in both directions it is called undirected, and otherwise directed. In general, we will treat all links as undirected.

The commodities transferred in a communication network are different types of data. Such a transfer of data will be called a flow (or path-flow to emphasize its path). Each flow is assumed to be induced by a demand, which is a request for traffic between a node-pair. A demand can realize its total flow on (potentially) several different paths (i.e., by several simultaneous path-flows) connecting the node-pair.

1.5 Subject of the thesis

The main topic of this thesis is a study of how a set of different demands (node-pairs) should share the network resources (link capacities) such that it is assured that this is done in a fair way. Criteria that accompany the primary fairness goal can be to keep a high network utilization and/or to obtain a solution that is resilient to link failures. The introduced problems are primarily intended to model large-scale networks.

1.6 Introductory examples

1.6.1 Fairness vs throughput

In order to establish some familiarity with the problems studied in this thesis, and also to motivate a deeper knowledge, we will investigate some very simple example problems.

Consider the simple 3-node network given in Figure 1.2. There are three demands, i.e., three node-pairs that require transmission capacity to be able to communicate, each with a single associated path. Demand 1 corresponds to node-pair $(1, 2)$, demand 2 to $(2, 3)$, and demand 3 to $(1, 3)$. Capacities of the two links are both equal to 1.5. We want to allocate portions of the link capacities to the demands, facilitating their flow transfer. Apparently, this may be done in several ways, depending on what objective that we have in mind. For a start, suppose that we want to maximize the total flow transfer, also called maximizing the network throughput. Let x_d be the flow allocated to demand d, $d = 1, 2, 3$. It is not hard to see that the throughput maximal

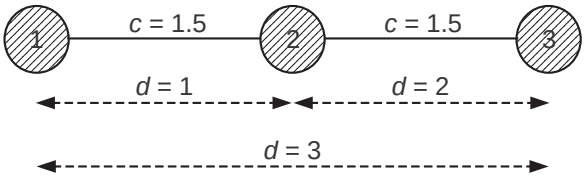


Figure 1.2: A simple linear network.

solution is to assign $x_1 = 1.5$, $x_2 = 1.5$, and $x_3 = 0$, giving a total flow throughput of $\sum_d x_d = 3$.

The objective of throughput maximization means that we do not care about the specific flow volume assigned to a demand, but rather that the total flow, allocated to all the demands, is made as large as possible. Clearly, this can be a desired property, especially from a network operator perspective. It is however evident that this objective treats the demands in an unfair way, as the third demand is allocated zero flow. We may instead take the opposite standpoint, and use as objective to share the two link capacities as fairly as possible, intiuitively meaning that the three demands get an equal share that is as large as possible. This results in a solution with $x_1 = 0.75$, $x_2 = 0.75$, and $x_3 = 0.75$, giving a total flow throughput of $\sum_d x_d = 2.25$. Thus, a maximally fair solution is obtained at the cost of reducing the total throughput.

A natural question to ask is if there is an easy-accessible compromise solution, i.e., a solution that is fairer than the one obtained by throughput maximization but has a better throughput than the maximally fair one. Indeed, such a soluton can be obtained by for instance maximizing the product of flows, $\prod_d x_d$. This yields a solution with $x_1 = 1$, $x_2 = 1$, and $x_3 = 0.5$, which is definitely more fair than the throughput maximal solution, and more throughput effective ($\sum_d x_d = 2.5$) than the maximally fair one.

To summarize, it is clear that in order to obtain a fair solution, one has to relax the requirement of a high total throughput and vice versa. We can also conclude that there is a lot of room for definitions of objectives that achieve more fairness than throughput maximization and more throughput-effective solutions than the maximally fair one.

1.6.2 The fairest solution

The notion of fairness is rather subjective, and we will therefore now try to give an intuitive understanding of what we mean by a solution that is as fair as possible. Consider the network given in Figure 1.3. There are 4 nodes connected by 4 links, and there are 6 demands for communication; demand 1 between nodes 1 and 2, demand 2 between nodes 1 and 3, demand 3 between nodes 2 and 3, demand 4 between nodes 3 and 4, demand 5 between nodes 3

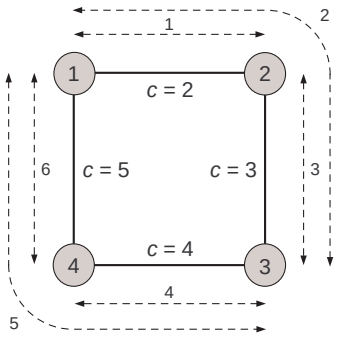


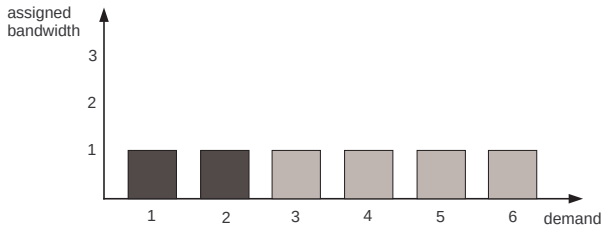
Figure 1.3: A simple square network.

and 1, and demand 6 between 4 and 1. Link capacities are 2 between nodes 1 and 2, 3 between 2 and 3, 4 between 3 and 4, and 5 between 4 and 1. Each demand has one predefined path, given by the figure. Suppose that we want to let the 6 demands share the link capacities as fairly as possible, i.e., that we want to obtain the fairest solution. We then start by assigning all the demands an equal, as large flow as possible. This flow is equal to 1. Demands 1 and 2 cannot be assigned a flow larger than 1 because the link between nodes 1 and 2 blocks them. Now the critical question is, should all the demands be assigned a flow of 1 as demands 1 and 2 cannot be assigned more? There is really no non-naive reason for this. As can be seen in Figure 1.4(a), although demands 1 and 2 cannot be further increased, because of the link between nodes 1 and 2 being saturated, there is still room to increase all the other demands. As depicted in Figure 1.4(b) demands 3, 4, 5, and 6 can be increased to a flow of 2. At this level, the links between nodes 2 and 3 and nodes 3 and 4 become saturated. Hence demands 3, 4, and 5 cannot be further increased. However, there is still unused capacity on the link connectiong nodes 1 and 4. We may thus finally assign a flow of 3 to demand 6, which results in that the final link becomes saturated, and we have reached what is often regarded as the fairest possible solution, called the *max-min fair* solution. The concept of max-min fairness will be formally introduced in the following chapter.

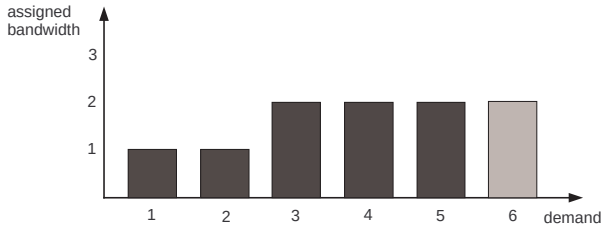
1.7 The framework

1.7.1 Mathematical programming

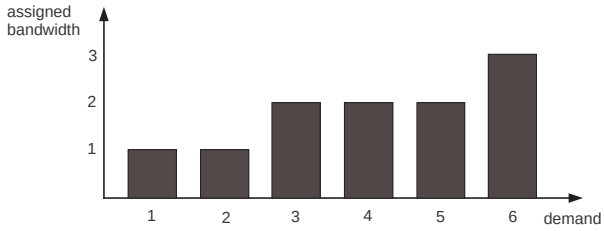
Most of the problems considered in this thesis will be formulated as constrained single-criteria optimization problems. Such problems are also referred to as mathematical programming problems. A problem of this type



(a) Link between nodes 1 and 2 saturated.



(b) Links between nodes 2 and 3 and nodes 3 and 4 saturated.



(c) Link between nodes 4 and 1 saturated.

Figure 1.4: Successively obtaining the fairest solution. Black bars indicate demands that cannot be increased more.

is usually written as

$$\begin{aligned} \max \quad & f(\boldsymbol{x}) && && (1.1)\\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq a_i && i = 1, 2, \ldots, k && (1.2)\\ & h_j(\boldsymbol{x}) = b_j && j = 1, 2, \ldots, l && (1.3)\\ & \boldsymbol{x} \in X, && && (1.4) \end{aligned}$$

where $f : X \to \mathbb{R}$ is called the objective function that should be maximized over a set of constraints determined by the real valued functions g_i and h_j and the set X. The constrained optimization problem thus specifies the task of finding an $\boldsymbol{x} \in X$ such that $g_i(\boldsymbol{x}) \leq a_i$ for all i, $i = 1, 2, \ldots, k$ and $h_j(\boldsymbol{x}) = b_j$ for all j, $j = 1, 2, \ldots, l$, and such that $f(\boldsymbol{x})$ is maximal. To simplify notation, we will use the "for all"-quantifier, $\forall$, when ranges for the constraints of an optimization problem are known. Using this way of writing, (1.1)-(1.4) can be put as

$$\begin{aligned} \max \quad & f(\boldsymbol{x}) && && (1.5)\\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq a_i && \forall\ i && (1.6)\\ & h_j(\boldsymbol{x}) = b_j && \forall\ j && (1.7)\\ & \boldsymbol{x} \in X. && && (1.8) \end{aligned}$$

A special type of constrained optimization problems are the linear programming problems. A linear programming problem (LP) is a mathematical programming problem where f, g_i, and h_j all are linear functions and $X = \mathbb{R}^n$. An integer programming problem (IP) is an LP, but with the set X restricted further: $X = \mathbb{Z}^n$. Finally, a mixed-integer programming problem (MIP) is an LP, but with the set X defined by $\boldsymbol{x} \in X$ if and only if $x_i \in \mathbb{R}$ for $i = 1, 2, \ldots, t$, and $x_t \in \mathbb{Z}$, for $i = t + 1, t + 2, \ldots, n$, for some positive integer t, $t < n$.

1.7.2 Notation

For all the network optimization problems considered in this thesis we will assume that the network graph is given. We will label the nodes of the network with index v, $v = 1, 2, \ldots, V$, and use e, $e = 1, 2, \ldots, E$, to label its links. Also assumed given will be the set of demands, labeled by index d, $d = 1, 2, \ldots, D$. For each demand d we will assume that there is given a set of paths labeled by p, $p = 1, 2, \ldots, P_d$. The link-path incidence relation will be expressed by the binary indicator δ_{edp}, which is 1 if link e belongs to path p of demand d, and 0 otherwise. Of course, it is possible that we have only one admissible path for each demand, i.e., $P_d = 1$ for all demands d; in this case we can substitute the link-path incidence coefficient δ_{edp} by δ_{ed}.

Example 1.1. Consider the network given in Figure 1.5. Nodes and links are numbered in the figure. Suppose that there is one demand between

nodes 1 and 3. There are 3 (simple) paths connecting this node-pair: path 1 constituted by links 1 and 2, path 2 constituted by link 3, and path 3 constituted by links 4 and 5. If we refer to the only demand as demand 1 we have $\delta_{111} = 1$, $\delta_{211} = 1$, $\delta_{312} = 1$, $\delta_{413} = 1$, and $\delta_{513} = 1$.

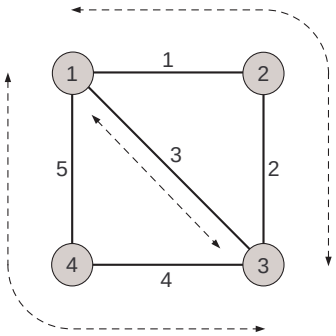


Figure 1.5: Three different paths between node-pair (1,3).

An optimization problem is always associated with a set of decision variables, i.e., a set of variables that should be chosen such that the solution is feasible and optimal. In the problems considered in this thesis, the decision variables are usually the flows, corresponding to variable z_{dp}, $d = 1, 2, \ldots, D$, $p = 1, 2, \ldots, P_d$, denoting flow allocated to demand d on path p. The total flow for demand d is given by $x_d = \sum_p z_{dp}$. If for demand d, z_{dp} is greater than zero for more than one path p, we say that the flow for demand d is *bifurcated*. On the other hand it is possible to require that for all demands d, that z_{dp} is greater than zero only for one path p, which defines an *unsplittable* flow problem. The total flows for all demands are usually collected in the so-called *allocation vector*, $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$. It is sometimes convenient to consider a sorted version of $\boldsymbol{x}$, written $\Theta(\boldsymbol{x})$, where Θ is the *ordering map*.

Definition 1.1. The ordering map Θ is the function $\Theta : \mathbb{R}^n \to \mathbb{R}^n$, such that $\Theta(\boldsymbol{x}) = (\theta_1(\boldsymbol{x}), \theta_2(\boldsymbol{x}), \ldots, \theta_n(\boldsymbol{x}))$, where $\theta_1(\boldsymbol{x}) \leq \theta_2(\boldsymbol{x}) \leq \ldots \leq \theta_n(\boldsymbol{x})$ and there exists a permutation $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ such that $\theta_i(\boldsymbol{x}) = x_{\pi(i)}$, for $i = 1, 2, \ldots, n$.

Network design problems considered in this thesis can be roughly divided into two categories – *allocation problems* and *dimensioning problems*. An allocation problem is a network optimization problem where link capacities are assumed to be given and the task is to allocate flows to demands. Such a problem is also referred to as a *capacitated problem*. In a dimensioning problem on the other hand, link capacities are treated as decision variables, together with the path flows. Dimensioning problems are also called *uncapacitated problems*.

Basic notation

$\Theta(\boldsymbol{x})$	the vector $\boldsymbol{x}$ sorted in non-decreasing order
$\theta_j(\boldsymbol{x})$	the j:th entry of $\Theta(\boldsymbol{x})$, $1 \leq j \leq n$
$\boldsymbol{x}[k]$	The subvector $(x_1, x_2, \ldots, x_k)$ of $\boldsymbol{x} = (x_1, x_2, \ldots, x_k, \ldots, x_n)$
$\boldsymbol{x} \geq \boldsymbol{x}'$	$x_i \geq x'_i$, $i = 1, 2, \ldots, n$
$\boldsymbol{x} \geqq \boldsymbol{x}'$	$x_i \geq x'_i$, $i = 1, 2, \ldots, n$, and $x_j > x'_j$ for at least one coordinate j, $1 \leq j \leq n$
$\boldsymbol{e}_j$	the identity vector, $e_j = 1$ and $e_i = 0,\ i = 1, 2, \ldots, n,\ i \neq j$
$\boldsymbol{u}$	the unit vector, $u_i = 1,\ i = 1, 2, \ldots, n$
$\mathbf{0}$	the zero vector $\mathbf{0} = (0, \ldots, 0)$
$\mathbb{R}_+$, $\mathbb{R}_{++}$	the non-negative / strictly positive real numbers
$\mathbb{Q}_+$, $\mathbb{Q}_{++}$	the non-negative / strictly positive rational numbers
$\mathbb{Z}_+$, $\mathbb{Z}_{++}$	the non-negative / strictly positive integral numbers

1.8 Thesis outline

The thesis is organized as follows. We start the main presentation in Chapter 2 by giving a formal introduction to the theory of fairness in resource sharing, which is fundamental for the problems considered in this thesis. This is done by defining the concept of fair rational preference relations, providing a unified perspective of fairness in general, and in communication networks in particular. In that chapter different types of fairness, such as max-min fairness, proportional fairness, α-norm fairness, etc., are viewed from the formal fair rational preference relation perspective. Moreover, as the existing literature is a bit ambiguous in the definition of the important max-min fairness notion, we classify the different definitions in terms of their strength. In the balance of Chapter 2, generic algorithms achieving max-min fairness are given.

In Chapter 3 we use the knowledge about max-min fairness described in Chapter 2 and apply it to allocation problems. Specifically, Chapter 3 deals with allocation problems that have convex structure, i.e., allocation problems for which the associated optimization problems are convex. This requirement excludes for instance the more complicated cases when decision variables can be discrete (integer variables, binary variables). Allocation problems are the network design problems that are characterized by the property that network topology as well as link capacities are given a priori. Such problems consist in determination of how a number of different (competing) users should share the network resources, so that some given criterion is met (e.g., an objective function is optimized). Thus, an allocation problem is a capacitated problem.

We proceed by considering max-min fairness allocation problems with non-convex sets of solutions, in Chapter 4. The studies of that chapter are thus devoted to the same type of max-min fair allocation problems as before,

but with complicating assumptions making the resulting optimization problems non-convex. Specifically, the problems considered in Chapter 4 do all include integer variables. More precisely, we are still considering problems that are classified under linear max-min fairness but with decision variables being restricted to integers. The reason for this is that in practice, decision variables, such as, e.g., flow allocated to a demand, cannot take on an arbitrary value. Another realistic requirement might be that a flow between a node-pair cannot be split arbitrarily between connecting paths, but can only use one path, selected in the optimization process. Also this makes it necessary to model the allocation problem with integer variables, rendering substantially more complicated optimization problems.

In Chapter 5 we deal with fairness in dimensioning problems. It is significant for a dimensioning problem that, as opposed to an allocation problem, the link capacities are assumed to be design variables. In the spirit of previous chapters, we consider dimensioniong problems for which fairness among demands is the preliminary objective. In this chapter, the type of fairness that is addressed is commonly known as proportional fairness, introduced in Chapter 2.

The final chapter is dedicated to a summary of the previously obtained results, along with a specification of which of the results that are contributions of this thesis.

1.9 List of publications

The contents of this thesis is based mainly but not exclusively upon the following publications.

1. P. Nilsson, M. Pioro, "Solving dimensioning tasks for proportionally fair networks carrying elastic traffic ", *Performance Evaluation*, 49(1-4):271–386, 2002.

2. P. Nilsson, M. Pioro, "Fair Bandwidth Allocation and Links' Dimensioning in Elastic Traffic Networks". In 16th Nordic Teletraffic Seminar (NTS 16), Espoo, Finland, pages 56–69, August 2002.

3. M. Pioro, P. Nilsson, E. Kubilinskas and G. Fodor, "On efficient maxmin fair routing algorithms". In IEEE International Symposium on Computers and Communications, pages 465–472, 2003.

4. P. Nilsson and M. Pioro, "Link Protection within an Existing Backbone Network". In International Network Optimization Conference, INOC'03, pages 435–440, 2003.

5. M. Pioro, E. Kubilinskas, P. Nilsson and M. Matuszewski, "Robust Dimensioning of Proportionally Fair Networks", *European Transactions on Telecommunications*, 13(1):241–251, 2005.

6. M.Pioro, M. Dzida, E. Kubilinskas, P. Nilsson, W. Ogryczak, A. Tomaszewski and Michal Zagozdzon, "Applications of the Max-Min Fairness Principle in Telecommunication Network Design". In First Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005), 2005.

7. P. Nilsson and M. Pioro, "Max-Min Fairness of Unsplittable Network Flows". Technical report, Dept. of Communication Sytems, Lund University, June 2005, CODEN: LUTEDX(TETS-7209)/1-21/(2005)&local 25.

8. P. Nilsson, "A transformation from INDEPENDENT SET to MM-SPF2". Technical report, Dept. of Communication Sytems, Lund University, June 2005, CODEN: LUTEDX(TETS-7210)/1-2/(2005)&local 18.

9. P. Nilsson and M. Pioro, "Unsplittable max-min demand allocation - a routing problem". In HETNETs'05, P26, 2005.

10. P. Nilsson, "Some simple special cases of FIXMMF-MF". Technical report, Dept. of Communication Sytems, Lund University", November 2005, CODEN: LUTEDX(TETS-7214)/1-3/(2005)&local 33".

11. P. Nilsson and M. Pioro, "Max-Min Fair Distribution of Modular Network Flows on Fixed Paths". In Networking'06 (Published in Springer LNCS), pages 916–927, 2006.

Specifically, Chapter 3 is based on publications 3, 4, and 6. Publications 7,8,9,10, and 11 cover the material presented in Chapter 4. Chapter 5 corresponds to publications 1,2, and 5.

Chapter 2

Optimization models for fairness

2.1 The notion of fairness

In order to define fairness of outcome vectors on some given set, we will adopt the terminology used by Ogryczak and Wierzbicki in [39].

2.1.1 Fair rational preference relations

Consider an m-dimensional set Q, $Q \subseteq \mathbb{R}^m$, on which there is defined a weak preference relation, $\succeq$. Such a weak preference states that if $\boldsymbol{y}', \boldsymbol{y}'' \in Q$, and $\boldsymbol{y}' \succeq \boldsymbol{y}''$ then the vector $\boldsymbol{y}'$ is at least as good as vector $\boldsymbol{y}''$. A weak preference will always be assumed to be complete, stating that it relates any two vectors of Q. This is in fact all that is needed to construct the notion of fairness. The relations of strict preference, $\succ$, and indifference, $\cong$, are defined as follows

$$\boldsymbol{y}' \succ \boldsymbol{y}'' \quad \Leftrightarrow \quad (\ \boldsymbol{y}' \succeq \boldsymbol{y}'' \quad \text{and} \quad \text{not} \quad \boldsymbol{y}'' \succeq \boldsymbol{y}'\ ) \tag{2.1}$$

$$\boldsymbol{y}' \cong \boldsymbol{y}'' \quad \Leftrightarrow \quad (\ \boldsymbol{y}' \succeq \boldsymbol{y}'' \quad \text{and} \quad \boldsymbol{y}'' \succeq \boldsymbol{y}'\ ) \tag{2.2}$$

We will use preference relations to define fairness on a superset, Q, of the set of outcome vectors, Y. The set Q will be assumed to have certain regularity:

Definition 2.1. A set $Q \subseteq \mathbb{R}^m$ is said to be *transferable* if for any $\boldsymbol{y} \in Q$, $\boldsymbol{y} - \epsilon \boldsymbol{e}_i + \epsilon \boldsymbol{e}_j \in Q$ for all i, j such that $y_i - y_j \geq \epsilon \geq 0$

Note that if a transferable set contains a vector $\boldsymbol{y}$, it must also contain all the permutations of $\boldsymbol{y}$.

Definition 2.2. A preference relation $\succeq$ defined on a transferable set Q is a *fair rational preference relation* (or *fair preference* in short) on Q if for any $\boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'', \boldsymbol{y}''' \in Q$ the following requirements are satisfied.

(*i*) *Transitivity*: $\boldsymbol{y}' \succeq \boldsymbol{y}''$ and $\boldsymbol{y}'' \succeq \boldsymbol{y}''' \Rightarrow \boldsymbol{y}' \succeq \boldsymbol{y}'''$.

(*ii*) *Strict monotonicity*: if $\boldsymbol{y}' = \boldsymbol{y} + \epsilon \boldsymbol{e}_i$ for some $\epsilon > 0$ and some $i = 1, 2, \ldots, m$, then $\boldsymbol{y}' \succ \boldsymbol{y}$.

(*iii*) *Symmetry*: for any permutation $\pi : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, m\}$, it holds that $(y_{\pi(1)}, y_{\pi(2)}, \ldots, y_{\pi(m)}) \cong (y_1, y_2, \ldots, y_m) = \boldsymbol{y}$.

(*iv*) *Equitable transfers*: if $y_i > y_j$ then $\boldsymbol{y} - \epsilon \boldsymbol{e}_i + \epsilon \boldsymbol{e}_j \succ \boldsymbol{y}$ for $0 < \epsilon < y_i - y_j$.

Remark 2.3. The requirement of symmetry implies that a fair preference is reflexive.

Remark 2.4. It is sometimes assumed that already a weak preference relation is transitive and reflexive, making it a preorder.

Remark 2.5. If $\succeq$ is a fair preference defined on the transferable set Q, it can also be regarded as a fair preference on any set $Y \subseteq Q$, by inheriting the fair preference defined on Q.

Example 2.1. Consider a preference relation, $\succeq_\times$, called the product order (or the *Nash product* [51]), defined by $\boldsymbol{y} \succeq_\times \boldsymbol{y}' \Leftrightarrow \prod_{i=1}^m y_i \geq \prod_{i=1}^m y_i'$. If we consider this preference relation on $\mathbb{R}^m_{++}$, it is obviously transitive, strictly monotonic, and symmetric. Let $\boldsymbol{y} \in \mathbb{R}^m_{++}$ be a vector that has entries i, j, for which $y_i > y_j$. For any ϵ, such that $0 < \epsilon < y_i - y_j$, we have that $(y_i - \epsilon)(y_j + \epsilon) = y_i y_j + \epsilon(y_i - y_j) - \epsilon^2 > y_i y_j$. Hence the product order defines a fair preference on $\mathbb{R}^m_{++}$.

Definition 2.6. An outcome vector $\boldsymbol{y}$ is said to fairly dominate outcome vector $\boldsymbol{y}'$, $\boldsymbol{y} \succ_e \boldsymbol{y}'$, if and only if $\boldsymbol{y} \succ \boldsymbol{y}'$ for all fair rational preference relations $\succeq$.

In many cases it can be cumbersome to compare solutions in terms of vectors. Typical solution concepts for multiple criteria problems are instead defined by aggregation functions.

Definition 2.7. An aggregation function for a preference relation $\succeq$ is a function $g : Q \to \mathbb{R}$ for which $\boldsymbol{y} \succeq \boldsymbol{y}' \Leftrightarrow g(\boldsymbol{y}) \geq g(\boldsymbol{y}')$.

This may of course be used in transforming a multi-criteria optimization problem to a single-objective problem. Further, this gives a tool to specify (implicitly) fair rational preference relations on a given set Q.

Property 2.1. *For any strictly concave, strictly increasing function $s : \mathbb{R} \to \mathbb{R}$ the function $g(\boldsymbol{y}) = \sum_{i=1}^m s(y_i)$ defines a fair rational preference relation $\succeq_g$ on Q.*

Proof. It is easy to see that g induces a preference relation that is transitive and symmetric. The strict monotonicity follows from that s is strictly increasing. For the equitable transfers, consider two entries j and i such that $y_j > y_i$, and an ϵ such that $y_j - y_i > \epsilon > 0$. Then there exists a λ, $0 < \lambda < 1$, such that $y_i + \epsilon = \lambda y_i + (1 - \lambda) y_j$. After some manipulation, the same expression reads $y_j - \epsilon = (1 - \lambda) y_i + \lambda y_j$. By the strict concavity of s we have that $s(y_i + \epsilon) > \lambda s(y_i) + (1 - \lambda) s(y_j)$ and $s(y_j - \epsilon) > (1 - \lambda) s(y_i) + \lambda s(y_j)$, respectively. Adding the two inequalities yields $s(y_i + \epsilon) + s(y_j - \epsilon) > s(y_i) + s(y_j)$. □

Even though the idea of aggregation functions appears very practical in defining fair preferences in general, it is not convenient for all types of fairness.

Definition 2.8. The preference relation of *leximin order* on a set Q, $\succeq_{lex}$, is given by, for any $\boldsymbol{y}, \boldsymbol{z} \in Q$, if $\boldsymbol{y} \succeq_{lex} \boldsymbol{z}$ then the first non-zero element of $\boldsymbol{y} - \boldsymbol{z}$ is positive or $\boldsymbol{y} = \boldsymbol{z}$. If $\boldsymbol{y} \succ_{lex} \boldsymbol{z}$ we say that $\boldsymbol{y}$ is lexicographically greater than $\boldsymbol{z}$.

It is quite obvious that the leximin order does not define a fair preference relation, as it lacks the property of symmetry. However, imposing the ordering map, $\Theta(\cdot)$, provides the remedy.

Definition 2.9. The *symmetrical leximin order*, $\succeq_{symlex}$, is the preference relation defined by $\boldsymbol{y} \succeq_{symlex} \boldsymbol{y}' \Leftrightarrow \Theta(\boldsymbol{y}) \succeq_{lex} \Theta(\boldsymbol{y}')$.

The following assertion is easily verified.

Property 2.2. *The symmetrical leximin order defines a fair rational preference relation on any transferable set Q.*

2.1.2 Fairly nondominated vectors

The concept of obtaining fairness is usually concerned with, given a fair rational preference relation, $\succeq$, on a transferable set Q, finding a vector $\boldsymbol{y}^* \in Y$, $Y \subseteq Q$, such that $\boldsymbol{y}^* \succeq \boldsymbol{y}$ for all vectors $\boldsymbol{y} \in Y$. According to Definition 2.6, this is a matter of finding a (specific) fairly non-dominated vector. The type of fair preference considered will determine which fairly non-dominated vector that is desired, as there in general are several. In the case of a fair preference, $\succeq_g$, defined by an aggregation function, $g : Q \to \mathbb{R}$, obtaining fairness is a single-criteria optimization problem; $\max\{g(\boldsymbol{y}) : \boldsymbol{y} \in Y\}$. As established by Property 2.1, any strictly concave, strictly increasing function $s : \mathbb{R} \to \mathbb{R}$ can be used to define a fair rational preference relation $\succeq_g$ on some superset of the outcomes, Q, through defining the aggregation function $g : Q \to \mathbb{R}$, as $g(\boldsymbol{y}) = \sum_{i=1}^{m} s(y_i)$. For instance, we may let $Q = \mathbb{R}^m_{++}$, and let $s(y_i) = \log(y_i)$, rendering $g(\boldsymbol{y}) = \sum_{i=1}^{m} \log y_i$, which if maximized is the principle of proportional fairness [15]. Note that the induced fair preference, $\succeq_g$, is actually the product order, $\succeq_\times$, considered in Example 2.1. Similarly, restricting Q in the same way we may obtain the fairness principle of minimum delay (dicussed in [24]), by letting $s(y_i) = -\frac{1}{y_i}$, and maximizing $g(\boldsymbol{y}) = -\sum_{i=1}^{m} \frac{1}{y_i}$. Clearly, the method of selecting an appropriate function s, offers a straightforward way of defining a fair rational preference relation on Q, and more examples of interesting (appropriate) functions s can be found in [39]. It is however far from obvious how to physically interpret the fair rational preference relation $\succeq_g$ on Q, induced by an arbitrary strictly concave, strictly increasing function s.

2.1.3 Fairness and Pareto-efficiency

From the ordered outcomes, $\Theta(\boldsymbol{y}) = (\theta_1(\boldsymbol{y}), \theta_2(\boldsymbol{y}), \ldots, \theta_m(\boldsymbol{y}))$, we may derive a linear cumulative map, called the *cumulative ordering map*,

$$\bar{\Theta}(\boldsymbol{y}) = (\bar{\theta}_1(\boldsymbol{y}), \bar{\theta}_2(\boldsymbol{y}), \ldots, \bar{\theta}_m(\boldsymbol{y})),$$

defined as

$$\bar{\theta}_i = \sum_{j=1}^{i} \theta_j(\boldsymbol{y}) \quad \text{for} \quad i = 1, 2, \ldots, m. \tag{2.3}$$

From the theory of majorization [23], the following result can be derived [19]. However, the proof provided here is based on first-order principles.

Theorem 2.3 (Kostreva and Ogryczak, 1999)**.** *Outcome vector $\boldsymbol{y}$ fairly dominates outcome vector $\boldsymbol{y}'$, if and only if $\bar{\Theta}(\boldsymbol{y}) \geqq \bar{\Theta}(\boldsymbol{y}')$.*

Proof. Suppose $\bar{\theta}_i(\boldsymbol{y}) \geq \bar{\theta}_i(\boldsymbol{y}')$ for $i = 1, 2, \ldots, m$, and that strict inequality holds for at least one entry. First of all we point out that for any two vectors $\boldsymbol{y}$ and $\boldsymbol{y}'$, if $\bar{\theta}_j(\boldsymbol{y}) = \bar{\theta}_j(\boldsymbol{y}')$, for $j = 1, 2, \ldots, n$, $n \leq m$, then $\theta_j(\boldsymbol{y}) = \theta_j(\boldsymbol{y}')$ for $j = 1, 2, \ldots, n$. Now assume that k is the smallest index for which $\bar{\theta}_k(\boldsymbol{y}) > \bar{\theta}_k(\boldsymbol{y}')$. If $k = m$, then the monotonic increase $\Theta(\boldsymbol{y}') + \boldsymbol{e}_m(\theta_m(\boldsymbol{y}) - \theta_m(\boldsymbol{y}'))$ is indifferent from $\boldsymbol{y}$. Thus $\boldsymbol{y} \succ_e \boldsymbol{y}'$ and we are done. On the other hand, assume $k < m$. Assign $\boldsymbol{y}'' = \Theta(\boldsymbol{y}')$. If $\bar{\theta}_m(\boldsymbol{y}) > \bar{\theta}_m(\boldsymbol{y}')$, assign $y''_m = y''_m + (\bar{\theta}_m(\boldsymbol{y}) - \bar{\theta}_m(\boldsymbol{y}'))$. We then have that $\bar{\theta}_m(\boldsymbol{y}) = \bar{\theta}_m(\boldsymbol{y}'')$, and that $\bar{\theta}_k(\boldsymbol{y}) > \bar{\theta}_k(\boldsymbol{y}'')$ for some $k < m$. Further it holds that $\theta_k(\boldsymbol{y}) > y''_k$, since k is the first such entry. As $\bar{\theta}_m(\boldsymbol{y}) = \bar{\theta}_m(\boldsymbol{y}'')$, there must exist entries $j > k$, such that $\theta_j(\boldsymbol{y}) < y''_j$. Because of that Θ is an ordering map (and the construction of $\boldsymbol{y}''$), it holds that $y''_j > y''_k$, for all these entries j. We can thus perform equitable transfers from these entries j to entry k, redefining $\boldsymbol{y}''$, until $y''_k = \theta_k(\boldsymbol{y})$. Repeating this procedure (obviously finitely many times), we will eventually get that $y''_i = \theta_i(\boldsymbol{y})$, for $i = 1, 2, \ldots, m$, implying that $\boldsymbol{y}'' \cong \Theta(\boldsymbol{y})$. Since $\boldsymbol{y}''$ is obtained from $\Theta(\boldsymbol{y}')$ as a series of (necessary) equitable transfers and possibly a monotonic increase in the largest component, we have that $\boldsymbol{y} \cong \Theta(\boldsymbol{y}) = \boldsymbol{y}'' \succ_e \Theta(\boldsymbol{y}') \cong \boldsymbol{y}'$. For the reversed implication we may easily verify that the relation $\bar{\Theta}(\boldsymbol{y}) \geq \bar{\Theta}(\boldsymbol{y}')$ defines a fair preference, $\succeq_{\bar{\theta}}$. Specifically $\boldsymbol{y} \succ_{\bar{\theta}} \boldsymbol{y}'$ if $\bar{\theta}_i(\boldsymbol{y}) \geq \bar{\theta}_i(\boldsymbol{y}')$, for $i = 1, 2, \ldots, m$ and $\bar{\theta}_k(\boldsymbol{y}) > \bar{\theta}_k(\boldsymbol{y}')$ for at least one entry k, $1 \leq k \leq m$. Hence if $\boldsymbol{y} \succ_e \boldsymbol{y}'$, it must hold that $\boldsymbol{y} \succ_{\bar{\theta}} \boldsymbol{y}'$. □

This gives the possibility of expressing all the fairly non-dominated solutions as the Pareto-efficient solutions to a multiple criteria problem, namely

$$\max\{(\eta_1, \eta_2, \ldots, \eta_m) : \eta_i = \bar{\theta}_i(\boldsymbol{y}) \text{ for } i = 1, 2, \ldots, m, \ \boldsymbol{y} \in Y\}. \tag{2.4}$$

A Pareto-efficient solution is a solution for which no Pareto improvements can be done, i.e., a solution for which no entry can be increased without making another entry worse off. Since quantities $\bar{\theta}_i(\boldsymbol{y})$ are quite complicated, the usefulness of this formula relies on an elegant result concerning the expressability of $\bar{\theta}_i(\boldsymbol{y})$ for a given outcome vector $\boldsymbol{y}$ [42]:

$$\bar{\theta}_k(\boldsymbol{y}) = \max \quad kt - \sum_{i=1}^{m} d_i \tag{2.5}$$

$$\text{s.t.} \quad t - y_i \leq d_i, \ d_i \geq 0 \qquad i = 1, 2, \ldots, m. \tag{2.6}$$

Exploting this, we may thus rewrite (2.4) as

$$\begin{aligned} \max \quad & (\eta_1, \eta_2, \ldots, \eta_m) && && (2.7)\\ \text{s.t.} \quad & \eta_k = kt_k - \sum_{i=1}^{m} d_{ik} && k = 1, 2, \ldots, m && (2.8)\\ & t_k - d_{ik} \leq y_i,\ d_{ik} \geq 0 && i, k = 1, 2, \ldots, m && (2.9)\\ & \boldsymbol{y} \in Y. && && (2.10) \end{aligned}$$

To summarize, any fairly non-dominated solution may be modeled as a Pareto-efficient solution to (2.7)-(2.10).

2.2 General ways of defining fairness

Besides the general use of proper aggregation functions there are several commonly used ways of defining different classes of fairness. Such a class typically defines a generic type of fairness in which some easily accessible parameters may be changed in order to vary the induced fair preference. Examples of such generic types of definitions are p-norm fairness, fairness induced by ordered weighted averaging, and α-fairness.

2.2.1 p-norm fairness

Consider the usual p-norm, $||\cdot||_p$ defined for arbitrary vectors $\boldsymbol{y} \in \mathbb{R}^n$;

$$||\boldsymbol{y}||_p = \left(\sum_{i=1}^{n} |y_i|^p \right)^{1/p} ,$$

where p is an integer, $p \geq 1$. To derive the notion of p-norm fairness we will need the following lemma:

Lemma 2.4. *Consider arbitrary numbers $c \geq x > y \geq 0$ and an integer p, $1 < p < \infty$. Let ϵ be any number such that $0 < \epsilon < x - y$. It holds that*

$$(c - x)^p + (c - y)^p > (c - (x - \epsilon))^p + (c - (y + \epsilon))^p .$$

Proof. The function $f(\epsilon) = (c - (x - \epsilon))^p + (c - (y + \epsilon))^p$, has exactly one stationary point for $\epsilon \in [0, x - y]$, namely $\epsilon = \frac{x-y}{2}$. As $f'(\epsilon) < 0$ for $\epsilon \in [0, \frac{x-y}{2})$, and $f'(\epsilon) > 0$ for $\epsilon \in (\frac{x-y}{2}, x - y]$, this stationary point must be a minimum. Furthermore $f(0) = f(x - y) = (c - x)^p + (c - y)^p$, which implies the result. □

Consider a vector $c\boldsymbol{u}$, $c > 0$, and let $Q = \{\boldsymbol{y} \in \mathbb{R}^n_+ : \boldsymbol{y} \leq c\boldsymbol{u}\}$.

Definition 2.10. The preference relation $\succeq^c_p$, called the p-norm preference with respect to c, is given by, for any $\boldsymbol{y}, \boldsymbol{y'} \in Q$, $\boldsymbol{y} \succeq^c_p \boldsymbol{y'} \Leftrightarrow ||c\boldsymbol{u} - \boldsymbol{y}||_p \leq ||c\boldsymbol{u} - \boldsymbol{y'}||_p$.

Property 2.5. *The p-norm preference (with respect to $c > 0$) defines a fair rational preference relation on $Q = \{\boldsymbol{y} \in \mathbb{R}^n_+ : \boldsymbol{y} \leq c\boldsymbol{u}\}$.*

Proof. Verification of transitivity, strict monotonicity, and symmetry is quite straightforward. The property of equitable transfers is though slightly harder to establish. Consider any vector $\boldsymbol{y} \in Q$ for which $y_i > y_j$. Let ϵ be any number suct that $0 < \epsilon < y_i - y_j$, and define $\boldsymbol{y}'$ as $\boldsymbol{y}' = \boldsymbol{y} + \epsilon\boldsymbol{e}_j - \epsilon\boldsymbol{e}_i$. We then have that $||c\boldsymbol{u} - \boldsymbol{y}'||_p =$

$$= \left(\sum_i^n (c - y_k)^p - (c - y_i)^p - (c - y_j)^p + (c - y_i + \epsilon)^p + (c - y_j - \epsilon)^p \right)^{1/p},$$

which, according to Lemma 2.4 and the fact that the root function is strictly increasing, is strictly smaller than $\left(\sum_i^n (c - y_k)^p \right)^{1/p} = ||c\boldsymbol{u} - \boldsymbol{y}||_p$. Thus we have equitable transfers. □

2.2.2 Ordered weighted averaging

The result that any Pareto-efficient solution $\boldsymbol{y}^0$ to (2.7)-(2.10) is a fairly non-dominated solution implies that there exists at least one fair rational preference relation $\succeq$, such that $\boldsymbol{y}^0 \succeq \boldsymbol{y}$, for all $\boldsymbol{y} \in Q$. We may thus use as a single objective function the weighted sum,

$$\sum_{i=1}^m w_i \eta_i = \sum_{i=1}^m w_i \bar{\theta}_i(\boldsymbol{y}), \ w_i > 0, \text{ for } i = 1, 2, \ldots, m, \tag{2.11}$$

which is certain to generate a Pareto-efficient solution to (2.7)-(2.10). As $\bar{\theta}_i(\boldsymbol{y}) = \sum_{j=1}^i \theta_j(\boldsymbol{y})$, we have

$$\sum_{i=1}^m w_i \bar{\theta}_i(\boldsymbol{y}) = \sum_{i=1}^m w_i \sum_{j=1}^i \theta_j(\boldsymbol{y}) = \sum_{i=1}^m v_i \theta_i(\boldsymbol{y}),$$

where $v_i = \sum_{j=i}^m w_j$. Using $\sum_{i=1}^m v_i \theta_i(\boldsymbol{y})$ as the single objective function is called Ordered Weighted Averaging (OWA), and was first introduced in [54]. Note that if we use this single objective function the Pareto-efficiency will be guaranteed if weights v_i are strictly decreasing and positive. Besides the direct consequence that if weights v_i are chosen so that they are strictly decreasing and positive then each solution to the OWA is a fairly non-dominated solution, there is a reversed implication. Namely, in the case of linear constraints, every fairly non-dominated solution can be identified as an optimal solution to some OWA problem [39]. The importance of this lies in that for any given fair rational preference relation $\succeq$ on Q, if for some outcome vector $\boldsymbol{y}^0 \in Q$, it holds that $\boldsymbol{y}^0 \succeq \boldsymbol{y}$, for all $\boldsymbol{y} \in Q$, then $\boldsymbol{y}^0$ must be fairly non-dominated and therefore an optimal solution to some OWA problem. Specifically, if the difference between the weights v_i tend to infinity, solving the OWA problem will be equivalent to finding a vector

$\boldsymbol{y}^0 \in Q$ such that $\boldsymbol{y}^0 \succeq_{symlex} \boldsymbol{y}$, for all $\boldsymbol{y} \in Q$, i.e., obtaining the fairness associated with the symmetrical leximin order. This type of fairness is often referred to as max-min fairness and will be extensively investigated in later sections.

2.2.3 α-fairness

The notion of α-fairness was introduced by Mo and Walrand in [28], and it defines a class of fairness relations through an aggregation function. In [28], α-fairness is defined through maximization and is not put in the context of preference relations. Hence they do not verify that its corresponding order, here referred to as the α-order, defines a fair rational preference relation.

Definition 2.11. For a given α, $\alpha > 0$, $\alpha \neq 1$, the α-order, $\succeq_\alpha$, is the preference relation defined by

$$\boldsymbol{y} \succeq_\alpha \boldsymbol{y}' \Leftrightarrow \sum_{i=1}^{m} \frac{y_i^{(1-\alpha)}}{1-\alpha} \geq \sum_{i=1}^{m} \frac{y_i'^{(1-\alpha)}}{1-\alpha}.$$

Property 2.6. *For a given α, $\alpha > 0$, $\alpha \neq 1$, the α-order defines a fair rational preference relation on $\mathbb{R}^m_{++}$.*

Proof. By Property 2.1, it is only necessary to verify that $s : \mathbb{R}_{++} \to \mathbb{R}$, $s(y) = \frac{y^{(1-\alpha)}}{1-\alpha}$, is strictly increasing and strictly concave for any proper choice of α. Consider the case when $0 < \alpha < 1$. Then $s(y) = Cy^{(1-\alpha)}$, for a constant $C = \frac{1}{1-\alpha} > 0$, so obviously s is strictly increasing. To ensure the concavity, we see that $\frac{d^2s}{dy^2} = -\alpha y^{-(\alpha+1)} < 0$, for any $y \in \mathbb{R}_{++}$. Now consider the case when $\alpha > 1$. Then $s(y) = C\frac{1}{y^\gamma}$, for some constant $C < 0$, and $\gamma > 0$, so s is strictly increasing. Further, $\frac{d^2s}{dy^2} = -\alpha y^{-(\alpha+1)} < 0$, for any $y \in \mathbb{R}_{++}$, so s is strictly concave. □

Remark 2.12. In the original definition of α-fairness [28], the proportional fairness is covered by defining $\boldsymbol{y} \succeq_\alpha \boldsymbol{y}' \Leftrightarrow \sum_{i=1}^m \log(y_i) \geq \sum_{i=1}^m \log(y_i')$ for $\alpha = 1$.

Remark 2.13. If $\alpha = 2$, then $\frac{y^{(1-\alpha)}}{1-\alpha} = -\frac{1}{y}$, so minimum delay fairness is a special case of α-fairness.

Interestingly it is shown in [28] that if α tends to infinity, the solution $\boldsymbol{y}^0$ to

$$\max \quad \sum_{i=1}^{m} \frac{y_i^{(1-\alpha)}}{1-\alpha} \tag{2.12}$$

$$\text{s.t.} \quad \boldsymbol{y} \in Y \ , \tag{2.13}$$

will have the property that $\boldsymbol{y}^0 \succeq_{symlex} \boldsymbol{y}$ for all $\boldsymbol{y} \in Y$, for nicely behaved sets of outcome vectors $Y \subseteq \mathbb{R}^m_{++}$ (see [28] for the exact restrictions put

on Y). However, this result may be misleading in the sense that one might get the impression that for a sufficiently large α there will be equivalence between $\boldsymbol{y} \succeq_\alpha \boldsymbol{y}'$ and $\boldsymbol{y} \succeq_{symlex} \boldsymbol{y}'$, for $\boldsymbol{y}, \boldsymbol{y}' \in Y$, which is not true. In fact, if Y is a non-discrete set (for instance $Y = \{\ \boldsymbol{y} \in \mathbb{R}^m : \sum_{i=1}^m y_i \leq L\ \}$), an $\alpha < \infty$ will never suffice for the equivalence. On the other hand, if discrete sets of outcome vectors are considered, and if it is possible to somehow bound each coordinate from above, a more practical assertion can be posed. More practical in the sense that we can find an upper bound on α for which the equivalence holds, and also in the sense that discrete outcome sets, as well as upper bound on coordinates, model real-world resource allocation in communication networks realistically.

Proposition 2.7. *If the set of outcome vectors Y has the property that for each $\boldsymbol{y} \in Y$, each coordinate y_k, $k = 1, 2, \ldots, m$, must be a strictly positive integer which can be bounded from above, $y_k \leq U$, then for any $\boldsymbol{y}, \boldsymbol{y}' \in Y$, $\boldsymbol{y} \succeq_{symlex} \boldsymbol{y}' \Leftrightarrow \sum_{i=1}^m \frac{1}{(y_i)^\beta} \leq \sum_{i=1}^m \frac{1}{(y_i')^\beta}$, for any $\beta > \frac{\log m}{\log \frac{U}{U-1}}$.*

Proof. If $\boldsymbol{y} \cong_{symlex} \boldsymbol{y}'$ then $\Theta(\boldsymbol{y}) = \Theta(\boldsymbol{y}')$ so it must be true that $\sum_{i=1}^m \frac{1}{(y_i)^\beta} = \sum_{i=1}^m \frac{1}{(y_i')^\beta}$, for any β. Suppose that $\boldsymbol{y} \succ_{symlex} \boldsymbol{y}'$. This implies that there is a first non-zero coordinate of $\Theta(\boldsymbol{y}) - \Theta(\boldsymbol{y}')$, which is necessarily positive. Let j be the index of this coordinate. We have that

$$\begin{aligned}\sum_{i=1}^m \frac{1}{(y_i)^\beta} = \quad & \sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}))^\beta} + \frac{1}{(\theta_j(\boldsymbol{y}))^\beta} + \sum_{i=j+1}^m \frac{1}{(\theta_i(\boldsymbol{y}))^\beta} \leq \\ & \leq \sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}))^\beta} + \frac{m-j+1}{(\theta_j(\boldsymbol{y}))^\beta}\ .\end{aligned}$$

Further,

$$\begin{aligned}\sum_{i=1}^m \frac{1}{(y_i')^\beta} = \quad & \sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}'))^\beta} + \frac{1}{(\theta_j(\boldsymbol{y}'))^\beta} + \sum_{i=j+1}^m \frac{1}{(\theta_i(\boldsymbol{y}'))^\beta} > \\ & > \sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}'))^\beta} + \frac{1}{(\theta_j(\boldsymbol{y}'))^\beta}\ .\end{aligned}$$

Thus if

$$\sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}))^\beta} + \frac{m-j+1}{(\theta_j(\boldsymbol{y}))^\beta} < \sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}'))^\beta} + \frac{1}{(\theta_j(\boldsymbol{y}'))^\beta}\ , \tag{2.14}$$

it must hold that $\sum_{i=1}^m \frac{1}{(y_i)^\beta} < \sum_{i=1}^m \frac{1}{(y_i')^\beta}$. Noting that the first two terms of the respective sides are equal, the requirement reads

$$\frac{m-j+1}{(\theta_j(\boldsymbol{y}))^\beta} < \frac{1}{(\theta_j(\boldsymbol{y}'))^\beta}\ , \tag{2.15}$$

which, after some manipulation, yields

$$\beta > \frac{\log(m-j+1)}{\log(\theta_j(\boldsymbol{y})) - \log(\theta_j(\boldsymbol{y}'))} \ . \tag{2.16}$$

Since $m-j+1 \leq m$ and $\frac{\theta_j(\boldsymbol{y})}{\theta_j(\boldsymbol{y}')} \geq \frac{U}{U-1}$, it suffices to require that $\beta > \frac{\log(m)}{\log \frac{U}{U-1}}$. For the opposite direction, first assume that $\sum_{i=1}^{m} \frac{1}{(y_i)^\beta} = \sum_{i=1}^{m} \frac{1}{(y'_i)^\beta}$ and that j is the first coordinate for which $\theta_j(\boldsymbol{y}) > \theta_j(\boldsymbol{y}')$. The assumption $\sum_{i=1}^{m} \frac{1}{(y_i)^\beta} = \sum_{i=1}^{m} \frac{1}{(y'_i)^\beta}$ implies that

$$\sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}))^\beta} + \frac{1}{(\theta_j(\boldsymbol{y}))^\beta} + \sum_{i=j+1}^{m} \frac{1}{(\theta_i(\boldsymbol{y}))^\beta} = \tag{2.17}$$

$$= \sum_{i=1}^{j-1} \frac{1}{(\theta_i(\boldsymbol{y}'))^\beta} + \frac{1}{(\theta_j(\boldsymbol{y}'))^\beta} + \sum_{i=j+1}^{m} \frac{1}{(\theta_i(\boldsymbol{y}'))^\beta} \ . \tag{2.18}$$

As before, the first terms on the respective sides cancel out. The second two terms on the left-hand side are not larger than $\frac{m-j+1}{(\theta_j(\boldsymbol{y}))^\beta}$, and the second two terms on the right-hand side are strictly larger than $\frac{1}{(\theta_j(\boldsymbol{y}'))^\beta}$. Hence, according to the previous reasoning, insertion of $\beta > \frac{\log(m)}{\log \frac{U}{U-1}}$ will violate the equality and we have a contradiction. Finally we have to prove that $\sum_{i=1}^{m} \frac{1}{(y_i)^\beta} < \sum_{i=1}^{m} \frac{1}{(y'_i)^\beta}$ implies that $\boldsymbol{y} \succ_{symlex} \boldsymbol{y}'$. This is however contrapositive (equivalent) to $\boldsymbol{y} \preceq \boldsymbol{y}' \Rightarrow \sum_{i=1}^{m} \frac{1}{(y_i)^\beta} \geq \sum_{i=1}^{m} \frac{1}{(y'_i)^\beta}$, which is already proved. □

It should be noted that Proposition 2.7 actually concerns the α-order as

$$\sum_{i=1}^{m} \frac{1}{(y_i)^\beta} \leq \sum_{i=1}^{m} \frac{1}{(y'_i)^\beta} \Leftrightarrow \sum_{i=1}^{m} \frac{y_i^{(1-\alpha)}}{1-\alpha} \geq \sum_{i=1}^{m} \frac{y_i'^{(1-\alpha)}}{1-\alpha} \ ,$$

for $\alpha = \beta + 1$.

2.3 Max-min fairness

The most thoroughly investigated type of fairness is undoubtedly the type called Max-Min Fairness (MMF). There are several, more or less general, ways of defining MMF. In order to give a non-ambigous introduction to this topic, will we make a distinction between weak and strong MMF. Throughout this section we will consider an m-dimensional outcome set Y, $Y \subseteq Q \subseteq \mathbb{R}^m$, facilitating a superset Q on which there is defined a preference relation.

2.3.1 Weak and strong max-min fairness

Definition 2.14. A vector $\boldsymbol{y}^0 \in Y$ for which it holds that $\boldsymbol{y}^0 \succeq_{lex} \boldsymbol{y}$, for all $\boldsymbol{y} \in Y$, is called lexicographically maximal on Y, written

$$\boldsymbol{y}^0 = \text{lex max}\{\boldsymbol{y} : \boldsymbol{y} \in Y\}.$$

It is quite intuitive to state an algorithm that finds a lexicographically maximal $\boldsymbol{y}^0 = (y_1^0, y_2^0, \ldots, y_m^0)$, $\boldsymbol{y}^0 \in Y$. Let $y_1^0 = \max\{y_1 : \boldsymbol{y} \in Y\}$ and define consecutively $y_j^0 = \max\{y_j : y_1 \geq y_1^0, \ldots, y_{j-1} \geq y_{j-1}^0, \boldsymbol{y} \in Y\}$, for $j = 2, 3, \ldots, m$. Carrying out this procedure, we successively get the entries of $\boldsymbol{y}^0$. Apparently, in lexicographical maximization, maximization of the first entry, y_1, has the highest priority. Second highest priority is maximization of y_2, and so on. In fact, many practical applications, including the ones adressed in this thesis, do not assume these priorities, but rather prescribe equal (fair) treatment of all the entries (i.e., assume symmetry of the preference relation). For this purpose we need the notion of sorted vectors or the symmetrical leximin order, $\succeq_{symlex}$.

Although the introduction of the fair rational preference relation, $\succeq_{symlex}$, fits the terminology of fairness well, we will stick to an equivalent consideration of the original leximin order on the set $\vec{Y} = \{\Theta(\boldsymbol{y}) : \boldsymbol{y} \in Y\}$, i.e., for each vector $\boldsymbol{y} \in Y$, evaluate a sorted permutation $\Theta(\boldsymbol{y})$. This convention is due to the leximin order as a well known and well accepted concept (as opposed to the symmetrical leximin order). Lexicographical maximality for sorted vectors is defined as follows.

Definition 2.15. A vector $\boldsymbol{y}^0 \in Y$ for which it holds that $\Theta(\boldsymbol{y}^0) \succeq_{lex} \Theta(\boldsymbol{y})$, for all $\boldsymbol{y} \in Y$, is called sorted lexicographically maximal on Y, written

$$\Theta(\boldsymbol{y}^0) = \text{lex max}\{\Theta(\boldsymbol{y}) : \boldsymbol{y} \in Y\}.$$

As this section will consider the preference relation of leximin order exclusively, we will from now on omit the subscript and write only $\succeq$ for $\succeq_{lex}$.

Definition 2.16. A vector $\boldsymbol{y}^0 \in Y$ which is sorted lexicographically maximal on Y, i.e., $\Theta(\boldsymbol{y}^0) = \text{lex max}\{\Theta(\boldsymbol{y}) : \boldsymbol{y} \in Y\}$, is called *weakly max-min fair* (or just max-min fair) on Y.

This is the definition that will be intended whenever we speak about max-min fairness without mentioning "weak" or "strong", i.e., we will sometimes denote weak max-min fairness only by "max-min fairness". Note that this type of fairness is exactly that induced by the symmetrical leximin order. The alternative way of defining max-min fairness is more restrictive (thus not equivalent), and is originally due to Bertsekas and Gallgaher [6]. Although less general, this definition has been adopted by several authors, e.g., [48],[46]. We will refer to this latter characteristic as strong max-min fairness.

Definition 2.17. The vector $\boldsymbol{y}^0$ is *strongly max-min fair* on Y if $\boldsymbol{y}^0 \in Y$ and $\forall \boldsymbol{y} \in Y$, if $\exists k \in \{1, 2, \ldots, m\} : y_k > y_k^0$, then $\exists j \in \{1, 2, \ldots, m\} : y_j < y_j^0 \leq y_k^0$.

Property 2.8. *A strongly max-min fair vector is unique.*

Proof. Suppose that there exist two strongly max-min fair vectors, $\boldsymbol{y}^1, \boldsymbol{y}^2 \in Y$, $\boldsymbol{y}^1 \neq \boldsymbol{y}^2$. Find the entry k for which $y_k^1 > y_k^2$, such that y_k^1 is as small as possible (if it does not exist, exchange the names of $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$). Then, since $\boldsymbol{y}^2$ is strongly max-min fair, there exists an entry $j \in \{1, 2, \ldots, m\} : y_j^1 < y_j^2 \leq y_k^2$. However, since also $\boldsymbol{y}^1$ is strongly max-min fair there must exist an entry $i \in \{1, 2, \ldots, m\} : y_i^2 < y_i^1 \leq y_j^1$, which implies existence of an entry $l \in \{1, 2, \ldots, m\} : y_l^1 < y_l^2 \leq y_i^2$, and so on. Eventually, we will run out of entries and get a contradiction. □

Remark 2.18. In [48] a variation of Definition 2.17 is used to define a preference relation to compare vectors. A vector $\boldsymbol{y}$ is said to be "relatively fairer" than a vector $\boldsymbol{y}'$ if either $\boldsymbol{y} \geqq \boldsymbol{y}'$ or if $\exists k \in \{1, 2, \ldots, m\} : y_k < y'_k$, then $\exists j \in \{1, 2, \ldots, m\} : y'_j < y_j \leq y_k$ (The term "relative fairness" was first introduced in [47]). A feasible vector $\boldsymbol{y}$ for which there exists no feasible vector $\boldsymbol{y}'$ that is relatively fairer than $\boldsymbol{y}$ is called "maximally fair". Even though being closely related to the symmetrical leximin order, this does not define a fair rational preference relation. Too see this, suppose that the vectors $(0, 2, 1)$ and $(3, 0, 0)$ belong to a feasible (transferable) set. The first vector is obtainable from the second one through an equitable transfer followed by a permutation. To be a fair rational preference relation it must thus hold that $(0, 2, 1)$ is preferable to $(3, 0, 0)$. However $(0, 2, 1)$ is *not* relatively fairer than $(3, 0, 0)$.

The exact relation between weak and strong max-min fairness is described by the following two properties

Property 2.9. *If $\boldsymbol{y}^0$ is strongly max-min fair on Y, then it is also weakly max-min fair on Y.*

Proof. We may assume that $\boldsymbol{y}^0$ is sorted, since if it is not we may reenumerate coordinates such that they appear in non-decreasing order. For any $\boldsymbol{y} \in Y$, $\boldsymbol{y} \neq \boldsymbol{y}^0$, pick the smallest k, such that $y_k > y_k^0$ (if such an entry does not exist, the result follows immediately as there then must exist an entry k for which $y_k < y_k^0$). If there are several entries e of $\boldsymbol{y}^0$ that are equal to y_k^0, then order them (i.e., order the coordinates) such that all entries e with $y_e \leq y_e^0$ $(= y_k^0)$ appear before all entries e with $y_e > y_e^0$ $(= y_k^0)$, which is again achievable by alternative enumeration of the coordinates of $\boldsymbol{y}^0$. Now redefine entry k as the first entry for which $y_k > y_k^0$. It then must hold that $y_i^0 \geq y_i$ for any entry $i < k$. Specifically, as $\boldsymbol{y}^0$ is strongly max-min fair, there is an entry j, $1 \leq j < k$, such that $y_j^0 > y_j$. Let $\boldsymbol{y}^0[k-1]$ be

the subvector constituted by the $k-1$ first entries of $\boldsymbol{y}^0$, and let $\boldsymbol{y}[k-1]$ be defined similarly. It is easy to see that $\boldsymbol{y}^0[k-1] \geq \boldsymbol{y}[k-1]$ implies $\boldsymbol{y}^0[k-1] \geq \Theta(\boldsymbol{y})[k-1]$ and that existence of an entry j, $1 \leq j < k$ such that $y_j^0[k-1] > y_j[k-1]$ implies that there exists an entry l, $1 \leq l < k$ such that $y_l^0[k-1] > \theta_l(\boldsymbol{y})[k-1]$. Thus $\boldsymbol{y}^0[k-1] \succ \Theta(\boldsymbol{y})[k-1]$, which implies that $\boldsymbol{y}^0 = \Theta(\boldsymbol{y}^0) \succ \Theta(\boldsymbol{y})$. □

Property 2.10. *If $\boldsymbol{y}^0 \in Y$ and Y is convex, then if $\boldsymbol{y}^0$ is weakly max-min fair on Y, it is also strongly max-min fair on Y.*

Proof. We prove the equivalent statement: if $\boldsymbol{y}^0$ is not strongly max-min fair on the convex set Y then $\Theta(\boldsymbol{y}^0) \prec_{lex} \text{lex max}\{\Theta(\boldsymbol{y}) : \boldsymbol{y} \in Y\}$. Suppose that $\boldsymbol{y}^0$ is not strongly max-min fair on Y. Then there exists a vector $\boldsymbol{y} \in Y$, which has an entry $k \in \{1, 2, \ldots, m\}$, for which $y_k > y_k^0$ but $y_j \geq y_j^0$ for all entries j such that $y_j^0 \leq y_k^0$ (if such entries j exist). Define $\boldsymbol{z} = (1-\lambda)\boldsymbol{y}^0 + \lambda\boldsymbol{y}$, where λ is a constant, $0 < \lambda < 1$. Assume that coordinates appear such that $\boldsymbol{y}^0 = \Theta(\boldsymbol{y}^0)$ (if not rename coordinates). For sufficiently small λ it also holds that $\boldsymbol{z} = \Theta(\boldsymbol{z})$. Moreover, there exists an entry e such that such that $z_e > y_e^0$ and $z_i \geq y_i^0$ for all indices $i < e$. Thus $\Theta(\boldsymbol{z}) \succ_{lex} \Theta(\boldsymbol{y}^0)$. By convexity, $\boldsymbol{z} \in Y$. □

2.3.2 Max-min fairness in the solution space

Let X be an n-dimensional subset of $\mathbb{R}^n$, $X \subseteq \mathbb{R}^n$, and consider a vector of real-valued functions $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$, with $f_j : X \to \mathbb{R}$, $j = 1, 2, \ldots, m$. Let the vector $\boldsymbol{f}$ be the vector of outcomes (sometimes also referred to as objectives or criteria).

Definition 2.19. The vector $\boldsymbol{x}_0 \in X$ is said to be a *max-min fair solution*, if and only if $\Theta(\boldsymbol{f}(\boldsymbol{x}^0)) = \text{lex max}\{\Theta(f(\boldsymbol{x})) : \boldsymbol{x} \in X\}$, i.e., if and only if $\boldsymbol{f}(\boldsymbol{x}^0)$ is (weakly) max-min fair on $\{\boldsymbol{f}(\boldsymbol{x}) : \boldsymbol{x} \in X\}$.

Define the set Y as

$$Y = \{\boldsymbol{y} : \boldsymbol{y} \leq \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in X\}. \tag{2.19}$$

Property 2.11. *Given a vector $\boldsymbol{x}^0 \in X$. If $\boldsymbol{y}^0 = \boldsymbol{f}(\boldsymbol{x}^0)$ is strongly max-min fair on Y, then $\boldsymbol{x}^0$ is a max-min fair solution.*

Proof. If $\boldsymbol{y}^0 = \boldsymbol{f}(\boldsymbol{x}^0)$ is strongly max-min fair on $Y = \{\boldsymbol{y} : \boldsymbol{y} \leq \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in X\}$, then, according to Property 2.9, $\Theta(\boldsymbol{y}^0) = \text{lex max}\{\Theta(\boldsymbol{y}) : \boldsymbol{y} \in Y\} = \text{lex max}\{\Theta(\boldsymbol{y}) : \boldsymbol{y} \leq \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in X\} = \text{lex max}\{\Theta(\boldsymbol{f}(\boldsymbol{x})), \boldsymbol{x} \in X\}$. Moreover $\boldsymbol{y}^0 = \boldsymbol{f}(\boldsymbol{x}^0)$ implies $\Theta(\boldsymbol{y}^0) = \Theta(\boldsymbol{f}(\boldsymbol{x}^0))$, so $\boldsymbol{x}^0$ is a max-min fair solution. □

Due to a relatively compact description, and possibly more appeal to intuition, some authors use Property 2.11 in defining the concept of max-min fairness (and associated solution) [46]. This would certainly be enough if

the vectors specified to be max-min fair solutions by Definition 2.19 but not by Property 2.11 were artificial and of no practical interest. However, the following example shows that this is not the case.

Example 2.2. Let $m = n$ and $f_j(\boldsymbol{x}) = x_j$ for $j = 1, 2, \ldots, n$. Suppose $X = \mathbb{B}^2$. This models an instance with two demands over one link with capacity one, where demand flows must be integral. As the only solutions, $(0, 1)$ and $(1, 0)$, are equal from a fairness viewpoint, we want a definition that establishes that both of them are max-min fair. This is accomplished by Definition 2.19. However, none of them are strongly max-min fair so none of them can be concluded to be MMF through Property 2.11.

Property 2.12. *If $\boldsymbol{x}^0$ is a max-min fair solution, $\boldsymbol{x}^0 \in X$, $f_1, f_2, \ldots, f_m$ are concave functions, and X is convex, then $\boldsymbol{y}^0 = \boldsymbol{f}(\boldsymbol{x}^0)$ is strongly max-min fair on Y.*

Proof. The facts that $f_1, f_2, \ldots, f_m$ are concave and X is convex imply that $Y = \{\boldsymbol{y} : \boldsymbol{y} \leq \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{x} \in X\}$ is convex. Further, $\boldsymbol{x}^0$ is a max-min fair solution on X so $\Theta(\boldsymbol{y}^0) = \Theta(\boldsymbol{f}(\boldsymbol{x}^0)) = \text{lex max}\{\Theta(\boldsymbol{f}(\boldsymbol{x})) : \boldsymbol{x} \in X\} = \text{lex max}\{\Theta(\boldsymbol{y}) : \boldsymbol{y} \in Y\}$. The statement now follows from Property 2.10. □

Linear max-min fairness

An important special case of MMF is when the functions $f_1, f_2, \ldots, f_m$ are linear and the feasible set X is a polyhedron (i.e, X is defined by linear constraints). One such case is for instance when the max-min fair outcome vector is the max-min fair solution, letting $m = n$ and defining $f_j(\boldsymbol{x}) = x_j$, $j = 1, 2, \ldots, n$ (assuming linear constraints). In telecommunications, this seemingly simple variant is by far the most thoroughly investigated. Actually, some authors do not take any other possibilities into account, and state their MMF definitions only for this case. We will denote this special case as *linear max-min fairness*, and when there is no risk for confusion, just max-min fairness. With the given assumptions, following Definition 2.19, a solution $\boldsymbol{x}^0$ will be a max-min fair solution in the linear context if an only if

$$\Theta(\boldsymbol{x}^0) = \text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\}. \tag{2.20}$$

Using the facts established by Property 2.12, and the fact that X is a polyhedron and thus convex, this will be equivalent to the statement that $\boldsymbol{x}^0$ is strongly max-min fair on X.

2.4 Algorithms that achieve max-min fairness

Given the set X, which is a subset of $\mathbb{R}^n$, $X \subseteq \mathbb{R}^n$, and a vector of real-valued functions $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)$, with $f_j : X \to \mathbb{R}$, $j = 1, 2, \ldots, m$,

obtaining max-min fairness is a matter of finding a solution $\boldsymbol{x}^0 \in X$ that has the property that

$$\Theta(\boldsymbol{f}(\boldsymbol{x}^0)) = \text{lex max}\{\Theta(\boldsymbol{f}(\boldsymbol{x})) : \boldsymbol{x} \in X\} \ .$$

There are certain generic algorithms for these kinds of optimization problems. For the case when the problem has convex structure we give algorithms that are generalizations of the algorithms presented in [30], [50], [44], and [43]. For the non-convex cases we essentially account for the general techniques presented in [40].

2.4.1 The convex case

For the convex case, i.e., when the feasible set X is convex and the functions f_i, $i = 1, 2, \ldots, m$, are concave, the following algorithm, based on successive convex programming, outputs a max-min fair solution.

Algorithm 2.1. (*max-min fairness, convex case*)

Step 0: Assign $\mathcal{N} := \{1, 2, \ldots, m\}$. Solve

$$\begin{aligned} \max \quad & t & & (2.21)\\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \geq t & \forall\ i & (2.22)\\ & \boldsymbol{x} \in X \ , & & (2.23) \end{aligned}$$

and let t^* and $\boldsymbol{x}^*$ be the optimal solution.

Step 1: For all $j \in \mathcal{N}$ solve

$$\begin{aligned} \mu_j = \max \quad & f_j(\boldsymbol{x}) & & (2.24)\\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \geq t_i & \forall\ i \notin \mathcal{N} & (2.25)\\ & f_i(\boldsymbol{x}) \geq t^* & \forall\ i \in \mathcal{N} & (2.26)\\ & \boldsymbol{x} \in X \ . & & (2.27) \end{aligned}$$

For all $j \in \mathcal{N}$, if $\mu_j = t^*$ then $\mathcal{N} := \mathcal{N} \setminus \{j\}$ and $t_j := t^*$. If $\mathcal{N} = \emptyset$ then stop, $\boldsymbol{x}^*$ is the max-min fair solution. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned} \max \quad & t & & (2.28)\\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \geq t_i & \forall\ i \notin \mathcal{N} & (2.29)\\ & f_i(\boldsymbol{x}) \geq t & \forall\ i \in \mathcal{N} & (2.30)\\ & \boldsymbol{x} \in X \ , & & (2.31) \end{aligned}$$

and let t^* and $\boldsymbol{x}^*$ be the optimal solution. Go to Step 1.

The above algorithm is completely relying on the convex structure of the involved optimization problem. Note first of all that we are able to put a constraint of the type

$$f_i(\boldsymbol{x}) \geq t_i, \ t_i \text{ fixed}, \tag{2.32}$$

specific for entry i, only because of concavity of f_i. The check of which entries that can be increased, performed in Step 1, is valid only as long as X is convex and the functions $f_1, f_2, \ldots, f_m$ are concave. The function of this step can be seen if we consider all $i \in \mathcal{N}$ such that $\mu_i > t^*$. Denote this set of indices by I. The value of the outcomes corresponding to $i \in I$ can all be increased simultaneously as there exists an $\bar{\boldsymbol{x}} \in X$ and a $\tau > t^*$, such that

$$f_i(\bar{\boldsymbol{x}}) \geq \tau \qquad \forall\ i \in I\ , \tag{2.33}$$

$$f_i(\bar{\boldsymbol{x}}) \geq t^* \qquad \forall\ i \in \mathcal{N} \setminus I\ , \tag{2.34}$$

$$f_i(\bar{\boldsymbol{x}}) \geq t_i \qquad \forall\ i \notin \mathcal{N}\ . \tag{2.35}$$

Such a solution $\bar{\boldsymbol{x}}$ is the convex combination, $\bar{\boldsymbol{x}} = \sum_{j \in I} \frac{1}{|I|} \boldsymbol{x}_j$, where $\boldsymbol{x}_j$ is the solution to the convex programme of Step 1, performed for entry $j \in I$. Let $\tau = \min\{\mu_i : i \in I\}$. Because of convexity of X, we have $\bar{\boldsymbol{x}} \in X$ and because of concavity of functions $f_1, f_2, \ldots, f_m$, that

$$f_i\Big(\ \textstyle\sum_{j \in I} \frac{1}{|I|} \boldsymbol{x}_j\ \Big) \geq \displaystyle\sum_{j \in I} \frac{1}{|I|} f_i(\boldsymbol{x}_j) \geq \tau \qquad \forall\ i \in I\ , \tag{2.36}$$

$$f_i\Big(\ \textstyle\sum_{j \in I} \frac{1}{|I|} \boldsymbol{x}_j\ \Big) \geq \displaystyle\sum_{j \in I} \frac{1}{|I|} f_i(\boldsymbol{x}_j) \geq t^* \qquad \forall\ i \in \mathcal{N} \setminus I\ , \tag{2.37}$$

$$f_i\Big(\ \textstyle\sum_{j \in I} \frac{1}{|I|} \boldsymbol{x}_j\ \Big) \geq \displaystyle\sum_{j \in I} \frac{1}{|I|} f_i(\boldsymbol{x}_j) \geq t_i \qquad \forall\ i \notin \mathcal{N}\ . \tag{2.38}$$

If it is possible to somehow access optimal dual multipliers for constraints (2.30), λ_i^*, $i \in \mathcal{N}$, the algorithm can be made substantially faster, as we then can omit the time-consuming Step 1. Let t^* and $\boldsymbol{x}^*$ be the primal optimal solution to (2.28)-(2.31), and let $\Lambda^* = [\lambda_i^*]_{i \in \mathcal{N}}$, be the corresponding vector of optimal dual multipliers for constraints (2.30).

Proposition 2.13. *If $\lambda_i^* > 0$ for $i \in \mathcal{N}$, then $f_i(\boldsymbol{x}^*) = t^*$ for any optimal solution $\boldsymbol{x}^*$ to* (2.28)-(2.31).

Proof. For any optimal primal-dual pair (saddle-point), $(\boldsymbol{x}^*, t^*; \Lambda^*)$, because of the convex structure, the complementary slackness property holds [20],

$$\lambda_i^*\ \big(\ t^* - f_i(\boldsymbol{x}^*)\ \big) = 0\ , \quad \text{for each } i \in \mathcal{N}. \tag{2.39}$$

Hence if $\lambda_i^* > 0$ then necessarily $f_i(\boldsymbol{x}^*) = t^*$, for any optimal primal solution $\boldsymbol{x}^*$. □

The improved algorithm is as follows.

Algorithm 2.2. (*max-min fairness, convex case*)

Step 0: Assign $\mathcal{N} := \{1, 2, \ldots, m\}$. Solve

$$\begin{aligned} \max \quad & t & & (2.40)\\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \geq t & \forall\ i & (2.41)\\ & \boldsymbol{x} \in X\ , & & (2.42) \end{aligned}$$

and let t^* and $\boldsymbol{x}^*$ be the optimal solution, and $\Lambda^* = [\lambda_i^*]_{i=1,2,\ldots,m}$ be the optimal dual multipliers corresponding to constraints (2.41).

Step 1: For all $i \in \mathcal{N} : \lambda_i^* > 0$ put $\mathcal{N} := \mathcal{N} \setminus \{i\}$ and $t_i := t^*$. If $\mathcal{N} = \emptyset$ then stop, $\boldsymbol{x}^*$ is the max-min fair solution. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned} \max \quad & t & & (2.43)\\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \geq t_i & \forall\ i \notin \mathcal{N} & (2.44)\\ & f_i(\boldsymbol{x}) \geq t & \forall\ i \in \mathcal{N} & (2.45)\\ & \boldsymbol{x} \in X\ , & & (2.46) \end{aligned}$$

and let t^* and $\boldsymbol{x}^*$ be the optimal solution, and $\Lambda^* = [\lambda_i^*]_{i\in\mathcal{N}}$ be the optimal dual multipliers corresponding to constraints (2.45). Go to Step 1.

Unfortunately, the fact that $\lambda_j^* = 0$ does not in general imply that $f_j(\boldsymbol{x})$ can be increased further. Nevertheless, Algorithm 2.2 will keep such an index in the set $\mathcal{N}$, i.e., treating it as it potentially could be increased, which of course is unnecessary if the corresponding constraint is binding. Such a strategy will work since there will always be at least one binding constraint i out of constraints (2.45) (and also out of constraints (2.41) in Step 0), and, as will be shown, $\sum_{i\in\mathcal{N}} \lambda_i^* = 1$ (and $\sum_i\ \lambda_i^* = 1$ in Step 0).

Property 2.14. $\sum_{i\in\mathcal{N}} \lambda_i^* = 1$.

Proof. Dualize constraints (2.45), and form the Lagrangian,

$$L(\boldsymbol{x}, \Lambda) = t - \sum_{i\in\mathcal{N}} \lambda_i \left(\ t - f_i(\boldsymbol{x})\ \right) = t \left(\ 1 - \textstyle\sum_{i\in\mathcal{N}} \lambda_i\ \right) + \displaystyle\sum_{i\in\mathcal{N}} \lambda_i f_i(\boldsymbol{x})\ , \tag{2.47}$$

and the dual function,

$$W(\Lambda) = \max_{\boldsymbol{x}\in X} L(\boldsymbol{x}, \Lambda)\ , \quad \Lambda \geq \mathbf{0}\ . \tag{2.48}$$

Suppose that $\sum_{i\in\mathcal{N}} \lambda_i^* \neq 1$. Then the bounded function $W(\Lambda)$ may be made infinitely large by assuming $t = +\infty$ or $t = -\infty$, which is a contradiction. □

The consequence of this treatment of indices j, for which $\lambda_j^* = 0$, will result in that it may happen that we sometimes carry out Step 2 without increasing t (i.e., not increasing with respect to the value of t obtained the last time Step 2 was visited). For practical instances the occurence of $\lambda_j^* = 0$ has however shown to be quite rare.

2.4.2 Non-convex cases

Whenever the feasible set, X, is not convex or if the criteria $\boldsymbol{f}$ are not concave, we cannot make use of neither Algorithm 2.1 nor Algorithm 2.2. When this is the case the problem will have a more complicated non-convex structure. This may for instance happen if X only contains integers.

Cumulated ordered outcomes

In the non-convex cases it is possible to make use of the expressability result for cumulated ordered outcomes, given by (2.5)-(2.6). It is fairly easy to see that a solution to

$$\text{lex max}\{\bar{\Theta}(\boldsymbol{f}(\boldsymbol{x})) : \boldsymbol{x} \in X\} \ ,$$

is also a solution to

$$\text{lex max}\{\Theta(\boldsymbol{f}(\boldsymbol{x})) : \boldsymbol{x} \in X\} \ ,$$

and vice versa. The latter should be recognized as the general max-min fairness problem. Given a vector of outcomes, $\boldsymbol{f}(\boldsymbol{x})) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$, the cumulated values, $\bar{\theta}_i(\boldsymbol{f}(\boldsymbol{x}))$, $i = 1, 2, \ldots, m$, can be expressed as

$$\begin{aligned}
\bar{\theta}_k(\boldsymbol{f}(\boldsymbol{x})) = \max \quad & kt - \sum_{i=1}^{m} b_i & & (2.49)\\
\text{s.t.} \quad & t - f_i(\boldsymbol{x}) \leq b_i & \forall\ i & (2.50)\\
& b_i \geq 0 & \forall\ i & (2.51)
\end{aligned}$$

which makes it possible to write any max-min fairness problem as

$$\begin{aligned}
\text{lex max} \quad & (\eta_1, \eta_2, \ldots, \eta_m) & & (2.52)\\
\text{s.t.} \quad & \eta_k \leq kt_k - \sum_{i=1}^{m} b_{ik} & \forall\ k & (2.53)\\
& t_k - b_{ik} \leq f_i(\boldsymbol{x}) & \forall\ i,k & (2.54)\\
& b_{ik} \geq 0 & \forall\ i,k & (2.55)\\
& \boldsymbol{x} \in X & & (2.56)
\end{aligned}$$

where $k = 1, 2, \ldots, m$. This multi-criteria optimization problem can be solved acccording to the following sequential algorithm.

Algorithm 2.3. (*max-min fairness, non-convex case*)

Step 0: Assign $k := 1$. Solve

$$\begin{aligned} \max \quad & \eta_1 && && (2.57)\\ \text{s.t.} \quad & \eta_1 \leq t - \sum_{i=1}^{m} b_i && && (2.58)\\ & t - f_i(\boldsymbol{x}) \leq b_i && \forall\ i && (2.59)\\ & b_i \geq 0 && \forall\ i && (2.60)\\ & \boldsymbol{x} \in X && && (2.61) \end{aligned}$$

and let η_1^* be the optimal objective value, and $\boldsymbol{x}^*$ the optimal solution.

Step 1: Assign $k := k + 1$. If $k > m$ then stop, $\boldsymbol{x}^*$ is the max-min fair solution. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned} \max \quad & \eta_k && && (2.62)\\ \text{s.t.} \quad & \eta_j^* \leq jt_j - \sum_{i=1}^{m} b_{ij} && j = 1, 2, \ldots, k-1 && (2.63)\\ & \eta_k \leq kt_k - \sum_{i=1}^{m} b_{ik} && && (2.64)\\ & t_j - b_{ij} \leq f_i(\boldsymbol{x}) && j = 1, 2, \ldots, k-1,\ \forall\ i && (2.65)\\ & t_k - b_{ik} \leq f_i(\boldsymbol{x}) && \forall\ i && (2.66)\\ & b_{ij} \geq 0 && j = 1, 2, \ldots, k,\ \forall\ i && (2.67)\\ & \boldsymbol{x} \in X && && (2.68) \end{aligned}$$

and let η_k^* be the optimal objective value, and $\boldsymbol{x}^*$ the optimal solution. Go to Step 1.

Provided that the optimization problems of Step 0 and Step 2 can be efficiently solved, Algorithm 2.3 actually provides a generic way of obtaining a max-min fair solution. On the other hand, if the feasible set X is non-convex, or if the outcomes $\boldsymbol{f}$ are non-concave, efficient solvers are rare.

Discrete feasible set

Some applications prescribe that the optimal outcomes $\boldsymbol{f}$ should belong to a discrete, finite set, $v = \{0, v_1, v_2, \ldots, v_r\}$, where it assumed that $0 < v_1 < v_2 < \ldots < v_r$. Finding a max-min fair solution is then a matter of solving

$$\begin{aligned} \text{lex max} \quad & \Theta(\boldsymbol{f}(\boldsymbol{x})) && && (2.69)\\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \in v && \forall\ i && (2.70)\\ & \boldsymbol{x} \in X\ . && && (2.71) \end{aligned}$$

It can then in many cases be useful to put this formulation as a lexicographical minimization (see Definition 2.15 for the definition of lexicographical maximization, lexicographical minimization is defined analogously), using the following straightforward Property.

Property 2.15. *A vector $\boldsymbol{x}^* \in X$ is an optimal solution to* (2.69)-(2.71) *if it is an optimal solution to*

$$\begin{aligned}
\text{lex min} \quad & \left(\textstyle\sum_i t_{1i}, \sum_i t_{2i}, \ldots, \sum_i t_{ri} \right) && && (2.72)\\
\text{s.t.} \quad & v_k - f_i(\boldsymbol{x}) \leq t_{ki} && k = 1, 2, \ldots, r, \ \forall\ i && (2.73)\\
& t_{ki} \geq 0 && k = 1, 2, \ldots, r, \ \forall\ i && (2.74)\\
& \boldsymbol{x} \in X && && (2.75)\\
& f_i(\boldsymbol{x}) \in v && \forall\ i && (2.76)
\end{aligned}$$

Proof. Let $\boldsymbol{x}^a$ be an optimal solution to (2.69)-(2.71) and $\boldsymbol{x}^b$ an optimal solution to (2.72)-(2.76). Property 2.15 states that $\Theta(\boldsymbol{f}(\boldsymbol{x}^a)) = \Theta(\boldsymbol{f}(\boldsymbol{x}^b))$. As we must have that $\Theta(\boldsymbol{f}(\boldsymbol{x}^a)) \succeq \Theta(\boldsymbol{f}(\boldsymbol{x}^b))$, we only need to prove that $\Theta(\boldsymbol{f}(\boldsymbol{x}^b)) \succeq \Theta(\boldsymbol{f}(\boldsymbol{x}^a))$. Suppose that $\Theta(\boldsymbol{f}(\boldsymbol{x}^a)) \succ \Theta(\boldsymbol{f}(\boldsymbol{x}^b))$. Then there exists an entry j such that $\theta_j(\boldsymbol{f}(\boldsymbol{x}^b)) < \theta_j(\boldsymbol{f}(\boldsymbol{x}^a)) = v_t \in v$, and $\theta_i(\boldsymbol{f}(\boldsymbol{x}^b)) = \theta_i(\boldsymbol{f}(\boldsymbol{x}^a))$ for all indices $i < j$ (if such indices exist). It then holds that

$$\sum_i \max\{v_s - f_i(\boldsymbol{x}^a), 0\} = \sum_i \max\{v_s - f_i(\boldsymbol{x}^b), 0\},$$

for all s such that $s < t$, and that

$$\sum_i \max\{v_t - f_i(\boldsymbol{x}^a), 0\} < \sum_i \max\{v_t - f_i(\boldsymbol{x}^b), 0\},$$

which contradicts that $\boldsymbol{x}^b$ is a solution to (2.72)-(2.76). □

2.5 Conclusions

In this chapter we have introduced the concept of fairness in a formal way. Particularly, the concept of one vector being more fair than another has been rigorously defined through the notion of preference relations. For a preference relation to define fairness, we have assumed that it has to fulfill certain regularity requirements, and is in this case called a fair rational preference relation. From the associated rules we have developed a number of important basic types of fairness.

The objective of obtaining fairness has been shown to be a matter of finding a vector that is (weakly) preferred to all other vectors, with respect to the considered fair rational preference relation. Such a vector is called fairly nondominated, and is often obtainable through some kind of (possibly multicriteria) maximization. This provides a natural connection to

Pareto-efficiency, as it could be shown that all fairly non-dominated vectors can be seen as Pareto-efficient solutions to a special type of multi-criteria maximization problem.

In order to define a fair rational preference relation there exist several possible generic methods. The first, and probably most natural one, is to use the idea of aggregation functions. Other types of generic fairness, i.e., classes of fairness where parameter-settings define the particular fairness, have been presented. For example we have shown how to define fairness through p-norms and ordered weighted averaging.

The most thoroughly investigated type of fairness is the one that is often reasonably argued to be intuitively the most fair, namely max-min fairness. We gave a formal definition of max-min fairness, using the fair rational preference relation of the symmetrical leximin order. We then related this definition to a stronger but less general definition, and showed that the definitions are equivalent in the case of a convex outcome space. We did also highlight that a solution that induces a max-min fair outcome vector is sometimes called max-min fair itself. To the extreme it may even happen that the solution variables are the outcomes, a case often referred to as linear max-min fairness.

Finally, this chapter turned the focus to generic algorithms that achieve max-min fairness. For the convex case we gave an algorithm that is based on the use of dual multipliers, and for the non-convex case an algorithm that makes use of cumulated ordered values. The presented algorithms will be extensively used in the following chapters, as obtaining max-min fairness will be one of the main objectives of the considered applications.

Chapter 3

Convex allocation problems

In this chapter, we apply the material presented on max-min fairness in Chapter 2 on communication network allocation problems. Specifically, this chapter deals with allocation problems that have convex structure, i.e., allocation problems for which the associated optimization problem is convex. This requirement excludes for instance the more complicated cases when decision variables can be discrete (integer variables, binary variables). Allocation problems are the network design problems that are characterized by the property that network topology as well as link capacities are given a priori. Such problems consist in determination of how a number of different (competing) users should share the network resources, so that some given criterion is met (e.g., an objective function is optimized). Thus, an allocation problem is always a capacitated problem. As this chapter will address max-min fairness, the preference relation of leximin order will be considered exclusively, allowing us to drop subscript and use $\succeq$ to denote $\succeq_{lex}$.

3.1 Max-min fair allocation problems

3.1.1 Max-min fair flows on fixed paths

Assume given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each link $e = 1, 2, \ldots, E$ is assigned a certain capacity c_e. For each demand $d = 1, 2, \ldots, D$, suppose that we are given a predetermined, connecting path, specified by the link-demand incidence relation, δ_{ed}. The simplest max-min fairness problem is concerned with allocation of link capacity (bandwidth) to demands on their respective paths such that the resulting allocation vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ is max-min fair on the feasible set X (where the feasible set X means the associated set of constraints). Thus the flow variables x_d, $d = 1, 2, \ldots, D$, are the decision variables of this problem. Note that each flow variable is defined as an outcome, so this is actually an application of *linear* max-min fairness. The considered problem is described by

$$\text{lex max} \quad \Theta(\boldsymbol{x}) \tag{3.1}$$

$$\text{s.t.} \quad \sum_d \delta_{ed} x_d \leq c_e \qquad \forall\ e \tag{3.2}$$

where Constraints (3.2) assure that for each link its load (left-hand side) does not exceed the given capacity.

Definition 3.1. An allocation vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ is said to be *max-saturating* if each demand is maximal on a saturated link on its path, i.e., if for each demand $d', d' = 1, 2, \ldots, D$ there exists a link e such that $\sum_d \delta_{ed} x_d = c_e$ and $\delta_{ed'} = 1$ for which $x_{d'} = \max\{x_d : d = 1, 2, \ldots, D,\ \delta_{ed} = 1\}$.

The following property is sometimes used for characterizing max-min fairness [6], but is not generalizable as it is relevant only for the fixed paths case.

Property 3.1. *An allocation vector* $\boldsymbol{x}^0 = (x_1^0, x_2^0, \ldots, x_D^0)$ *solves* (3.1)-(3.2) *if and only if* $\boldsymbol{x}^0$ *is max-saturating. Moreover, such an allocation vector is unique.*

Proof. Suppose that $\boldsymbol{x}^0$ is not max-saturating. Then either there is a demand for which there is no saturated link on its path, or there is a demand that has saturated links on its path but is not maximal on any of them. The first possibility clearly contradicts that $\Theta(\boldsymbol{x}^0)$ is lexicographically maximal. The second possibility implies that on the demand's path there only exist saturated links on which another demand is dominating (is maximal). Hence by decreasing these dominating demands it is possible to increase the original smaller demand, which again contradicts that $\Theta(\boldsymbol{x}^0)$ is lexicographically maximal. Thus if $\boldsymbol{x}^0$ is a solution to (3.1)-(3.2) it is certainly max-saturating. Suppose that $\boldsymbol{x}^0$ is max-saturating and that there exists a feasible allocation vector $\boldsymbol{x}^1$ such that $\Theta(\boldsymbol{x}^1) \succ \Theta(\boldsymbol{x}^0)$. Then there must exist a demand d such that $x_d^1 > x_d^0$. Let d be such a demand, with the smallest possible value of x_d^0. Define the set S as $S = \{d' : x_{d'}^0 \leq x_d^0\}$. It holds that $x_{d'}^0 \leq x_{d'}^1$ for any $d' \in S$, because otherwise we would not have $\Theta(\boldsymbol{x}^1) \succ \Theta(\boldsymbol{x}^0)$. Consider the saturated link e on which demand d is maximal (for allocation $\boldsymbol{x}^0$). Clearly, $x_d^0 + \sum_{d' \in S} \delta_{ed'} x_{d'}^0 = c_e$. However, this link load is strictly less than the one implied by allocation $\boldsymbol{x}^1$, namely $x_d^1 + \sum_{d' \in S} \delta_{ed'} x_{d'}^1$, which contradicts the existence of $\boldsymbol{x}^1$. Concluding, if $\boldsymbol{x}^0$ is max-saturating it must be a solution to (3.1)-(3.2). It is easily checked that the feasible set X is convex. According to Property 2.12 if $\boldsymbol{x}^0$ is a solution to (3.1)-(3.2) (i.e., MMF on X), it must be strongly max-min fair. A strongly max-min fair vector is, according to Property 2.8, unique. □

The unique solution of problem (3.1)-(3.2) can be found by a well known procedure given by Algorithm 3.1. The algorithm is sometimes referred to as "waterfilling" [46]. To simplify notation, we let $\mathcal{N}$ denote the set of demands that can be further increased and $\mathcal{S}$ the set of saturated links.

Algorithm 3.1. (*for solving* (3.1)-(3.2))

Step 0: $\boldsymbol{x} := \boldsymbol{0}$; $\mathcal{N} := \{1, 2, \ldots, D\}$; $\mathcal{S} := \emptyset$;

Step 1: $t := \min \left\{ \frac{c_e}{\sum_{d \in \mathcal{N}} \delta_{ed}} : e \notin S \right\}$.

Step 2: $c_e := c_e - t(\sum_{d \in \mathcal{N}} \delta_{ed})$ for all $e : e \notin \mathcal{S}$, $x_d := x_d + t$ for all d such that $d \in N$. For all e with $c_e = 0$ and $e \notin \mathcal{S}$, $S := S \cup \{e\}$. For each link e' that has been added to $\mathcal{S}$, for each demand d' for which $\delta_{e'd'} = 1$, put $\delta_{ed'} = 0$ for $e = 1, 2, \ldots, E$ and $\mathcal{D} := \mathcal{D} \setminus \{d'\}$.

Step 3: If $\mathcal{D} = \emptyset$, then stop, otherwise go to Step 1.

The function of Algorithm 3.1 is very simple: the flows allocated to each demand are increased at an equal rate until a link is saturated. Then all the demands that use this link become "blocked", and cannot be increased further. The flows of these demands have reached their final value. The procedure is then repeated with the demands that are not blocked. This continues until all demands are blocked, i.e., until all flows have reached their final values. Thus Algorithm 3.1 constructs an allocation vector $\boldsymbol{x}$ that is max-saturating and therefore, by Property 3.1, the only solution to (3.1)-(3.2).

It should be noted that the quantity t, calculated in Step 1 of Algorithm 3.1, is the solution of the following LP:

$$\begin{aligned} \max \quad & t & & (3.3) \\ \text{s.t.} \quad & t\left(\ \textstyle\sum_d \delta_{ed}\ \right) \le c_e & \forall\ e & (3.4) \end{aligned}$$

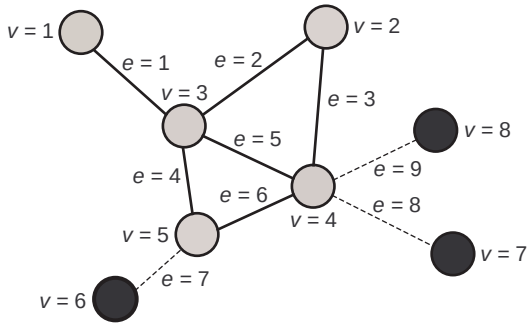
This simple observation will show to be useful later when more complicated MMF problems are addressed. A more general version of problem (3.1)-(3.2) is obtained if the constraints

$$h_d \le x_d \le H_d\ , \quad d = 1, 2, \ldots, D\ , \tag{3.5}$$

are added to the formulation. Here h_d and H_d are lower and upper bounds on the flow allowed to assign to demand d. There is an easy way of accounting for the upper bounds H_d. This can be done by adjusting the network graph before applying Algorithm 3.1. For each demand d an auxiliary leaf node and an extra link of capacity H_d are added; the link is then added to the path of demand d.

Example 3.1. Consider the network in Figure 3.1. Initially the network has $V = 5$ nodes, $E = 6$ links and $D = 3$ demands. To account for the upper bounds, H_d, $d = 1, 2, 3$, three new nodes, $v = 6, 7, 8$, and links, $e = 7, 8, 9$, with appropriate capacities are added, resulting in the augmented network.

To cope with the lower bounds h_d, modifications to Algorithm 3.1 have to be done. Although using identical ideas, such a modified algorithm is more difficult to describe as it involves more technical details. The algorithm can however be informally outlined as follows: Start by assigning all demands, d, their lower bound values, $x_d := h_d$. If for some link e, $\sum_d \delta_{ed} h_d > c_e$, the problem is infeasible. Otherwise we let $C = \{C_1, C_2, \ldots, C_K\}$ be the ordered set of distinct values h_d, $d = 1, 2, \ldots, D$. We define a set $\mathcal{D}$ of demands that should be increased. Initially, let $\mathcal{D} = \{d : h_d = C_1\}$. Then all the demands contained in $\mathcal{D}$ are increased at the same rate until a demand gets blocked or until the next element of C is reached. If a demand is blocked (by a saturated link), we simply leave this demand behind by removing it from $\mathcal{D}$



demand	node	node	path	leaf node	new path
$d = 1$	$v = 1$	$v = 5$	$\{1, 5, 6\}$	$v = 6$	$\{1, 5, 6, 7\}$
$d = 2$	$v = 1$	$v = 4$	$\{1, 4, 6\}$	$v = 7$	$\{1, 4, 6, 8\}$
$d = 3$	$v = 2$	$v = 4$	$\{3\}$	$v = 8$	$\{3, 9\}$

Figure 3.1: Augmented network, $c_7 = H_1$, $c_8 = H_2$, $c_9 = H_3$.

and continue. This demand has reached its final value. If the next element of C, call it C_k, is reached then all the demands d for which $h_d = C_k$ enter $\mathcal{D}$ and we continue the procedure. It may happen that $\mathcal{D}$ becomes empty before C_k is reached. In such a case we let $\mathcal{D} = \{d : h_d = C_k\}$. The procedure is continued until all demands have reached their final values.

3.1.2 Max-min fair flows on optimized paths

In this section we investigate how to obtain max-min fairness if there are several allowable paths on the routing list assigned to each demand, and the flow associated with a demand is allowed to split over its paths. This differs from the simpler problem considered in the previous section, as it was then assumed that each demand is restricted to one single fixed path. In general, flows realizing each demand volume, x_d, being allowed to split among the allowable paths will of course result in a sorted allocation vector, $\Theta(\boldsymbol{x})$, lexicographically greater than the one solving the fixed paths case. Formally we consider the following problem, where z_{dp}, $d = 1, 2 \ldots, D$, $p = 1, 2, \ldots, P_d$, are the considered variables.

$$\text{lex max} \quad \Theta(\boldsymbol{x}) \tag{3.6}$$

$$\text{s.t.} \quad \sum_p z_{dp} = x_d \qquad \forall\ d \tag{3.7}$$

$$\sum_d \delta_{edp} z_{dp} \leq c_e \qquad \forall\ e \tag{3.8}$$

The multi-criteria optimization problem formulated by (3.6)-(3.8) is substantially more difficult than its fixed path counterpart (3.1)-(3.2), because

in general the quantity t can in this case not be computed by a unique-solution LP, as is done by (3.3)-(3.4). Instead, to compute t, the following LP has to be devised:

$$\begin{aligned} \max \quad & t && && (3.9)\\ \text{s.t.} \quad & \sum_p z_{dp} = x_d && \forall\ d && (3.10)\\ & t - x_d \leq 0 && \forall\ d && (3.11)\\ & \sum_d \delta_{edp} z_{dp} \leq c_e. && \forall\ e && (3.12) \end{aligned}$$

The following example shows that (3.9)-(3.12) may have several solutions.

Example 3.2. Consider the network in Figure 3.2. Assume given two demands; $d = 1$ between $v = 1$ and $v = 2$, and $d = 2$ between $v = 1$ and $v = 4$. Resolution of the LP formulated by (3.9)-(3.12) may result in two different solutions: *i*) $z_{11} = 1$ on path $\{2\}$, $z_{21} = 1$ on path $\{1, 4\}$, and *ii*) $z_{11} = 1$ on path $\{1, 3\}$, and $z_{21} = 1$ on path $\{2, 3, 4\}$. Solution *ii*) does not leave room for any improvement. However, solution *i*) leaves room for further increment of demand $d = 1$, facilitating the optimal (optimal for (3.6)-(3.8)) allocation vector: $z^0_{11} = 1$ (on path $\{2\}$), $z^0_{12} = 1$ (on path $\{1, 3\}$), and $z^0_{21} = 1$ (on path $\{1, 4\}$). The optimal allocation vector, $\boldsymbol{x}^0$, is equal to $(2, 1)$.

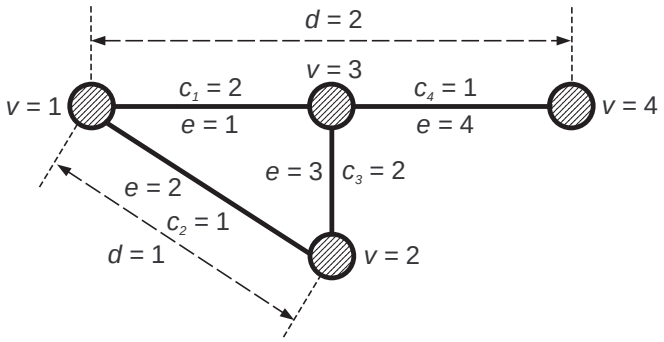


Figure 3.2: Multiple optimal allocation vectors.

Hence we can conclude that Algorithm 3.1 cannot be generalized to solve (3.6)-(3.8). Nevertheless, the feasible set is convex (and outcomes are linear and thus concave), so the general method for convex MMF problems given by Algorithm 2.1 can be applied. This general algorithm is application-specified by Algorithm 3.2.

Algorithm 3.2. (*for solving* (3.6)-(3.8))

Step 0: Assign $\mathcal{N} := \{1, 2, \ldots, D\}$. Solve

$$\begin{aligned} \max \quad & t && && (3.13)\\ \text{s.t.} \quad & \sum_p z_{dp} = x_d && \forall\ d && (3.14)\\ & t - x_d \leq 0 && \forall\ d && (3.15)\\ & \sum_d \sum_p \delta_{edp} z_{dp} \leq c_e && \forall\ e && (3.16) \end{aligned}$$

and let t^* and $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$ be the optimal solution.

Step 1: For all $d' \in \mathcal{N}$ solve

$$\begin{aligned} \mu_{d'} = \max \quad & x_{d'} && && (3.17)\\ \text{s.t.} \quad & x_d \geq t_d && \forall\ d \notin \mathcal{N} && (3.18)\\ & x_d \geq t^* && \forall\ d \in \mathcal{N} && (3.19)\\ & \sum_p z_{dp} = x_d && \forall\ d && (3.20)\\ & \sum_d \sum_p \delta_{edp} z_{dp} \leq c_e && \forall\ e && (3.21) \end{aligned}$$

For all $d' \in \mathcal{N}$, if $\mu_{d'} = t^*$ then $\mathcal{N} := \mathcal{N} \setminus \{d'\}$ and $t_{d'} := t^*$. If $\mathcal{N} = \emptyset$ then stop, $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$ is the max-min fair solution. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned} \max \quad & t && && (3.22)\\ \text{s.t.} \quad & x_d \geq t_d && \forall\ d \notin \mathcal{N} && (3.23)\\ & x_d \geq t && \forall\ d \in \mathcal{N} && (3.24)\\ & \sum_p z_{dp} = x_d && \forall\ d && (3.25)\\ & \sum_d \sum_p \delta_{edp} z_{dp} \leq c_e && \forall\ e && (3.26) \end{aligned}$$

and let t^* and $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$ be the optimal solution. Go to Step 1.

The function of Step 1 is to test whether a demand $d \in \mathcal{N}$ can be increased above the latest obtained t^* or not. Note that we are therefore not interested in the actual value of μ_d, but only whether it is larger or equal to t^*. It may of course happen that while solving (3.17)-(3.21) for a certain

demand $d' \in \mathcal{N}$, the resulting solution will have $x_{d''} > t^*$ for some other demand $d'' \in \mathcal{N}$. In that case we may just assign $\mu_{d''} := x_{d''}$ and neglect solving (3.17)-(3.21) for $d'' \in \mathcal{N}$.

Obviously, the LP solving-procedure associated with Step 1 should be performed using the basic LP solution resulting from Step 0 (or Step 2). As indicated in Section 2.4, the tests performed by Step 1 can be very time consuming. Particularly, with the application at hand, we may look at large networks with, say, thousand demands. In such a situation more efficient tests are called for. Besides the dual-multiplier based improvement described by Algorithm 2.2, it is possible to base a more efficient test on the properties of the Simplex tableau. The test of Step 1 of whether a demand $d \in \mathcal{N}$ can be increased above the latest obtained t^* or not can be done directly by examining the Simplex tableau corresponding to the final solution of Step 0 (or Step 2). We just have to examine the slack variable associated with the constraint in (3.15) (or (3.24)), corresponding to the currently considered demand $d' \in \mathcal{N}$. We will not give the details for such a tableau-based test, but instead give an application-specified version of Algorithm 2.2 for the considered problem. However, before specifying the algorithm based on dual multipliers we will make some assertions concerning the solution $\boldsymbol{x}^*$ of (3.6)-(3.8).

Property 3.2. *The allocation vector,* $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$, *being the final solution of problem* (3.22)-(3.26) *in Step 2, is unique. In general the unique vector* $\boldsymbol{x}^*$ *can be achieved by more than one optimal configuration of the individual flows,* z^*_{dp}, $d = 1, 2, ..., D$, $p = 1, 2, ..., P_d$.

This follows from the convexity of the constraint set (c.f. Property 2.12 and Property 2.8). Another property, implied by the number of basic variables in (3.22)-(3.26) and the operation of Simplex reads:

Property 3.3. *There exists a solution of* (3.22)-(3.26) *with at most* $D + E$ *non-zero flows,* z^*_{dp}, $d = 1, 2, ..., D$, $p = 1, 2, ..., P_d$.

A more complicated result is considered with the number of distinct entries of the optimal allocation vector $\boldsymbol{x}^0$, if all possible paths between node-pairs can be used [31].

Property 3.4 (Nace and Doan, 2005)**.** *If for each demand* $d = 1, 2, \ldots, D$ *all possible paths can be used, then the optimal allocation vector,* $\boldsymbol{x}^0$, *has at most* $V - 1$ *distinct entries.*

Proof. The usefulness of this result depends on if for the considered instance $V - 1 < D$, since there can obviously not be more distinct entries than there are entries. However, there are potentially as many as $V(V-1)/2$ demands in the network, so many important cases will be covered.

Let ϕ_1 be the smallest entry of $\boldsymbol{x}^0$, i.e., $\phi_1 = \theta_1(\boldsymbol{x}^0)$. Define X_1 as the set of all feasible solutions for which the smallest entry is not less than ϕ_1, i.e.,

$$X_1 = \{\boldsymbol{x} \in X : \theta_1(\boldsymbol{x}) \geq \phi_1\} \ , \tag{3.27}$$

where X is the set of all feasible solutions. Define further $s(\boldsymbol{x})$ as the set of links that are saturated by a solution $\boldsymbol{x} \in X$. For any $\boldsymbol{x} \in X_1$ suppose that $s(\boldsymbol{x})$ does not define a cut (for the notion of cuts, see [1]). Then it will be possible to increase any coordinate of $\boldsymbol{x}$, as subtracting the capacity occupied by $\boldsymbol{x}$ will leave room for additional flow between any node-pair, which contradicts that $\boldsymbol{x} \in X_1$. Hence $s(\boldsymbol{x})$ defines a cut for any $\boldsymbol{x} \in X_1$.

Define further

$$S_1 = \bigcap_{\boldsymbol{x} \in X_1} s(\boldsymbol{x}) \ . \tag{3.28}$$

Generate a set Y_1 as follows. Start by assigning $Y_1 = \emptyset$. For each cut c that is generated by some $\boldsymbol{x} \in X$, i.e., a cut c for which $c = s(\boldsymbol{x})$ for some $\boldsymbol{x} \in X_1$, pick exactly one representative solution $\boldsymbol{x}' \in X_1$ such that $c = s(\boldsymbol{x}')$, and add it to Y_1, $Y_1 := Y_1 \cup \{\boldsymbol{x}'\}$. This will give a finite set of solutions $Y_1 \subseteq X_1$ (as there are finitely many cuts in a graph), and it will hold that

$$S_1 = \bigcap_{\boldsymbol{x} \in X_1} s(\boldsymbol{x}) = \bigcap_{\boldsymbol{x} \in Y_1} s(\boldsymbol{x}) \ . \tag{3.29}$$

Suppose that S_1 does not define a cut. This implies that $s(\boldsymbol{x}^c)$, $\boldsymbol{x}^c = \sum_{\boldsymbol{x} \in Y_1} \frac{1}{|Y_1|} \boldsymbol{x}$, does not define a cut. But since X_1 is convex, $\boldsymbol{x}^c \in X_1$ and $\boldsymbol{x}^c$ must define a cut, which is a contradiction. Thus S_1 defines a cut.

Let ϕ_k be the k:th distinct entry of $\Theta(\boldsymbol{x}^0)$. For each ϕ_k define a set X_k as

$$X_k = \left\{ \boldsymbol{x} \in X : \begin{array}{ll} \theta_j(\boldsymbol{x}) = \theta_j(\boldsymbol{x}^0) & \text{for } j \text{ such that } \theta_j(\boldsymbol{x}^0) \leq \phi_k \\ \theta_j(\boldsymbol{x}) \geq \phi_k & \text{for } j \text{ such that } \theta_j(\boldsymbol{x}^0) > \phi_k \end{array} \right\} \tag{3.30}$$

In other words, the set X_k is the set of solutions $\boldsymbol{x} \in X$ for which $\Theta(\boldsymbol{x})$ is lexicographically greater or equal to

$$(\theta_1(\boldsymbol{x}^0), \theta_2(\boldsymbol{x}^0), \ldots, \theta_t(\boldsymbol{x}^0), \phi_k, \ldots, \phi_k) \ ,$$

where t is the largest index for which $\theta_t(\boldsymbol{x}^0) < \phi_k$. Let

$$S_k = \bigcap_{\boldsymbol{x} \in X_k} s(\boldsymbol{x}) \ . \tag{3.31}$$

Now assume that $\boldsymbol{x}^0$ has more than n distinct entries and suppose that the graph $\mathcal{G}_n = (\mathcal{V}, \mathcal{E} \setminus S_n)$ has K components (disjoint subgraphs), and consider any solution $\boldsymbol{x} \in X_{n+1}$. Then it must hold that the graph $(\mathcal{V}, \mathcal{E} \setminus s(\boldsymbol{x}))$ has at least $K + 1$ components. Because suppose that it has K components.

This must then be the K components induced by S_n, as $\boldsymbol{x} \in X_{n+1}$ implies $\boldsymbol{x} \in X_n$. Clearly, such an $\boldsymbol{x}$ will be possible to increase in any coordinate for which the corresponding node-pair is not cut by S_n. This contradicts that $\boldsymbol{x} \in X_{n+1}$.

Generate a set Y_{n+1} as follows. Start by assigning $Y_{n+1} = \emptyset$. For each partition P of the graph generated by any solution $\boldsymbol{x} \in X_{n+1}$, i.e., $P = s(\boldsymbol{x})$, $\boldsymbol{x} \in X_{n+1}$, pick exactly one representative solution $\boldsymbol{x}' \in X_{n+1}$ such that $P = s(\boldsymbol{x}')$ and add it to Y_{n+1}, $Y_{n+1} := Y_{n+1} \cup \{\boldsymbol{x}'\}$. This will give a finite set of solutions $Y_{n+1} \subseteq X_{n+1}$, and it holds that

$$S_{n+1} = \bigcap_{\boldsymbol{x} \in X_{n+1}} s(\boldsymbol{x}) = \bigcap_{\boldsymbol{x} \in Y_{n+1}} s(\boldsymbol{x}) \ . \tag{3.32}$$

As $S_n \subseteq S_{n+1}$ the graph $\mathcal{G}_{n+1} = (\mathcal{V}, \mathcal{E} \setminus S_{n+1})$ must have at least K components. Suppose that $\mathcal{G}_{n+1}$ has exactly K components. This implies that $s(\boldsymbol{x}^c)$, $\boldsymbol{x}^c = \sum_{\boldsymbol{x} \in Y_{n+1}} \frac{1}{|Y_{n+1}|} s(\boldsymbol{x})$, induces K components. But due to convexity of X_{n+1}, $\boldsymbol{x}^c \in X_{n+1}$, and $s(\boldsymbol{x}^c)$ must thus induce at least $K+1$ components, so this is a contradiction.

Per definition, every set X_k , $k = 1, 2, \ldots M$, corresponds to a distinct entry ϕ_k of $\Theta(\boldsymbol{x}^0)$. Suppose that $M > V - 1$ and that X_M is non-empty and thus that X_{V-1} is non-empty. If X_{V-1} is non-empty we have, for any $\boldsymbol{x} \in X_{V-1}$, that $s(\boldsymbol{x})$ induces V components, meaning that $s(\boldsymbol{x}) = \mathcal{E}$. Thus there are no solutions $\boldsymbol{x}' \in X$ such that $\Theta(\boldsymbol{x}') \succ \Theta(\boldsymbol{x})$, meaning that the set X_M must be empty, and we have a contradiction. This completes the proof. □

Improving the algorithm using dual multipliers

The vector $\Lambda^* = (\lambda_1^*, \lambda_2^*, \ldots, \lambda_D^*)$ of the optimal dual variables corresponding to constraints (3.15) (or the vector $\Lambda^* = [\lambda_d^*]$, $d \in \mathcal{N}$, corresponding to (3.24)) constitutes the key tool in improving Algorithm 3.2. The general dual-multiplier based algorithm (Algorithm 2.2) is application-specified for optimization problem (3.6)-(3.8) by the following algorithm.

Algorithm 3.3. (*for solving* (3.6)-(3.8))

Step 0: Assign $\mathcal{N} := \{1, 2, \ldots, D\}$. Solve

$$\max \quad t \tag{3.33}$$

$$\text{s.t.} \quad \sum_p z_{dp} = x_d \qquad \forall\ d \tag{3.34}$$

$$t - x_d \leq 0 \qquad \forall\ d \tag{3.35}$$

$$\sum_d \sum_p \delta_{edp} z_{dp} \leq c_e \qquad \forall\ e \tag{3.36}$$

and let t^* and $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$ be the optimal solution, and $\Lambda^* = [\lambda^*_d]_{d=1,2,\ldots,D}$ be the optimal dual multipliers corresponding to constraints (3.35).

Step 1: For all $d \in \mathcal{N} : \lambda^*_d > 0$ assign $\mathcal{N} := \mathcal{N} \setminus \{d\}$ and $t_d := t^*$. If $\mathcal{N} = \emptyset$ then stop, $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$ is the max-min fair solution. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned}
\max \quad & t && && (3.37)\\
\text{s.t.} \quad & x_d \geq t_d && \forall\ d \notin \mathcal{N} && (3.38)\\
& x_d \geq t && \forall\ d \in \mathcal{N} && (3.39)\\
& \sum_p z_{dp} = x_d && \forall\ d && (3.40)\\
& \sum_d \sum_p \delta_{edp} z_{dp} \leq c_e && \forall\ e && (3.41)
\end{aligned}$$

and let t^* and $\boldsymbol{x}^* = \left(\sum_p z^*_{1p}, \sum_p z^*_{2p}, \ldots, \sum_p z^*_{Dp} \right)$ be the optimal solution, and $\Lambda^* = [\lambda^*_d]_{d \in \mathcal{N}}$ be the optimal dual multipliers corresponding to constraints (3.39). Go to Step 1.

As problem (3.6)-(3.8) is entirely linear, the optimzation problems solved in Step 0 and Step 2 will be linear programming problems. This has the obvious advantage that, since practically all LP solvers provide optimal dual multipliers along with the primal optimal solution, we can assume that we have easy access to dual multipliers, making Algorithm 3.3 an appealing way of solving the considered problem. It is however a fact that examining the numerical value of an optimal dual multiplier λ^*_d, $d \in \mathcal{N}$, will be useful only if $\lambda^*_d > 0$. In the case when $\lambda^*_d = 0$ we cannot say anything about the possibilities of increasing x_d. This is illustrated by the following example.

Example 3.3. Consider the network from Figure 3.3. In this case (3.6)-(3.8) will be solved already by Step 0 of Algorithm 3.3, yielding $x^*_1 = z^*_{11} = x^*_2 = z^*_{21} = x^*_3 = z^*_{31} = 1$ (there is only one path for each demand). Although none of the three demands can be further increased, the Simplex algorithm will always set to 0 one of the two first values in the vector $\Lambda^* = (\lambda^*_1, \lambda^*_2, \lambda^*_3)$, of the optimal multipliers corresponding to constraints (3.35), yielding one of the two optimal dual solutions: $\Lambda^* = (0, \frac{1}{2}, \frac{1}{2})$ or $\Lambda^* = (\frac{1}{2}, 0, \frac{1}{2})$.

Summing up, investigating dual multipliers is really a method of detecting the demands that are blocking (not possible to increase further), and not the ones that are non-blocking (possible to increase further), which would be even better. However, as is also explained in Section 2.4, assuming that all demands that are not shown to be blocking by the dual multiplier test

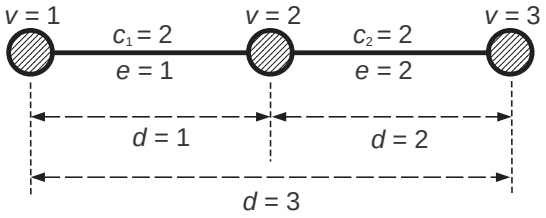


Figure 3.3: Two-link network.

are indeed non-blocking, will work even though we risk not to improve t^* in two consecutive iterations. This is because that $\sum_d \lambda_d^* = 1$, resulting in that each iteration will reveal at least one blocking demand, so we will eventually reach an iteration where every demand d for which $\lambda_d^* = 0$ is non-blocking.

Efficiency issues of Algorithm 3.3

Algorithm 3.3 combined with an efficient LP solver, provides a time-efficient and easy to implement tool for solving (3.6)-(3.8). Moreover, the maximal number of iterations of Algorithm 3.3, required to achieve the final solution is theoretically equal to the number of demands, D. In practice the algorithm rarely requires more than E (the number of links) iterations, as blocking demands with $\lambda_d^* = 0$ occur very rarely.

In the problem formulations considered in this section we have used the so called link-path formulation, assuming a limited list of paths currently predefined for each demand (paths $p = 1, 2, \ldots, P_d$ for demand d). In general, such lists do not contain (deliberately) all the paths allowable for the demand. For instance, if the allowable paths are, say, all the paths with up to ten hops, then there can exist thousands of such paths for a demand, still, typically, the current path list will contain a relatively small number of paths in order to limit the number of variables. Hence, it can happen that the current set of lists of predefined paths is not sufficient for reaching the optimal solution achievable within the full set of all allowable paths. To avoid this problem, it is of course possible to instead put the optimization problem in a so-called node-link formulation and account for each possible path for each demand, done e.g. in [30]. Taking each possible path into account can however be done in an equivalent, and substantially less computationally exhaustive way, using the efficient path generation[1] [9].

[1] In general linear programming this is called *column generation*.

Applying the algorithms on a numerical example

To illustrate the performance and function of Algorithm 3.3, we have considered a network with $V = 12$ nodes, $E = 18$ links, and $D = 66$ demands (demands are all undirected S-D pairs, numbered in the natural order: $\{1,2\},\{1,3\},\ldots,\{11,12\}$). Each demand is assigned the routing sequence of all simple paths, resulting in P_d values ranging from 6 to 13. To perform the numerical study, a large-scale interior-point solver provided by MATLAB Optimization Toolbox is used. Both Algorithm 3.1 (single pre-defined path case) and Algorithm 3.3 (optimized paths case) have been implemented. Figure 3.4 shows the successive demand volume allocations

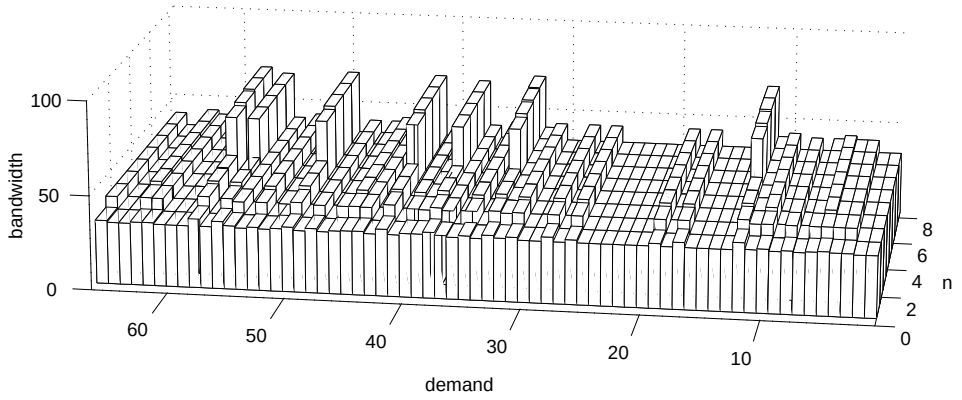


Figure 3.4: Evolution of allocated demand volumes using Algorithm 3.3.

x_d of Algorithm 3.3. In total, the experiment required 8 iterations and took less than 6 seconds of the CPU time on a PC with an Intel PIII-1GHz CPU, RAM of 256 MB, and Windows 2000 OS. The check of dual variables has been sufficient in all iterations (i.e., each iteration gave improvement of t^*). Note that the allocations are non-decreasing over the iterations, and that after Step 0 approximately equal volume is allocated to all demands. The benefit from using optimized paths for this network is illustrated in Figure 3.5. The left diagram illustrates the final demand volume allocation after applying Algorithm 3.1 (the predefined paths are the shortest paths in terms of the number of hops). This should be compared to the right diagram, which is the final allocation pattern of Algorithm 3.3 (equal to the last row in Figure 3.4). The throughput, standard deviation, and mean is equal to 2.2e3, 10.9, 33.0 and 2.6e3, 10.0, 39.6, for single and flexible paths cases, respectively. Notably, the total throughput of the flexible solution is greater than the throughput of the fixed-path solution by 16%.

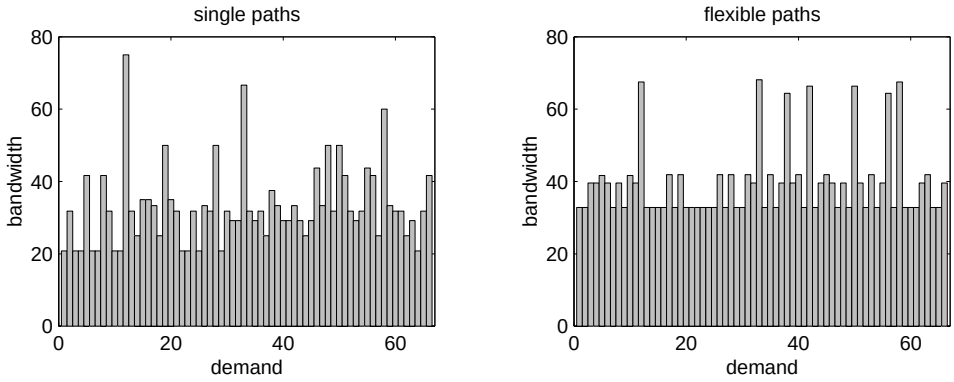


Figure 3.5: Volume allocated to each demand using single (fixed) and flexible (optimized) paths.

3.2 Max-min fair reallocation

Conventionally, research efforts targeting the issue of network robustness focus either on imposing resilience in the network design phase, or on augmenting existing network resources. For a given backbone transport network, both these approaches can be inadequate, simply because the network is in an operational phase and augmenting its resources may not be feasible. In this section we present ideas that take another approach, achieving robustness through a link protection mechanism, obtainable through reducing and rerouting the existing nominal demand flows, without altering neither network link capacities nor topology. We will show that the scheme of doing this can in a natural way be based upon the notion of max-min fairness.

The problem addressed in this section occurs when a network, designed and allocated in a non-robust way, encounters a link failure. Characteristically, such a network can be highly vulnerable, i.e., large fractions of connections disrupt due to events of this kind. The idea is to provide means of protecting (make robust) this type of network within its given resources, at the cost of fairly redistributing (reduce and possibly reroute) individual demand flows, still maintaining the overall demand volume relations. With this as the desired goal, we formulate and solve an optimization problem, maximizing the resemblence beteween original and reduced demand volumes, constrained on that the reduced demand volumes are fully protected for any single link failure. The solution to such an optimization problem may be further improved, and therefore we introduce a max-min fair improvement procedure. The suggested methodology is application of the convex-case max-min fairness algorithms.

3.2.1 Problem description

The methodology studied in this section is essentially concerned with investigating means for redistributing flows in an existing network with preallocated flows, *within* given link capacities, so that network robustness to link failures is introduced. Certain realistic assumptions about mean time between failures and mean time to repair in transport networks, justify the assumption that only one link fails at the time [53]. Adopting this reasoning advocates interpretation of the term "network robustness" as flow survival upon single link failures. In the preallocated network each source-destination (s-d) node pair is assigned a certain flow. The objective is to uniformly scale (and if necessary reroute) the preallocated flows such that in the event of a single failing link, each one of these (scaled) flows will be entirely restored using the link protection mechanism. By scaling the flows, we will obtain certain spare capacity on each link. This capacity, denoted *protection capacity*, is used to protect failing links, i.e., if a link fails, flows going through this link will be carried on protection capacity of other links, between the two nodes connected by the failed link. The rearranged network guarantees 100% Traffic Restoration Level (TRL) [2].

3.2.2 Notation and definitions

Consider a network given by a set of nodes and a set of connecting links that are also specified in terms of capacity c_e. It is assumed that we are given a set of demands (node-pairs), a set of paths for each demand (implicitly given by the link-path incidence relation, δ_{edp}), and a feasible allocation $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_D)$. The allocation $\hat{\boldsymbol{x}}$ is assumed to be feasible, meaning that there exists $\hat{z}_{dp}$ such that $\sum_p \hat{z}_{dp} = \hat{x}_d$, $d = 1, 2, \ldots, D$ and

$$\sum_d \sum_p \delta_{edp} \hat{z}_{dp} \leq c_e \ , \quad e = 1, 2, \ldots, E \ .$$

We want to find a number $t > 0$, uniformly scaling the preallocation and thereby liberating link capacity, necessary to ensure full restoration of any single link, i.e., to obtain compensation for the total failure of each link e, $e = 1, 2, \ldots, E$. We restrict ourselves to the study of pure link protection, meaning that in case of link failure, the aggregated flow carried by the link is rerouted on protection paths, thus requiring a link connectivity of at least 2 from the network, and that each link e is assigned a set of protection paths, identified by $q = 1, 2, \ldots, Q_e$. The links used by each protection path are determined by the link-path incidence relation, β_{feq}, where $\beta_{feq} = 1$ if link f belongs to path q, protecting link e, and 0 otherwise. For link e, consider a protecting flow, r_{eq}, i.e., a flow that protects link e on path q.

Definition 3.2. Let $\boldsymbol{x}$ be a feasible allocation. We say that $\boldsymbol{x}$ is fully protected by flows r_{eq} if

$$\sum_d \sum_p \delta_{edp} z_{dp} + \sum_q \beta_{efq} r_{fq} \leq c_e \ , \ e, f = 1, 2, \ldots, E, \ e \neq f \tag{3.42}$$

and

$$\sum_q r_{eq} \geq \sum_d \sum_p \delta_{edp} z_{dp} \ , \ e = 1, 2, \ldots, E \ . \tag{3.43}$$

Note that (3.42) relies on the single link failure assumption, and that (3.43) ensures that a fully protected allocation will have a TRL of 100%.

Definition 3.3. Assume a given feasible allocation $\hat{\boldsymbol{x}}$, and an allocation $\boldsymbol{x}$ that is fully protected by flows r_{eq}. We define the scaling factor, $t(\boldsymbol{x})$, with respect to $\hat{\boldsymbol{x}}$ as

$$t(\boldsymbol{x}) = \min \left\{ \ \frac{x_d}{\hat{x}_d} : d = 1, 2, \ldots, D \ \right\} \ . \tag{3.44}$$

For a given preallocation, $\hat{\boldsymbol{x}}$, we want to determine a protected allocation $\boldsymbol{x}$, including protecting flows, r_{eq}, so that the the protected allocation resembles the preallocation as much as possible. Therefore, maximization of the scaling factor, t, will be of major importance.

3.2.3 A lower bound

Before formulating an appropriate optimization problem, certain conclusions regarding how large scaling factor that can be obtained for a given allocation $\hat{\boldsymbol{x}}$, may be drawn.

Proposition 3.5. *For each link e, $e = 1, 2, \ldots, E$, denote by c_e the capacity of link e, by v and u the two nodes connected by link e, $v, u = 1, 2, \ldots, V$, $v \neq u$. In the unloaded network, let π_e denote the largest flow possible to transfer between v and u (the max-flow) on the protection paths of link e. There exists a protected allocation $\boldsymbol{x}$ such that*

$$t \geq \min \left\{ \ \frac{\pi_e}{c_e + \pi_e} : e = 1, 2, \ldots, E \ \right\} \ . \tag{3.45}$$

Proof. First, we note that it is sufficient to prove the existence of a restoration such that

$$t \geq \frac{\pi_e}{c_e + \pi_e} \ , \tag{3.46}$$

for all links e. Assume that we delete an arbitrary link e, of capacity c_e, carrying the flow ω_e in the preallocation, and form an $s - d$ cut between v and w and over the residual network. Furthermore, assume that we can measure the aggregated (preallocated) flow intersecting this cut, and that

we are able to discern the share incident on paths protecting link e. Denote this share by ϕ. Now, place the cut such that the share is minimal, and denote this flow by ϕ^* (cf. the notion of min-cuts [1]). By scaling all the flows incident on the protection paths by $\frac{\pi_e}{c_e+\pi_e}$, we open up for a connection between v and w of capacity

$$c'_e = \pi_e - \phi^* \frac{\pi_e}{c_e + \pi_e} \ . \tag{3.47}$$

Moreover, from the *Max-flow Min-cut theorem* we know that it certainly holds that $\phi^* \leq \pi_e$ [1], so

$$c'_e \geq \pi_e - \pi_e \frac{\pi_e}{c_e + \pi_e} = c_e \frac{\pi_e}{c_e + \pi_e} \geq \omega_e \frac{\pi_e}{c_e + \pi_e} \ .$$

Thus we have a restoration satisfying (3.46). □

It turns out that if the network is saturated by demands that each have one, single, allowable path, then the bound given by Proposition 3.5 is exact. Note that in this case the preallocated flows cannot be rearranged, but only thinned.

Proposition 3.6. *If the network is saturated by single-path demands, then for any protected allocation* $\boldsymbol{x}$*,*

$$t = \min \left\{ \ \tfrac{\pi_e}{c_e+\pi_e} : e = 1, 2, \ldots, E \ \right\} \ . \tag{3.48}$$

Proof. For a saturated network we have $\phi^* = \pi_e$, and $\omega_e = c_e$. Pick a link e for which $\min \left\{ \ \frac{\pi_e}{c_e+\pi_e} : e = 1, 2, \ldots, E \ \right\}$ is assumed. Assume that we restore link e by a factor $\alpha_1 > \frac{\pi_e}{c_e+\pi_e}$. This implies that we are forced to scale flows carried by links on link e's protection paths by $\alpha_2 < \frac{\pi_e}{c_e+\pi_e}$, since we must satisfy $c_e\alpha_1 + \pi_e\alpha_2 = \pi_e$. By assumption these links realize demands uniquely, so this would imply

$$t < \frac{\pi_e}{c_e + \pi_e} \ , \tag{3.49}$$

which contradicts Proposition 3.5. □

Under certain circumstances it is possible to find a scaling factor, t, close to 1, as the following example shows.

Example 3.4. Suppose that $c_e = C$ for all links e, and that for each link e we have at least n disjoint protection paths, i.e., that none of these n paths share a link. Then

$$t \geq \min \left\{ \ \frac{\pi_e}{c_e + \pi_e} : e = 1, 2, \ldots, E \ \right\} \geq \frac{Cn}{C + Cn} = \frac{n}{1 + n} \ .$$

If the number of protection paths, Q_e, for each link is reasonably small, determination of the corresponding max-flows, π_e, is a fairly simple process, requiring mere consideration of the network topology, with its corresponding link capacities. However, in a more complex network, with a large number of (possibly non-disjoint) protection paths for each link, we may have to use e.g. the Ford-Fulkerson algorithm [6] (for the original algorithm, see [11]). For its relative simplicity, the lower bound provided by Proposition 3.5 can be valuable in estimating a worst case redistribution loss. Given an *optimal* scaling factor t^* (see the corresponding LP in the following section), it also exposes links constituting *protectional bottlenecks*, i.e., links e, for which $\frac{\pi_e}{c_e+\pi_e} = t^*$. By improving these links' protection properties, e.g. by extension of corresponding sets of protection paths, (such that $\frac{\pi_e}{c_e+\pi_e} > t^*$ for all $e = 1, 2, \ldots, E$), the proposition guarantees a larger optimal scaling factor.

3.2.4 Optimal solutions

A linear programming formulation

The problem of resembling the preallocation, $\hat{\boldsymbol{x}}$, with the fully protected allocation, $\boldsymbol{x}$ (with protecting flows r_{eq}), can be addressed by formulation of an appropriate optimization problem. Using the notation introduced in the previous section, together with the auxiliary variables w_e and y_e, denoting protection capacity and nominal capacity for link e, $e = 1, 2, \ldots, E$, respectively, an LP with the decision variables z_{dp} and r_{eq}, can be formulated in the following way (where $f = 1, 2, \ldots, E$):

$$\begin{aligned}
\max \quad & t && && (3.50)\\
\text{s.t.} \quad & t \leq \frac{x_d}{\hat{x}_d} && \forall\ d && (3.51)\\
& x_d = \sum_p z_{dp} && \forall\ d && (3.52)\\
& w_e + y_e \leq c_e && \forall\ e && (3.53)\\
& y_e \leq \sum_q r_{eq} && \forall\ e && (3.54)\\
& \sum_q \beta_{feq} r_{eq} \leq w_f && \forall\ f, e : f \neq e && (3.55)\\
& \sum_d \sum_p \delta_{edp} z_{dp} = y_e && \forall\ e && (3.56)
\end{aligned}$$

Here, $\hat{x}_d = \sum_p \hat{z}_{dp}$ are the given preallocated demand volumes, and they are without loss of generality assumed to satisfy $\hat{x}_d > 0$, for all $d = 1, 2, \ldots, D$. Since this LP is bounded and feasible, it is clearly solvable, and will provide

a protected allocation $\boldsymbol{x}$ and associated protecting flows r_{eq}, with a maximal protection factor t, with respect to the preallocation $\hat{\boldsymbol{x}}$.

A max-min fair improvement

Since the solution to (3.50)-(3.56) is usually non-unique, there is some freedom in which solution to choose. Note that rearranging the network strictly according to the solution of (3.50)-(3.56), will most likely result in a poor link utilization, because there will be a lot of unused capacity. However, for the considered application, there is a natural way of improving this, exploiting the idea of max-min fairness. In doing so we first define the constraint set F. For simpler notation, denote by S a collection of flows, z_{dp}, and protecting flows, r_{eq}. We say that $S \in F$ (S is feasible) if and only if it holds that the associated flows, z_{dp}, and protecing flows, r_{eq}, satisfy (3.52)-(3.56). The LP of (3.50)-(3.56) can then be rewritten as:

$$\begin{aligned} \max \quad & t && && (3.57)\\ \text{s.t.} \quad & t \leq \frac{x_d}{\hat{x}_d} && \forall\ d && (3.58)\\ & S \in F && && (3.59) \end{aligned}$$

Now assume that (t^*, S^*) is an optimal solution to (3.50)-(3.56), and that $t^* = \frac{x_d^*}{\hat{x}_d}$, for some demand d, and that there exists a demand Δ, $\Delta \neq d$, such that $t^* < \frac{x_\Delta^*}{\hat{x}_\Delta}$. We may then decrease x_Δ^* such that $t^* = \frac{x_\Delta^*}{\hat{x}_\Delta}$, and still have an optimal solution. This observation has the following implication for our solutions: given an optimal solution (t^*, S^*) to (3.50)-(3.56), it may be possible to further increase some aggregated flows (corresponding to demands), x_d^*, (certainly not all of them), and still have an optimal solution. We will refer to the demands possible to further increase as *non-blocking*. Such an increment is of course desirable, and for this purpose we propose the following algorithm, being an application-specified version of Algorithm 2.2, in order to obtain a better solution than that obtained by only solving (3.50)-(3.56).

Algorithm 3.4.

Step 0: Assign $\mathcal{N} := \{1, 2, \ldots, D\}$. Solve

$$\begin{aligned} \max \quad & t && && (3.60)\\ \text{s.t.} \quad & t \leq \frac{x_d}{\hat{x}_d} && \forall\ d && (3.61)\\ & S \in F && && (3.62) \end{aligned}$$

and let t^*, z_{dp^*}, and r_{eq}^* be the optimal solution, and $\Lambda^* = [\lambda_d^*]_{d=1,2,\ldots,D}$ be the optimal dual multipliers corresponding to constraints (3.61).

Step 1: For all $d \in \mathcal{N} : \lambda_d^* > 0$ assign $\mathcal{N} := \mathcal{N} \setminus \{d\}$ and $t_d := t^*$. If $\mathcal{N} = \emptyset$ then stop, $x_d^* = \sum_p z_{dp}^*$, $d = 1, 2, \ldots, D$ are the flows corresponding to a maximal scaling factor, improved in a max-min fair way. Single link failures are entirely protected by (protecting) flows r_{eq}^*. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned} \max \quad & t & & (3.63)\\ \text{s.t.} \quad & t \leq \frac{x_d}{\hat{x}_d} & \forall\ d \in \mathcal{N} & (3.64)\\ & t_d \leq \frac{x_d}{\hat{x}_d} & \forall\ d \notin \mathcal{N} & (3.65)\\ & S \in F & & (3.66) \end{aligned}$$

and let t^*, z_{dp^*}, and r_{eq}^* be the optimal solution, and $\Lambda^* = [\lambda_d^*]_{d \in \mathcal{N}}$ be the optimal dual multipliers corresponding to constraints (3.64). Go to Step 1.

The function of this algorithm relies, as it is an application of Algorithm 2.2, on complementary slackness and the fact that

$$\sum_{d \in \mathcal{N}} \lambda_d = 1. \tag{3.67}$$

As all involved optimization problems are LPs, we may consider the optimal dual multipliers easily accessible. Note that, just as was the case of Algorithm 3.3, in the case of a demand $d \in \mathcal{N}$ for which $\lambda_d = 0$ we can not be certain of that this demand can be increased further (i.e., that it really is a non-blocking demand). This implies that it may happen that we do not increase t^* in two consecutive iterations. However, since (3.67) always holds, each iteration will reveal at least one new blocking demand.

Algorithm 3.4 provides a max-min fair distribution of degree of resemblence to the corresponding preallocated demand flows. This is conditioned on that capacity has to be reserved so that in case of failure of any single link, the link can be completely restored by protecting flows.

3.2.5 Numerical examples

To illustrate the sharpness of the lower bound provided by Proposition 3.5, Table 3.1 gives a comparison between optimal t, computed by (3.50)-(3.56), and the corresponding lower bound for t, for 11 different randomly generated networks with different settings. For all the included networks, the individual link capacities are set according to $c_e = 30 \pm \rho,\ e = 1, 2, \ldots, E$, where ρ is randomly chosen. For networks 1-10 (denoted net 1-10), $\rho \in \{0, 1, 2, 3, 4, 5\}$, and for network 11 (denoted net 11), $\rho \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. In all

the cases, the preallocation saturates the network, since it is obtained by application of the MMF allocation rule, as described in Section 3.1.2, to the network with demands between all pairs of nodes. It should be noted that

net	#nodes	#links	ν_{max}	ν_{min}	M	L	optimal t	bound	error (%)
1	6	8	3	2	1	2	0.4167	0.4167	0
2	8	10	3	2	2	2	0.4492	0.4333	3.5
3	10	17	4	2	3	3	0.4915	0.4237	14
4	12	19	4	2	4	4	0.5498	0.4407	25
5	14	23	4	2	5	4	0.4776	0.4444	7
6	16	30	5	2	5	5	0.4445	0.4333	3
7	18	25	5	2	6	5	0.3726	0.3265	14
8	20	35	5	2	7	5	0.4754	0.4386	8
9	22	39	5	2	7	5	0.4426	0.4426	0
10	24	46	5	2	7	5	0.4531	0.4333	5
11	13	23	5	2	5	4	0.4541	0.4032	13

Table 3.1: Example networks. $M = \max_d\{P_d\},\ L = \max_e\{Q_e\},\ \nu$ =node degree.

the first row of the table (network 1) verifies Proposition 3.6 (each demand has one admissible path and the link capacities are saturated by the preallocation, hence the optimal t and the lower bound must be equal). The rightmost column of Table 3.1 suggests that the lower bound is rather sharp.

Figure 3.6 illustrates the dynamics of Algorithm 3.4. The algorithm is applied to network 11 of Table 3.1. The bar diagram of Figure 3.6 shows evolution of aggregated flows, x_d, for all the demands, throughout the 5 required iterations. The innermost (white) row diplays the associated preallocation. It should be noted, that for each demand the height of the bars is non-decreasing, with the number of iterations. In the diagram of Figure 3.7, the resulting nominal and protection capacities – the auxiliary variables y_e and w_e – are shown together with the fixed link capacities c_e. It should be emphasized that link capacities are not always fully exploited. Since we require 100% TRL (which is the most restrictive possible setting), we cannot exploit the fraction of link capacity (c_e) that is impossible to fully protect. On average, 57.5% and 40.3% of the fixed link capacities are used for nominal- and protection capacity, respectively. Further, for this example, the fully protected allocated demand constitutes 56.8% of the preallocated demand (on average, minimally by 45.4% and maximally by 74.5%).

3.3 Conclusions

This chapter has been devoted to a study of allocation problems where the objective is (linear) max-min fairness. All of the considered allocation problems have convex structure, meaning that the feasible set of solutions

is convex and that the criteria are concave.

The most basic problem addresses max-min fairness of flow distribution between demands that have one fixed path each. We showed that in this case max-min fairness of flows is equivalent to that the allocation vector is max-saturating, and gave an algorithm which is fundamentally dependent on this equivalence. We then showed how this algorithm could be extended in order to take prerequired upper- and lower flow-bounds of demands into account.

A more difficult problem is encountered if each demand may use several paths and is allowed to split its flow over them. In this case we say that paths are optimized. It is then not possible to use the max-saturating property, and more complicated LP-based algorithms have to be devised. It was illustrated how the generic (convex-case) algorithms from Chapter 2 could be applied, and we gave examples showing their operation. In addition, we have derived some properties of the resulting problem solution, particularly concerned with the relation between different demand flows. The section was closed by giving an example that shows what can be gained in terms of fairness if paths are optimized (i.e. flows may split) and not kept fixed.

In the balance, we have introduced a reallocation scheme with the purpose of facilitating resilience. The reallocation scheme is based on max-min fairness. Also in this problem we assume that all flows may split over several paths. We assume that we are given a network that is preallocated, i.e., demands' flows and paths are given a priori, and that the task is to reallocate and reroute these flows such that we facilitate a special type of resilience to network failures. The type of resilience considered is full protection of every possible single link failure. The goal is to reallocate the (preallocated) flows such that they as much as possible resemble the preallocation, but also such that we make room (reserve capacity) for protection paths for a failing link. We gave a special measure for the resemblence and an accompanying lower bound. The lower bound may be useful in approximate calculations as well as in exposing weakly protected links. Finally, it has been shown and examplified how Algorithm 2.2 solves the stated problem.

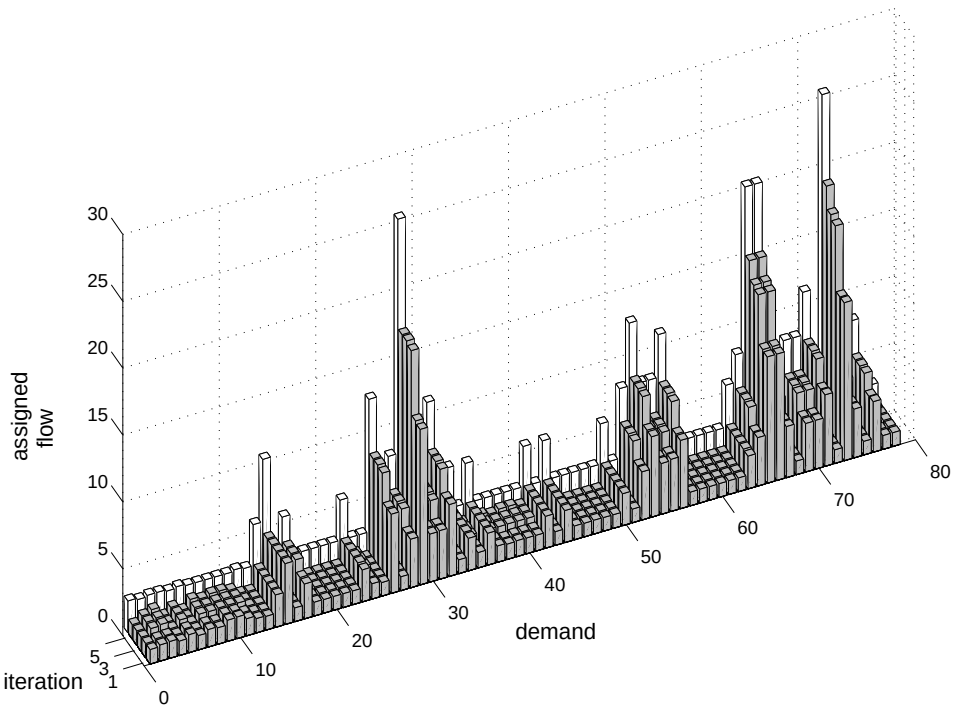


Figure 3.6: The evolution of flows allocated to demands.

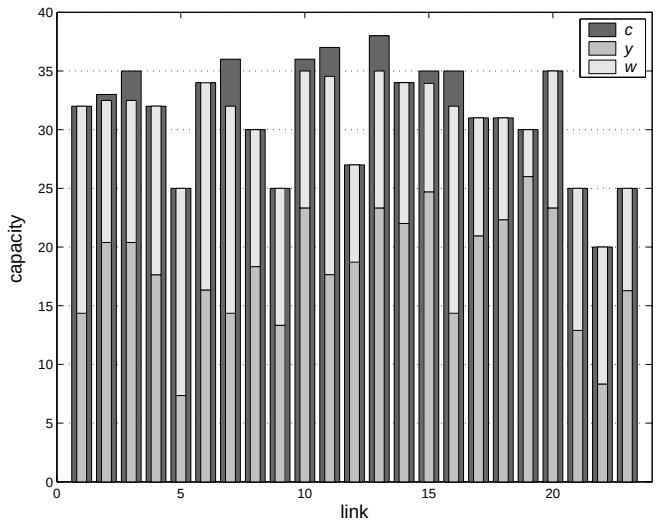


Figure 3.7: Resulting nominal- and protection capacities.

Chapter 4

Non-convex allocation problems

In Chapter 3 we considered allocation problems that had convex structure, targeting max-min fair distributions of demand flows or max-min fairness in sharing residual resources (after reducing original allocation). This chapter is devoted to the same type of max-min fair allocation problems, but with complicating assumptions making the resulting optimization problems non-convex. Specifically, all the problems considered in this chapter do all include the use of integer variables, in some form. More precisely, we are still considering problems that are classified under linear max-min fairness but with decision variables being restricted to integers. The reason for this is that in practice, decision variables, such as e.g. flow allocated to a demand, cannot take on an arbitrary value. Another realistic requirement might be that a flow between a node-pair cannot be split arbitrarily between connecting paths, but can only use one path, selected in the optimization process. Also this makes it necessary to model the allocation problem with integer variables, rendering substantially more complicated optimization problems.

4.1 Max-min fairness of modular flows

This section concerns max-min fair allocation of bandwidth to demands (users) in a communication network, when one (fixed in advance) path per demand is used, and when demands can be assigned bandwidth only in multiples of a predefined module. Although not to a great extent, this particular problem has been studied before. In [48] a polynomial time approximation (based on the notion of a preference relation induced by strong max-min fairness, see Remark 2.18) was presented, and more recently in [21] a metaheuristic was proposed. Both these approaches suggest that the problem is difficult to solve to optimality.

The consdidered problem is a practical extension of the frequently cited MMF problem addressed in [6], which was the simplest problem treated in Chapter 3 (more precisely the optimization problem formulated by (3.1)-(3.2)). Integral flow volumes can be well motivated from a practical viewpoint. This requirement takes into account that each demand volume must be a multiple of a predefined module, and is a consequence of that in a real network there is always a smallest trading unit, prohibiting flows from being continuous. As in Section 3.1.1, it will be assumed that a problem instance is given as a network with given link capacities, a set of source-destination node-pairs (S-D pairs), where each S-D pair represents a requirement for bandwidth (demand), and a path for each demand. For such an instance, the demand between each S-D pair must be assigned a *modular* flow volume, such that the sum of flows on each link does not violate the link capacity. The distribution of flows among the S-D pairs should obey the MMF principle.

4.1.1 Problem description

For the considered problem each demand d is assumed to be associated with one selected simple path. The binary indicator, δ_{ed}, is used for the link-demand incidence relation. The total flow allocated to demand d (on its corresponding path) will be identified by x_d, which will be the decision variable. As earlier, we will let $\succeq$ denote the leximin order, $\succeq_{lex}$.

The study undertaken in this section concerns the problem of assigning modular flows to demands in a capacitated network, such that the distribution of flows among demands is MMF. As has been described in Chapter 2, the MMF principle is to first assure that the demand that gets the least flow gets as much as possible, then that the demand that gets the second least flow gets as much as possible, and so on. Recall that obtaining a (linear) MMF allocation of flows is equivalent to solving

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\}, \tag{4.1}$$

where X is the set of feasible solutions. Put in words this means that in an MMF allocation of flows each demand is assigned a flow such that it holds for the sorted allocation vector that an entry can be increased only at the cost of decreasing a previous entry, or by making the allocation vector infeasible[1]. In this section, the set of feasible solutions X is defined by the following two requirements:

$$\sum_d \delta_{ed} x_d \leq c_e \ , \ e = 1, 2, \ldots, E, \text{ and} \tag{4.2}$$

$$x_d \in \mathbb{Z}_+ \ , \ d = 1, 2, \ldots, D, \tag{4.3}$$

where $\mathbb{Z}_+$ is the non-negative integers. The above two constraints mean in turn that the sum of flows on an link cannot exceed the link's capacity, and that each flow must assume a non-negative integer. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ be an allocation vector feasible for (4.2), and let $\boldsymbol{x}^z = (x_1^z, x_2^z, \ldots, x_D^z)$ be an allocation vector feasible for (4.2) and (4.3). We will denote an allocation vector $\boldsymbol{x}$ optimal for (4.1) constrained by (4.2) an *optimal continuous solution* (OCS), and allocation vector $\boldsymbol{x}^z$ optimal for (4.1) constrained by (4.2) *and* (4.3) an *optimal integral solution* (OIS). Note that solving for OCS is precisely what is addressed in [6], and is well known to be accomplished by Algorithm 3.1 (as it is exactly the problem described by (3.1)-(3.2)). Note further that if $\boldsymbol{x}$ is the OCS and $\boldsymbol{x}^z$ the OIS for the same instance, it must hold that $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^z)$.

Example 4.1. Consider the network given in Figure 4.1. One demand between each node-pair is assumed. Paths are evident. Link capacities are given in the figure. The sorted OCS is $\Theta(\boldsymbol{x}) = (0.5, 0.5, 0.5)$, whereas the sorted OIS is $\Theta(\boldsymbol{x}^z) = (0, 1, 1)$.

[1] Since this is a property of the sorted allocation vector entries and not for the specific demands, it is valid even for non-convex versions of the problem.

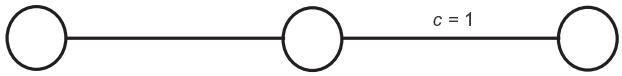


Figure 4.1: A simple instance.

4.1.2 The assumption of modular flows

In a communication network, a link typically cannot be assigned an arbitrary capacity, but is installed in multiples of a predefined module, M (cf. the standards of SONET/SDH [13]). Moreover, it is reasonable to assume that for each demand d, $d = 1, 2, \ldots, D$, the demand's flow, x_d, must be a multiple of the same module. Without loss of generality we may, just changing units, assume that $M = 1$, and thus that $c_e \in \mathbb{Z}_{++}$, and, as is accomplished by (4.3), require that $x_d \in \mathbb{Z}_+$ for all demands $d = 1, 2, \ldots, D$. Hence, modular flows can be treated as integral flows.

4.1.3 Properties of the optimal integral solution

The properties of the optimal continuous solution are, mainly due to the *max-saturating* characterization following from Property 3.1, very well known and understood [6],[43],[44]. We will therefore in this section address characterization of the optimal integral solution by comparing it to the optimal continuous solution. We will assume that $\boldsymbol{x}$ is the OCS and $\boldsymbol{x}^z$ is the OIS, for some given instance of the problem.

Property 4.1. *If* $\Theta(\boldsymbol{x}^z) \neq \Theta(\boldsymbol{x})$ *then there exists an entry* k *for which* $\theta_k(\boldsymbol{x}^z) > \theta_k(\boldsymbol{x})$.

Proof. It is easy to see that $\Theta(\boldsymbol{x}^z) \neq \Theta(\boldsymbol{x})$ implies that there exists a demand d such that $x_d - \lfloor x_d \rfloor > 0$, because if $x_d \in \mathbb{Z}_+$, for all $d = 1, 2, \ldots, D$, then the OCS would be an OIS and necessarily $\Theta(\boldsymbol{x}^z) = \Theta(\boldsymbol{x})$. Without loss of generality we may assume that $\Theta(\boldsymbol{x}) = \boldsymbol{x}$ (this may be obtained by alternative enumeration of demands). Consider the non-integral entries $x_i, x_{i+1}, \ldots, x_{i+m}$ for which it holds that for all entries k, $k < i$, (if any) $x_k \in \mathbb{Z}_+$, and if there exists an entry $i+m+1$ then $x_{i+m+1} \in \mathbb{Z}_+$. Construct a solution $\boldsymbol{x}^a$ by truncating all demand volumes of $\boldsymbol{x}$, except demand $i+m$, which is rounded up. The idea is illustrated in Figure 4.2. This solution must be feasible since link capacities are integral. Moreover $\boldsymbol{x}^a$ is an integral solution with the property that $\boldsymbol{x}^a = \Theta(\boldsymbol{x}^a)$. As it must hold that $\Theta(\boldsymbol{x}^z) \succeq \Theta(\boldsymbol{x}^a)$, we must have that either $\theta_j(\boldsymbol{x}^z) > \theta_j(\boldsymbol{x}^a)$ for some j, $i \leq j < i+m$, or that $\theta_j(\boldsymbol{x}^z) = \theta_j(\boldsymbol{x}^a)$ for all j, $i \leq j < i+m$ and that $\theta_{i+m}(\boldsymbol{x}^z) \geq \theta_{i+m}(\boldsymbol{x}^a)$. Both cases imply that there exists an index j, $i \leq j \leq i+m$, such that $\theta_j(\boldsymbol{x}^z) > \theta_j(\boldsymbol{x})$. □

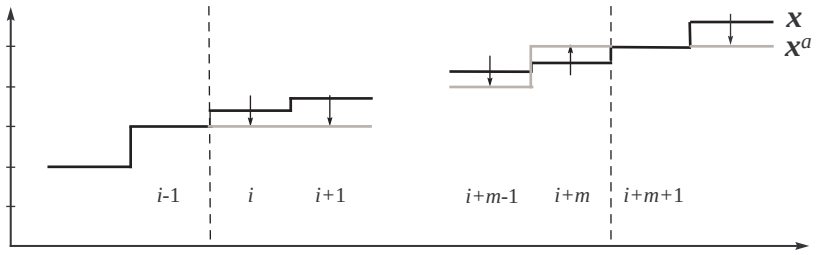


Figure 4.2: How to obtain $\boldsymbol{x}^a$ from $\boldsymbol{x}$.

Property 4.2. *If for some entry k, $\theta_k(\boldsymbol{x}^z) > \theta_k(\boldsymbol{x})$, then there exists a demand d, such that $x^z_d > x_d$.*

Proof. Without loss of generality assume that $\boldsymbol{x} = \Theta(\boldsymbol{x})$ (if this is not true, reenumerate the demands). Suppose $\boldsymbol{x} \geq \boldsymbol{x}^z$ and consider all entries m and n, $1 \leq m < n \leq D$, such that $x^z_n < x^z_m$. Interchanging all such elements in $\boldsymbol{x}^z$, we will eventually arrive at $\Theta(\boldsymbol{x}^z)$. However, we have that $x_m \geq x^z_m > x^z_n$ and $x_n \geq x_m \geq x^z_m$, so it must be true that $\boldsymbol{x} = \Theta(\boldsymbol{x}) \geq \Theta(\boldsymbol{x}^z)$, which is a contradiction. □

Property 4.3. *Let j be the largest integer for which it is true that $\theta_k(\boldsymbol{x}) - \theta_k(\boldsymbol{x}^z) \geq 0$, $1 \leq k \leq j$. Then, $j \geq 1$ and it holds that $\theta_k(\boldsymbol{x}) - \theta_k(\boldsymbol{x}^z) < 1$.*

Proof. By definition such an entry j must exist. Suppose that for some k, $1 \leq k \leq j$, $\theta_k(\boldsymbol{x}) - \theta_k(\boldsymbol{x}^z) \geq 1$. Then we can find a feasible integral solution by just truncating the OCS. Call this solution $\boldsymbol{x}^t$. Apparently, $\Theta(\boldsymbol{x}^t) \succ \Theta(\boldsymbol{x}^z)$, which is a contradiction proving the second part of the statement. □

The following two properties are analogies to the max-saturating property of the OCS.

Property 4.4. *For each demand d', there exists at least one saturated link e for which $x^z_{d'} \geq \max_d\{x^z_d : \delta_{ed} = 1\} - 1$.*

Proof. It is obvious that it is possible to find at least one saturated link for each demand, since otherwise that demand could be increased. Denote by

$$\hat{x}^z_e = \max_d\{x^z_d : \delta_{ed} = 1\},$$

the flow of the maximal demand on link e, and suppose, contradictory to the statement, that it holds for all such saturated links e, for demand d', that $x^z_{d'} < \hat{x}^z_e - 1$. Then, for these saturated links we have $\hat{x}^z_e > x^z_{d'} + 1$. Thus for each of the links with this property, reassigning the currently maximal flow a value of $\hat{x}^z_e - 1$ and demand d' a value of $x^z_{d'} + 1$ give a lexicographically larger solution, which is a contradiction. □

It should be noted that Property 4.4 implies that for each demand d', if there does not exist a saturated link e for which $x^z_{d'} = \max_d\{x^z_d : \delta_{ed} = 1\}$, then there must exist a saturated link e for which

$$x^z_{d'} = \max_d\{x^z_d : \delta_{ed} = 1\} - 1.$$

Property 4.5. *For a feasible allocation vector $\boldsymbol{x}^m$, $\boldsymbol{x}^m \in \mathbb{Z}^D_+$, if it holds for each demand d', $d' = 1, 2, \ldots, D$, that $x^m_{d'} = \max_d\{x^m_d : \delta_{ed} = 1\}$ on at least one saturated link e for which $\delta_{ed'} = 1$, then $\boldsymbol{x}^m$ is the unique OIS.*

Proof. According to Property 3.1 the result is valid for the continuous flows case, i.e., a feasible allocation vector, $\boldsymbol{x}$, for which it holds that for each demand d', $x_{d'} = \max_d\{x_d : \delta_{ed} = 1\}$ on at least one saturated link e for which $\delta_{ed'} = 1$, is the unique OCS. Now since $\boldsymbol{x}^m \in \mathbb{R}^D_+$, $\boldsymbol{x}^m$ is the unique OCS and therefore the unique OIS. □

The following examples illustrate that it may happen, considering the OCS ($\boldsymbol{x}$) and the OIS ($\boldsymbol{x}^z$) for a given instance, that $x^z_d < \lfloor x_d \rfloor$ and that $x^z_d > \lceil x_d \rceil$ for some demand d. They also show that there is no certain throughput domination, i.e., there exist both instances for which $\sum_d x^z_d < \sum_d x_d$, and instances for which $\sum_d x^z_d > \sum_d x_d$.

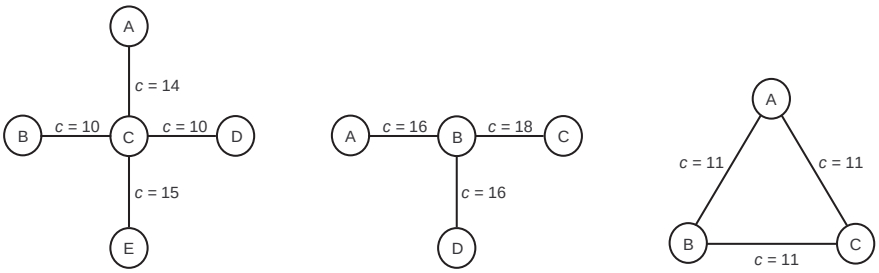
Example 4.2. Consider the network given in Figure 4.3(a). There are 2 demands between nodes A and B, 2 between A and E, 2 between A and D, 1 between C and B, 1 between C and E, and 1 between C and D. The sorted OCS is $\Theta(\boldsymbol{x}) = (\underbrace{7/3, \ldots, 7/3}_{6}, 16/3, 16/3, 31/3)$ and the sorted OIS is $\Theta(\boldsymbol{x}^z) = (2, 2, 2, 2, 3, 3, 6, 6, 9)$. The throughput is 35 for both solutions.

Example 4.3. Consider the network given in Figure 4.3(b). There are 4 demands between verticess A and B, 2 between A and C, 4 between D and B, 2 between D and C, and 1 between D and B. The sorted OCS is $\Theta(\boldsymbol{x}) = (\underbrace{8/3, \ldots, 8/3}_{12}, 22/3)$ and the sorted OIS is $\Theta(\boldsymbol{x}^z) = (2, 2, 2, 2, \underbrace{3, \ldots, 3}_{8}, 10)$. The throughputs are $39\frac{1}{3}$ and 42.

Example 4.4. Consider the network given in Figure 4.3(c). Edge capacities are given in the figure. There is one demand between each vertex-pair. Each demand is using the associated simple two-edge path. The sorted OCS is $\Theta(\boldsymbol{x}) = (5.5, 5.5, 5.5)$ and the sorted OIS is $\Theta(\boldsymbol{x}^z) = (5, 5, 6)$. The throughputs are $16\frac{1}{2}$ and 16.

4.1.4 Computational complexity

In this section it will be shown that optimization problem (4.1)-(4.3) is $\mathcal{NP}$-hard. This was already stated in [48], although a complete proof was not



(a) Example 2. (b) Example 3. (c) Example 4.

Figure 4.3: Example instances comparing the OCS and the OIS.

given there. In their discussion, the authors of that paper suggest that a transformation from INDEPENDENT SET shows this. The proof given here does not use this assumption (and was derived independently).

The negative result of $\mathcal{NP}$-hardness means that it is unrealistic to aim for a general polynomial time algorithm that obtains an MMF flow distribution, when integer flows on fixed paths are required. Mind that the considered optimization problem is exactly that considered in [6], but with integer-valued flows. As can be verified in Examples 4.3 and 4.4, the basic "waterfilling" algorithm (Algorithm 3.1) is in general not applicable for the integer flows case. An attempt to use this basic procedure will show that certain non-trivial, discrete decisions occasionaly have to be taken. So there are certainly reasons to conjecture that this multi-criteria optimization problem is computationally hard. We will call the associated decision problem FIXED PATHS MMF – MODULAR FLOWS (FIXMMF-MF):

Decision Problem 4.1 (FIXMMF-MF)**.**

INSTANCE: A link capacity $c_e \in \mathbb{Z}_+$ for each link $e = 1, 2, \ldots, E$, a binary link-demand incidence coefficient, δ_{ed}, for each demand $d = 1, 2, \ldots, D$, and a target vector $\boldsymbol{x}^T \in \mathbb{Z}_+^D$.

QUESTION: Is there an assignment of flow, $x_d \in \mathbb{Z}_+$, for each demand d, such that $\sum_d \delta_{ed} x_d \leq c_e$ for each link e, and such that if $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$, then $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^T)$?

Proposition 4.6. *FIXMMF-MF is $\mathcal{NP}$-complete.*

Proof. A nondeterministic algorithm needs only to guess an integral flow for each demand and check if the links have the required capacity and if it holds for the resulting allocation vector, $\boldsymbol{x}$, that $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^T)$. Thus clearly, FIXMMF-MF is in $\mathcal{NP}$. We will transform the decision problem of SET PACKING into an instance of FIXMMF-MF. It is trivial to verify $\mathcal{NP}$-completeness of the former, restricting it to EXACT COVER BY 3-SETS, shown to be $\mathcal{NP}$-complete in [12].

Decision Problem 4.2 (SET PACKING)**.**

INSTANCE: A collection C of finite sets and a positive integer $K \leq |C|$.

QUESTION: Does C contain at least K mutually disjoint sets?

Consider an arbitrary instance of SET PACKING. A collection C of n finite sets is given, $C = \{A_1, A_2, \ldots, A_n\}$. We will let each set A_i constitute a demand and the elements of each such set a chain of links that is a path for that demand. Assume that there is N distinct elements in total in all of the sets A_i. For each such element a_k, $k = 1, 2, \ldots, N$, construct two nodes connected by one link as in Figure 4.4. These links will be referred to as

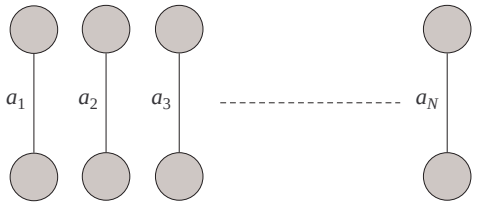


Figure 4.4: Element links.

the element links. For each set $A_i \in C$ perform the following operations. Construct a source node s_i and a sink node t_i. Label the elements of set A_i such that $A_i = \{z_1, z_2, \ldots, z_m\}$, and note that there is one-to-one correspondence between these labeled elements and m of the element links. Denote the upper node of the element link corresponding to z_j, $j = 1, 2, \ldots, m$ by v_j^u, and the lower node by v_j^l. Add new links connecting s_i to v_1^u, v_1^l to v_2^u, v_2^l to v_3^u, and so on. Finally, add a link that connects v_m^l with the sink node t_i. This constitutes a path between s_i and t_i, traversing all element links of A_i. Note that all links on this path that are not element links can only be used by node-pair (demand) (s_i, t_i). Assign a capacity of 1 to all links. Let $\boldsymbol{x}^T = (\underbrace{0, \ldots, 0}_{n-K}, \underbrace{1, \ldots, 1}_{K})$. This constitutes an instance of FIXMMF-MF, with n demands. Suppose that we have a positive answer to SET PACKING. This implies that there exist at least K mutually disjoint sets A_i. Assigning a flow of 1 to each of the corresponding node-pairs (s_i, t_i), and 0 to the rest will give $\Theta(\boldsymbol{x}) = (\underbrace{0, \ldots, 0}_{n-H}, \underbrace{1, \ldots, 1}_{H})$, $H \geq K$. Thus $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^T)$, and we have a positive answer to FIXMMF-MF.

Conversely, suppose that we have a positive answer to FIXMMF-MF, i.e., that there exists a feasible allocation $\boldsymbol{x}$, with $\Theta(\boldsymbol{x}) \succeq (\underbrace{0, \ldots, 0}_{n-K}, \underbrace{1, \ldots, 1}_{K})$. Since capacities are equal to 1, no links can be shared by demands and

$x_d \leq 1$ for all $d = 1, 2, \ldots, D$. This implies that paths of demands that are assigned flow 1 must be disjoint. As constructed, these paths define at least K mutually disjoint sets A_i, and a positive answer to SET PACKING follows. Hence, since SET PACKING is $\mathcal{NP}$-complete, FIXMMF-MF is $\mathcal{NP}$-complete. □

Knowledge of Algorithm 3.1 and a bit of reflection reveals that the above result will not hold (unless $\mathcal{P} = \mathcal{NP}$) if $\boldsymbol{x}^T = c \cdot \boldsymbol{u}$, where $\boldsymbol{u}$ is the unity vector of size D, and c is a positive integer (corresponding to the optimization problem of finding the maximal first entry of the sorted allocation vector). Solving the problem is then just a matter of assigning $x_d = c$ for $d = 1, 2, \ldots, D$, and test feasibility. It is a bit more cumbersome to deduce that if each demand's path has at most one shared link, the problem is also solvable in polynomial time. Algorithm 4.1, which is a modification of Algorithm 3.1, solves this task. The function "any(A)" returns any element of set A.

Algorithm 4.1.

for $d := 1$ to D **do**
  $x_d := 0$
end for
$S := \emptyset$
while $|S| \neq D$ **do**
  for all d such that $d \notin S$ **do**
    $x_d := x_d + 1$;
  end for
  for all e such that $\sum_d \delta_{ed} x_d > c_e$ **do**
    $A := \{d : \delta_{ed} = 1,\ d \notin S\}$;
    repeat
      $d :=$any(A);
      $x_d := x_d - 1$;
      $A := A \setminus \{d\}$;
    until $\sum_d \delta_{ed} x_d = c_e$;
  end for
  for all e such that $\sum_d \delta_{ed} x_d = c_e$ **do**
    for all d such that $\delta_{ed} = 1$ and $d \notin S$ **do**
      $S := S \cup \{d\}$;
    end for
  end for
end while

Algorithm 4.1 uses a set S to denote the set of blocked demands, and a (temporary) set A to keep track of "violating demands". The crucial property for its operation is that only one link is shared per demand, which

eliminates (makes arbitrary) the selection difficulty in distributing capacity of a link to competing demands.

4.1.5 An algorithm

The proved difficulty of the considered problem slightly relaxes the requirements that are reasonable to put on its resolution methods. The approach taken here is in essence exploiting the "distribution approach" described for MMF problems that have discrete outcomes in general in [40]. Without using this naming convention, we have presented the idea, fundamental for the distribution approach, in Section 2.4.2 of Chapter 2. Specifically, adjusting parameters for the application at hand, Property 2.15 states that a solution to

$$\begin{aligned}
\text{lex min} \quad & \left(\textstyle\sum_d t_{1d}, \sum_d t_{2d}, \ldots, \sum_d t_{rd} \right) && && (4.4)\\
\text{s.t.} \quad & k - x_d \leq t_{kd} && k = 1, 2, \ldots, r, \ \forall\ d && (4.5)\\
& t_{kd} \geq 0 && k = 1, 2, \ldots, r, \ \forall\ d && (4.6)\\
& x_d \in \{0, 1, \ldots, r\} && \forall\ d && (4.7)\\
& \boldsymbol{x} \in X\ , && && (4.8)
\end{aligned}$$

is a solution to

$$\begin{aligned}
\text{lex max} \quad & \Theta(\boldsymbol{x}) && && (4.9)\\
\text{s.t.} \quad & x_d \in \{0, 1, \ldots, r\} && \forall\ d && (4.10)\\
& \boldsymbol{x} \in X\ . && && (4.11)
\end{aligned}$$

If we let $r = \max_e\{c_e\}$ and define the feasible set X by the link-load constraints, $\sum_d \delta_{ed} x_d \leq c_e$, $e = 1, 2, \ldots, E$, problem (4.9)-(4.11) is equivalent to the considered problem, (4.1)-(4.3). Hence by solving (4.4)-(4.8) by a conventional, iterative procedure, we have a solution method for (4.1)-(4.3). Such a procedure is described by the following algorithm.

Algorithm 4.2. (*for solving* (4.1)-(4.3))

Step 0: Assign $k := 1$. Solve

$$\begin{aligned}
\tau_1 = \min \quad & \sum_d t_{1d} && && (4.12)\\
\text{s.t.} \quad & 1 - x_d \leq t_{1d} && \forall\ d && (4.13)\\
& \sum_d \delta_{ed} x_d \leq c_e && \forall\ e && (4.14)\\
& t_{1d} \geq 0 && && (4.15)\\
& x_d \in \mathbb{Z}_+ && \forall\ d && (4.16)
\end{aligned}$$

and let τ_1^* be the optimal objective value, and $\boldsymbol{x}^*$ the optimal solution.

Step 1: If $k > \max_d\{x_d^* : d = 1, 2, \ldots, D\}$ then stop, $\boldsymbol{x}^*$ is the MMF solution. Otherwise assign $k := k+1$. If $k > \max\{c_e : e = 1, 2, \ldots, E\}$ then stop, $\boldsymbol{x}^*$ is the max-min fair solution. Otherwise proceed to Step 2.

Step 2: Solve

$$\tau_k = \min \quad \sum_d t_{kd} \tag{4.17}$$

$$\text{s.t.} \quad j - x_d \leq t_{jd} \qquad j = 1, 2, \ldots, k, \ \forall\ d \tag{4.18}$$

$$\sum_d t_{jd} \leq \tau_j^* \qquad j = 1, 2, \ldots, k-1 \tag{4.19}$$

$$\delta_{ed} x_d \leq c_e \qquad \forall\ e \tag{4.20}$$

$$t_{jd} \geq 0 \qquad j = 1, 2, \ldots, k, \ \forall\ d \tag{4.21}$$

$$x_d \in \mathbb{Z}_+ \qquad \forall\ d \tag{4.22}$$

and let τ_k^* be the optimal objective value, and $\boldsymbol{x}^*$ the optimal solution. Go to Step 1.

Note that both (4.12)-(4.16) of Step 0 and (4.17)-(4.22) of Step 2 are MIPs. They may thus be very difficult to solve efficiently for large instances. As entirely integral solutions are desired, it is possible to substitute constraints (4.19) by

$$\sum_d t_{jd} \leq \lceil \tau_j \rceil, \quad j = 1, 2, \ldots, k-1. \tag{4.23}$$

Such a substitution is useful if we relax the integrality requirements, by omitting constraints (4.16) and (4.22), and solve what are called the *linear relaxations* of optimzation problems (4.12)-(4.16) and (4.17)-(4.22), as we then do not unnecessarily force a non-integral solution implied by a previous iteration. Consider Algorithm 4.2 modified by the above substitution, and linearly relaxed in its optimzation problems. We will call this the relaxed version of the algorithm. Since $\mathbb{Z}_+ \subset \mathbb{Q}_+$, if the output $\boldsymbol{x}^*$ of the relaxed version of the algorithm is entirely integral, this will clearly be a solution to (4.1)-(4.3). As will be shown in the following section, if the relaxed problems are solved by Simplex (the relaxed optimization problems of Step 0 and Step 2 are LPs), entirely integral outputs of the relaxed algorithm occur quite frequently.

There are some implementational issues of the relaxed version of Algorithm 4.2 that ought to be mentioned. First of all, it is convenient to recycle the sparse constraint matrices of the successive LPs, as they change only marginally between consecutive steps. Secondly, special care should be taken in the rounding of τ. For large instances (many variables), aggregation

of small numerical errors in the computed variables may cause an erroneous rounding of τ (typically making the right-hand side of (4.23) too large).

Finally, it is essential that the LPs are solved for vertex solutions, as is done by Simplex. This is easily realized if one considers a network of two vertices connected by one edge of capacity 1. Assume that there are two demands between the vertices. As opposed to Simplex, an interior point solution to this instance cannot belong to $\{0,1\}^2$.

4.1.6 A numerical experiment

In this section we apply the relaxed version of Algorithm 4.2 to randomly generated problem instances ranging from 36 to 435 demands (corresponding to demands between all node-pairs in a 9-node network to all node-pairs in a 30-node network). In all instances $r = 50$ and each link capacity, c_e, $e = 1, 2, \ldots, E$ belongs to $\{5, 10, 15, \ldots, 50\}$. The results can be found in Table 4.1. The first 4 columns give, in turn, the number of nodes, the number of links, the number of demands, and the average length (hops) of a path. The fifth column indicates if the algorithm halts with an integral solution. Column 6 gives the number of solved LP:s (iterations) that did not produce an integral solution, and the 7:th column gives the number of iterations for which τ was rounded up, i.e., when optimal τ was non-integral (due to rounding errors there is no one-to-one correspondence between rounded τ's and non-integral solutions). The two final columns give the total running time and the required number of iterations, respectively. The computations were carried out on a PC with an Intel PIII-1GHz CPU, RAM of 256 MB, and Windows 2000 OS. The algorithm was implemented in MATLAB6.5, and the LPs are solved using a MATLAB interface (mex-function) to CPLEX 9 (Simplex LP-solver).

Although the relaxed version of Algorithm 4.2 performs satisfactorily on all of the instances considered in Table 4.1, there exist instances for which it fails, as e.g. the following example.

Example 4.5. Consider the network given in Figure 4.3(c). Suppose that the same demands and paths as in Example 4.4 are given, and that link capacities are all equal to 1. Then the (sorted) solution generated by Algorithm 1 is $\Theta(\boldsymbol{x}') = (0.5, 0.5, 0.5)$ but the true OIS is $\Theta(\boldsymbol{x}^z) = (0, 0, 1)$.

4.2 Max-min fairness of unsplittable flows

In this section we study a difficult version of the MMF allocation problem when only non-bifurcated (unsplittable) flows are allowed. This assumption, also called requirement of single-path flows, leads, as mentioned in the beginning of this chapter, to non-convex problem formulations which are inherently hard [17] (general single-source unsplittable flow problems are

V	E	D	$\mathbb{E}(\|p\|)$	t-int	n-int	roundings	time (s)	iterations
9	20	36	5.3	yes	0	0	2.50	26
10	20	45	4.7	yes	0	0	7.03	43
11	21	55	6.4	yes	1	1	9.47	46
12	27	66	6.8	yes	4	1	11.56	41
13	28	78	7.2	yes	0	7	14.16	48
14	34	91	7.7	yes	1	1	10.27	35
15	29	105	9.0	yes	0	0	11.29	37
16	34	120	8.6	yes	2	2	17.42	38
17	39	136	8.8	yes	0	0	16.86	34
18	43	153	10.3	yes	2	7	16.59	30
19	45	171	10.3	yes	0	0	18.74	36
20	41	190	10.8	yes	0	0	14.95	28
21	50	210	11.8	yes	1	1	47.22	42
22	55	231	11.7	yes	0	1	40.28	36
23	59	253	13.6	yes	1	14	53.77	40
24	54	279	13.7	yes	2	10	55.41	37
25	54	300	13.7	yes	0	0	43.43	32
26	72	325	14.4	yes	0	0	147.34	50
27	66	351	14.6	yes	0	0	38.77	26
28	69	378	15.7	yes	0	0	144.45	42
29	59	406	7.3	yes	0	0	351.72	49
30	65	435	8.5	yes	1	0	426.30	49

Table 4.1: Testing the algorithm.

studied in [17], continuing the work on single-source splittable flow problems carried out in [25]). Nevertheless, unsplittable flows is often a realistic restriction due to the used routing protocol, or simply an explicit management requirement, stipulating avoidance of packet resequencing in receiving nodes. The network is assumed to be given in terms of topology and link capacities, and the problem is to associate the demand between each S-D node-pair with a single (optimized) path such that a sufficient volume of flow can be routed on demands' single paths simultaneously, without violating the link capacities. By "sufficient volume" a volume that is MMF among different S-D pairs is addressed. Consequently, once each S-D pair is assigned a single path, the problem is reduced to max-min fair sharing of corresponding link capacities, for which Algorithm 3.1 can be used. Thus essentially, the considered problem amounts to the difficult task of appropriate single path selection.

4.2.1 Problem description

The considered problem is characterized by a capacitated network, where a set of demands is assumed to be given. As usual, a demand d is an unspecified requirement for bandwidth (flow) between a node-pair in the network, and c_e is used to denote the capacity associated with link e. For each demand d, $d = 1, 2, \ldots, D$, a list of allowable paths is predefined. Each path p for demand d, $p = 1, 2, \ldots, P_d$ defines a cycle-free set of links that connects vertex-pair d. The binary indicator, δ_{edp}, is used for the link-path incidence relation. The flow allocated to demand d will be identified by x_d, and z_{dp} is used to denote its part that is allocated to path p, i.e., $\sum_p z_{dp} = x_d$. The decision variables of the considered problem will be the flow variables z_{dp}. As in Section 4.1, the problem is compactly described by

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\}. \tag{4.24}$$

However, in this section, the feasible set X is different – a solution $\boldsymbol{x} \geq \mathbf{0}$ is considered feasible, $\boldsymbol{x} \in X$, if and only if

$$\sum_d \sum_p \delta_{edp} z_{dp} \leq c_e \ , \ e = 1, 2, \ldots, E, \text{ and} \tag{4.25}$$

$$\text{for all } d, \ d = 1, 2, \ldots, D, \ z_{dp'} = x_d \text{ for some path } p', \text{ and}$$
$$z_{dp} = 0 \text{ for all other paths } p, \ p = 1, 2, \ldots, P_d, \ p \neq p'. \tag{4.26}$$

So an allocation is feasible if and only if the sum of flows on a link does not exceed the link's capacity, and no demand has flows on more than one path.

4.2.2 Computational complexity

To motivate resolution techniques that are computationally heavy, this section will have as main purpose to show that solving

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\},$$

with the feasible set X constituted by (4.25) and (4.26), is $\mathcal{NP}$-hard. This is accomplished by proving that the decision problem corresponding to one of its subproblems (or the problem itself) is $\mathcal{NP}$-complete. Specifiaclly, we will sometimes consider the subproblem of obtaining the largest possible minimal value among the entries of the allocation vector. It will be convenient to divide the problem into two cases: the case when a demand may use any one path to connect its node-pair – the *unlimited path-sets case*, and the case when a demand may select a path only from a limited, predefined set of paths – the *limited path-sets case*. A somewhat different version of the former decision problem was proven $\mathcal{NP}$-complete in [18]. However, that study concerned a directed graph with a single-source multiple-sinks demand. Their proof is based on a nontrivial transformation from a scheduling problem. Here we only consider undirected graphs.

Unlimited path-sets

When a demand may use any (simple) path that connects its node-pair, there is clearly no reason to predefine its path-set, as it is directly implied by the network. This is the case for all demands if path-sets are unlimited. We will refer to the decision problem corresponding to the optimization problem of obtaining the maximal smallest demand (i.e., the smallest entry of the allocation vector) on single paths with unlimited path-sets as MAXIMIZING SMALLEST DEMAND- SINGLE PATH FLOW (MSD-SPF). If the set of feasible solutions, X, admits usage of any (single) path for each demand, such an optimization problem is equivalent to finding a maximal first entry of $\Theta(\boldsymbol{x})$, $\boldsymbol{x} \in X$, i.e., certainly a subproblem of (4.24).

Decision Problem 4.3 (MSD-SPF)**.**

INSTANCE: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a link capacity c_e for each e, $e = 1, 2, \ldots, E$, a set of demands (node-pairs), and a number $K > 0$.

QUESTION: Is there a set of simple paths, containing exactly one path for each demand d, such that for each demand d it is possible to assign a flow x_d, with $x_d \geq K$, on the path corresponding to demand d, without violating any link capacity c_e, $e = 1, 2, \ldots, E$?

As can be seen, MSD-SPF puts no restrictions on the number of admissible paths, since any one (cycle-free) set of links that connects demand d, i.e., constitutes a simple path between the associated node-pair, may be selected.

Proposition 4.7. *MSD-SPF is $\mathcal{NP}$-complete.*

Proof. A nondeterministic algorithm needs only to guess one of the paths for each demand, assign flow $x_d = K$ to each demand and check if the links forming these paths have the required capacity. Thus clearly, MSD-SPF is in $\mathcal{NP}$. If we restrict MSD-SPF to instances with $K = 1$ and $c_e = 1$, for all links e, $e = 1, 2, \ldots, E$, we immediately get a decision problem which is equivalent to the problem of LINK-DISJOINT PATHS.

Decision Problem 4.4 (LINK-DISJOINT PATHS)**.**

INSTANCE: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a collection of node-pairs $(s_1, t_1), \ldots, (s_k, t_k)$.

QUESTION: Does $\mathcal{G}$ contain k mutually link-disjoint paths, one connecting s_i and t_i for each i, $1 \leq i \leq k$?

LINK-DISJOINT PATHS is known to be $\mathcal{NP}$-complete [52], [26], and the polynomial time restriction shows that LINK-DISJOINT PATHS $\propto$ MSD-SPF, so MSD-SPF is also $\mathcal{NP}$-complete. □

As MSD-SPF is the decision version of a subproblem to the optimization problem of (4.24), where the feasible set X admits usage of any path for each demand, this result certifies that the latter problem is $\mathcal{NP}$-hard.

Limited path-sets

The complexity of a more specific version of the studied problem will now be investigated. For unlimited path-sets the assumption was that the selection of a single path could be made from all possible paths for a demand. In many cases such an assumption might be too generous and not practical. It may be more realistic to assume that each demand is given with a predefined, bounded (in terms of cardinality) path-set. Clearly, if the sizes of the predefined path-sets are very large, this will differ only academically from the unlimited path-sets case. However, in practice this is rarely the case. It will be shown that obtaining the maximal smallest demand (i.e., the smallest entry of the allcoation vector) on single paths with a cardinality of path-sets limited to as little as 3 is $\mathcal{NP}$-hard. However, it is easier to prove $\mathcal{NP}$-hardness of the full problem (4.24), with X admitting cardinality of path-sets being 2, so this is where we will start. Note that the subproblem of determining the maximal smallest demand is, from a practical viewpoint, an interesting problem in its own right, and we will return to this in a while.

The decision problem corresponding to (4.24), with X limiting (in both directions) cardinality of path-sets to 2 is called MMF OF SINGLE PATH FLOWS 2 (MMF-SPF2), and is formalized as follows.

Decision Problem 4.5 (MMF-SPF2)**.**

INSTANCE: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a link capacity c_e for each e, $e = 1, 2, \ldots, E$, a set of demands (node-pairs), two simple paths for each demand, and a target vector, $\boldsymbol{x}^T$, of length D

QUESTION: Is there a set of paths, containing exactly one of the two admissible paths for each demand d, $d = 1, 2, \ldots, D$, such that for each demand d it is possible to assign a flow x_d on the path corresponding to demand d, without violating any link capacity c_e, and such that if $\boldsymbol{x} = [x_d]_{d=1,2,\ldots,D}$, then $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^T)$?

Proof. MMF-SPF2 belongs to $\mathcal{NP}$ since a nondeterministic algorithm needs only to guess one of the two paths for each demand and apply Algorithm 3.1 (which is an algorithm of polynomial time) to obtain the allocation vector $\boldsymbol{x}$, and check if $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^T)$. We will transform PARTITION into a single-source multiple-sinks instance of MMF-SPF2, essentially using an idea from [16].

Decision Problem 4.6 (PARTITION)**.**

INSTANCE: Finite set A of items, and a size $s(a) \in \mathbb{Z}_+$ for each $a \in A$.

QUESTION: Is there a subset $A' \subset A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

Consider an arbitrary instance of PARTITION and construct a graph as follows: let two "core nodes", κ_l and κ_r, be connected by 2 "core links", e and e', both of which have capacity $\frac{1}{2}\sum_{a\in A} s(a)$. For each element $a \in A$, connect by a link of capacity $s(a)$ a node t_a, to the right core node, κ_r. Let $\boldsymbol{x}^T = [s(a)]_{a\in A}$. Now consider each (κ_l, t_a)-pair, $a \in A$, in the resulting graph as a demand (see Figure 4.5). Each (κ_l, t_a)-pair has exactly two

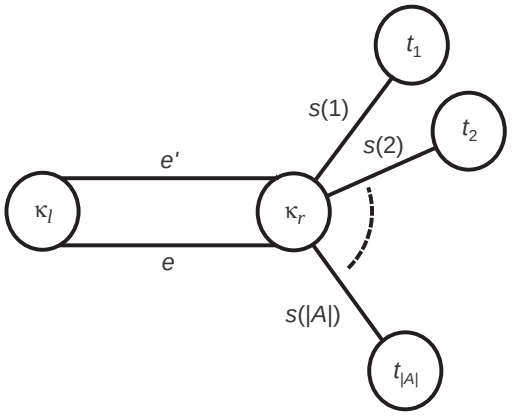


Figure 4.5: The resulting graph.

admissible paths – one traversing e and one traversing e'. This construction, which is apparently done in polynomial time, is a valid instance of MMF-SPF2 (with $|A|$ demands). Now assume that there is a positive answer to PARTITION. This implies existence of a subset $A' \subset A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a)$. Assign to each (κ_l, t_a)-pair a flow of $x_a = s(a)$. If $a \in A'$ let the demand use link e', and if $a \in A\setminus A'$, let it use link e together with its dedicated link connecting κ_r and t_a. As $c_e = c_{e'} = \frac{1}{2}\sum_{a\in A} s(a)$, no link capacities are exceeded. We have thus an allocation $\boldsymbol{x}$, for which $x_a = s(a)$ for all $a \in A$, and consequently that $\Theta(\boldsymbol{x}) = \Theta(\boldsymbol{x}^T)$, implying a positive answer to MMF-SPF2. Conversely, suppose that there is a positive answer for the constructed instance of MMF-SPF2, i.e., that there exists a set of single paths for which an allocation vector $\boldsymbol{x}$, with $\Theta(\boldsymbol{x}) \succeq \Theta(\boldsymbol{x}^T)$, is obtainable. In particular, this means that there exist flows x_a, such that $x_a = s(a)$ for all $a \in A$. Thus the total flow between κ_l and κ_r is $\sum_{a\in A} s(a)$. But $c_e = c_{e'} = \frac{1}{2}\sum_{a\in A} s(a)$ and flows are unsplittable so there must exist a subset $A' \subset A$, such that $\sum_{a\in A'} s(a) = \sum_{a\in A\setminus A'} s(a) = \frac{1}{2}\sum_{a\in A} s(a)$, which answers the PARTITION question positively. Therefore, since PARTITION is $\mathcal{NP}$-complete [12], MMF-SPF2 must also be $\mathcal{NP}$-complete. □

In the unlimited path-sets case it was possible to prove $\mathcal{NP}$-hardnness even of the subproblem of obtaining the maximal smallest demand (i.e., the

smallest entry of the allcoation vector) on single paths. A natural question to ask is if this is possible also if path-sets are limited. This is in fact possible if cardinalities of path-sets are limited to 3. There is obvious practical importance in this subproblem – which is the largest amount of bandwidth possible to assign to all demands? As mentioned, we have already shown that if all possible paths are allowed, this problem is $\mathcal{NP}$-hard. The question is if the difficulty remains if we are more restrictive regarding possible selection of paths. In order to show $\mathcal{NP}$-hardness we define the associated decision problem, denoted MAXIMIZING SMALLEST DEMAND - SINGLE PATH FLOW 3 (MSD-SPF3) as follows:

Decision Problem 4.7 (MSD-SPF3)**.**

INSTANCE: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a link capacity c_e for each e, $e = 1, 2, \ldots, E$, a set of demands (node-pairs). For each demand d, $d = 1, 2, \ldots, D$, three simple paths. A number $K > 0$.

QUESTION: Is there a selection of exactly one of the three paths for each demand d, such that for each demand d it is possible to assign a flow x_d, with $x_d \geq K$, on the path corresponding to demand d, without violating any link capacity c_e?

The requirement of simple paths is important here. Since a communication network demand is for obvious reasons never assigned a path containing a cycle, this is a reasonable assumption. If on the other hand cyclic paths were allowed, it rather straightforward to see how an $\mathcal{NP}$-completeness result can be obtained even for path-set cardinalities restricted to 2, e.g. by a transformation from INDEPENDENT SET [33]. To the best of our knowledge it is an open question if this result holds in the case of 2 simple paths per demand.

Proposition 4.8. *MSD-SPF3 is $\mathcal{NP}$-complete.*

Proof. A nondeterministic algorithm needs only to guess one of the paths for each demand, assign flow $x_d = K$ to each demand and check if the links forming these paths have the required capacity, so clearly MSD-SPF3 is in $\mathcal{NP}$. We will transform 3-SATISFIABILITY (3SAT) to MSD-SPF3. The problem of 3SAT is one of the most fundamental $\mathcal{NP}$-complete problems [12]. The proof starts by the construction of a polynomial-time transformation. Then it is shown that the constructed instance of MSD-SPF3 will have a positive answer if and only if 3SAT has a positive answer.

Consider an arbitrary instance of 3SAT. Such an instance consists of a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of clauses on a finite set U of boolean variables such that $|C_i| = 3$ for $1 \leq i \leq m$. For each $u \in U$, let k_u be the number of appearances of variable u in the clauses of $\mathcal{C}$, and $k_{\bar{u}}$ be the number of appearances of its negation, $\bar{u}$. For each variable $u \in U$, construct

two associated nodes, s_u and t_u. Each such node-pair will be connected by three disjoint paths; they will be referred to as the left-path, the slack-path and the right-path. The left-path and the right-path are constructed from chains of links, whereas the slack-path will be one single link. The number of links of the right-path, α_u, and the left-path, $\alpha_{\bar{u}}$, will depend on k_u and $k_{\bar{u}}$, respectively. Specifically, $\alpha_u = 1$ if $k_u = 0$, and $\alpha_u = k_u$ otherwise. Similarly, $\alpha_{\bar{u}} = 1$ if $k_{\bar{u}} = 0$, and $\alpha_{\bar{u}} = k_{\bar{u}}$ otherwise. The subgraph made up by node-pair (s_u, t_u) and its three connecting paths is said to constitute the truth-setting component for variable u. An example can be seen in Figure 4.6. Clearly, there will be exactly $|U|$ truth-setting components.

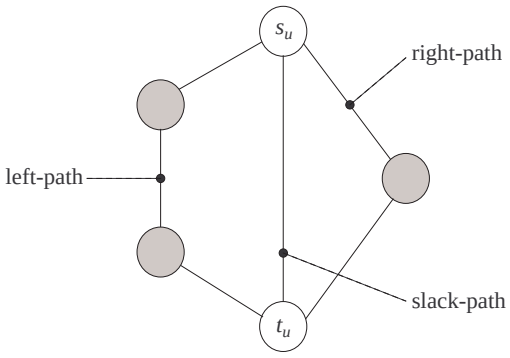


Figure 4.6: Truth-setting component for variable $u \in U$. It is assumed that $k_u = 2$ and that $k_{\bar{u}} = 3$.

Now consider a clause $C_i \in \mathcal{C}$. Construct and associate with the clause two nodes, v_i and w_i. These two nodes will be connected to exactly three of the truth-setting components, namely those whose associated variables, negated or non-negated, occur in clause C_i. For each literal l in C_i, connect them as follows: if l is a non-negated variable, say u, connect v_i and w_i by one link each to the right-path of the truth-setting component corresponding to variable u, such that the created path between v_i and w_i has exactly three links, and such that the end-nodes of the link of this path that is part of the truth-setting component are *not both* connected to another (one) pair, v_j and w_j, associated with some other clause, $C_j \in \mathcal{C}$. If l is a negated variable, say $\bar{u}$, take the corresponding action on the left-path of the truth-setting component of u. Note that this will always be possible since there are at least k_u links on the right-path, and $k_{\bar{u}}$ links on the left-path of the truth-setting component of variable u. An example of the connection associated with a clause is shown in Fig. 4.7. Making these connections for every $C_i \in \mathcal{C}$, completes the construction of the supply graph $\mathcal{G}$ for the instance of MSD-SPF3. What remains is to specify demands and associated paths, and also the link capacities. We impose a demand for each node-pair

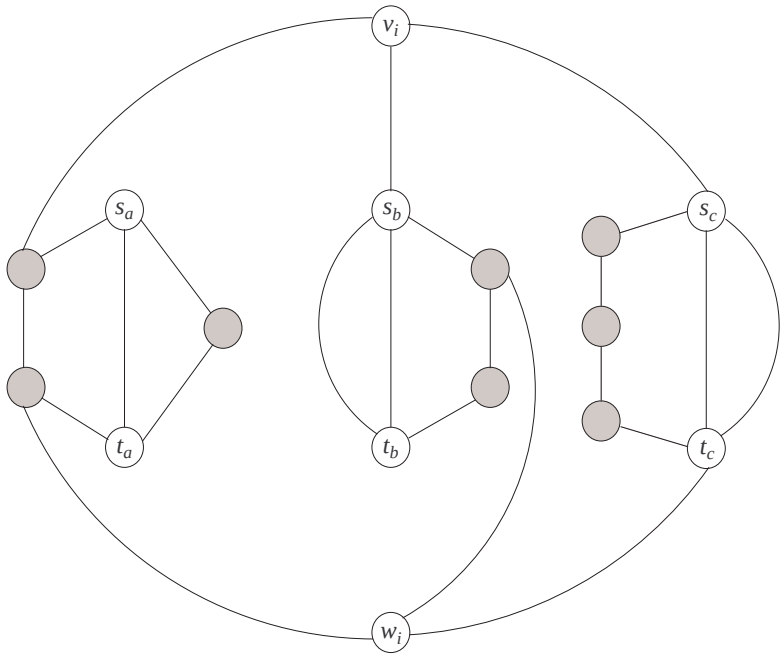


Figure 4.7: Subgraph corresponding to clause $C = (\bar{a}, b, c) \in \mathcal{C}$, and truth-setting components for variables $a, b, c \in U$.

(v_i, w_i), with three possible 3-link paths, all of which traverse exactly one truth-setting component on that side (left or right) that is determined by the corresponding variables' appearance (negated or not) in the associated clause C_i. These demands are referred to as the long demands. Further, assume that there is a demand between each of the pairs (s_u, t_u), $u \in U$, with the three obvious possible paths. These demands constitute what will be called the short demands. In total, this gives $|\mathcal{C}| + |U|$ demands. Finally, let every link corresponding to a slack-path have capacity $K/2$. Assign to all other links capacity K. This completes the transformation from 3SAT to MSD-SPF3, which is carried out in polynomial time.

Suppose that we have a positive answer to 3SAT. This means that we have a function, $T : U \to \{true, false\}$, that satisfies every clause $C_i \in \mathcal{C}$. For each variable $u \in U$ for which $T(u) = true$, assign the demand (s_u, t_u) the left-path of the truth-setting component for u. For each $u \in U$ for which $T(u) = false$, assign the demand (s_u, t_u) the right-path. Since T is a satisfying truth-assignment, there will be at least one literal in each clause for which the corresponding short demand is assigned a path that is disjoint with the path of the long demand associated with the clause. Thus chosing an arbitrary such path for the long demand gives a single-path pattern with neither shared links, nor usage of slack paths. This implies that a flow

$x_d = K$ may be allocated to each demand.

Conversely, suppose that we have a positive answer to the constructed instance of MSD-SPF3, i.e., that there exists a single path routing for which $x_d \geq K$ is possible for each demand d. Since $c_e \leq K$ for all links e, it can immediately be deduced that no slack-paths are used. Consequently, every short demand path corresponds to a truth assignment of the corresponding variable u; if (s_u, t_u) is routed on the right-path, then $T(u) := false$, and if (s_u, t_u) is routed on the left-path, then $T(u) := true$. As $x_d \geq K$ for all demands d, there must be at least one of the three paths for a long demand with the property that this path does not coincide with a path used for a short demand. In particular, this implies that this literal is set true by T. Thus all clauses are satisfiable. Hence 3SAT $\propto$ MSD-SPF3 and MSD-SPF3 must be $\mathcal{NP}$-complete. □

In many cases it is easier to understand a problem transformation if a complete example is given. We believe that this is indeed the case for the proof of Proposition 4.8. A complete example of how an instance of 3SAT is transformed into an instance of MSD-SPF3 is given in the following example.

Example 4.6. Consider the following instance of 3SAT;

$$U = \{a, b, c, d, e, f, g\} \text{ and } C = \{c_1, c_2, c_3, c_4\},$$

where $c_1 = \{\bar{a}, b, c\}$, $c_2 = \{\bar{a}, d, g\}$, $c_3 = \{\bar{a}, f, \bar{g}\}$ and $c_4 = \{c, e, \bar{f}\}$. Table 4.2 is determined from the given clauses and provides sufficient information to construct the supply graph of the MSD-SPF3 instance. The supply graph of

	k_u	α_u	$k_{\bar{u}}$	$\alpha_{\bar{u}}$
a	0	1	3	3
b	1	1	0	1
c	2	2	0	1
d	1	1	0	1
e	1	1	0	1
f	1	1	1	1
g	1	1	1	1

Table 4.2: The information needed to transform 3SAT to MSD-SPF3 .

the MSD-SPF3 instance corresponding to the given instance of 3SAT, can be seen in Figure 4.8. In this figure, every link is assigned a distinct number, given immediately to the left of the link. Note that each (v_i, w_i)-node-pair constitutes a long demand and corresponds to clause c_i, and that each (s_u, t_u)-node-pair constitutes a short demand and corresponds to variable u. Note also that each demand (node-pair) has exactly three allowed paths (from which only one can be selected). The paths for each demand are

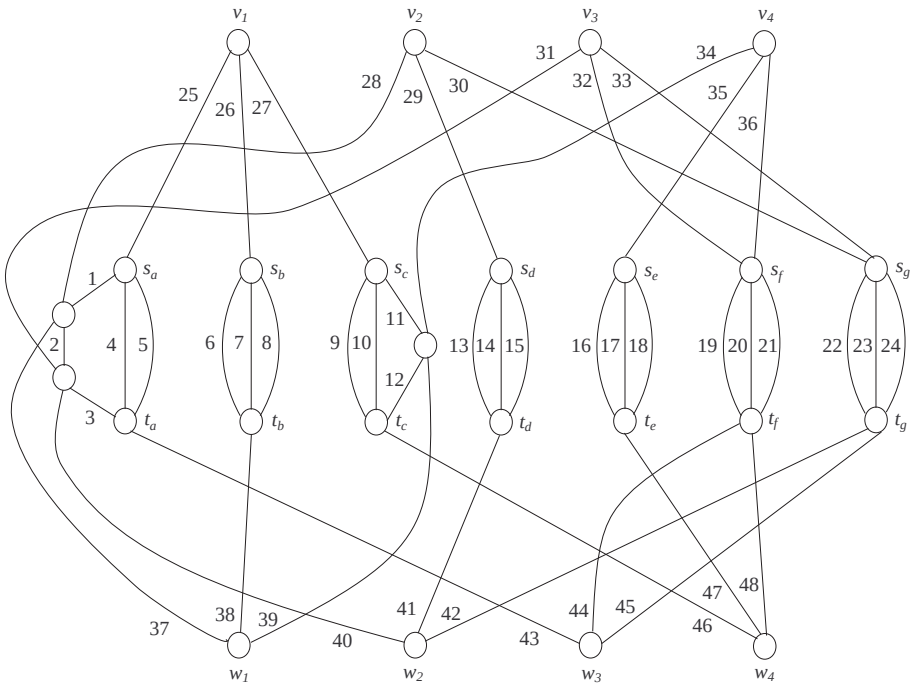


Figure 4.8: Supply graph for the instance of MSD-SPF3, given by the 3SAT instance.

listed in Table 4.3. A path is given as a set of integers, where each integer corresponds to a link according to Figure 4.8.

4.2.3 Resolution algorithms

When the feasible set X models the single-path constraint, it is not at all obvious how a resolution algorithm for

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\}, \tag{4.27}$$

should be formulated. Because of the non-convexity it is not possible to exploit the notion of "blocked demands" as was done in Section 3.1.2 and in Section 3.2.4. Note that with the complicating single-path constraint, a demand may be blocking for a specific collection of single paths, but be possible to increase for another collection of single paths. We will restrict our attempts to formulate resolution techniques to the problems where path-sets are limited. The approach generally taken is greatly inspired by the resolution techniques for the convex counterpart of the studied problem. Focus will be put on representing subproblems of (4.27), as MIPs. Particularly, each MIP is designed to compute a specific entry, say entry j, of $\Theta(\boldsymbol{x})$. We

Demand	Corresponds to	Path 1	Path 2	Path 3
(v_1, w_1)	clause c_1	$\{25, 1, 37\}$	$\{26, 8, 38\}$	$\{27, 11, 39\}$
(v_2, w_2)	clause c_2	$\{28, 2, 40\}$	$\{29, 15, 41\}$	$\{30, 24, 42\}$
(v_3, w_3)	clause c_3	$\{31, 3, 43\}$	$\{32, 21, 44\}$	$\{33, 22, 45\}$
(v_4, w_4)	clause c_4	$\{34, 12, 46\}$	$\{35, 18, 47\}$	$\{36, 19, 48\}$
(s_a, t_a)	variable a	$\{1, 2, 3\}$	$\{4\}$	$\{5\}$
(s_b, t_b)	variable b	$\{6\}$	$\{7\}$	$\{8\}$
(s_c, t_c)	variable c	$\{9\}$	$\{10\}$	$\{11, 12\}$
(s_d, t_d)	variable d	$\{13\}$	$\{14\}$	$\{15\}$
(s_e, t_e)	variable e	$\{16\}$	$\{17\}$	$\{18\}$
(s_f, t_f)	variable f	$\{19\}$	$\{20\}$	$\{21\}$
(s_g, t_g)	variable g	$\{22\}$	$\{23\}$	$\{23\}$

Table 4.3: The demands and their allowed paths. Paths are given as sets of integers, where each integer corresponds to a link in Figure 4.8.

will sometimes denote this entry by the j:th *level*, as $\Theta(\boldsymbol{x})$ is non-decreasing in the coordinates. It will become clear that the full problem can be solved (in some different ways) essentially by resolving a sequence of these MIPs.

The first level

The first level, i.e, the first element of $\Theta(\boldsymbol{x})$ (i.e, θ_1), can be obtained by solving

$$\begin{aligned}
\max \quad & \theta_1 && && (4.28)\\
\text{s.t.} \quad & \sum_p u_{dp} = 1 && \forall\ d && (4.29)\\
& \sum_d \sum_p \delta_{edp} u_{dp} \theta_1 \le c_e && \forall\ e && (4.30)\\
& \theta_1 \ge 0, \quad u_{dp} \in \{0, 1\} && \forall\ p, d && (4.31)
\end{aligned}$$

Apparently, this formulation has a difficulty of being nonlinear, containing multiplication of two variables (constraint (4.30)). However, by defining a new variable, $\mu = \frac{1}{\theta_1}$, this problem can be avoided. Of course, it is then necessary to presume that $\theta_1 > 0$, which is reasonable (if $\theta_1 = 0$ there are demands that have no paths). Clearly, μ will be minimal when (and only when) θ_1 is maximal. Further, if we restate the problem in variable μ, the nonlinearity is eliminated, making use of that neither θ_1 nor μ are

demand-dependent;

$$\begin{aligned}
\min \quad & \mu & & (4.32)\\
\text{s.t.} \quad & \sum_p u_{dp} = 1 & \forall\ d & (4.33)\\
& \sum_d \sum_p \delta_{edp} u_{dp} \le \mu c_e & \forall\ e & (4.34)\\
& \mu \ge 0, \quad u_{dp} \in \{0,1\} & \forall\ p,d & (4.35)
\end{aligned}$$

The last formulation is a (linear) mixed 0-1 program. The advantages of the "inverted approach" taken in (4.32)-(4.35) have been exploited in [30], although for the splittable paths case (which is convex).

Finding all levels – an application-specific approach

Assuming knowledge of levels $1, 2, \ldots, k-1$ (i.e., entries $\theta_1, \theta_2, \ldots, \theta_{k-1}$), it is possible to formulate a MIP that computes level k. Performed iteratively, with the first level obtained from (4.32)-(4.35), this provides a resolution technique for the considered problem. We will start by giving a MIP formulation to compute the second level, assuming that the first level , θ_1^*, is known. For this purpose a variable, $y_2 = \theta_2 - \theta_1^*$, describing adjacent level difference, is defined.

$$\begin{aligned}
\max \quad & y_2 & & (4.36)\\
\text{s.t.} \quad & \sum_d \sum_p \sigma_{dp} = (D-1) y_2 & & (4.37)\\
& \sum_p \sigma_{dp} \le y_2 & \forall\ d & (4.38)\\
& \sigma_{dp} \le M w_{dp} & \forall\ p,d & (4.39)\\
& \sum_d \sum_p w_{dp} = D - 1 & & (4.40)\\
& w_{dp} \le u_{dp} & \forall\ p,d & (4.41)\\
& \sum_p u_{dp} = 1 & \forall\ d & (4.42)\\
& \sum_d \sum_p \delta_{edp} (u_{dp} \theta_1^* + \sigma_{dp}) \le c_e & \forall\ e & (4.43)\\
& u_{dp} \in \{0,1\},\ \ w_{dp}, \sigma_{dp}, y_2 \ge 0 & \forall\ p,d & (4.44)
\end{aligned}$$

where M is a sufficiently large number, e.g. $M = \max_e\{c_e\}$. Here, constraints (4.37) and (4.38) establish that $D-1$ demands reach θ_2. Constraints (4.39)-(4.42) assure that demands use only one path each, and constraint (4.43) is the usual link-load constraint. Note that it is possible that $y_2^* = 0$,

in which case $\theta_2^* = \theta_1^*$. Given an optimal solution y_2^* to (4.36)-(4.44), the second level is computed as $\theta_2^* = y_2^* + \theta_1^*$. Defining $y_k = \theta_k - (\sum_{i=2}^{k-1} y_i^* + \theta_1^*)$, this formulation can be extended to find an arbitrary level (provided that the lower levels are known), generalizing the MIP of (4.36)-(4.44);

$$\max \quad y_k \tag{4.45}$$

$$\text{s.t.} \quad \sum_d \sum_p \sigma_{dp} = (D - (k-1))y_k \tag{4.46}$$

$$\sum_p \sigma_{dp} \leq y_k \qquad \forall\ d \tag{4.47}$$

$$\sigma_{dp} \leq M w_{dp}^{(k)} \qquad \forall\ p, d \tag{4.48}$$

$$\sum_d \sum_p w_{dp}^{(k)} = D - (k-1) \tag{4.49}$$

$$w_{dp}^{(2)} \leq u_{dp} \qquad \forall\ p, d \tag{4.50}$$

$$w_{dp}^{(r+1)} \leq w_{dp}^{(r)} \qquad \forall\ p, d, r \tag{4.51}$$

$$\sum_d \sum_p w_{dp}^{(r)} = D - (r-1) \qquad \forall\ r \tag{4.52}$$

$$\sum_p u_{dp} = 1 \qquad \forall\ d \tag{4.53}$$

$$\sum_d \sum_p \delta_{edp}(u_{dp}\theta_1^* + \sum_{r=2}^{k-1} w_{dp}^{(r)} y_r^* + \sigma_{dp}) \leq c_e \qquad \forall\ e \tag{4.54}$$

$$u_{dp} \in \{0,1\},\ w_{dp}^{(r)}, \sigma_{dp}, y_k \geq 0 \qquad \forall\ p, d, r \tag{4.55}$$

Here, $r = 2, 3, \ldots, k-1$. It should be noted that the correctness of (4.45)-(4.55) relies entirely on that $w_{dp}^{(i)}$ is forced binary for all $i = 2, \ldots, k$, although there is no such explicit constraint.

Property 4.9. *If θ_1^* and all y_i^*, $i = 2, \ldots, k-1$, are optimal, then for any feasible solution to (4.45)-(4.55) it holds that $w_{dp}^{(i)} \in \{0,1\}$, for all $i = 2, \ldots, k$.*

Proof. Suppose that $w_{dp}^{(r-1)} \in \{0,1\}$, for all $p = 1, 2, \ldots, P_d$, $d = 1, 2, \ldots, D$, for some r, $3 \leq r \leq k$. As $w_{dp}^{(r-1)} \in \{0,1\}$, and since $\sum_d \sum_p w_{dp}^{(r-1)} = D - r + 2$, there exist exactly $D - r + 2$ non-zero variables $w_{dp}^{(r-1)}$, all of which must be equal to 1. By (4.51), $w_{dp}^{(r-1)} \geq w_{dp}^{(r)}$, so $w_{dp}^{(r-1)} = 0$ implies $w_{dp}^{(r)} = 0$. Now for y_{r-1}^* to be optimal there must exist a (d, p), such that $w_{dp}^{(r-1)} = 1$ and $w_{dp}^{(r)} = 0$ (otherwise y_{r-1}^* could be increased). Combined with the constraint $\sum_p w_{dp}^{(r)} = D - r + 1$, this implies that there must exist $D - r + 1$ non-zero

variables $w_{dp}^{(r)}$, that sum up to $D - r + 1$. Since $0 \leq w_{dp}^{(r)} \leq 1$, it follows that $w_{dp}^{(r)} \in \{0, 1\}$. An identical argument can be used showing the same relation beween $w_{dp}^{(2)}$ and u_{dp}, implying that $w_{dp}^{(2)} \in \{0, 1\}$ as $u_{dp} \in \{0, 1\}$ by definition. Thus by induction over r the general result follows. □

Summarizing the above findings, it is possible to formulate a procedure that solves

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\}, \tag{4.56}$$

as is done by the following algorithm.

Algorithm 4.3. (*for solving* (4.24)-(4.26))

Step 0: Assign $k := 1$. Solve (4.32)-(4.35) for the first level, $\theta_1^* = \frac{1}{\mu^*}$ and optimal u_{dp}. If $D = 1$ then stop. The MMF solution is $\boldsymbol{x} = \theta_1^* \boldsymbol{u}$, and the used paths are given by optimal u_{dp}. Otherwise proceed to Step 1.

Step 1: Assign $k := k + 1$. If $k > D$ then stop. Then the MMF allocation is given by, for each demand d, $x_d^* = \theta_1^* + \sum_p \left(\sum_{r=2}^{D} w_{dp}^{(r)} y_r^* + \sigma_{dp} \right)$, and the optimal paths are given by optimal u_{dp}. The optimal levels are given by $\theta_i(\boldsymbol{x}) = \theta_1^* + \sum_{j=2}^{i} y_j^*$. Otherwise proceed to Step 2.

Step 2: Solve (4.45)-(4.55) for optimal objective, y_k^*, optimal variables $w_{dp}^{(r)}$, u_{dp}, and σ_{dp}. Go to Step 1.

Finding all levels – a generic method

The MIP model outlined in this section is application of the method of formulating a non-convex MMF problem by cumulated ordered values, described in Section 2.4.2. It is referred to as a *generic method*, since its original form constitutes a general way of expressing a wide range of non-convex MMF problems. Let us repeat the derivation of this formulation, for the considered application. Define the k:th *cumulated ordered value*, $\bar{\theta}_k(\boldsymbol{x})$, $1 \leq k \leq D$, as $\bar{\theta}_k(\boldsymbol{x}) = \sum_{j=1}^{k} \theta_j(\boldsymbol{x})$. Then, for a given outcome vector $\boldsymbol{x}$, this entity can be computed as

$$\bar{\theta}_k = \max \quad kr_k - \sum_d b_{kd} \tag{4.57}$$

$$\text{s.t.} \quad r_k - x_d \leq b_{kd} \qquad \forall\, d \tag{4.58}$$

$$b_{kd} \geq 0 \qquad \forall\, d \tag{4.59}$$

which is linear even if $\boldsymbol{x}$ is a variable. Note that this is application of the expressability result described by (2.5)-(2.6) in Section 2.1.3. Using the possibility of rewriting (2.4) as (2.7)-(2.10) we may rewrite

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\}$$

as

$$\begin{aligned}
\text{lex max} \quad & \bar{\Theta}(\boldsymbol{x}) && && (4.60)\\
\text{s.t.} \quad & \bar{\theta}_k = kr_k - \sum_d b_{kd} && \forall k && (4.61)\\
& r_k - x_d \le b_{kd} && \forall\ k,d && (4.62)\\
& b_{kd} \ge 0 && \forall\ k,d && (4.63)\\
& \boldsymbol{x} \in X && && (4.64)
\end{aligned}$$

where $k = 1, 2, \ldots, D$. For the purpose of solving problem (4.60)-(4.64), with X constituted by the single-path and the link-load constraints, we may apply Algorithm 2.3, expressed for the considered application by the following algorithm.

Algorithm 4.4. (*for solving* (4.24)-(4.26))

Step 0: Assign $k := 1$. Solve (4.32)-(4.35) for the first level, $\theta_1^* = \frac{1}{\mu^*}$ and optimal u_{dp}. If $D = 1$ then stop. The MMF solution is $\boldsymbol{x} = \theta_1^* \boldsymbol{u}$, and the used paths are given by optimal u_{dp}. Otherwise proceed to Step 1.

Step 1: Assign $k := k + 1$. If $k > D$ then stop. Then the MMF allocation is given by, for each demand d, $x_d^* = \sum_p z_{dp}$ (optimal variables z_{dp}), and the optimal paths are given by optimal u_{dp}. Otherwise proceed to Step 2.

Step 2: Let $l = 1, 2, \ldots, k$, and let M be a sufficiently large number, e.g. $M = \max_e\{c_e\}$. Solve

$$\begin{aligned}
\text{max} \quad & \bar{\theta}_k && && (4.65)\\
\text{s.t.} \quad & \bar{\theta}_k \le kr_k - \sum_d b_{kd} && && (4.66)\\
& \bar{\theta}_l^* \le lr_l - \sum_d b_{ld} && \forall\ l : l \ne k && (4.67)\\
& r_l - \sum_p z_{dp} \le b_{ld} && \forall\ l,d && (4.68)\\
& z_{dp} \le u_{dp} M && \forall\ p,d && (4.69)\\
& \sum_p u_{dp} = 1 && \forall\ d && (4.70)\\
& \sum_d \sum_p \delta_{edp} z_{dp} \le c_e && \forall\ e && (4.71)\\
& z_{dp} \ge 0,\ b_{ld} \ge 0,\ u_{dp} \in \{0,1\} && \forall\ p,l,d && (4.72)
\end{aligned}$$

for optimal objective $\bar{\theta}_k^*$ and optimal variables z_{dp} and u_{dp}. Go to Step 1.

Finding all levels – assuming a finite outcome set

In many cases, especially in the communication network context, it is very reasonable to assume that the flow volume allocated to demand d may take on only a limited set of discrete values. This was the explicit requirement in Section 4.1.5, where we on the other hand assumed predefined single paths. Recall that the assumption of discrete flow values is very reasonable if flows can only be assigned multiples of a predefined module. The restriction results in essentially the same problem, but with a *finite outcome set*. Introducing this restriction to the considered problem yields the following:

$$\begin{aligned} \text{lex max} \quad & \Theta(\boldsymbol{x}) && && (4.73)\\ \text{s.t.} \quad & \boldsymbol{x} \in X && && (4.74)\\ & x_d \in v && \forall\ d && (4.75) \end{aligned}$$

where $v = \{0, v_1, v_2, \ldots, v_r\}$, $0 < v_1 < v_2 < \ldots < v_r$, is a finite (ordered) set of possible flow values. From Property 2.15 we know that a solution to (4.73)-(4.75) is obtained through solving the following lexicographical minimization:

$$\begin{aligned} \text{lex min} \quad & \left(\textstyle\sum_d t_{1d}, \sum_d t_{2d}, \ldots, \sum_d t_{rd} \right) && && (4.76)\\ \text{s.t.} \quad & v_k - x_d \leq t_{kd} && k = 1, 2, \ldots, r,\ \forall\ d && (4.77)\\ & t_{kd} \geq 0 && k = 1, 2, \ldots, r,\ \forall\ d && (4.78)\\ & \boldsymbol{x} \in X && && (4.79)\\ & x_d \in v\ . && \forall\ d && (4.80) \end{aligned}$$

Intuitively, it is not hard to see that (4.76)-(4.80) first assures that as many demands as possible are raised to the first non-zero level, v_1. Keeping this many demand volumes greater or equal than v_1, it is then assured that as many demands as possible are raised to v_2, and so on. Now since our primary interest is to enforce unsplittable flows, not necessarily requiring that flows belong to the predefined set of values, the idea is to try to solve (4.76)-(4.80), omitting the explicit value constraint (4.80). Such an approach may seem far-fetched but is motivated from positive experience with running times of this approach applied to related location problems [41]. Resolution of (4.76)-(4.79) is accomplished by the following algorithm.

Algorithm 4.5. (*for approximately solving* (4.24)-(4.26))

Step 0: Assign $k := 1$. Let M be a sufficiently large number, e.g. $M = v_1$.

Solve

$$\tau_1 = \min \quad \sum_d t_{1d} \tag{4.81}$$

$$\text{s.t.} \quad v_1 - \sum_p z_{dp} \le t_{1d} \qquad \forall\ d \tag{4.82}$$

$$z_{dp} \le u_{dp} M \qquad \forall\ p, d \tag{4.83}$$

$$u_{dp} = 1 \qquad \forall\ d \tag{4.84}$$

$$\sum_d \sum_p \delta_{edp} x_d \le c_e \qquad \forall\ e \tag{4.85}$$

$$z_{dp} \ge 0, t_{1d} \ge 0, u_{dp} \in \{0,1\} \qquad \forall\ d, p \tag{4.86}$$

and let τ_1^* be the optimal objective value, z_{dp} and u_{dp} the optimal solution. If $D = 1$ then stop. The MMF solution is then given by optimal variables z_{dp}, and the used paths are given by optimal u_{dp}. Otherwise proceed to Step 1.

Step 1: Assign $k := k + 1$. If $v_{k-1} > \max_d\{\sum_p z_{dp} : d = 1, 2, \ldots, D\}$ or $k > r$ then stop, optimal z_{dp} give the (approximate) MMF solution and optimal u_{dp} give the used paths. Otherwise proceed to Step 2.

Step 2: Let $l = 1, 2, 3, \ldots, k-1$, and let M be a sufficiently large number, e.g. $M = v_k$. Solve

$$\tau_k = \min \quad \sum_d t_{kd} \tag{4.87}$$

$$\text{s.t.} \quad v_l - \sum_p z_{dp} \le t_{ld} \qquad \forall\ l, d \tag{4.88}$$

$$\sum_d t_{ld} \le \tau_l^* \qquad \forall\ l \tag{4.89}$$

$$v_k - \sum_p z_{dp} \le t_{kd} \qquad \forall\ d \tag{4.90}$$

$$z_{dp} \le u_{dp} M \qquad \forall\ p, d \tag{4.91}$$

$$\sum_p u_{dp} = 1 \qquad \forall\ d \tag{4.92}$$

$$\sum_d \sum_p \delta_{edp} z_{dp} \le c_e \qquad \forall\ e \tag{4.93}$$

$$z_{dp} \ge 0,\ t_{ld} \ge 0,\ u_{dp} \in \{0,1\} \qquad \forall\ l, p, d \tag{4.94}$$

and let τ_k^* be the optimal objective value, z_{dp} and u_{dp} the optimal solution. Go to Step 1.

It should be noted that, since we are interested in the case where continuous flows are allowed, iteratively solving (4.87)-(4.94) will not give an

exact solution to the considered problem, but, as we shall see, in many cases a good approximation.

Upper and lower bounds

In a communication network, it is often a reasonable assumption that each demand d is given with acceptation limits, h_d and H_d, where it is required that the allocation vector satisfies $h_d \leq x_d \leq H_d$, for all d, $d = 1, 2, \ldots, D$. The upper limit, H_d, can be incorporated in a natural fashion in all of the presented MIP models. This is just a matter of substituting the so-called "big-M constraints", $z_{dp} \leq u_{dp}M$, with $z_{dp} \leq u_{dp}H_d$ for all paths $p = 1, 2, \ldots, P_d$, for all demands $d = 1, 2, \ldots, D$. The lower limit, h_d, has to be added as explicit constraints, $h_d \leq x_d$ for all demands. Note that the acceptation limits have to be carefully a priori determined, since otherwise there is a large risk that they make the MIP model infeasible.

4.2.4 Numerical experiments

As has been shown, the studied problem suffers from heavy computational complexity. This is mainly due to the single-paths requirement and the fact that it embodies a multi-criteria optimization problem. Therefore, it is interesting to study and compare computation times for different instances and for different resolution techniques.

Exact methods

We will start by giving a flavour of the computation times associated with solving

$$\text{lex max}\{\Theta(\boldsymbol{x}) : \boldsymbol{x} \in X\},$$

where the feasible set X models the single-path and the link-load constraints, in an exact manner. Two of the presented approaches solve the problem to optimality – one is called *application-specific* (Algorithm 4.3), and the other will be referred to as *generic* (Algorithm 4.4). Both methods require solution of a sequence of MIPs. The frameworks of the algorithms are implemented in MATLAB6.5, and the MIPs are solved using a MATLAB interface to CPLEX 9 (mex-function), called CPLEXINT (downloadable freeware [5]). This means that the generic CPLEX 9 MIP-solver is used for resolving MIPs (4.32)-(4.35), (4.45)-(4.55), and (4.65)-(4.72). The computations were carried out on a PC with an Intel PIII-1GHz CPU, RAM of 256 MB, and Windows 2000 OS. Table 4.4 contains computation times for the two different methods, for a number of small instances. The instances are characterized by that there is a demand between every node-pair and that link capacities are uniformly distributed over $\{10, 20, 30, 40, 50\}$. For the first 9 instances

there are two paths per demand, and for the 5 last instances there are three paths per demand.

#nodes	#links	#paths/ demand	computation times (s)	
			app-specific	generic
3	3	2	0.1090	0.0940
4	6	2	0.0940	0.0470
5	8	2	0.8130	0.4710
6	12	2	1.1230	1.3730
7	12	2	10.098	4.7190
8	13	2	16.420	21.139
9	18	2	$1.62{\cdot}10^3$	327.92
10	18	2	$1.61{\cdot}10^3$	327.11
11	19	2	$1.92{\cdot}10^3$	107.41
4	6	3	0.4220	0.1080
5	8	3	1.1560	0.7020
6	12	3	8.8550	13.077
7	17	3	26.738	28.999
8	12	3	236.49	46.704

Table 4.4: Comparison of the application-specific and generic methods for some small instances.

The results of Table 4.4 suggest that the generic method is slightly faster than that of the application-specific method. We may sum up the usage of variables and constraints for the MIP corresponding to iteration k in the two different approaches, as is done in Table 4.5. Knowing that both algorithms performs $D-1$ iterations ($2 \leq k \leq D$), the generic method has a clear advantage.

	application-specific	generic
B	$\sum_d P_d$	$\sum_d P_d$
R	$k\sum_d P_d + 1$	$k\|D\| + 2\sum_d P_d$
C	$1 + 2\|D\| + k\sum_d P_d + E$	$k + (k+1)D + \sum_d P_d + E$

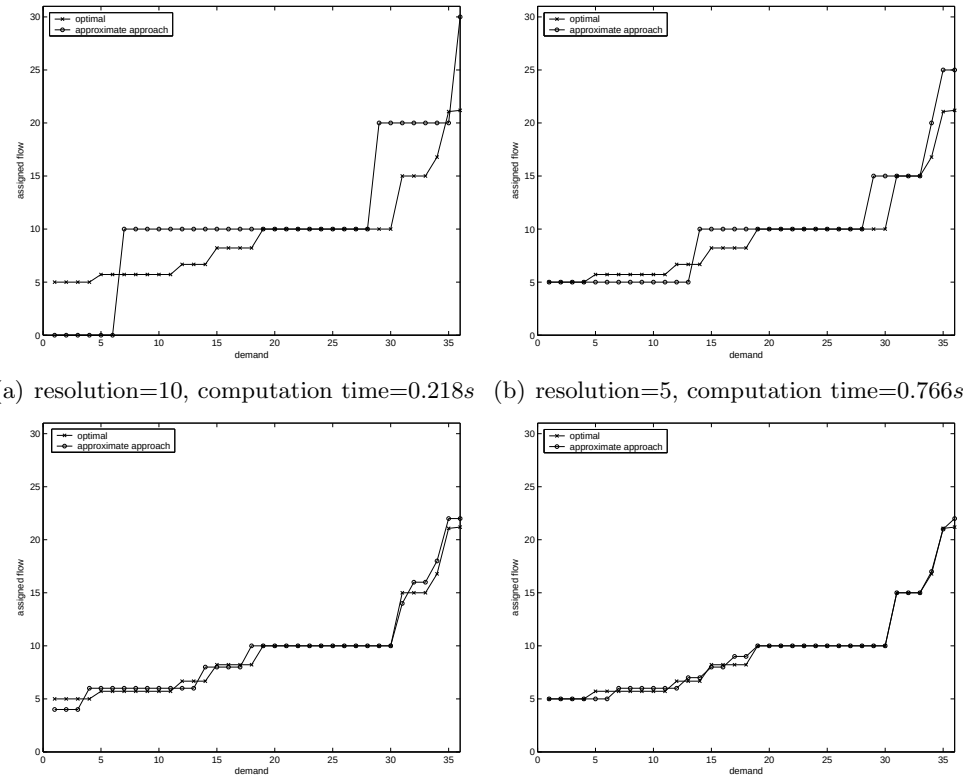
Table 4.5: Comparison of number of variables and constraints for step k in the application-specific and generic methods. B, R, and C indicate the number of binary variables, continuous variables, and constraints, respectively.

Assuming finite outcomes

The approach based on the assumption that the possible outcomes are finite can be seen as an approximate method for solving the studied problem. From this viewpoint it is interesting too see, if replacing an exact method by Algorithm 4.5, how much computation time that can be saved, and to what cost, in terms of deviation from optimum, this may be done. We then have to predefine a set of values, v, which the method will use trying to assign its values to flows. The hardware and software constellations are identical to those of the numerical experiments of the exact methods. In Figure 4.9 the (sorted) allocation vector for a 10-node, 18-link, 2 paths/demand instance is shown (entry 8 of Table 4.4). Link capacities are uniformly distributed over $\{10, 20, 30, 40, 50\}$. The lines with crosses are the allocation vectors obtained by the exact generic algorithm (Algorithm 4.4), whereas the circled lines are the allocation vectors obtained by the approximate approach (Algorithm 4.5) with 4 different set of values, v. In Figure 4.9(a), modules (resolution) are of size 10, i.e., the elements of the predefined grid, $v_1, v_2, \ldots, v_r$, are all the numbers v_i that can be written as $v_i = 10 \cdot z$, $z \in \mathbb{Z}$, $0 \leq v_i \leq 50$. The grid associated with Figures 4.9(b), 4.9(c), and 4.9(d) are defined analogously, but with module sizes of 5, 2, and 1, respectively. It should be noted that for the most dense grid considered (Figure 4.9(d)), the computation time is merely 14.54 seconds. The exact method solves the same instance in about 5 minutes.

In Table 4.6, the approximate approach is applied to a number of different instances. Again, link capacities are distributed over $\{10, 20, 30, 40, 50\}$, and there is a demand between every node-pair in the network. The computation times are given for Algorithm 4.5 with module size 2 (dist-2), module size 5 (dist-5), and also for the best exact method – the generic method (Algorithm 4.4). Dashed entries correspond to runs that did not finish within half an hour. Comparison of the exact methods and the approximate approach, by the computation times given in Table 4.6, strongly suggests that the approximate approach should be used whenever its deviation from the optimum is acceptable. Certainly, such an error tolerance will depend on the details of the application. From our numerical results it is tempting to conjecture that for sorted allocation vectors, if we consider an arbitrary entry, the absolute difference between distribution approach and the exact solution can be at most the size of the module. It is however easy to come up with a counterexample

Example 4.7. Consider the network given in Fig. 4.10. In this graph there are two demands; (A, A') and (B, B'). All link capacities are equal to 15, except for link e that has a link capacity of 5. The exact solution is $\Theta(\boldsymbol{x}) = (7.5, 7, 5)$. Suppose that our predefined grid is $\{0, 5, 10, 15\}$. Then the approximate approach solution is $\Theta(\boldsymbol{x}^{\Delta}) = (5, 15)$. Thus $\theta_2(\boldsymbol{x}^{\Delta}) - \theta_2(\boldsymbol{x}) = 7.5 > v = 5$.



(a) resolution=10, computation time=0.218s (b) resolution=5, computation time=0.766s

(c) resolution=2, computation time=5.84s (d) resolution=1, computation time=14.54s

Figure 4.9: Comparison of different resolutions for the approximate approach. The exact solution was computed in 327 seconds.

4.3 Conclusions

In Chapter 4 we have considered (linear) max-min fairness allocation problems, with non-convex problem structure. The non-convexity of the studied problems results from discrete solution spaces, and complicates both formulation and resolution. In Chapter 3 we did not encounter this difficulty as all decision variables were allowed to take on rational numbers, and we could in fact exploit the convex problem structure to formulate solution algorithms.

The first non-convex MMF problem studied was almost identical to the most basic one described in Chapter 3 – when each demand has exactly one fixed path and flows have to be max-min fairly distributed. The difference lies in that we now only allow flows to assume multiples of a predefined module. It was shown that, changing units, this requirement can be viewed as a requirement of flows being integral. We compared the solution of the integral-flow problem with the basic problem, and derived a number of relational properties. Moreover, an analogy to the max-saturating property was given. In order to motivate suspicion of non-existence of polynomial

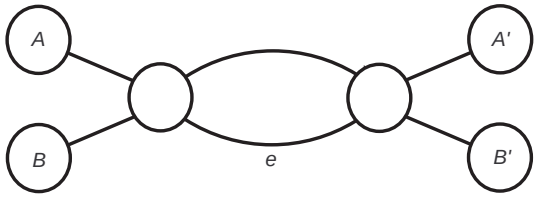


Figure 4.10: A contradicting example.

time algorithms we then proved that obtaining MMF of modular flows is $\mathcal{NP}$-hard. Nevertheless, using the classical approach of waterfilling (Algorithm 3.1), some subproblems could be shown to be of polynomial time. For resolution of the considered problem, we proposed an algorithm that solves an application-specific linear relaxation of a lexicographical minimization problem ((2.72)-(2.76)). Strictly speaking, since being a relaxation, this consituted an approximation algorithm. However, it was shown to reach an optimal solution in practically all of the randomly generated problem instances, and this at acceptable running times, even for relatively large instances.

We then took on the problem of obtaining a max-min fair distribution of flows if each demand may use no more than one path that has to be selected in the problem solution process. This problem is often refererred to as max-min fairness of unsplittable flows. General unsplittable-flow problems are known to be difficult, so it was reasonable to expect provable difficulty also from this problem. We separated the problem into two cases – the case when each demand may use any path, and the case when each demand may use one path from a predefined list. With the objective of obtaining max-min fairness, both cases were shown $\mathcal{NP}$-hard. In fact we proved that just obtaining the first entry of the sorted allocation vector is $\mathcal{NP}$-hard in itself, in both cases.

With a deeper understanding of the complexity of the unsplitabble flow MMF problem, the focus was then put on resolution techniques. We presented two exact, MIP-based algorithms, where one was derived specifically for this application, and the other a generic MMF problem resolution technique. In addition we gave an approximation algorithm that was based on (a relaxation of) the assumption that the outcome set is discrete and finite. The numerical experiment, which was carried out on network instances ranging from 3 nodes and 3 demands to 20 nodes and 190 demands, showed that, in general, the generic algorithm is usually faster than the

application-specific one, if the problem should be solved to optimality. However, both approaches did indeed exhibit vast time-consumption when problem instances were scaled up. Finally, we applied the approximate method to our randomly generated problem instances. Even though being based on successive resolution of MIPs, this approach obtained suboptimal solutions of quite high quality at acceptable running times in cases where the exact methods did not solve the problem in less than half an hour.

#nodes	#links	#paths/ demand	computation times (s)		
			generic method	dist-2	dist-5
3	3	2	0.1090	0.0940	0.0780
4	6	2	0.0940	0.0470	0.0770
5	8	2	0.8130	0.4710	0.2810
6	12	2	1.3730	0.5920	0.1550
7	12	2	4.7190	0.7490	0.2640
8	13	2	21.139	1.6750	0.2970
9	18	2	327.92	6.0000	0.7510
10	18	2	327.11	6.0630	0.7170
11	19	2	107.41	7.5480	0.5150
4	6	3	0.1080	0.2950	0.1560
5	10	3	0.9840	0.9380	0.2810
6	12	3	13.077	4.8110	0.4380
7	17	3	28.999	4.9530	1.6090
8	17	3	76.045	10.518	1.2370
9	16	3	$1.239{\cdot}10^3$	31.809	0.6860
10	22	3	–	58.518	5.3940
11	23	3	–	180.81	2.5340
12	24	3	–	50.954	1.5780
13	26	3	–	51.781	2.5780
14	24	3	–	16.640	1.2200
15	25	3	–	47.485	2.4700
16	33	3	–	59.623	5.6410
16	34	3	–	29.400	3.1260
17	36	3	–	99.595	12.333
18	40	3	–	358.72	5.5310
19	38	3	–	$1.838{\cdot}10^3$	4.9680
20	36	3	–	108.31	8.0770

Table 4.6: Comparison of computation times of the approximate approach and the exact (generic) method, for various instances.

Chapter 5

Dimensioning problems

This chapter deals with so-called dimensioning problems. It is significant for a dimensioning problem that, as opposed to an allocation problem, the link capacities are assumed to be design variables. The designation for network optimization problems for which link capacities are to be determined, is uncapacitated problems. In the spirit of the previous chapters, we will consider dimensioniong problems for which fairness among demands is a preliminary objective. The type of fairness that will be addressed is commonly known as proportional fairness, introduced in Section 2.1.2.

5.1 Proportionally fair dimensioning

The essence of Proportional Fairness (PF) is to assign flows to paths so that the sum of logarithms of the total flows is maximized. The use of the logarithmic function, instead of e.g., a linear function (which in particular could lead to throughput maximization), makes it impossible to assign zero flow to any demand and, on the other hand, makes it unprofitable to assign too much flow to any individual demand. A more formal investigation (as was carried out in Section 2.1.2) establishes that use of the logarithmic function has the important property of defining a fair rational preference relation on the strictly positive orthant, $\mathbb{R}^m_{++}$. A number of studies considering the capacitated flow allocation problems for PF have been carried out before [10, 14, 15]. In the capacitated problems, link capacities are given and the optimal flow allocation pattern is to be found. The capacitated problem with a fixed path for each demand is considered in [15]. In [10] and [14], a more general problem of simultaneously finding demands' paths and a corresponding flow allocation pattern is investigated. It is shown in [43] how the capacitated problem can be approached by piece-wise linear approximation of the logarithm. Here, we concentrate on the PF dimensioning problem, assuming that we are given a set of demands (each with an associated list of paths) and the topology of the network. The problem is to assign flows to demands and capacity to links such that the resulting allocation vector is proportionally fair, and such that the aggregated flow traversing a link does not exceed the capacity. To bound the problem, we assume that we are given a marginal cost for each link e, ρ_e, and some budget, B. It should then hold for the link capacities, y_e, (which now are decision variables) that

$$\sum_e \rho_e y_e \leq B. \tag{5.1}$$

Assuming that each demand is given a fixed path, the basic PF dimensioning problem takes the following shape:

$$f(\boldsymbol{x}) = \max \quad \sum_d w_d \log x_d \tag{5.2}$$

$$\text{s.t.} \quad \sum_e \rho_e y_e \leq B \tag{5.3}$$

$$\sum_d \delta_{ed} x_d \leq y_e \qquad \forall\ e \tag{5.4}$$

$$\boldsymbol{x} \geq 0,\ \boldsymbol{y} \geq 0\ . \tag{5.5}$$

Here, $\boldsymbol{y} = (y_1, y_2, \ldots, y_E)$, and w_d is the value of a unit of demand d (for pure PF, $w_d = 1$ for all demands d). We know from Chapter 2 that usage of the logarithmic objective function induces a fair rational preference relation. From a more intuitive perspective the interpretation of the objective function (5.2) can be as follows. The client associated with demand d does not pay for x_d (assigned bandwidth) but for the logarithm of x_d. This implies that the revenue from a client can be negative if he is assigned too little bandwidth. With function (5.2) this happens if the flow assigned to a client is less than one.

The idea underlying the above interpretation was introduced as early as in 1854 by Hermann Heinrich Gossen in his book "The Development of the Laws of Exchange among Men and of the Consequent Rules of Human Action". Gossen's First Law states that the pleasure obtained from each additional amount consumed by the same commodity diminishes until satiety is reached [4]. Another side of this law could be a statement that the pleasure approaches minus infinity as the amount of consumed commodity tends to zero.

Proposition 5.1. *Let $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ be the solution of* (5.2)-(5.5)*. Then the following equalities hold:*

$$\begin{aligned} f(\boldsymbol{x}^*) &= \log(B) \sum_d w_d - \sum_d w_d \log(\xi_d) + \\ &\quad + \sum_d w_d \log(w_d) - \log\left(\ \textstyle\sum_d w_d\ \right) \sum_d w_d\ , \end{aligned} \tag{5.6}$$

$$x_d^* = \frac{w_d B}{\xi_d \sum_d w_d}\ ,\ \textit{and} \tag{5.7}$$

$$y_e^* = \sum_d \delta_{ed} x_d^*\ , \tag{5.8}$$

where ξ_d is the cost of the path realizing demand d, $\xi_d = \sum_e \rho_e \delta_{ed}$.

Proof. The formulae follow from the explicit solution of the dual. To obtain the solution we note that for an optimal solution equalities must hold for

(5.3) and (5.4), resulting in

$$\sum_e \rho_e \sum_d \delta_{ed} x_d = B \ . \tag{5.9}$$

We dualize (5.9), and form the Lagrangian,

$$L(\boldsymbol{x};\sigma) = -\sum_d w_d \log x_d + \sigma \left(\ \textstyle\sum_e \rho_e \sum_d \delta_{ed} x_d - B \ \right) \ , \tag{5.10}$$

and rewrite it as

$$L(\boldsymbol{x};\sigma) = \sum_d \left(\ \left(\ \sigma \textstyle\sum_e \rho_e \delta_{ed} \ \right) x_d - w_d \log x_d \ \right) - \sigma B \ . \tag{5.11}$$

The dual function is given by

$$W(\sigma) = \min_{\boldsymbol{x} \geq \mathbf{0}} L(\boldsymbol{x};\sigma) \ , \tag{5.12}$$

which should be maximized with respect to the dual multiplier σ, in order to find a solution to the dual. Let σ^* be such an optimal multiplier, i.e.,

$$W(\sigma^*) = \max_{\sigma} W(\sigma) \ , \tag{5.13}$$

where

$$W(\sigma) = \min_{\boldsymbol{x} \geq \mathbf{0}} \sum_d \left(\ \sigma \xi_d x_d - w_d \log x_d \ \right) - \sigma B \ . \tag{5.14}$$

The flows, $x_d(\sigma)$, minimizing the Lagrangian for a fixed σ are obtained from the stationary point of (5.14). The consecutive flows are obtained by differentiating the terms of the sum in (5.14), and putting the consecutive derivatives equal to zero,

$$\frac{w_d}{x_d(\sigma)} - \sigma \xi_d = 0 \ , \tag{5.15}$$

resulting in

$$x_d(\sigma) = \frac{w_d}{\sigma \xi_d} \ . \tag{5.16}$$

Inserting (5.16) into (5.14) we finally arrive at

$$W(\sigma) = \sum_d \left(\ w_d - w_d \log \left(\ \tfrac{w_d}{\sigma \xi_d} \ \right) \ \right) - \sigma B \ . \tag{5.17}$$

The maximum of the dual function is attained at the stationary point of (5.17) with respect to the multiplier σ, resulting in the equality

$$\sum_d \frac{w_d}{\sigma} - B = 0 \ . \tag{5.18}$$

Hence, the optimal multiplier σ^* is given by

$$\sigma^* = \sum_d \frac{w_d}{B} \ , \tag{5.19}$$

which immediately implies (5.6) and (5.7). □

Of course, the maximum of the objective function (5.2) depends only on B (provided that the rest of the constraints are fixed); (5.6) implies that this maximum is of the form

$$f(B) = a \log(B) + b \ . \tag{5.20}$$

Furthermore, from formula (5.2) it is deduced that when paths are also subject to optimization, i.e., when each demand is not restricted to a single fixed path, then the optimal solution will assign each demand flow x_d to its shortest path. This is because only the second term on the right hand side of (5.6) depends on path selection. The same term is minimized when the shortest paths are used.

5.1.1 Bounded flows

As was discussed in the balance of Section 4.2.3, it is often very reasonable to assume that a demand can be assigned a flow only within a predefined range,

$$h_d \le x_d \le H_d \ , \quad d = 1, 2, \ldots, D \ . \tag{5.21}$$

To make sense, it must hold for such bounds that $0 \le h_d < H_d$, and further that $\sum_d h_d \xi_d < B < \sum_d H_d \xi_d$. We will continue the investigation of the PF dimensioning problem by adding (5.21) to optimzation problem (5.2)-(5.5), resulting in

$$\begin{aligned}
f(\boldsymbol{x}) = \ \max \quad & \sum_d w_d \log x_d && && (5.22)\\
\text{s.t.} \quad & \sum_e \rho_e y_e \le B && && (5.23)\\
& \sum_d \delta_{ed} x_d \le y_e && \forall\ e && (5.24)\\
& h_d \le x_d \le H_d && \forall\ d && (5.25)\\
& \boldsymbol{x} \ge 0, \ \boldsymbol{y} \ge 0 \ . && && (5.26)
\end{aligned}$$

The solution of the optimization problem defined by (5.22)-(5.26) can, just as (5.2)-(5.5), be successfully determined with the aid of dualization. The additional constraints need not necessarily be dualized but can be taken into

account explicitly in the minimization of the Lagrangian (5.14) for a fixed multiplier σ. This implies that formula (5.16) takes the form

$$x_d(\sigma) = \begin{cases} h_d \ , & \frac{w_d}{\sigma\xi_d} < h_d \\ \frac{w_d}{\sigma\xi_d} \ , & h_d \le \frac{w_d}{\sigma\xi_d} \le H_d \\ H_d \ , & H_d < \frac{w_d}{\sigma\xi_d} \ . \end{cases} \tag{5.27}$$

In this case the dual function can be maximized in an exact way, but explicit formulae like (5.6) and (5.7) are not available. To maximize the dual function we first calculate the threshold σ:s following from (5.27):

$$\sigma_d^1 = \frac{w_d}{H_d\xi_d}, \tag{5.28}$$

$$\sigma_d^2 = \frac{w_d}{h_d\xi_d} \ . \tag{5.29}$$

Note that, for a given demand d, $\sigma_d^1 < \sigma_d^2$ because $h_d < H_d$, and also that $\sigma_d^1 = 0$ if $H_d = +\infty$, and $\sigma_d^2 = +\infty$ if $h_d = 0$. Now the relations (5.27) can be rewritten as:

$$x_d(\sigma) = \begin{cases} H_d \ , & 0 \le \sigma \le \sigma_d^1 \\ \frac{w_d}{\sigma\xi_d} \ , & \sigma_d^1 \le \sigma \le \sigma_d^2 \\ h_d \ , & \sigma \ge \sigma_d^2 \ . \end{cases} \tag{5.30}$$

The next step is to sort all σ_d^1 and σ_d^2, $d = 1, 2, \ldots, D$, in non-decreasing order (in the sorted sequence some of the elements can be equal). Then, from each subsequence of equal elements all but one elements are deleted. This yields a shorter sequence $(s_1, s_2, \ldots, s_n)$, with the property

$$s_1 < s_2 < \ldots < s_n \ , \tag{5.31}$$

where s_1 may be equal 0 and s_n to $+\infty$. Now we form the following $n-1$ intervals:

$$[s_1, s_2], [s_2, s_3], \ldots, [s_{n-1}, s_n] \ , \tag{5.32}$$

and for each of them ($[s_j, s_{j+1}]$, $j = 1, 2, \ldots, n-1$) we introduce three sets of indices (demands);

$$\begin{aligned} L_j &= \{\ d \ : \ x_d(\sigma) \ = \ h_d \ \text{ for } \sigma \in [s_j, s_{j+1}]\ \} \ , \\ F_j &= \{\ d \ : \ x_d(\sigma) \ = \ \frac{w_d}{\sigma\xi_d} \ \text{ for } \sigma \in [s_j, s_{j+1}]\ \} \ , \text{ and} \\ U_j &= \{\ d \ : \ x_d(\sigma) \ = \ H_d \ \text{ for } \sigma \in [s_j, s_{j+1}]\ \} \ . \end{aligned} \tag{5.33}$$

Note that $L_j \cup F_j \cup U_j = \{1, 2, \ldots, D\}$ and $L_j \cap F_j = L_j \cap U_j = F_j \cap U_j = \emptyset$ for $j = 1, 2, \ldots, n-1$, i.e, the demands are partitioned into three disjoint sets.

Note also that $F_i \neq F_j$ for $i \neq j,\ i,j = 2,3,\ldots,n-2$ (provided $F_i, F_j \neq \emptyset$). For each interval $[s_j, s_{j+1}]$ the dual function takes the form

$$\begin{aligned} W(\sigma) &= \min_{\boldsymbol{x}\geq \mathbf{0}} \Big\{ \textstyle\sum_{d\in F_j} \big(\sigma\xi_d x_d - w_d \log(x_d) \big) - \\ &\quad -\sigma \Big(B - \textstyle\sum_{d\in L_j} \xi_d h_d - \sum_{d\in U_j} \xi_d H_d \Big) - \\ &\quad - \textstyle\sum_{d\in L_j} w_d \log(h_d) - \sum_{d\in U_j} w_d \log(H_d) \Big\} \quad , \end{aligned} \tag{5.34}$$

or equivalently,

$$\begin{aligned} W(\sigma) &= \sum_{d\in F_j} \Big(w_d - w_d \log(\tfrac{w_d}{\sigma\xi_d}) \Big) - \\ &\quad -\sigma \Big(B - \textstyle\sum_{d\in L_j} \xi_d h_d - \sum_{d\in U_j} \xi_d H_d \Big) - \\ &\quad - \sum_{d\in L_j} w_d \log(h_d) - \sum_{d\in U_j} w_d \log(H_d) \ . \end{aligned} \tag{5.35}$$

Example 5.1. Figure 5.1 illustrates the dual function for optimization problem (5.22)-(5.26) for a simple network with $D = 3$ demands and with cost coefficients $\xi_1 = \xi_2 = \xi_3 = 1$, reward coefficients $w_1 = 1$, $w_2 = 2$, $w_3 = 10$, and the budget $B = 13$. The bounds are as follows: $h_1 = 3$, $h_2 = 2$, $h_3 = 0$, $H_1 = +\infty$, $H_2 = +\infty$, $H_3 = 5$. The parameters imply that the intervals (5.32) are: $[0, \frac{1}{3}]$, $[\frac{1}{3}, 1]$, $[1, 2]$ and $[2, +\infty)$.

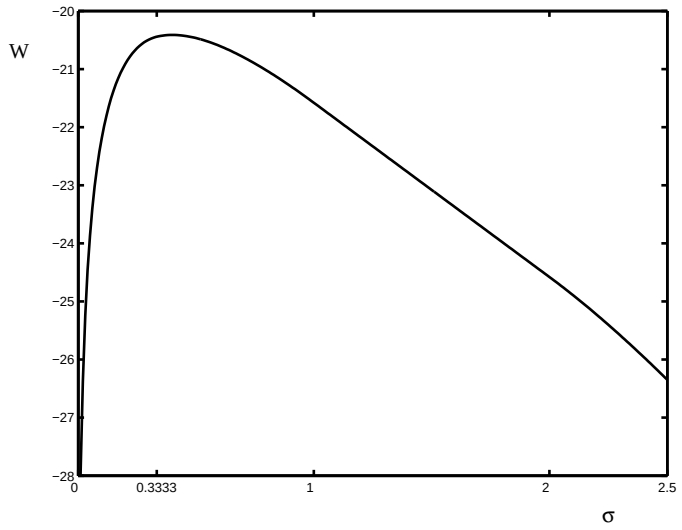


Figure 5.1: The dual function for a simple network.

Of course, function (5.35), as the dual function of the convex problem (5.22)-(5.26), is concave and thus continuous [27]. Figure 5.1 suggests that

it is also differentiable, which is actually the case. Note that according to (5.35), the first derivative of the dual function in interval $[s_j, s_{j+1}]$ is equal to

$$\frac{dW(\sigma)}{d\sigma} = -B + \sum_{d\in L_j} \xi_d h_d + \sum_{d\in U_j} \xi_d H_d - \sum_{d\in F_j} \frac{w_d}{\sigma} \ . \tag{5.36}$$

Consider a point s_j, $(j \geq 2)$. Because of (5.30), for all demands d that change category from U_j to F_j at this point, the equalities

$$H_d = \frac{w_d}{s_j \xi_d} \tag{5.37}$$

hold, so the terms $\xi_d H_d$ and $\frac{w_d}{s_j}$ cancel out in (5.36). The same holds for F_j and L_j and the lower bound h_d at the point s_j. Hence the left and right derivatives of the dual fuction are equal at each point s_j, $j = 2, 3, \ldots, n-1$.

Note that the dual function (5.35) is not twice differentiable. According to (5.36), in interval $[s_j, s_{j+1}]$ its second derivative is equal to

$$\frac{d^2W(\sigma)}{d\sigma^2} = \sum_{d\in F_j} \frac{w_d}{\sigma^2} \ , \tag{5.38}$$

so in general it is discontinuous at the ends of the intervals (5.32) where sets F_j change. Note also that in the case when the set F_j is empty the dual function $W(\sigma)$ is linear in $[s_j, s_{j+1}]$ implying that $\frac{dW(\sigma)}{d\sigma}$ is constant and $\frac{d^2W(\sigma)}{d\sigma^2} = 0$, for $\sigma \in [s_j, s_{j+1}]$ (cf. (5.36) and (5.38)).

The dual function is concave and differentiable, so its maximum is attained at a stationary point σ^* resulting from the equation

$$\frac{dW(\sigma)}{d\sigma} = 0 \ , \quad \sigma \in [0, +\infty) \ . \tag{5.39}$$

Of course, the stationary point σ^* can belong only to one of the intervals (5.32). Let j be the index of interval $[s_j, s_{j+1}]$ such that $\sigma^* \in [s_j, s_{j+1}]$. This interval has the unique property that the stationary point(s) resulting from (5.39) belongs to $[s_j, s_{j+1}]$. The considered stationary point is given by

$$\sigma^*|_{F_j \neq \emptyset} = \frac{\sum_{d\in F_j} w_d}{B - \sum_{d\in L_j} \xi_d h_d - \sum_{d\in U_j} \xi_d H_d} \ , \text{ and}$$

$$\sigma^*|_{F_j = \emptyset} = \begin{cases} \text{any } \sigma \in [0, +\infty) \ , & B = \sum_{d\in L_j} \xi_d h_d + \sum_{d\in U_j} \xi_d H_d \\ \text{does not exist} \ , & B \neq \sum_{d\in L_j} \xi_d h_d + \sum_{d\in U_j} \xi_d H_d \end{cases} \ . \tag{5.40}$$

Note that the unique interval j with the property $\sigma^* \in [s_j, s_{j+1}]$ can be identified by the properties

$$\frac{dW(\sigma)}{d\sigma} \Big|_{\sigma=s_j} \geq 0 \ \text{ and } \ \frac{dW(\sigma)}{d\sigma} \Big|_{\sigma=s_{j+1}} \leq 0 \ , \tag{5.41}$$

and that it can happen that in this interval the dual function is constant. The optimal multiplier defines the optimal demand flows via formula (5.30).

Example 5.2. Again consider the instance given by Example 5.1. It can easily be verified that for the consecutive intervals $[0, \frac{1}{3}]$, $[\frac{1}{3}, 1]$, $[1, 2]$, $[2, +\infty)$, the respective stationary points given by (5.40) are: $\frac{3}{8}$, $\frac{2}{5}$, no stationary point, and $\frac{5}{4}$. Thus, as can be expected considering Figure 5.1, only the second interval has the property that the stationary point of the dual belongs to it: $\frac{2}{5} \in [\frac{1}{3}, 1]$. The corresponding optimal flows are $x_1 = 3$, $x_2 = 5$ and $x_3 = 5$. Note that the dual function is linear in interval $[1, 2]$ and therefore it has no stationary point there.

5.1.2 An objective function extension

A practical variation of the logarithmic objective function could be to extend it by a term that pennalizes the cost induced by installation of link capacities. The influence of such an additional term can then be adjusted by the reward coefficients w_d.

Unbounded flows

Introducing the pennalizing term for the case when flows are unbounded results in the following optimization problem,

$$f(\boldsymbol{x}) = \max \quad \sum_d w_d \log x_d - \sum_e \rho_e y_e \tag{5.42}$$

$$\text{s.t.} \quad \sum_e \rho_e y_e \le B_0 \tag{5.43}$$

$$\sum_d \delta_{ed} x_d \le y_e \qquad \forall\ e \tag{5.44}$$

$$\boldsymbol{x} \ge 0,\ \boldsymbol{y} \ge 0\ . \tag{5.45}$$

Due to (5.6) and (5.20), the maximal value of objective function (5.42) is attained at the maximum, with respect to variable B, of the one-variable function (constant b in (5.20) is skipped)

$$g(B) = \log(B) \sum_d w_d - B\ , \tag{5.46}$$

over $0 \le B \le B_0$. The optimum of (5.46) is attained either at $B = \sum_d w_d$ (if $\sum_d w_d \le B_0$) or at B_0 (if $\sum_d w_d > B_0$). Clearly, the optimal B is the optimal total cost of links $\sum_e \rho_e y_e$. Furthermore from (5.6) it follows that if $\sum_d w_d \le B_0$ the optimal value of (5.43) is equal to

$$f^* = \sum_d w_d \log\left(\tfrac{w_d}{\xi_d} \right) - \sum_d w_d\ , \tag{5.47}$$

and the optimal flows are given by

$$x_d^* = \frac{w_d}{\xi_d}\ ,\quad d = 1, 2, \ldots, D\ . \tag{5.48}$$

Note that an equivalent formulation of optimization problem (5.43)-(5.45) is

$$f(\boldsymbol{x}) = \max \quad \sum_d \left(\; w_d \log x_d - \xi_d x_d \; \right) \tag{5.49}$$

$$\text{s.t.} \quad \sum_d \xi_d x_d \leq B_0 \tag{5.50}$$

$$\boldsymbol{x} \geq 0 \; . \tag{5.51}$$

The objective function (5.49) is strictly convex and the unique solution, not taking the budget constraint (5.50) into account, is obviously given by (5.48). A problem arises, however, when the budget constraint is not satisfied at such a solution, i.e. when $\sum_d \xi_d x_d^* > B_0$. If this is the case then, in order to find the optimal solution for (5.49)-(5.51), we have to solve (5.2)-(5.5) for the fixed budget B_0.

Bounded flows

In the case of the bounded flows, constraints (5.21) are added to (5.49)-(5.51). The unique solution of the resulting optimization problem, again not taking the budget constraint (5.50) into account, is given by

$$x_d^0 = \begin{cases} h_d \; , & \frac{w_d}{\xi_d} < h_d \\ \frac{w_d}{\xi_d} \; , & h_d \leq \frac{w_d}{\xi_d} \leq H_d \\ H_d \; , & \frac{w_d}{\xi_d} \geq H_d \end{cases} \qquad d = 1, 2, \ldots, D \quad . \tag{5.52}$$

Again, a problem arises when the budget constraint is not satisfied at this solution. Then the optimal solution is found by solving (5.22)-(5.26) for the fixed budget B_0. Of course, we will have to assume that $\sum_d h_d \xi_d < B < \sum_d H_d \xi_d$, in order to make this optimization problem feasible and non-trivial.

5.1.3 Numerical examples

To show the character of the introduced solutions we will proceed by giving some larger numerical examples. We start, however, with an example showing the difference between the concepts of Proportional Fairness and Max-Min Fairness, in terms of throughput for a solution to a network dimensioning problem.

Comparison of PF and MMF in dimensioning

Consider optimization problem (5.2)-(5.5) for the special case when $w_d = 1$ for all demands d, $d = 1, 2, \ldots, D$, and $\rho_e = 1$ for all links e, $e = 1, 2, \ldots, E$,

and suppose that $B = D$. Application of formula (5.7) yields the proportionally fair flows,

$$x_d^* = \frac{1}{n_d} \ , \quad d = 1, 2, \ldots, D \ , \tag{5.53}$$

where n_d is the length of the shortest path of demand d. The form of (5.53) justifies the name "proportionally fair", in the sense that the more links that are needed for a demand to connect its node-pair, the less flow is allocated to that demand. Now consider the max-min fairness counterpart of optimzation problem (5.2)-(5.5), for the same special case. Trivially, the solution to the corresponding MMF dimensioning problem will be to assign an equal, maximal flow to all demands, resulting in

$$x_d^* = \frac{D}{\sum_d n_d} \ , \quad d = 1, 2, \ldots, D \ . \tag{5.54}$$

Summing up we get that the throughput associated with the PF solution is

$$\sum_d \frac{1}{n_d} \ ,$$

whereas the throughput of the MMF solution is

$$\frac{D^2}{\sum_d n_d} \ .$$

Using a well known inequality,

$$\left(\ \sum_d \tfrac{1}{n_d} \ \right) \left(\ \sum_d n_d \ \right) \geq D^2 \ ,$$

it is established that, with the given presumptions, the PF solution dominates the MMF solution counterpart in terms of throughput. Note that equality will hold if and only if the shortest paths for all the demands are of equal length.

A backbone network

Consider the 12-node (polish) backbone network depicted in Figure 5.2. There are $E = 18$ links and $D = 66$ demands. Consider the special case with all w_d:s and c_e:s equal to 1. For these coefficients, the objective function (5.42) is shown in Figure 5.3, revealing that the maximum is attained at $B = 66$. Optimal flows are equal to reciprocals of the lengths of the shortest paths measured as the number of links in a path. The dual function (5.17) for the optimal budget $B = 66$ is depicted in Figure 5.4. The optimal dual multiplier is $\sigma = 1$. Note that $f(66) = -W(1) - D$.

In the next example, described in Tables 5.1 and 5.2, we consider the backbone network with links' unit costs proportional to their length (column 4 in Tables 5.1 and 5.2) resulting in the shortest paths of the length

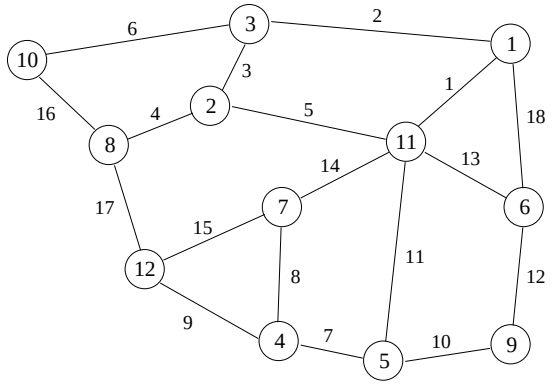


Figure 5.2: A backbone network.

given in column 7. Three experiments have been carried out for this case. The first experiment corresponds to optimization problem (5.42)-(5.45) — the unbounded case with the extended objective function and the budget constraint (5.43) skipped. The optimal flows are given in column 3 (the optimal budget is $B = 66$).

In the next two experiments bounds (5.21) have been added to (5.42)-(5.45). The bounds have been generated randomly and are given in columns 5 and 6 (in effect, $\sum_d h_d \xi_d = 245.64$). In the second experiment we have assumed $B_0 = 250$. In this case the solution of (5.22)-(5.26) applies (for $B = 250$). The optimal multiplier for budget $B = 250$ is $\sigma = 2.12$ (cf. Figure 5.5) and the maximum of the primal function (5.2) is $f(x) = -30.65$ (the optimal flows are given in column 8). Note that assuming $x_d = h_d$ for $d = 1, 2, \ldots, 66$, we would get $f(x) = -47.09$.

Finally, in the third experiment we have assumed the budget constraint large enough ($B_0 = 300$) and in the resulting solution of (5.42)-(5.45) extended by (5.21) (column 9) the optimal budget $\sum_d \xi_d x_d$ equals 259.49, and the optimal primal solution is $f(x) = -17.27$.

Studying the numerical results of Tables 5.1 and 5.2, some useful obsevations can be made:

- It can be verified that flows with small lower bounds are likely to exceed their lower bounds so that $x_d > h_d$. This is natural, looking at the objective function.

- In experiments 2 and 3, the number of flows x_d different from h_d and H_d in the optimal solution is approximately 30%. This relatively small percentage should be compared to the objective function gain, using optimal flows instead of the lower bounds.

- An obvious way to proceed if the solution of (5.42)-(5.45) with the

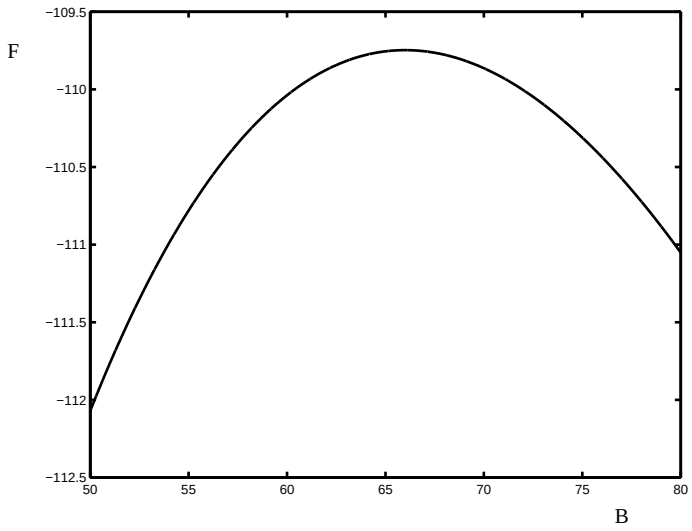


Figure 5.3: The primal function of (5.42)-(5.45).

added bounds constraint (5.21) is addressed, is to first assign all flows according to (5.52) and then compute $\sum_d \xi_d x_d$ and $\sum_d \xi_d h_d$. If the resulting budget $B_{opt} = \sum_d \xi_d x_d$ is acceptable, then the corresponding solution can be applied. Otherwise some $B_0 > \sum_d \xi_d h_d$ should be chosen and (5.22)-(5.26) applied. It can then be checked what is the actual loss with respect to the objective function choosing this B_0 instead of B_{opt}.

5.2 Resilient PF dimensioning

In the dimensioning process it is quite common to require, apart from the realization of the demands, robustness (resilience) to network failures. A network failure can occur if capacity of a link is lost (through e.g., a duct cut), either partially or completely. Another possible failure is that of a node, which however through special topology modifications can be considered as a link failure. It is a reasonable and common methodology to dimension the network such that i) the prescribed allocation requirements are met, and ii) such that the function of the network is maintained (possibly slightly deteriorated) under failure conditions. In this section we investigate a network dimensioning problem that intends to allocate the demands in a proportionally fair way, and at the same time forces resilience to certain predefined, potential link failures. In fact, the combination of fair bandwidth allocation and resilience is to the best of our knowledge novel in the context of dimensioning problems.

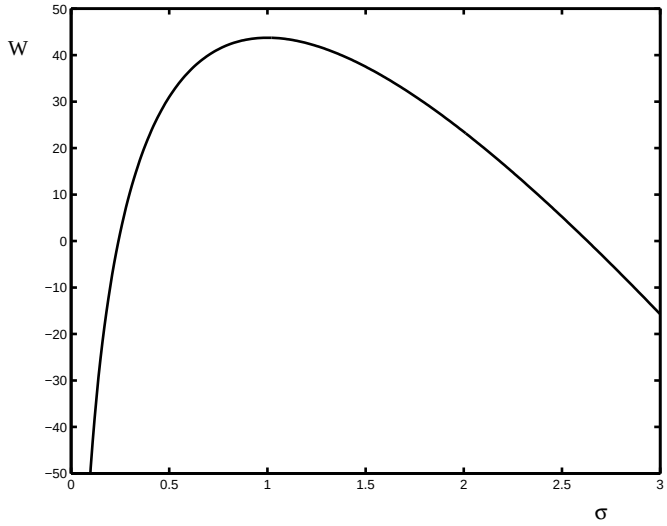


Figure 5.4: The dual function of (5.2)-(5.5) for the optimal budget $B = 66$.

The introduced problem formulation and its solution algorithms are based on the concepts presented in Section 2.4 and combine the ideas of the PF and the max-min fairness principles. To be strict, the considered problem is a non-linear max-min fairness problem, i.e., a max-min fairness problem for which the criteria are non-linear. To practically visualize the methodology, one can think of a scenario where for each failure situation, users and network operator agree to assign bandwidth to demands according to the PF principle, i.e., through maximizing the logarithmic revenue. This enables acceptable fairness as well as reasonable total network throughput. Since besides this, the interest is to keep the bandwidth distribution fair also over the failure situations, the total logarithmic revenue for individual situations is allocated in the max-min fair way. In this manner, none of the situations are favored. Technically put, the demand bandwidth allocation generated by the resulting algorithm is proportionally fair over the demand flows for the individual situations, and max-min fair for the logarithmic revenues over all situations. The algorithm is based on linear programming (using linear approximation of the logarithmic objective function implied by PF) applied in an iterative way.

5.2.1 Problem formulation

To formulate the problem we will need some notation that has not been previously used. Assume that a network is given in terms of topology. Also, consider as given the set of demands, $d = 1, 2, \ldots, D$, (node-pairs). For each demand, d, there is an associated set of paths, $p = 1, 2, \ldots, P_d$. As

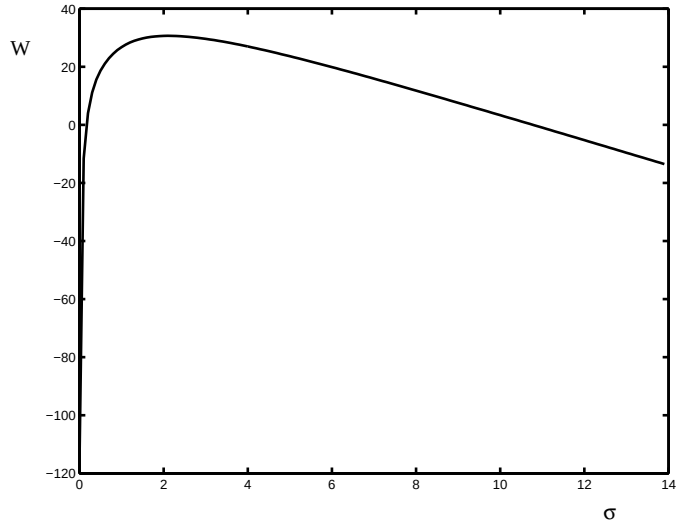


Figure 5.5: The dual function for the optimal budget in (5.22)-(5.26).

usual, let δ_{edp} denote the edge-path incidence relation. Further, a set of possible failure situations, indexed by $s = 1, 2, \ldots, S$, is given. Predefined are also the reward coefficients, w_{ds}, intended to be dependent on both demand and situation, and a set of availability coefficients, α_{es}, indicating the availability of link e in situation s, $0 \leq \alpha_{es} \leq 1$. As in Section 5.1, we assume that each link is given with an associated marginal cost, ρ_e, and that it is required that $\sum_e \rho_e y_e \leq B$, for some given budget B. We wish to determine z_{dps}, i.e., the flow allocated to path p of demand d in situation s, as well as the link capacities, y_e. These will thus be the problem decision variables. Let $x_{ds} = \sum_p z_{dps}$, and define the allocation array $\boldsymbol{x}$ as $\boldsymbol{x} = [x_{ds} : d = 1, 2, \ldots, D,\ s = 1, 2, \ldots, S]$. The revenue in situation s for allocation array $\boldsymbol{x}$ is given by $R_s(\boldsymbol{x}) = \sum_d w_{ds} \log(x_{ds})$, and the revenue vector, $\boldsymbol{R}$, by $\boldsymbol{R} = (R_1(\boldsymbol{x}), R_1(\boldsymbol{x}), \ldots, R_S(\boldsymbol{x}))$.

The proportional fairness resilient dimensioning problem is defined as the following optimization problem:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
node	node	x, no bounds	ρ, direct link	h	H	ξ	x, $B_0 = 250$	x, $B_0 = 300$
1	2	0.24	∞	0.56	10.0	4.15	h	h
1	3	0.29	3.40	1.43	2.06	3.40	h	h
1	4	0.22	∞	0.05	0.58	4.70	0.10	0.21
1	5	0.24	∞	1.03	100	4.15	h	h
1	6	0.59	1.70	0.96	1.83	1.70	h	h
1	7	0.32	∞	0.75	100	3.10	h	h
1	8	0.18	∞	1.43	100	5.60	h	h
1	9	0.32	∞	0.75	100	3.10	h	h
1	10	0.54	∞	1.62	100	6.30	h	h
1	11	0.50	1.85	0.40	100	1.85	h	0.54
1	12	0.19	∞	1.97	3.33	5.40	h	h
2	3	1.00	1.00	0.77	2.23	1.00	h	h
2	4	0.20	∞	1.86	3.75	5.00	h	h
2	5	0.22	∞	1.15	100	4.60	h	h
2	6	0.25	∞	0.98	100	3.95	h	h
2	7	0.28	∞	1.47	2.79	3.55	h	h
2	8	0.69	1.45	0.30	100	1.45	0.33	0.69
2	9	0.19	∞	0.67	10.0	5.35	h	h
2	10	0.33	∞	0.72	100	3.00	h	h
2	11	0.43	2.30	1.13	2.79	2.30	h	h
2	12	0.36	∞	0.05	0.32	2.80	0.17	H
3	4	0.17	∞	2.14	3.23	6.00	h	h
3	5	0.18	∞	0.97	10.0	5.60	h	h
3	6	0.20	∞	0.05	0.89	4.95	0.10	0.20
3	7	0.22	∞	1.74	2.65	4.55	h	h
3	8	0.41	∞	0.16	10.0	2.45	0.19	0.41
3	9	0.16	∞	2.23	3.65	6.35	h	h
3	10	0.35	2.90	0.69	100	2.90	h	h
3	11	0.30	∞	0.70	10.0	3.30	h	h
3	12	0.26	∞	0.94	100	3.80	h	h
4	5	1.00	1.00	0.05	0.05	1.00	H	H

Table 5.1: Numerical results.

$$\text{lex max} \quad \Theta(\boldsymbol{R}(\boldsymbol{x})) \tag{5.55}$$

$$\text{s.t.} \quad x_{ds} = \sum_{p} z_{dps} \qquad \forall\, d, s \tag{5.56}$$

$$\sum_{e} \rho_e y_e \leq B \tag{5.57}$$

$$\sum_{d} \sum_{p} \delta_{edp} \leq \alpha_{es} y_e \qquad \forall\, e, s \tag{5.58}$$

$$z_{dps} \geq 0,\ \boldsymbol{y} \geq 0 \qquad \forall\, d, p, s \tag{5.59}$$

5.2.2 Resolution methods

The resilient dimensioning problem formulated by (5.55)-(5.59) is non-trivial as it involves lexicographical maximization of a sorted vector of non-linear functions. Nevertheless, the formal defintion (Definition 2.19) of max-min

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
node	node	x, no bounds	ρ, direct link	h	H	ξ	x, $B_0 = 250$	x, $B_0 = 300$
4	6	0.26	∞	0.96	100	3.90	h	h
4	7	0.63	1.60	0.05	0.95	1.60	0.30	0.63
4	8	0.28	∞	0.44	10.0	3.55	h	h
4	9	0.40	∞	0.05	0.63	2.50	0.19	0.40
4	10	0.20	∞	1.89	2.02	5.10	h	h
4	11	0.35	∞	1.28	2.06	2.85	h	h
4	12	0.45	2.20	0.05	0.44	2.20	0.21	H
5	6	0.34	∞	1.29	2.61	2.90	h	h
5	7	0.38	∞	1.21	2.29	2.60	h	h
5	8	0.22	∞	1.14	100	4.55	h	h
5	9	0.67	1.50	0.31	100	1.50	0.31	0.67
5	10	0.16	∞	1.56	100	6.10	h	h
5	11	0.43	2.30	0.05	0.57	2.30	0.21	0.43
5	12	0.31	∞	0.06	10.0	3.20	0.15	0.31
6	7	0.34	∞	0.05	0.21	2.90	0.16	H
6	8	0.19	∞	1.97	2.57	5.40	h	h
6	9	0.71	1.40	0.05	0.53	1.40	0.34	H
6	10	0.14	∞	1.80	100	6.95	h	h
6	11	0.61	1.65	0.35	100	1.65	h	0.61
6	12	0.19	∞	1.32	100	5.20	h	h
7	8	0.27	∞	1.50	2.65	3.65	h	h
7	9	0.24	∞	1.62	2.28	4.10	h	h
7	10	0.19	∞	0.05	1.04	5.20	0.09	0.19
7	11	0.80	1.25	0.84	0.84	1.25	h	h
7	12	0.43	2.30	0.38	10.0	2.30	h	0.43
8	9	0.17	∞	1.55	100	6.05	h	h
8	10	0.65	1.55	0.92	2.28	1.55	h	h
8	11	0.27	∞	1.52	1.59	3.75	h	h
8	12	0.74	1.35	0.05	1.02	1.35	0.35	0.74
9	10	0.13	∞	0.05	0.40	7.60	0.06	0.13
9	11	0.33	∞	1.33	2.55	3.05	h	h
9	12	0.21	∞	0.05	0.60	4.70	0.10	0.21
10	11	0.19	∞	0.84	10.0	5.30	h	h
10	12	0.34	∞	0.25	10.0	2.90	h	0.34
11	12	0.28	∞	0.87	100	3.55	h	h

Table 5.2: Numerical results, continued.

fairness applies. Moreover is it possible to make use of the convex structure of the problem. Note that functions $R_s(\boldsymbol{x})$, $s = 1, 2, \ldots, S$ are concave, and that the feasible set is convex. We may thus use an application-specific variant of either Algorithm 2.1 or Algorithm 2.2. We will omit the exact adaptation of Algorithm 2.1 and concentrate on an application-specific version of the more efficient Algorithm 2.2, which bases its function on the use of dual multipliers.

Algorithm 5.1. (*for solving* (5.55)-(5.59))

Step 0: Assign $\mathcal{N} := \{1, 2, \ldots, S\}$. Solve

$$\begin{aligned}
\max \quad & t && && (5.60)\\
\text{s.t.} \quad & x_{ds} = \sum_p z_{dps} && \forall\ d, s && (5.61)\\
& \sum_e \rho_e y_e \leq B && && (5.62)\\
& \sum_d \sum_p \delta_{edp} \leq \alpha_{es} y_e && \forall\ e, s && (5.63)\\
& R_s(\boldsymbol{x}) = \sum_d w_{ds} \log(x_{ds}) \geq t && \forall\ s && (5.64)\\
& z_{dps} \geq 0,\ \boldsymbol{y} \geq 0 && \forall\ d, p, s && (5.65)
\end{aligned}$$

and let t^*, $\boldsymbol{y}^*$ and $\boldsymbol{x}^*$ be the optimal solution, and $\Lambda^* = [\lambda_s^*]_{s=1,2,\ldots,S}$ be the optimal dual multipliers corresponding to constraints (5.64).

Step 1: For all $s \in \mathcal{N} : \lambda_s^* > 0$ assign $\mathcal{N} := \mathcal{N} \setminus \{s\}$ and $t_s := t^*$. If $\mathcal{N} = \emptyset$ then stop, the components of $\boldsymbol{x}^*$ gives the max-min fair solution, and $\boldsymbol{y}^*$ holds the corresponding link cpacities. Otherwise proceed to Step 2.

Step 2: Solve

$$\begin{aligned}
\max \quad & t && && (5.66)\\
\text{s.t.} \quad & x_{ds} = \sum_p z_{dps} && \forall\ d, s && (5.67)\\
& \sum_e \rho_e y_e \leq B && && (5.68)\\
& \sum_d \sum_p \delta_{edp} \leq \alpha_{es} y_e && \forall\ e, s && (5.69)\\
& R_s(\boldsymbol{x}) = \sum_d w_{ds} \log(x_{ds}) \geq t_s && \forall\ s \notin \mathcal{N} && (5.70)\\
& R_s(\boldsymbol{x}) = \sum_d w_{ds} \log(x_{ds}) \geq t && \forall\ s \in \mathcal{N} && (5.71)\\
& z_{dps} \geq 0,\ \boldsymbol{y} \geq 0 && \forall\ d, p, s && (5.72)
\end{aligned}$$

and let t^*, $\boldsymbol{y}^*$ and $\boldsymbol{x}^*$ be the optimal solution, and $\Lambda^* = [\lambda_s^*]_{s \in \mathcal{N}}$ be the optimal dual multipliers corresponding to constraints (5.71). Go to Step 1.

Notably, although it would be rather difficult to find optimal multipliers for the optimization problems necessary to solve in Algorithm 5.1, a

linear approximation makes this practicable, since many linear programming solvers provide the optimal dual variables along with primal optimal solution. As mentioned before in the context of applying Algorithm 2.2, investigating dual multipliers is really a method of detecting the situations for which the corresponding revenue (R_s) is blocking (not possible to increase further), and not the ones that are non-blocking (possible to increase further), which would be even better. However, as is also explained in Section 2.4, assuming that all the situations for which the corresponding revenue is not shown to be blocking by the dual multiplier test are indeed non-blocking, will work even though we risk not to improve t^* in two consecutive iterations. This is because that $\sum_s \lambda_s^* = 1$, resulting in that each iteration will reveal at least one blocking situtation, so we will eventually reach an iteration where every situation s for which $\lambda_s^* = 0$ is non-blocking.

5.2.3 Linear approximation

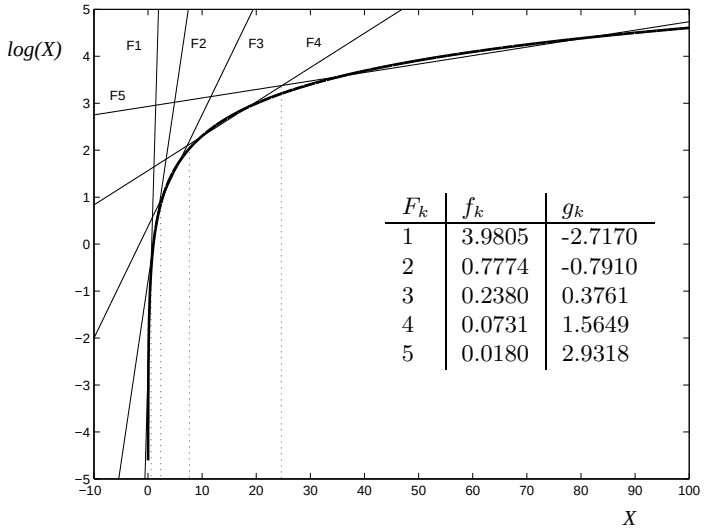
In the implementations of Algorithm 5.1 a piece-wise approximation of the logarithmic function is used, resulting in LP approximations of all the considered convex optimization problems. This is done by introducing additional variables, r_{ds}, corresponding to total flows, x_{ds}, which along with additional sets of constraints,

$$r_{ds} \leq f_k x_{ds} + g_k \quad \forall\, d, s, k, \tag{5.73}$$

replaces the logarithm of the flow, $\log(x_{ds})$. Here, $k = 1, 2, \ldots, K$, and K is the number of linear pieces of the approximation, and f_k and g_k are the coefficients of the consecutive linear pieces. Figure 5.6 illustrates an example of the approximation (actually the approximation that is used in the numerical examples in Section 5.2.5).

Using this approximation the convex optimization problem of Step 0 assumes the following LP form:

$$\begin{aligned}
\max \quad & t && && (5.74)\\
\text{s.t.} \quad & x_{ds} = \sum_p z_{dps} && \forall\, d, s && (5.75)\\
& \sum_e \rho_e y_e \leq B && && (5.76)\\
& \sum_d \sum_p \delta_{edp} \leq \alpha_{es} y_e && \forall\, e, s && (5.77)\\
& R_s(\boldsymbol{x}) = \sum_d w_{ds} r_{ds} \geq t && \forall\, s && (5.78)\\
& r_{ds} \leq f_k x_{ds} + g_k && \forall\, d, s, k && (5.79)\\
& z_{dps} \geq 0,\ \boldsymbol{y} \geq 0 && \forall\, d, p, s && (5.80)
\end{aligned}$$



F_k	f_k	g_k
1	3.9805	-2.7170
2	0.7774	-0.7910
3	0.2380	0.3761
4	0.0731	1.5649
5	0.0180	2.9318

Figure 5.6: Piece-wise approximation of the logarithmic function

5.2.4 Extensions of the basic problem

The proportional fairness robust dimensioning problem (5.55)-(5.59) can be extended in several ways. Such variations of the problem can be achieved by introducing additional constraints characterizing the solution set. First, we may assume that the capacities of links, y_e, are limited (e.g., because of the shortage of transmission capacity), adding constraints

$$y_e \leq c_e \quad \forall\ e\ , \tag{5.81}$$

where c_e are fixed upper bounds on link capacities. Secondly, we can introduce lower and upper bounds on the total demand flows realized in failure situations;

$$h_{ds} \leq x_{ds} \leq H_{ds} \quad \forall\ d, s\ , \tag{5.82}$$

where h_{ds} and H_{ds} are the given bounds on total flows. Finally, we can impose constraints on the flow reconfiguration in the case of failure. So far we have assumed that individual flows, z_{dps}, are only situation-dependent and can be freely reconfigured using surviving resources. This allows us to not explicitly distinguish the the normal state, i.e., the state with all α_{es} equal to 1), in the formulation of the problem. If we wish to prohibit flows that are not affected by a failure (i.e., the flows that survive a given failure situation s) to be disconnected and rerouted we proceed as follows. We label the "true" failure situations with $s = 1, 2, ..., S$, and use the index $s = 0$ for the normal state. Then we assume that if a link fails then it fails totally

(i.e., $\alpha_{es} \in \{0,1\}$), and introduce the binary path availability coefficients:

$$\theta_{dps} = \prod_{\{e:\delta_{edp}=1\}} \alpha_{es} \quad \forall\, d,p,s\ .$$

Now, the considered requirement for flow reallocation can be taken into account by introducing new constraints:

$$z_{dps} \geq \theta_{dps} z_{dp0} \quad \forall\, d,p,s$$

It should be noted that all the above described modifications allow for the application of the introduced algorithm without any special adjustments.

5.2.5 Numerical results

The introduced algorithm has been implemented in three variants, all on the same hardware, a Dell Precision 220 PC, with an Intel Pentium III-1GHz CPU, RAM of 256 MB, a Quantum Atlas10K2-TY092L SCSI HDD and the Windows 2000 Pro SP2 OS. The different software implementations are given in Table 5.3. A number of experiments were carried out to test the al-

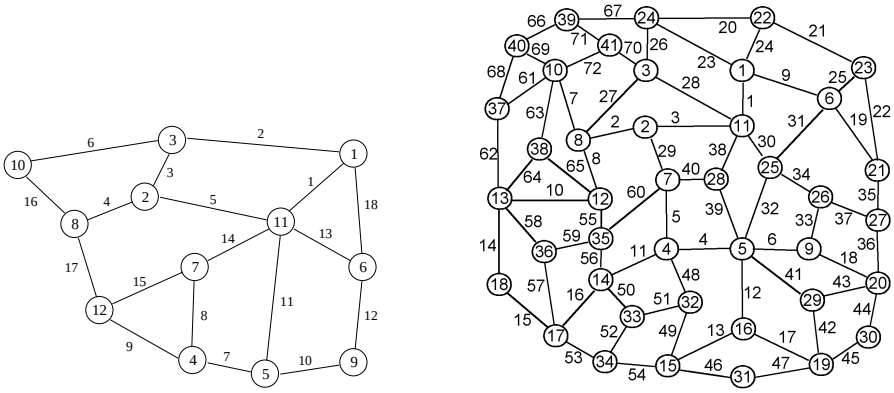
Reference code	Programming environment	Solver
C++	Microsoft Visual C++ 6.0; Callable CPLEX libraries	CPLEX 7.5.0
AMPL	AMPL ver. 20010215	CPLEX 7.5.0
Matlab	MATLAB 6.1	MATLAB Optimization toolbox

Table 5.3: Software implementations of the algorithms.

gorithm for different network examples. In fact, in order to have a reference, a problem-specific version of Algorithm 2.1 has been implemeted in paralell. The following subsections contain selected input data and the results of the algorithms for three different-sized networks. The network instances used are presented in Table 5.4. Obviously, the networks of particular interest are the "Polish backbone network", N_{12}, and the artificial network (randomly generated), N_{41}, shown in figures 5.7(a) and 5.7(b), respectively. To show the dynamics of the considered algorithms we also study a small, triangular (toy) network, N_3, for which we will give complete numerical results. A summary of selected numerical values obtained by the algorithms applied to N_{12} will also be given. For N_{41} only execution time is reported. For the problem instances associated with N_3 and N_{12} the budget B is equal to 1000, while for N_{41} it is $1E6$.

ref. code	# nodes	# links	# demands	# paths per demand	# failure situations
N_3	3	3	3	2	4
N_{12}	12	18	66	6-13	19
N_{41}	41	72	100	3	35

Table 5.4: The networks used for experiments.



(a) The N_{12} network (b) The N_{41} network.

Figure 5.7: Two larger example instances.

The simple network, N_3

We start by presenting results achieved by the algorithms applied to the simple 3-node network, N_3. In this experiment the revenue coefficients w_{ds}, and cost coefficients ρ_e are all put equal to 1. Tables 5.5 and 5.6 give the input data. Results from these trials are found in Table 5.7 (empty entries

Demand		$d=1$	$d=2$	$d=3$
Nodes		1-2	1-3	2-3
Links (e)	$p=1$	1	2	3
	$p=2$	2,3	1,3	1,2

Table 5.5: Demands and paths for network N_3

indicate that an entity remains unchanged with respect to the preceding step). Values of flows, z_{dps}, are given only if not equal to zero. Final results (after the algorithm completion) are shown in the rightmost position. Note that $R_{s'}$ (in contrast to R_s) is relevant only for the basic algorithm (Algorithm 2.1), and that λ_s on the other hand is relevant only for Algorithm 5.1.

Links	α_{es}				ρ_e
	$s=1$	$s=2$	$s=3$	$s=4$	
$e=1$	1	0	1	1	1
$e=2$	1	1	0	1	1
$e=3$	1	1	1	0	1
w_{ds}	1	1	1	1	

Table 5.6: α_{es}, w_{ds} and c_e for network N_3

n	0	1		2
Instant	After Step 0	After Step 1	After Step 2	After Step 1
t^*	20.35		26.8	
t_s	$t_s = 20.35$, $s = 0,1,2,3$		$t_0 = 26.8$, $t_s = 20.35$, $s = 1,2,3$	
R_s	$R_s = 20.35$, $s = 0,1,2,3$	$R_{0'} = 26.80$, $R_{s'} = 20.35$, $s' = 1,2,3$	$R_0 = 26.8$, $R_s = 20.35$, $s = 1,2,3$	$R_{0'} = 26.8$
y	$y_1 = y_2 = y_3 = = 333.33$		$y_1 = y_2 = y_3 = = 333.33$	
z_{dps}	$z_{110} = 24.81$ $z_{210} = 24.81$ $z_{310} = 283.72$ $z_{320} = 308.53$ $z_{121} = 24.81$ $z_{211} = 308.53$ $z_{311} = 308.53$ $z_{112} = 308.53$ $z_{222} = 24.81$ $z_{312} = 308.53$ $z_{113} = 308.53$ $z_{213} = 308.53$ $z_{323} = 24.81$		$z_{110} = 333.33$ $z_{210} = 333.33$ $z_{310} = 333.33$ $z_{121} = 24.81$ $z_{211} = 308.53$ $z_{311} = 308.53$ $z_{112} = 308.53$ $z_{222} = 24.81$ $z_{312} = 308.53$ $z_{113} = 308.53$ $z_{213} = 308.53$ $z_{323} = 24.81$	
Z_1	$s = 0,1,2,3$	$s = 0$		$\emptyset$
Z_0	$\emptyset$	$s = 1,2,3$		$s = 0,1,2,3$
λ_s	For constraint (5.64): $\lambda_0 = 0$, $\lambda_1 = \lambda_2 = \lambda_3 = 0.33$		For constraint (5.71): $\lambda_0 = 1$	

Table 5.7: Numerical results applying Algorithms 2.1 and 5.1 to instance N_3

The Backbone Network, N_{12}

Already network N_{12} (medium-size Polish backbone network) is quite challenging for the algorithms in terms of computation times. Since for instance the number of the non-zero flows, z_{dps}, is quite big in this trial, only selected numerical results are given. As in the N_3 case, the revenue coefficients, w_{ds},

are all put equal to 1. Link costs are given in Table 5.8. Situations are defined in accordance to the N_3 case, i.e., each situation corresponds to one exclusively, completely failed link, except for the normal situation, when no links are broken. In effect, α_{es} are simply obtained by expanding Table 5.6. The evolution of R_s over (selected) iterations n, along with the evolution of the auxiliary variable t and the resulting link capacities y_e, is illustrated in Table 5.9. Table 5.9 reveals at which iteration (n) each indi-

e	1	2	3	4	5	6	7	8	9
ρ_e	1.85	3.4	1	1.45	2.3	2.9	1	1.6	2.2
e	10	11	12	13	14	15	16	17	18
ρ_e	1.5	2.3	1.4	1.65	1.25	2.3	1.55	1.35	1.7

Table 5.8: Link marginal costs for network N_{12}.

vidual situation was blocked. By studying the second column (t^*) and the 8th column ($R_s, n = 15$) it is concluded that situations 5,6,15 and 18 are blocked at $n = 0$, $t^* = 91.62$. Situation 8 is blocked for $n = 1$, $t^* = 92.15$ and 17 and 3 for $n = 2, 3$, $t^* = 92.32$, $t^* = 92.33$, respectively, and so on. It is of interest to illustrate the computational efficiency of the algorithms

n	t^*	R_s						e	y_e
		s	n						
			0	1	2	14	15		
0	91.62	0	91.62	92.15	92.32	97.83	101.64	1	25.89
1	92.15	1	91.62	92.15	92.32	96.22	96.22	2	33.32
2	92.32	2	91.62	92.18	92.32	92.33	92.33	3	42.54
3	92.33	3	91.62	92.15	92.32	93.96	93.96	4	49.76
4	93.11	4	91.62	91.62	91.62	91.62	91.62	5	40.46
5	93.56	5	91.62	91.62	91.62	91.62	91.62	6	13.70
6	93.96	6	91.62	92.15	92.32	97.37	97.37	7	54.97
7	94.06	7	91.62	92.15	92.15	92.15	92.15	8	27.97
8	94.27	8	91.62	92.15	92.32	95.93	95.93	9	22.43
9	95.93	9	91.62	92.15	92.32	94.27	94.27	10	31.64
10	96.22	10	91.62	92.15	92.32	93.56	93.56	11	16.55
11	96.26	11	91.62	92.15	92.32	97.83	97.83	12	38.13
12	96.68	12	91.62	92.15	92.32	93.11	93.11	13	22.43
13	97.37	13	91.62	92.15	92.32	96.68	96.68	14	55.60
14	97.83	14	91.62	91.62	91.62	91.62	91.62	15	22.24
15	101.64	15	91.62	92.15	92.32	96.26	96.26	16	25.24
		16	91.62	92.15	92.32	92.32	92.32	17	40.45
		17	91.62	91.62	91.62	91.62	91.62	18	29.56
		18	91.62	92.15	92.32	94.06	94.06		

Table 5.9: Selected results and parameters for network N_{12}

and their implementations in terms of execution time. The corresponding data is found in Table 5.10. Tests for N_{41} have been performed using only AMPL implementations. Therefore, some entries for N_{41} in Table 5.10 are

Reference code	Execution times (seconds)				
	Implementations of Algorithm 2.1			Implementations of Algorithm 5.1	
	C++	AMPL	Matlab	C++	AMPL
N_3	< 1	2	4	< 1	< 1
N_{12}	3859	390	6.4e4	179	325
N_{41}		24945			6960

Table 5.10: Execution times

left blank. The choice to use the AMPL implementations for these tests is based on the observation that these implementations exhibited the smallest difference between execution times of the basic and modified algorithms for the other networks considered. Therefore, since the computation times for large networks such as N_{41} are quite long for all the considered implementations, we have chosen to use the AMPL implementations to illustrate the differences in time performance of the two considered algorithms.

Remarks

As can be seen in Table 5.10, for the network examples considered, Algorithm 5.1 performs, as expected, much better than Algorithm 2.1. One can also notice that the performance advantage of Algorithm 5.1 increases with the size of the considered instance.

The revenues, R_s, for different situations after the final iteration (algorithm completion) may differ only by a small number, although making actual flow differences significant, since such a revenue difference is logarithmically related to the aggregated flow difference.

5.3 Conclusions

In this chapter we have dealt with so-called dimensioning problems. A dimensioning problem is, as oppposed to an allocation problem, concerned with determination of link capacities. The two problems we have considered both have convex structure, as they are aimed to distribute flows between demands in the proportionally fair way. The objective of proportional fairness was introduced in Chapter 2 and has been shown to define a fair rational preference relation on $\mathbb{R}^m_{++}$.

First we studied the problem of achieving a proportionally fair distribution of flows between demands, using both demand flows and link capacities as variables. The problem is bounded from above by a predefined budget, not to be exceeded by the aggregated cost of link capacity installation. We derived an analytical solution for this problem, implying that optimal link capacities and demand flows can be explicitly computed. As the optimal

solution always will use the shortest paths (with respect to links' predefined marginal cost), this solution is valid both if each demand has one fixed path and if each demand may split its flow over several paths. We then introduced the assumption that each demand is given with associated lower and upper flow bounds, and showed how the optimal flows can be computed in this case. In the balance of this section we discuss how the logarithmic objective function can be extended by a term that pennalizes on the link installation cost, and how this affects the optimal solution. In order to illustrate that the optimal solution is worthwhile finding, a large numerical example closes this section.

We then concentrated on a dimensioning problem that, besides proportionally fair distribution of flows, has the objective of obtaining a network design that is resilient to link failures. For this purpose we introduced a number of potential failure situations. The goal was to keep the network operability for each situation, measured in sense of the value of the proportional fairness objective function for that situation, max-min fair. For this purpose we formulated a lexicographical maximization problem, and showed how it could be solved by means of Algorithm 2.2, provided that the logarithmic function is linearly approximated. The application-specific version of Algorithm 2.2 was implemented in various settings, and evaluated on 2 different real-sized networks. The experiments exhibited acceptable running times, relating them to the corresponding instance sizes.

Chapter 6

Summary and contribution

In this chapter we summarize the main results of this thesis. In addition, we point out where and why the studies presented in this thesis contribute to the general network optimization research knowledge. In case a presented result has appeared in a published paper for which the author of this thesis was co-author, we consider the result as a *partial contribution* of this thesis. On the other hand, if a result did appear in a published paper for which the author of this thesis was the first auhtor, we simply refer to it as *contribution*. Finally, if a result has not been previously published it is denoted an *original contribution*. In case a presented result is not due to the author this will be clearly indicated.

6.1 Optimization models for fairness (Chapter 2)

In Chapter 2 we give a formal introduction to the concept of fairness. Particularly, the concept of one vector being more fair than another is rigorously defined through the notion of preference relations. For a preference relation to define fairness, we show that it has to fulfill certain regularity requirements, and is in this case called a fair rational preference relation. From the associated rules we develop a number of important basic types of fairness. The regularity requirements, as well as the idea of formalizing fairness through preference relations are concepts borrowed from [39], which is also true for the concept of fair domination (Definition 2.6). Property 2.1, giving a possibility of evaluating fairness of a vector as one value, is well known [23]. Although well known, the provided proof of Property 2.1 is an original contribution.

The objective of obtaining fairness is shown to be a matter of finding a vector that is (weakly) preferred to all other vectors, with respect to the considered fair rational preference relation. Such a vector is called fairly nondominated, and is often obtainable through some kind of (possibly multicriteria) maximization. This provides a natural connection to Pareto-efficiency, as it can be shown that all fairly non-dominated vectors can be seen as Pareto-efficient solutions to a special type of multi-criteria maximization problem. The connection between fairness and Pareto-efficiency was established in [19], and accordingly Theorem 2.3 is due to [19]. Nevertheless, the provided proof is an original contribution.

There exist several possible generic methods, for defining a fair rational preference relation. The first, and probably the most natural one, is to use the idea of aggregation functions. Other types of generic fairness, i.e., classes of fairness where parameter-settings define a particular fairness, have been presented. For example we have shown how to define fairness through p-norms and ordered weighted averaging. It is well known that p-norms define a class of fairness, as is stated by Property 2.5. The provided proof of this is due to the author. The α-fairness, first introduced in [28], is put

in the formal fair preference relation context in Section 2.2.3, which has not been done before and is therefore considered an original contribution. Specifically, Property 2.6 and Proposition 2.7 are (along with their proofs) due to the author. The discussion on ordered weighted averaging is on the other hand entirely based on material from [39].

The most thoroughly investigated type of fairness is the one that is often reasonably argued to be the most intutively fair, namely max-min fairness. A formal definition of max-min fairness is given, using the fair rational preference relation of the symmetrical leximin order. We then relate this definition to a stronger but less general definition, and show that the definitions are equivalent in the case of a convex outcome space. The distinction between strong and weak max-min fairness is an original contribution of this thesis. The properties relating these two notions (i.e., Properties 2.8, 2.9, and 2.10) are known [46], still, to the best of our knowledge, have not been published before, at least not accompanied with their proofs. Hence, their collection, as well as their arrangement in a proper context, is considered an original contribution.

Further on the topic of max-min fainess, we did also highlight that a solution that induces a max-min fair outcome vector is sometimes called max-min fair itself. To the extreme it may even happen that the solution variables are the outcomes, a case often referred to as linear max-min fairness.

Finally, Chapter 2 describes generic algorithms that achieve max-min fairness. For the convex case we gave an algorithm that is based on the use of dual multipliers. This algorithm (and the more basic algorithm constituting its origin) is considered partially contributional for this thesis, as it has been published by the author in the generic shape in [29], and in different applied forms in [45] and [44]. For the non-convex case an algorithm that makes use of cumulated ordered values, which is a concept borrowed from [40], has been presented.

6.2 Convex allocation problems (Chapter 3)

Chapter 3 is devoted to a study of allocation problems where the objective is (linear) max-min fairness. All of the considered allocation problems have so-called convex structure, meaning that the feasible set of solutions is convex and that the criteria are concave (actually in Chapter 3 all criteria are linear).

The most elementary problem addresses max-min fairness of flow distribution between demands that have one fixed path each. We show that in this case max-min fairness of flows is equivalent to that the allocation vector is max-saturating, and gave an algorithm which is fundamentally dependent on this equivalence. Constituting the basis for the well known algorithm,

the equivalence is a well understood result. However, the formal proof of this was published by the author in [44], which makes it a partial contribution of this thesis. The fact that the algorithm could be extended in order to take prerequired upper- and lower flow-bounds of demands into account, was mentioned in [44] but the complete extension is given in this thesis, as an original contribution.

A more difficult problem is encountered if each demand may use several paths and is allowed to split its flow over them. In this case we say that paths are optimized. It is then not possible to use the max-saturating property, and more complicated LP-based algorithms have to be devised. It is illustrated how the generic (convex-case) algorithms from Chapter 2 could be applied, and we give examples showing their operation. In addition, we derive some properties of the resulting problem solution, particularly concerned with the relation between different demand flows. The most important (and complicated) relation is given by Property 3.4 and is due to [31]. The provided proof is however due to the author, greatly inspired by proof ideas stemming from the first author of [31]. Section 3.1 is concluded by giving an example that shows what can be gained in terms of fairness if paths are optimized (i.e., flows may be split) and not kept fixed.

Besides the exceptions mentioned above, the contents of Section 3.1 are considered partial contributions of this thesis, as they have been published by the author in [44].

In the second section of Chapter 3, we introduce a reallocation scheme with the purpose of facilitating network resilience. All the material presented in this section has been published by the author in [35], and is thus a contribution of this thesis. The reallocation scheme is based on max-min fairness. Also in this problem we assume that all flows may be split over several paths. We assume that we are given a network that is preallocated, i.e., flows and paths are given a priori, and that the task is to reallocate and reroute these flows such that we facilitate a special type of resilience to network failures. The type of resilience considered is full protection of every possible single link failure. The goal is to reallocate the (preallocated) flows such that they as much as possible resemble the preallocation, but also such that we make room (reserve capacity) for protection paths for a failing link. We gave a special measure for the resemblence and an accompanying lower bound. The lower bound may be useful in approximate calculations as well as in exposing weakly protected links. Finally it is shown and examplified how Algorithm 2.2 solves the stated problem.

6.3 Non-convex allocation problems (Chapter 4)

In Chapter 4 we study (mixed-integer) max-min fairness allocation problems, with non-convex problem structure. The non-convexity of the studied

problems results from the discrete solution space, and complicates both formulation and resolution. In Chapter 3 we did not encounter this difficulty as all decision variables were allowed to take on continuous values, and we could in fact exploit the convex problem structure to formulate resolution algorithms.

The first non-convex MMF problem studied is in a way similar to the most basic one described in Chapter 3 – when each demand has exactly one predefined fixed path and flows have to be max-min fairly distributed. The crucial difference lies in that we now only allow flows to assume multiples of a predefined module. Section 4.1 is devoted to this problem. As the results of this section have previously been published by the author in [38] (Algorithm 4.1 was published in [32]), the entire section is considered a contribution for the thesis. It should however be noted that the idea of using lexicographical minimization to approach problems concerning (discrete-valued) lexicographical maximization is borrowed from [40].

Considering the mentioned problem it was first of all shown that, changing units, the requirement of modular flows can be viewed as a requirement of flows being integral. We compare the solution of the integral-flow problem with the basic problem, and derive a number of properties that explain their relation. Moreover, an analogy to the max-saturating property is given. In order to motivate suspicion of non-existence of polynomial time algorithms we then prove that obtaining MMF of modular flows is $\mathcal{NP}$-hard (this was suggested already in [48], although without a proof). Nevertheless, using the classical approach of waterfilling (Algorithm 3.1), some subproblems can be shown to be of polynomial time. For resolution of the considered problem, we propose an algorithm that solves an application-specific linear relaxation of a lexicographical minimization problem ((2.72)-(2.76)). Strictly speaking, as it is a relaxation, this constitutes an approximation algorithm. However, this relaxation is shown to reach an optimal solution in practically all of the randomly generated problem instances, and this at acceptable running times, even for relatively large instances.

We then take on the problem of obtaining a max-min fair distribution of flows if each demand may use exactly one path that has to be selected in the problem solution process. This problem is often referred to as max-min fairness of unsplittable flows, being the main theme of Section 4.2. All the findings presented in this section have appeared in publications due to the author, namely [37], [36], and [33], and are thus contributions of the thesis. However, which is also mentioned in the text, the ideas constituting the fundaments of Algorithms 4.4 and 4.5, are due to [40] and [41].

General unsplittable-flow problems are known to be difficult, so it is reasonable to expect provable difficulty also from this problem. We separate the problem into two cases – the case when each demand may use any path, and the case when each demand may use one path from a predefined list. With the objective of obtaining max-min fairness, both cases are shown to

be $\mathcal{NP}$-hard. In fact we prove that just obtaining the first entry of the sorted allocation vector is $\mathcal{NP}$-hard in itself, in both cases.

With a deeper understanding of the complexity of the unsplitabble flow MMF problem, the focus is then put on resolution techniques. We present two exact, MIP-based algorithms, where one was derived specifically for this application, and the other is a generic MMF problem resolution technique. In addition we give an approximation algorithm that was based on the assumption that the outcome set is discrete and finite. The numerical study, which is carried out on network instances ranging from 3 nodes and 3 demands to 20 nodes and 190 demands, shows that, in general, the generic algorithm is usually faster than the application-specific one, if the problem should be solved to optimality. However, both approaches exhibit vast time-consumption when problem instances are scaled up. Finally, we apply the approximate method to our randomly generated problem instances. Even though being based on successive resolution of MIPs, this approach obtained suboptimal solutions of quite high quality at acceptable running times in cases where the exact methods do not solve the problem in less than half an hour.

6.4 Dimensioning problems (Chapter 5)

In Chapter 5 we deal with so-called dimensioning problems. A dimensioning problem is, as oppposed to an allocation problem, concerned with determination of link capacities. The two problems we consider both have convex structure, and are aimed to distribute flows between demands in a proportionally fair way. The objective of proportional fairness was introduced in Chapter 2 and has been shown to define a fair rational preference relation in $\mathbb{R}^m_{++}$.

All the findings presented in this chapter have been previously published by the author – Section 5.1 is based on [34], and Section 5.2 on [45]. Section 5.1 and Section 5.2 are therefore considered contribution and partial contribution, respectively.

First we study the problem of achieving a proportionally fair distribution of flows between demands, using both demand flows and link capacities as variables. The problem is bounded from above by a predefined budget, not to be exceeded by the aggregated cost of link capacity. We derive an analytical solution for this problem, implying that optimal link capacities and demand flows can be explicitly computed. As the optimal solution will always use the shortest paths (with respect to links' predefined marginal cost), this solution is valid both if each demand has one fixed path and if each demand may split its flow over several paths. We then introduce the assumption that each demand is given with associated lower and upper flow bounds, and show how the optimal flows can be computed in this case. In

the balance of this section we discuss how the logarithmic objective function can be extended by a term that pennalizes on the link capacity cost, and how this affects the optimal solution. A large numerical example closes this section.

We then in Section 5.2 concentrate on a dimensioning problem that, besides proportionally fair distribution of flows, has the objective of obtaining a network design that is resilient to link failures. For this purpose we introduce a number of potential failure situations. The goal is to keep the network operability for each situation, measured in sense of the value of the proportional fairness objective function for that situation, max-min fair. For this purpose we formulate a lexicographical maximization problem, and show how it could be solved by means of Algorithm 2.2, provided that the logarithmic function is piece-wise linearly approximated. The application-specific version of Algorithm 2.2 is implemented in various settings, and evaluated on two different real-sized networks. The experiments exhibit acceptable running times, relating them to the corresponding instance sizes.

Bibliography

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice Hall, 1993.

[2] G.R. Ash, F. Chang, and D. Medhi. Robust traffic design for dynamic routing networks. In *INFOCOM*, pages 508–514, 1991.

[3] A. Banerjee, J. Drake, J. P. Lang, B. Turner, K. Kompella, and Y. Rekhter. Generalized multiprotocol label switching: An overview of routing and management enhancements. *IEEE Communications Magazine*, pages 144–150, January 2001.

[4] G. Bannock, R.E. Baxter, and E. Davies. *The Penguin Dictionary of Economics*. Penguin Books Ltd, 1998.

[5] M. Baotic. Documentation of cplexint. `http://control.ee.ethz.ch/hybrid/cplexint.php`.

[6] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.

[7] D. Cavendish. Evolution of optical transport technologies: From sonet/sdh to wdm. *IEEE Communications Magazine*, pages 164–172, June 2000.

[8] D.E. Comer. *Internetworking with TCP/IP, Principles, Protocols, and Architectures*. Prentice Hall, 2000.

[9] M. Dzida, M. Pioro, and M. Zagozdzon. The application of max-min fairness rule to bandwidth allocation in telecommunication networks. In *3:rd Polish-German Teletraffic Symposium (PGTS)*, 2004.

[10] G. Fodor, G. Malicsko, M. Pioro, and T. Szymanski. Path optimisation for elastic traffic under fairness constraints. In *ITC 17*, 2001.

[11] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton Universuty Press, 1962.

[12] M. Garey and D. Johnson. *Computers and intractability – a guide to the theory of NP-completeness*. Freeman, 1979.

[13] W. Grover. *Mesh-Based Survivable Networks*. Prentice Hall, 2004.

[14] K. Kar, S. Sarkar, and L. Tassiulas. Optimisation based rate control for multipath sessions. In *ITC 17*, 2001.

[15] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:206–217, 1998.

[16] J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, MIT, 1996.

[17] J. Kleinberg. Single-source unsplittable flow. In *IEEE Symposium on Foundations of Computer Science*, pages 68–77, 1996.

[18] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. *Journal of Computer and System Sciences*, 63(1):2–20, 2001.

[19] M.M. Kostreva and W. Ogryczak. Linear optimization with multiple equitable criteria. *RAIRO Operations Research*, 33:275–297, 1999.

[20] L.S. Lasdon. *Optimization Theory for Large Systems*. MacMillan, 1970.

[21] C. Y. Lee, Y. P. Moon, and Y. J. Cho. A lexocographicall fair allocation of discrete bandwidth for multirate multicast traffics. *Computer & Operations Research*, 31:2349–2363, 2004.

[22] A. Lindstrom. Sonet celebrates its 15th anniversary. *America's Network*, 103:32–35, 1999.

[23] A.W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, 1979.

[24] L. Massoulié and J. Roberts. Bandwidth sharing: Objectives and algorithms. In *INFOCOM*, pages 1395–1403, 1999.

[25] N. Megiddo. Optimal flows in networks with sources and sinks. *Mathematical Programming*, 7, 1974.

[26] M. Middendorf and F. Pfeiffer. On the complexity of the disjoint path problem. *Combinatorica*, 13:97–107, 1993.

[27] M. Minoux. *Mathematical Programming: Theory and Algorithms*. John Wiley & Sons, 1986.

[28] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000.

[29] M.Pioro, M. Dzida, E. Kubilinskas, P. Nilsson, W. Ogryczak, A. Tomaszewski, and Michal Zagozdzon. Applications of the max-min fairness principle in telecommunication network design. In *First Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005)*, 2005.

[30] D. Nace. A linear programming based approach for computing optimal fair splittable routing. In *IEEE International Symposium on Computers and Communications*, pages 468–474, 2002.

[31] D. Nace and L.N. Doan. A polynomial approach to the max-min fair multi-commodity flow problem. In *5th International Conference on Decision Support for Telecommunications and Information Society*, 2005.

[32] P. Nilsson. Some simple special cases of FIXMMF-MF. Technical report, Dept. of Communication Sytems, Lund University, November 2005. CODEN: LUTEDX(TETS-7214)/1-3/(2005)&local 33.

[33] P. Nilsson. A transformation from INDEPENDENT SET to MMSPF2. Technical report, Dept. of Communication Sytems, Lund University, June 2005. CODEN: LUTEDX(TETS-7210)/1-2/(2005)&local 18.

[34] P. Nilsson and M. Pioro. Solving dimensioning tasks for proportionally fair networks carrying elastic traffic. *Performance Evaluation*, 49(1-4):371–386, 2002.

[35] P. Nilsson and M. Pioro. Link protection within an existing backbone network. In *International Network Optimization Conference, INOC'03*, pages 435–440, 2003.

[36] P. Nilsson and M. Pioro. Max-min fairness of unsplittable network flows. Technical report, Dept. of Communication Sytems, Lund University, June 2005. CODEN: LUTEDX(TETS-7209)/1-21/(2005)&local 25.

[37] P. Nilsson and M. Pioro. Unsplittable max-min demand allocation - a routing problem. In *HETNETs'05*, 2005. P26.

[38] P. Nilsson and M. Pioro. Max-min fair distribution of modular network flows on fixed paths. In *Networking*, pages 916–927, 2006.

[39] W. Ogryczak and A.Wierzbicki. On multi-criteria approaches to bandwidth allocation. *Control and Cybernetics*, 33:427–448, 2004.

[40] W. Ogryczak, M. Pioro, and A. Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of telecommunications and information technology*, 3, 2005.

[41] W. Ogryczak and T. Sliwinski. On direct methods for lexicographic min-max optimization. Technical report, Institute of Control and Computation Engineering, Warsaw University of technology, 2005.

[42] W. Ogryczak and A. Tamir. Minimizing the sum of the k largest functions in linear time. *Inf. Process. Lett.*, 85(3):117–122, 2003.

[43] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann (Elsevier), 2004.

[44] M. Pióro, P. Nilsson, E. Kubilinskas, and G. Fodor. On efficient max-min fair routing algorithms. In *IEEE International Symposium on Computers and Communications*, pages 465–472, 2003.

[45] Michal Pioro, Eligijus Kubilinskas, Pål Nilsson, and Marcin Matuszewski. Robust dimensioning of proportionally fair networks. *European Transactions on Telecommunications*, 13(1):241–251, 2005.

[46] B. Radunovic and J. Le Boudec. A unified framework for max-min and min-max fairness with applications, 2002.

[47] S. Sarkar and K. N. Sivarajan. Fairness in cellular mobile networks. In *34th Annual Allerton Conference on Communication, Control and Computing*, pages 457–469, 1996.

[48] S. Sarkar and L. Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In *INFOCOM*, pages 1491–1500, 2000.

[49] W. Stallings. *Data and Computer Communications*. Prentice Hall, 2004.

[50] A. Tomaszewski. A polynomial time algorithm for solving a general max-min fairness problem. In *2nd Polish-German Teletraffic Symposium*, 2002.

[51] W. Trockel. On the meaning of the nash product. Technical report, Bielefeld University, 2003. Paper no. 354.

[52] J. Vygen. NP-completeness of some edge-disjoint paths problems. *Discrete Applied Mathematics*, 61:83–90, 1995.

[53] T-H. Wu. *Fiber Network Service Survivability*. Artech House, 1992.

[54] R.R. Yager. On the analytic representation of the leximin ordering and its application to flexible constraint propagation. *European J. Oper. Res.*, 102:176–192, 1997.

Reports on Communication Systems

101. **On Overload Control of SPC-systems**
Ulf Körner, Bengt Wallström and Christian Nyberg, 1989.
102. **Two Short Papers on Overload Control of Switching Nodes**
Christian Nyberg, Ulf Körner and Bengt Wallström, 1990.
103. **Priorities in Circuit Switched Networks**
Åke Arvidsson, Ph.D. thesis, 1990.
104. **Estimations of Software Fault Content for Telecommunication Systems**
Bo Lennselius, Lic. thesis, 1990.
105. **Reusability of Software in Telecommunication Systems**
Anders Sixtensson, Lic. thesis, 1990.
106. **Software Reliability and Performance Modelling for Telecommunication Systems**
Claes Wohlin, Ph.D. thesis, 1991.
107. **Service Protection and Overflow in Circuit Switched Networks**
Lars Reneby, Ph.D. thesis, 1991.
108. **Queueing Models of the Window Flow Control Mechanism**
Lars Falk, Lic. thesis, 1991.
109. **On Efficiency and Optimality in Overload Control of SPC Systems**
Tobias Rydén, Lic. thesis, 1991.
110. **Enhancements of Communication Resources**
Johan M Karlsson, Ph.D. thesis, 1992.
111. **On Overload Control in Telecommunication Systems**
Christian Nyberg, Ph.D. thesis, 1992.
112. **Black Box Specification Language for Software Systems**
Henrik Cosmo, Lic. thesis, 1994.
113. **Queueing Models of Window Flow Control and DQDB Analysis**
Lars Falk, Ph.D. thesis, 1995.
114. **End to End Transport Protocols over ATM**
Thomas Holmström, Lic. thesis, 1995.
115. **An Efficient Analysis of Service Interactions in Telecommunications**
Kristoffer Kimbler, Lic. thesis, 1995.
116. **Usage Specifications for Certification of Software Reliability**
Per Runeson, Lic. thesis, May 1996.
117. **Achieving an Early Software Reliability Estimate**
Anders Wesslén, Lic. thesis, May 1996.

118. **On Overload Control in Intelligent Networks**
Maria Kihl, Lic. thesis, June 1996.

119. **Overload Control in Distributed-Memory Systems**
Ulf Ahlfors, Lic. thesis, June 1996.

120. **Hierarchical Use Case Modelling for Requirements Engineering**
Björn Regnell, Lic. thesis, September 1996.

121. **Performance Analysis and Optimization via Simulation**
Anders Svensson, Ph.D. thesis, September 1996.

122. **On Network Oriented Overload Control in Intelligent Networks**
Lars Angelin, Lic. thesis, October 1996.

123. **Network Oriented Load Control in Intelligent Networks Based on Optimal Decisions**
Stefan Pettersson, Lic. thesis, October 1996.

124. **Impact Analysis in Software Process Improvement**
Martin Höst, Lic. thesis, December 1996.

125. **Towards Local Certifiability in Software Design**
Peter Molin, Lic. thesis, February 1997.

126. **Models for Estimation of Software Faults and Failures in Inspection and Test**
Per Runeson, Ph.D. thesis, January 1998.

127. **Reactive Congestion Control in ATM Networks**
Per Johansson, Lic. thesis, January 1998.

128. **Switch Performance and Mobility Aspects in ATM Networks**
Daniel Søbirk, Lic. thesis, June 1998.

129. **VPC Management in ATM Networks**
Sven-Olof Larsson, Lic. thesis, June 1998.

130. **On TCP/IP Traffic Modeling**
Pär Karlsson, Lic. thesis, February 1999.

131. **Overload Control Strategies for Distributed Communication Networks**
Maria Kihl, Ph.D. thesis, March 1999.

132. **Requirements Engineering with Use Cases - a Basis for Software Development**
Björn Regnell, Ph.D. thesis, April 1999.

133. **Utilisation of Historical Data for Controlling and Improving Software Development**
Magnus C. Ohlsson, Lic. thesis, May 1999.

134. **Early Evaluation of Software Process Change Proposals**
Martin Höst, Ph.D. thesis, June 1999.

135. **Improving Software Quality through Understanding and Early Estimations**
Anders Wesslén, Ph.D. thesis, June 1999.

136. **Performance Analysis of Bluetooth**
Niklas Johansson, Lic. thesis, March 2000.

137. **Controlling Software Quality through Inspections and Fault Content Estimations**
Thomas Thelin, Lic. thesis, May 2000

138. **On Fault Content Estimations Applied to Software Inspections and Testing**
Håkan Petersson, Lic. thesis, May 2000.

139. **Modeling and Evaluation of Internet Applications**
Ajit K. Jena, Lic. thesis, June 2000.

140. **Dynamic traffic Control in Multiservice Networks - Applications of Decision Mo-dels**
Ulf Ahlfors, Ph.D. thesis, October 2000.

141. **ATM Networks Performance - Charging and Wireless Protocols**
Torgny Holmberg, Lic. thesis, October 2000.

142. **Improving Product Quality through Effective Validation Methods**
Tomas Berling, Lic. thesis, December 2000.

143. **Controlling Fault-Prone Components for Software Evolution**
Magnus C. Ohlsson, Ph.D. thesis, June 2001.

144. **Performance of Distributed Information Systems**
Niklas Widell, Lic. thesis, February 2002.

145. **Quality Improvement in Software Platform Development**
Enrico Johansson, Lic. thesis, April 2002.

146. **Elicitation and Management of User Requirements in Market-Driven Software Development**
Johan Natt och Dag, Lic. thesis, June 2002.

147. **Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis**
Håkan Petersson, Ph.D. thesis, September 2002.

148. **Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections**
Thomas Thelin, Ph.D. thesis, September 2002.

149. **Software Information Management in Requirements and Test Documentation**
Thomas Olsson, Lic. thesis, October 2002.

150. **Increasing Involvement and Acceptance in Software Process Improvement**
Daniel Karlström, Lic. thesis, November 2002.

151. **Changes to Processes and Architectures; Suggested, Implemented and Analyzed from a Project viewpoint**
Josef Nedstam, Lic. thesis, November 2002.

152. **Resource Management in Cellular Networks -Handover Prioritization and Load Balancing Procedures**
Roland Zander, Lic. thesis, March 2003.

153. **On Optimisation of Fair and Robust Backbone Networks**
Pål Nilsson, Lic. thesis, October 2003.

154. **Exploring the Software Verification and Validation Process with Focus on Efficient Fault Detection**
Carina Andersson, Lic. thesis, November 2003.

155. **Improving Requirements Selection Quality in Market-Driven Software Development**
Lena Karlsson, Lic. thesis, November 2003.

156. **Fair Scheduling and Resource Allocation in Packet Based Radio Access Networks**
Torgny Holmberg, Ph.D. thesis, November 2003.

157. **Increasing Product Quality by Verification and Validation Improvements in an Industrial Setting**
Tomas Berling, Ph.D. thesis, December 2003.

158. **Some Topics in Web Performance Analysis**
Jianhua Cao, Lic. thesis, June 2004.

159. **Overload Control and Performance Evaluation in a Parlay/OSA Environment**
Jens K. Andersson, Lic. thesis, August 2004.

160. **Performance Modeling and Control of Web Servers**
Mikael Andersson, Lic. thesis, September 2004.

161. **Integrating Management and Engineering Processes in Software Product Deve-lopment**
Daniel Karlström, Ph.D. thesis, December 2004.

162. **Managing Natural Language Requirements in Large-Scale Software Development**
Johan Natt och Dag, Ph.D. thesis, February 2005.

163. **Designing Resilient and Fair Multi-layer Telecommunication Networks**
Eligijus Kubilinskas, Lic. thesis, February 2005.

164. **Internet Access and Performance in Ad hoc Networks**
Anders Nilsson, Lic. thesis, April 2005.

165. **Active Resource Management in Middleware and Self-oriented Architectures**
Niklas Widell, Ph.D. thesis, May 2005.

166. **Quality Improvement with Focus on Performance in Software Platform Development**
Enrico Johansson, Ph.D. thesis, June 2005.

167. **On Inter-System Handover in a Wireless Hierarchical Structure**
Henrik Persson, Lic. thesis, September 2005.

168. **Prioritization Procedures for Resource Management in Cellular Networks**
Roland Zander, Ph.D. thesis, November 2005.

169. **Strategies for Management of Architectural Change and Evolution**
Josef Nedstam, Ph.D. thesis, December 2005.

170. **Internet Access and QoS in Ad Hoc Networks**
Ali Hamidian, Lic. thesis, April 2006.

171. **Managing Software Quality through Empirical Analysis of Fault Detection**
Carina Andersson, Ph.D. thesis, May 2006.

172. **Fairness in communication and computer networks design**
Pål Nilsson, Ph.D. thesis, September 2006.