LCA-REPORT-2006-001
available from http://infoscience.epfl.ch

A Unified Framework for Max-Min and Min-Max Fairness with Applications

Božidar Radunović and Jean-Yves Le Boudec
bozidar.radunovic@epfl.ch, jean-yves.leboudec@epfl.ch
EPFL, CH-1015 Lausanne, Switzerland *

January 31, 2006

Abstract

Max-min fairness is widely used in various areas of networking. In every case where it is used, there is a proof of existence and one or several algorithms for computing the max-min fair allocation; in most, but not all cases, they are based on the notion of bottlenecks. In spite of this wide applicability, there are still examples, arising in the context of wireless or peer-to-peer networks, where the existing theories do not seem to apply directly. In this paper, we give a unifying treatment of max-min fairness, which encompasses all existing results in a simplifying framework, and extend its applicability to new examples. First, we observe that the existence of max-min fairness is actually a geometric property of the set of feasible allocations (uniqueness always holds). There exist sets on which max-min fairness does not exist, and we describe a large class of sets on which a max-min fair allocation does exist. This class contains, but is not limited to the compact, convex sets of $\mathbb{R}^N$. Second, we give a general purpose centralized algorithm, called Max-min Programming, for computing the max-min fair allocation in all cases where it exists (whether the set of feasible allocations is in our class or not). Its complexity is of the order of N linear programming steps in $\mathbb{R}^N$, in the case where the feasible set is defined by linear constraints. We show that, if the set of feasible allocations has the free-disposal property, then Max-min Programming reduces to a simpler algorithm, called Water Filling, whose complexity is much lower. Free disposal corresponds to the cases where a bottleneck argument can be made, and Water Filling is the general form of all previously known centralized algorithms for such cases. Our derivations are based on the relation between max-min fairness and leximin ordering. All our results apply mutatis mutandis to min-max fairness. Our results apply to weighted, unweighted and util-max-min and min-max fairness. Distributed algorithms for the computation of max-min fair allocations are outside the scope of this paper.

1 Introduction

1.1 Max-min Fairness

Max-min fairness is a simple, well-recognized approach to define fairness in networks [6]; it aims at allocating as much as possible to users with low rates, and, at the same time, not unnecessarily wasting

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

resources (see Section 2.1 for a formal definition). It was used in window flow control protocols [8], then became very popular in the context of bandwidth sharing policies for ABR service in ATM networks [3]. It is now widely used in various areas of networking [27, 29, 28, 13, 10, 7, 19, 16, 8, 1].

One of the simplest max-min fairness examples, given in [6], is single-path rate allocation. Suppose we have a network consisting of links with fixed capacities, and a set of source destination pairs that communicate over a single path each, and with fixed routing. The problem is to allocate a rate to each source-destination pair, while keeping the rate on each link below capacity. Here, we call a rate allocation max-min fair if one cannot increase the rate of a flow without decreasing the rate of an already smaller flow. A set of feasible rate allocations for a simple two source example is given in Figure 1. A definition dual to a max-min fair allocation is min-max fair allocation, and is used in the context of workload distribution, where the goal is to spread a given workload evenly to all the parties (see [15]) and where rates have to be allocated to available links as evenly as possible.

1.2 Microeconomic Approaches to Fairness

Microeconomic theories of social welfare functions and social optima discuss a fair choice of alternatives (such as goods distribution or policy making) [2]. Each possible alternative is assigned a utility, that represents its value to each individual in the system. A social welfare function is a way to aggregate individual utilities into a social utility. The optimal choice of the alternative is the one that maximizes the social welfare function [2].

There are numerous ways to define social welfare functions. One is a maximin or Rawlsian social welfare function [22] that maximizes the utility of the worst-off individual. It has been widely used in the design of communication systems (see for example [18]).

The main problem of the maximin social welfare function is that the optimal alternative is not necessarily Pareto optimal. In other words, starting from the maximin optimal alternative one can increase the utility of one individual without decreasing utilities of the others, and this is clearly not a desirable property of an efficient alternative.

A leximin social welfare ordering is a refinement of the maximin social welfare function [5, 4]. It is based on the notion of the *leximin ordering*: one vector is said to be leximin larger or equal than the other if its ordered permutation is lexicographically larger or equal to the ordered permutation of the other vector (a precise definition is given in Definition 4 in Section 2.2). The leximin social welfare optimum is always Pareto optimal [2].

The fairness criteria in networking are based on findings from social welfare theory. Max-min fairness is closely related to leximin ordering. We discuss this issue in depth in Section 2.2. Other concepts of fairness, like proportional fairness [9], are also based on social welfare theory. We discuss them in more detail in Section 2.4.

Another important concept from microeconomics used in this paper is the *free disposal property*. In economics, it is defined as the right of each user to dispose of an arbitrary amount of owned commodities [2], or alternatively, to consume fewer resources than maximally allowed. The formal definition is given in Definition 6 in Section 3.2.

1.3 Bottleneck and Water-Filling

Most of the existing works on max-min fairness rely on the notion of bottleneck link. Referring again to the single-path rate allocation example given in Figure 1, we say that a link is a bottleneck for a given flow if the flow uses the link, if the link is fully utilized, and if the flow has the maximal rate among all the flows that use the link. It is shown in [6] for the above example that if each flow has

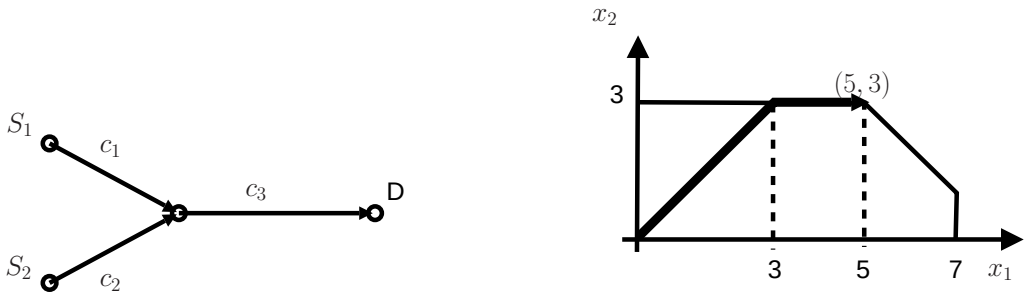

Figure 1: An example of a network with its feasible rate set, and max-min allocation obtained by water-filling. On the left, a network of 3 links is given. Flow x_1 connects S_1 and D and passes over links 1 and 3, and flow x_2 connects S_2 and D and traverses links 2 and 3. The set of feasible rates (x_1, x_2) is depicted by the inequalities: $0 \leq x_1 \leq c_1$, $0 \leq x_2 \leq c_2$ and $x_1 + x_2 \leq c_3$, and it is given on the right ($c_1 = 7, c_2 = 3, c_3 = 8$). The water-filling, as explained in [6], is depicted by the bold arrow: first, rates on both flows are increased, until flow x_2 hits the limit $x_2 = 3$. Then, x_2 is fixed to 3, and x_1 is further increased until we reach the equality $x_1 + x_2 = 8$. The max-min fair rate allocation is $(5, 3)$.

a bottleneck link, then the rate allocation is max-min fair. This finding, which we call the bottleneck argument, is often used to prove the existence of max-min fairness.

The most widely used algorithm for obtaining max-min fairness is the *water-filling* algorithm (WF) [6]. The principles of WF are the following: rates of all flows are increased at the same pace, until one or more links are saturated. The rates of flows passing over saturated links are then frozen, and the other flows continue to increase rates. The algorithm is repeated until all rates are frozen. A more precise description of WF algorithm is given in Section 3.2. It is proven in [6] that the output of WF, applied on a wired network, yields max-min fair allocation.

A simple example of WF in two dimensions on a wired network with single-path routing is given in Figure 1. We see in the example that although WF, as defined in [6], is related to the network topology, max-min fair allocation itself is solely a property of the set of feasible rates.

An extension of this scenario is introduced, for example, in [13] and [28]. Each flow is separately guaranteed a minimal rate. The algorithm used in [13] and [28] for computing the max-min fair rate allocation is a modified WF. Specifically, all rates are set to their minimal guaranteed values, and only the lowest rates are increased. A simple 2-dimensional example with an illustration of WF is given on the left of Figure 2.

Max-min fairness for single-rate multicast sessions is defined in [10]. This is generalized to multi-rate multicast sessions in [7]. Rates are again upper-bounded by links' capacities, and here we are interested in max-min fair allocation of receivers rates. A set of feasible allocations is linearly constrained, and a WF approach can be used. The geometric shape of the feasible set is essentially the same as in single-path routing.

The aforementioned scenarios have in common that the linearity of the constraints defining the feasible set. In [29], a single-path routing scenario is considered, and each source is assigned a utility, which is an increasing and concave function of its rate. Instead of searching for a max-min fair rate allocation, the authors of [29] look for max-min fair utility allocation. This approach is generalized in [7], where a max-min fair utility allocation is considered in the context of a multicast network. Here, the authors only required that a utility function be a strictly increasing but not necessarily concave function of rate, hence the feasible set is not necessarily convex. A simple 2-dimensional example is given in the right hand side of Figure 2. The WF algorithm can be used in this case as well.

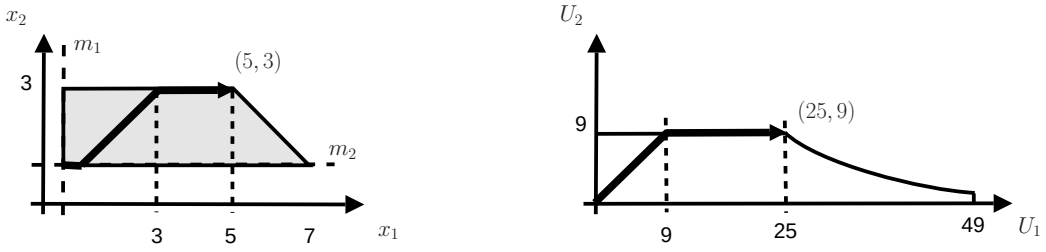

Figure 2: More examples of feasible rate sets. We consider again the topology given on the left of Figure 1. We first assume there are minimum rates guaranteed, $m_1 = 0.5$ and $m_2 = 1$, for flows x_1 and x_2 respectively. The feasible set for this case is depicted on the left. The water-filling, as explained in [13, 28], is represented with the bold arrow: we start by setting both flows on their minimal values ($x_1 = m_1, x_2 = m_2$). We start by increasing the smaller rate, x_1, until we reach $x_1 = x_2 = 1$. The rest is the same as in the example in Figure 1. On the right we consider utility max-min fairness, presented in [29, 7]. The network is the same as in Figure 1, and we assume that the utility function is $U(x) = x^2$. The set of feasible utilities is depicted on the right. It is a non-convex set. The max-min fair utility allocation can be obtained by water-filling, similarly as in Figure 1: utilities on both flows are increased, until flow x_2 hits the limit $x_2 = 3$. Then, x_2 is frozen, and x_1 is further increased until we reach the equality $x_1 + x_2 = 8$. The max-min fair utility allocation is $(25, 9)$ which corresponds to the rate allocation $(5, 3)$.

1.4 When Bottleneck and Water-Filling Become Less Obvious

It is not always obvious how to generalize the notion of a bottleneck link and the water-filling approach to an arbitrary problem. To see why, consider a point-to-point multi-path routing scenario, where, to our knowledge, max-min fairness was not studied before. We look at the same set-up as above, but now allow for multiple paths to be used by a single source-destination pair. The end-to-end rate of communication between a source and a destination is equal to the sum of the rates over all used paths. An example is given in Figure 3: when node 1 talks to node 4, it transmit using the direct path over link 1-4 and in parallel it can relay through node 3. The end-to-end rate of communication between 1 and 4 equals to the sum of rates over paths 1-4 and 1-3-4.

We are interested in a max-min fair rate allocation of end-to-end source-destination rates. In other words, an end-to-end source-destination rate allocation is max-min fair if one cannot increase the end-to-end rate of a source-destination pair without decreasing the rate of some other pair, which is already smaller.

Consider again the example in Figure 3, and assume that all links in the example have capacity 1. Then, by increasing all the rates at the same pace, we will have rates of all paths equal to 1/2 when link 3-4 saturates. Now, if we continue increasing the rate over path 1-4, the rate of source-destination pair 1 will be higher than the rate of source destination 2, and path 2-3-4 will loose its bottleneck since it is no longer the biggest end-to-end flow that uses 3-4. If we change the definition of the bottleneck, and instead of taking the biggest end-to-end flow, we consider the path with the highest rate, we obtain the max-min fair path rate allocation that differs from the end-to-end max-min fair rate allocation.

A first question that arises is how to define a bottleneck, such that the water-filling algorithm finds max-min fair end-to-end rate allocation, if it is possible at all. Also, it is not clear if for a given definition of a bottleneck we can still claim that if each path has a bottleneck, the allocation is max-min fair. Finally, we do not even know, using the existing state of the art, if the max-min fair end-to-end rate allocation exists on an arbitrary multi-path network.

This example can be solved by observing that the max-min fair allocation depends only on the set of feasible rates. Consider again the example in Figure 3, left. Assume that all links have unit capacity.

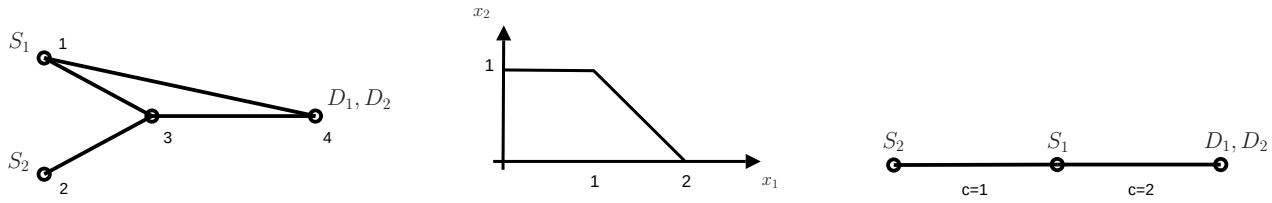

Figure 3: A simple multi-path example. Left: S1 sends to D1 over two paths, 1-3-4 and 1-4, while S2 sends to D2 over a single path 2-3-4. Center: the set of feasible rates. Right: the corresponding virtual single path problem.

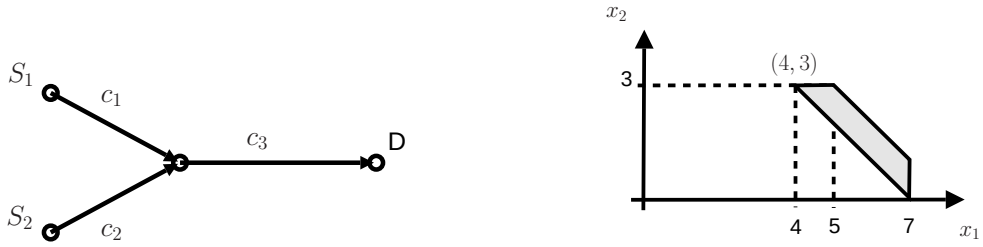

Figure 4: When water-filling does not work - consider the network topology on the left ($c_1 = 7, c_2 = 3, c_3 = 8$). Suppose that node D receives parts of the same stream from both S_1 and S_2, through flows x_1 and x_2, and suppose it needs a minimal total rate of $x_1 + x_2 \geq 7$. We want to minimize loads of servers S_1 and S_2, and we are interested in min-max fair allocation of (x_1, x_2). The feasible rates set is given on the right. Min-max fair allocation exists, and it is $(4, 3)$.

Call $x_1 = y_1 + y_2$ the rate of source 1, and x_2 the rate of source 2, where y_1 is the rate of source 1 on path $1 - 4$, and y_2 on path $1 - 3 - 4$. The set of feasible rates is the set of $(x_1 \geq 0, x_2 \geq 0)$ such that there exist slack variables $y_1 \geq 0, y_2 \geq 0$ with $y_1 \leq 1, y_2 + x_2 \leq 1$ and $x_1 = y_1 + y_2$. This is an *implicit* definition, which can be made explicit by eliminating the slack variables; this gives the conditions $x_1 \leq 1, x_1 + x_2 \leq 2$ (Figure 3, center). The set is convex, with a linear boundary, as in Figure 1, left. We can re-interpret the original multi-path problem as a virtual single path problem (Figure 3, right), and apply the existing WF algorithms. On the virtual single-path problem we can define bottlenecks in a usual way. Note however that the concept of bottleneck in the virtual single path problem has lost its physical interpretation on the original problem.

1.5 When Bottleneck and Water-Filling Do Not Work

Unfortunately, the approach with a virtual bottleneck does not always work. Consider the following workload distribution example: servers in a peer-to-peer network send data to a client; every client receives data from multiple servers, and has a guaranteed minimal rate of reception. Each flow from a server to a client is constrained by link capacities. Our goal is to equalize load on the servers while satisfying the capacity constraints.

A natural definition of fairness in this setting is min-max fairness, where we try to give the least possible work to the most loaded server. We say that a load on the servers is min-max fair if we cannot decrease a load on a server without increasing a load of another server that already has a higher load. A 2-dimensional example is given and explained in Figure 4. One can verify that is not possible to define a virtual bottleneck in this case. We discuss this example in more detail in Section 3.2.2 and Section 4.2.

A similar, but simpler, example is given in [15], which focuses on finding a leximax minimal

allocation (we show in Section 3 that the leximax minimal allocation obtained in [15] is in fact min-max fair). Its complexity is of the order of N polynomial steps in $\mathbb{R}^N$, in the case where the feasible set is defined by linear constraints.

In Section 4.3 we present another example where water-filling does not work. We consider the lifetime of nodes in a sensor network, inspired by the example introduced in [14], which studied the minimum lifetime. The lifetime of a node is a time until a node exhausts its battery, and it depends on the routing policy of a network. Unlike in [14], we study the routing strategy that achieves the min-max fair allocation of lifetimes of nodes. We characterize the set of lifetimes that can be achieved with any possible routing strategy, and we show that the min-max fair lifetime allocation exists. However, as we also show, it is not possible to obtain it by water-filling.

1.6 Our Findings

Our first finding is on the existence of max-min fairness. We give a large class of continuous sets on which a max-min fair allocation does exist, and we theoretically prove the existence. This class contains, but is not limited to the all compact, convex subsets of an arbitrary dimension Euclidean space $\mathbb{R}^N$. We also illustrate in a few examples that there are sets on which max-min fairness does not exist, thus that our result is not trivial.

Our second finding is on algorithms to locate the max-min fair allocation. In Section 3, we give a general purpose, centralized algorithm, called *Max-min Programming (MP)*, and prove that it finds the max-min fair allocation in all cases where it exists. Its complexity is of the order of N linear programming steps in $\mathbb{R}^N$, in general, whenever the feasible set is defined by linear constraints.

The third finding is on the relation between the general MP algorithm and the existing WF algorithm. We recall the definition of the free disposal property and show that, whenever it holds, Max-min programming (MP) degenerates to the simpler Water-filling (WF) algorithm (originally defined in [6]), whose complexity is much lower. The free-disposal property corresponds to cases where a bottleneck argument can be made, all previously known centralized algorithms for such cases rely on the water-filling approach. We note that WF requires the feasible set to be given in explicit form, unlike MP, and we discuss the case of an implicit feasible set with the free-disposal property.

We use a novel approach to analyze properties of max-min fairness. Instead of considering a specific networking problem with an underlying network topology, we focus only on the feasible rate sets. Therefore, our framework does not depend on a specific problem; it is general and it unifies the existing approaches that analyze max-min fairness.

In Section 4 we show applications of the results for three networking examples. We give specific, numerical examples where max-min fair allocation exists, but the feasible sets do not have the free-disposal property, hence a classical water-filling cannot be used. We show in these examples how MP does find max-min fair allocation even when the free-disposal does not hold. This way, we verify that our framework unifies previous results, and extends the applicability of max-min fairness to new scenarios.

All our results are given for max-min fairness; they apply mutatis mutandis to min-max fairness. They are valid for weighted and unweighted max-min and min-max fairness, using the transformation given in Section 2.1. Distributed algorithms for the computation of max-min fair allocations [8, 1] are left outside the scope of this paper.

1.7 Organization of The Paper

In Section 2 we define our framework (max-min and min-max fairness in N continuous variables). We mention a number of elementary results, such as the uniqueness and the reduction of weighted

max-min fairness to the unweighted case. We recall the definition of leximin ordering that we use in a latter analysis. We prove our first main result about the existence of max-min fairness. At the end of the section we also show that max-min fairness is a limiting case of utility fairness whenever a feasible set is compact.

In Section 3, we give the definitions of the two analyzed algorithms: Max-min Programming (MP) and Water-filling (WF), and we discuss the other two main findings. In Section 4 we illustrate our framework on three networking examples. We conclude in Section 5. Proofs are in the appendix.

2 Max-Min and Min-Max Fairness in Euclidean Spaces

In this section we provide a precise definition of max-min and min-max fairness and give results on their existence.

2.1 Definitions and Uniqueness

Consider a set $\mathcal{X} \subset \mathbb{R}^N$. We define the max-min and min-max fair vectors with respect to set $\mathcal{X}$ as follows:

Definition 1 *[6] A vector $\vec{x}$ is "max-min fair on set $\mathcal{X}$" if and only if*

$$(\forall \vec{y} \in \mathcal{X})(\exists s \in \{1, ..., N\})\ y_s > x_s \quad \Rightarrow \quad (\exists t \in \{1, ..., N\})\ y_t < x_t \leq x_s \tag{1}$$

i.e. increasing some component x_s must be at the expense of decreasing some already smaller or equal component x_t.

Definition 2 *A vector $\vec{x}$ is "min-max fair on set $\mathcal{X}$" if and only if*

$$(\forall \vec{y} \in \mathcal{X})(\exists s \in \{1, ..., N\})\ y_s < x_s \quad \Rightarrow \quad (\exists t \in \{1, ..., N\})\ y_t > x_t \geq x_s \tag{2}$$

i.e. decreasing some component x_s must be at the expense of increasing some already larger component x_t.

It is easy to verify that if $\vec{x}$ is a min-max fair vector on $\mathcal{X}$, then $-\vec{x}$ is max-min fair on $-\mathcal{X}$ and vice versa. If we show properties of max-min fairness on set $-\mathcal{X}$, the same properties will hold for min-max fairness on set $\mathcal{X}$. Thus, in the remainder of the paper, we give theoretical results only for max-min fairness.

Uniqueness of max-min fairness is assured by the following proposition:

Proposition 1 *[6] If a max-min fair vector exists on a set $\mathcal{X}$, then it is unique.*

The proof of the proposition is given in [6].

Weighted min-max fairness is a classical variation of max-min fairness, defined as follows. Given some positive constants w_i (called the "weights"), a vector $\vec{x}$ is "weighted-max-min fair" on set $\mathcal{X}$, if and only if increasing one component x_s must be at the expense of decreasing some other component x_t such that $x_t/w_t \leq x_s/w_s$ [6]. This is generalized in [7], which introduces the concept of "util max-min fairness": given N increasing functions $\phi_i : \mathbb{R} \to \mathbb{R}$, interpreted as utility functions, a vector $\vec{x}$ is "util-max-min fair" on set $\mathcal{X}$ if and only if increasing one component x_s must be at the expense of decreasing some other component x_t such that $\phi_t(x_t) \leq \phi_s(x_s)$ (this is also called "weighted max-min fairness" in [19]). Consider the mapping ϕ defined by

$$(x_1, \cdots, x_N) \to (\phi_1(x_1), \cdots, \phi_N(x_N)) \tag{3}$$

It follows immediately that a vector $\vec{x}$ is util-max-min fair on set $\mathcal{X}$ if and only if $\phi(\vec{x})$ is max-min fair on the set $\phi(\mathcal{X})$, the case of weighted max-min fairness corresponding to $\phi_i(x_i) = x_i/w_i$. Thus, we now restrict our attention to unweighted max-min fairness.

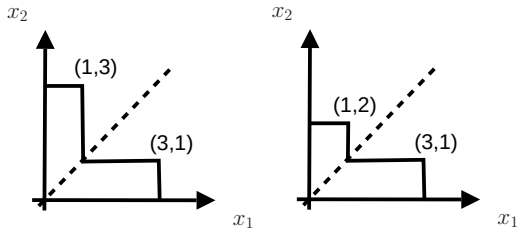

Figure 5: Examples of 2-dimensional sets that do not have max-min fair allocation. Point $(1,3)$ is not max-min fair in the example on the left since there exists point $(3,1)$ that contradicts with definition Definition 1. Both points $(1,3)$ and $(3,1)$ are leximin maximal in this example. In the example on the right, point points $(3,1)$ is the single leximin maximal point. Still, it is not the max-min fair point. Note that there exist no real networking example we are aware of that has these feasible rate sets – these sets are only artificial examples that illustrate properties of leximin ordering.

2.2 Max-Min Fairness and Leximin Ordering

In the rest of our paper we will extensively use leximin ordering, a concept we borrow from economics, and which we now recall. Let us define the "order mapping" $\mathcal{T} : \mathbb{R}^N \to \mathbb{R}^N$ as the mapping that sorts $\vec{x}$ in non-decreasing order, that is: $\mathcal{T}(x_1, \cdots, x_n) = (x_{(1)}, \cdots, x_{(n)})$, with $x_{(1)} \leq x_{(2)} \cdots \leq x_{(n)}$ and for all i, $x_{(i)}$ is one of the x_js. Let us also define the lexicographic ordering of vectors in $\mathcal{X}$ by $\vec{x} \stackrel{lex}{>} \vec{y}$ if and only if $(\exists i)\, x_i > y_i \text{ and } (\forall j < i)\, x_j = y_j$. We also say that $\vec{x} \stackrel{lex}{\geq} \vec{y}$ if and only if $\vec{x} \stackrel{lex}{>} \vec{y}$ or $\vec{x} = \vec{y}$. This latter relation is a total order on $\mathbb{R}^N$.

Definition 3 *[2] Vector $\vec{x}$ is* leximin larger *than or equal to $\vec{y}$ if $\mathcal{T}(\vec{x}) \stackrel{lex}{\geq} \mathcal{T}(\vec{y})$.*

Definition 4 *[2] Vector $\vec{x} \in \mathcal{X}$ is* leximin maximal *on a set $\mathcal{X}$ if for all $\vec{y} \in \mathcal{X}$ we have $\mathcal{T}(\vec{x}) \stackrel{lex}{\geq} \mathcal{T}(\vec{y})$.*

Note that a leximin maximum is not necessarily unique. See Figure 5 on the left for a counter-example.

Proposition 2 *[24] Any compact subset of $\mathbb{R}^n$ has a leximin maximal vector.*

It has been observed in [29, 13, 7] that a max-min fair allocation is also leximin maximal, for the feasible sets defined in these papers. It is generalized to an arbitrary feasible set in [24], as follows.

Proposition 3 *[24] If a max-min fair vector exists on a set $\mathcal{X}$, then it is the unique leximin maximal vector on $\mathcal{X}$.*

Thus, the existence of a max-min fair vector implies the uniqueness of a leximin maximum. The converse is not true: see Figure 5, right, for an example of a set with unique leximin maximal vector which is not max-min achievable. [24] defines a weaker version of max-min fairness, "maximal fairness"; it corresponds to the notion of leximin maximal vector, hence it is not unique, and exists on a larger class of feasible sets. We leave this weaker version outside the scope of this paper.

It is shown in [2] that if a vector is leximin maximal, it is also Pareto optimal. Therefore, from Proposition 3 it follows that the max-min fair vector, if it exists, is Pareto optimal. The converse is not necessarily true.

2.3 Existence and Max-Min Achievable Sets

As already mentioned, a number of papers showed the existence of max-min fair allocation in many cases, using different methods. We give here a generalized proof that holds on a larger class of continuous sets that incorporates, but is not limited to convex sets. This class of continuous sets includes the feasible sets of all the networking applications we are aware of. Note that a max-min fair vector does not exist on all feasible sets, even sets that are compact and connected. Simple counter-examples are given in Figure 5. However, these counter-examples are hand-crafted and do not correspond to any networking scenario. In the reminder of this section we give a sufficient condition for the existence of a max-min vector.

Definition 5 *A set $\mathcal{X}$ is max-min achievable if there exists a max-min fair vector on $\mathcal{X}$.*

Theorem 1 *Consider a mapping ϕ defined as in Equation 3. Assume that ϕ_i is increasing and continuous for all i. If the set $\mathcal{X}$ is convex and compact, then $\phi(\mathcal{X})$ is max-min achievable.*

The proof is in the appendix. As a special case, obtained by letting $\phi_i(x) = x$, we conclude that all convex and compact sets are max-min achievable. Taking $\phi_i(x) = x/w_i$, we also conclude that weighted max-min fairness exists on all compact, convex sets. More generally, util-max-min fairness exists on all compact, convex sets, if the utility functions are continuous (and increasing).

Note that if $\vec{x}^*$ is max-min fair on $\mathcal{X}$ and $\vec{y}^*$ is max-min fair on $\phi(\mathcal{X})$, then in general $\vec{y}^* \neq \phi(\vec{x}^*)$. To see this, consider an example of a wired network with a single link of capacity c, with two flows, x_1 and x_2 passing over the link. The max-min fair rate allocation in this case is $x_1 = x_2 = c/2$. Let us next define the utility functions $\phi_i(x) = x/w_i$, as above, and let $w_1 = 1, w_2 = 2$. We then have $u_1 = x_1, u_2 = x_2/2$, and the max-min fair utility allocation is going to be $u_1 = u_2 = c/3$ and the resulting rate allocation is going to be $x_1 = c/3, x_2 = 2c/3$, which differs from the max-min fair rate allocation in the first case.

In [29], the utility functions ϕ_i are arbitrary, continuous, increasing and concave functions. With these assumptions, the set $\phi(\mathcal{X})$ is also convex and compact. Note that in general, though, the set $\phi(\mathcal{X})$ used in Theorem 1 is not necessarily convex. Examples with non-convex sets are provided in [19] and [7].

2.4 Max-min Fairness as A Limiting Case of Utility Fairness

As explained in Section 1.2, utility fairness is an another popular class of fairness objectives in networking, derived from social welfare theory. If $\mathcal{X} \subset \mathbb{R}^N$ is the set of feasible rate allocation, and $U_i : \mathbb{R} \to \mathbb{R}$ for all $i \in \{1 \cdots N\}$, then utility fair rate allocation is the one that maximizes $\sum_{i=1}^{N} U_i(x_i)$ over $x \in \mathcal{X}$. The best-known example of utility fairness is proportional fairness [9], where $U_i(x) = \log(x)$. There exist many distributed algorithms for solving the convex optimization problem of finding the utility fair rate allocation [12, 17].

It is known that no continuous utility function exists that can represent leximin ordering [2], hence we cannot directly use the existing utility-based algorithms to find a max-min fair vector. However, it is shown in [12] that for wired networks with single-path routing, and for a class of utility functions defined as

$$f_m(x) = \begin{cases} (1-m)^{-1}x^{1-m} & \text{if } m \neq 1 \\ \log(x) & \text{if } m = 1 \end{cases}, \tag{4}$$

the utility fair rate allocation converges to max-min fair one as m goes to infinity. Therefore, by choosing an appropriate m, we can approach arbitrarily close to the max-min fair allocation using the existing algorithms [12, 17].

The above convergence property of max-min fair allocation is shown to hold, in [12], only in the case of wired networks with single-path routing, and the proof of it heavily depends on the notion of bottleneck. It is not known if the same result holds for an arbitrary problem where max-min fair allocation applies. We show here that it holds whenever the feasible allocation set is convex.

Let f_m be a family of increasing, concave and differentiable functions defined on $\mathbb{R}^+$. Assume that, for any fixed numbers x and δ,

$$\lim_{m\to+\infty} \frac{f_m(x)}{f_m(x+\delta)} = 0, \tag{5}$$

$$\lim_{m\to+\infty} \frac{f'_m(x)}{f'_m(x+\delta)} = 0. \tag{6}$$

The assumptions (5) and (6) are satisfied if for example f_m is defined as in (4).

Let us consider a convex set $\mathcal{X}$ and let $\vec{x}^m$ be the utility-fair vector on $\mathcal{X}$, that is the unique vector that maximizes $\sum_{i=1}^{N} f_m(x_i)$ The following theorem shows that a max-min fair vector is then a limiting case of utility fair vector for the set of utilities we constructed above:

Theorem 2 *The set of utility-fair vectors $\vec{x}^m$ converges toward the max-min fair vector as m tends to $+\infty$.*

The proof is in the appendix.

3 Max-Min Programming and Water-Filling

In the following section present the max-min programming (MP) algorithm, which finds the max-min fair vector on any feasible set, if it exists. We also define a condition called a free-disposal property, and show that, under that conditions, a commonly used water-filling (WF) algorithm coincides with the MP algorithm, and is guaranteed to find the max-min fair allocation.

3.1 The Max-Min Programming (MP) Algorithm

The idea of the MP algorithm is first to find the smallest component of the max-min fair vector, which is done by maximizing the minimal coordinate. Once this is done, the minimal coordinate is fixed, and the dimension corresponding to the minimal coordinate is removed. This step is repeated until all coordinates are fixed, and we show that a vector obtained in such way is indeed the max-min fair one. A precise definition of the algorithm is given below.

1. let $S^0 = \{1, ..., N\}, \mathcal{X}^0 = \mathcal{X}, \mathcal{R}^0 = \mathcal{X}$ and $n = 0$
2. **do**
3. $n = n + 1$
4. Problem MP^n: maximize T^n subject to:
   $$\begin{cases} (\forall i \in S^{n-1}) & x_i \geq T^n \\ \text{for some} & \vec{x} \in \mathcal{X}^{n-1} \end{cases}$$
5. let $\mathcal{X}^n = \{\vec{x} \in \mathcal{X}^{n-1} \mid (\forall i \in S^{n-1})\, x_i \geq T^n, (\exists i \in S^{n-1})\, x_i > T^n\}$,
   $\mathcal{R}^n = \{\vec{x} \in \mathcal{X}^{n-1} \mid (\forall i \in S^{n-1})\, x_i \geq T^n\}$
   and $S^n = \{i \in \{1, ..., N\} \mid (\forall \vec{x} \in \mathcal{X}^n)\, x_i > T^n\}$
6. **until** $S^n = \emptyset$
7. return the only element in $\mathcal{R}^n$

The algorithm maximizes in each step the minimal coordinate of the feasible vector, until all coordinates are processed. The n-th step of the algorithm is a minimization problem, called MP^n, where $\mathcal{X}^n$

represents the remaining search space, S^n represents the direction of search, in terms of coordinates that can be further increased, and $\mathcal{R}^n$ is the set that will, in the end, contain a single rate allocation, the max-min fair one.

3.1.1 Proof of Correctness

The algorithm always terminates if $\mathcal{X}$ is compact and max-min achievable, and $\mathcal{X}^n$ is reduced to one single element, which is the required max-min fair vector, as is proved in the following theorem:

Theorem 3 *If $\mathcal{X}$ is compact and max-min achievable, the above algorithm terminates and finds the max-min fair vector on $\mathcal{X}$ in at most N steps.*

The proof is in the appendix. Note that the theorem requires set $\mathcal{X}$ to be compact but this usually just a technical assumption since in most of the practical examples the feasible sets are compact.

The algorithm presented in [15] for calculating the leximax minimal allocation is a particular implementation of MP. In each step, this algorithm maximizes the minimum rate of links, which is exactly step 4 of the MP algorithm, tailored to the problem considered. The overall complexity of the algorithm in [15] is thus the same as the complexity of MP. Since the feasible set considered there is compact convex, it follows from Theorem 1 and Proposition 3 that the leximax minimal allocation obtained in [15] is in fact a min-max fair allocation.

3.1.2 Numerical Examples

In order to illustrate the behaviour of MP, we consider two simple examples. The first one is the network from Figure 1. The set of feasible rates is

$$\mathcal{X} = \{(x_1, x_2) \,|\, 0 \leq x_1 \leq 7, 0 \leq x_2 \leq 3, x_1 + x_2 \leq 8\}, \tag{7}$$

and it is depicted on the right of Figure 1. We are looking for the max-min fair rate allocation.

In the first step of the algorithm we have $\mathcal{X}^0 = \mathcal{X}, \mathcal{R}^0 = \mathcal{X}, S^0 = \{1, 2\}$. By solving the linear program in step 4, we obtain $T^1 = 3$. We further have $\mathcal{X}^1 = \{(x_1, 3) \,|\, 3 < x_1 \leq 5\}, \mathcal{R}^1 = \{(x_1, 3) \,|\, 3 \leq x_1 \leq 5\}, S^0 = \{1\}$. Again by solving the linear program in step 4 we obtain $T^2 = 5$. Now we have $\mathcal{X}^2 = \emptyset, \mathcal{R}^2 = \{(5, 3)\}, S^2 = \emptyset$. The algorithm terminates and set $\mathcal{R}^2$ contains only the max-min fair rate allocation.

The second example we consider is the load distribution example from Figure 4. The set of feasible rates is

$$\mathcal{X} = \{(x_1, x_2) \,|\, 0 \leq x_1 \leq 7, 0 \leq x_2 \leq 3, 7 \leq x_1 + x_2 \leq 8\}, \tag{8}$$

and it is depicted on the right of Figure 4. We are looking for the min-max fair rate allocation on set $\mathcal{X}$, which is equivalent of finding max-min fair rate allocation on set $-\mathcal{X}$, as discussed in Section 2.1.

In the first step of the algorithm we have $\mathcal{X}^0 = -\mathcal{X}, \mathcal{R}^0 = -\mathcal{X}, S^0 = \{-1, -2\}$. By solving the linear program in step 4 we obtain $T^1 = -4$. We then have $\mathcal{X}^1 = \{(-4, -3)\}, \mathcal{R}^1 = \{(-4, -3)\}, S^0 = \emptyset$. The algorithm terminates and set $\mathcal{R}^2$ contains a single allocation which. The min-max fair rate allocation is thus $(4, 3)$.

Note that when the max-min fair allocation does not exist, MP will not give one of the leximin maximal points, as one might expect. To see this, consider the examples from Figure 5. In both examples, in the first step of MP, we will have $T^1 = 1$ and $S^1 = \emptyset$, and the algorithm will return $(1, 1)$ as the optimal point. This point is neither leximin maximal, nor Pareto optimal.

.

Before applying MP to a specific class of problems, it is thus important to verify, e.g. using results from Section 2, that max-min fairness exists. This has to be done only once, since the existence of max-min fairness depends on the nature of the problem. On the contrary, once the existence is verified, the

MP algorithm can be further applied on any instance of the problem and will always yield the correct result.

3.2 The Water-Filling (WF) Algorithm

We now compare MP with the water-filling approach used in the traditional setting [6]. We here present a generalized version that includes minimal rate guarantees, as in [28].

We first introduce the concept of free disposal property. It is defined in economics as the right of each user to dispose of an arbitrary amount of owned commodities [2], or alternatively, to consume fewer resources than maximally allowed. We then modify it slightly, as follows. Call $\vec{e}_i$ a unitary vector $(\vec{e}_i)_j = \delta_{ij}$.

Definition 6 *We say that a set $\mathcal{X}$ has the free-disposal property if (1) there exists $\vec{m}$ with $x_i \geq m_i$ for all $\vec{x} \in \mathcal{X}$ and (2) for all $i \in \{1, ..., N\}$ and for all α such that $\vec{x} - \alpha\vec{e}_i \geq \vec{m}$, we have $\vec{x} - \alpha\vec{e}_i \in \mathcal{X}$.*

Informally, free disposal applies to sets where each coordinate is independently lower-bounded, and requires that we can always decrease a feasible vector, as long as we remain above the lower bounds. We now describe the Water-Filling algorithm.

0. Assume $\mathcal{X}$ is free-disposal
1. let $S^0 = \{1, ..., N\}, \mathcal{X}^0 = \mathcal{X}, \mathcal{R}^0 = \mathcal{R}$ and $n = 0$
2. **do**
3. $n = n + 1$
4. Problem WF^n: maximize T^n subject to:
$$\begin{cases} (\forall i \in S^{n-1}) & x_i = \max(T^n, m_i) \\ \text{for some} & \vec{x} \in \mathcal{X}^{n-1} \end{cases}$$
5. let $\mathcal{X}^n = \{\vec{x} \in \mathcal{X}^{n-1} \mid (\forall i \in S^{n-1})\, x_i \geq T^n, (\exists i \in S^{n-1})\, x_i > T^n\}$,
$\mathcal{R}^n = \{\vec{x} \in \mathcal{X}^{n-1} \mid (\forall i \in S^{n-1})\, x_i \geq T^n\}$
and $S^n = \{i \in \{1, ..., N\} \mid (\forall \vec{x} \in \mathcal{X}^n)\, x_i > T^n\}$
6. **until** $S^n = \emptyset$
7. return the only element in $\mathcal{X}^n$

3.2.1 Equivalence of WF and MP

The following theorem demonstrates the equivalence of MP and WF on free-disposal sets.

Theorem 4 *Let $\mathcal{X}$ be a max-min achievable set that satisfies the free-disposal property. Then, at every step n, the solutions to problems WF^n and MP^n are the same.*

The proof is in the appendix. Thus, under the conditions of the theorem, WF terminates and returns the same result as MP, namely the max-min fair vector if it exists. The theorem is actually stronger, since the two algorithms provide the same result at every step. However, if the free-disposal property does not hold, then WF may not compute the max-min fair allocation. We refer to Section 3.2.2 for such an example.

The examples previously mentioned of single path unicast routing [6], multicast util-max-min fairness [10, 7] and minimal rate guarantee [28, 13] all have the free-disposal property. Thus, the water-filling algorithm can be used, as is done in all the mentioned references. In contrast, the load distribution example [15] is not free-disposal, and all we can do is use MP, as is done in [15] in a specific example.

The multi-path routing example also has the free-disposal property, but the feasible set is defined implicitly. We discuss the implications of this in the next section.

3.2.2 Numerical Examples

To illustrate the behaviour of WF, we consider again the same two examples as in Section 3.1.2. In the first example, depicted in Figure 4, the feasible rate set, described by (7), has the free-disposal property. It is easy to verify that sets $\{\mathcal{X}^i\}_{i=1\cdots3}, \{\mathcal{R}^i\}_{i=1\cdots3}, \{S^i\}_{i=1\cdots3}$ are taking exactly the same values as in the case of MP, described in Section 3.1.2. This confirms the findings of Theorem 4.

The second example we consider is the load distribution example depicted in Figure 4 and described by (7). For this type of problem we cannot a priori set the upper limits in $\vec{m}$, as [13, 28], as they are not universal (they would need to depend on given network topology and are not known in advance). Then, it is easy to verify that the linear program in step 4 (with minimization instead of maximization since we are looking for min-max fairness) has no solution. Therefore, in this case, WF cannot find the min-max fair rate allocation.

Note that the free-disposal property is a sufficient but not a necessary condition for MP to degenerate to WF. This becomes evident when considering again the example from Figure 4. Suppose that $c_1 = 3, c_2 = 3, c_3 = 4$, and, in addition, the minimum rate constraint is $x_1 + x_2 \geq 3$. The feasible rate set in this example has the same shape and orientation as in Figure 4, but it is translated to the left such that it touches both x_1 and x_2 axes. In this case, it is easy to verify that the set still does not have the free-disposal property, since the shape and the orientation of the feasible set are the same. However WF finds the min-max allocation in a single step.

3.3 Complexity Of The Algorithms In Case Of Linear Constraints

Let us now assume that $\mathcal{X}$ is an n-dimensional feasible set defined by m linear inequalities. Each of the n steps of the MP algorithm is a linear programming problem, hence the overall complexity is $O(nLP(n,m))$, where $LP(n,m)$ is the complexity of linear programming. The WF algorithm also has n steps, each of complexity $O(m)$ (since in step 4 we have to find the maximum value of T that satisfies the equality in each of the m inequalities, and take as the result the smallest of those). Hence the complexity of WF is $O(nm)$. Linear programming has solutions of exponential complexity in the worst case, however in most practical cases there are solutions with polynomial complexity.

Assume next that $\mathcal{X}$ is defined implicitly, with an l-dimensional slack variable (for an example scenario, see multi-path case on Figure 3). We can use MP, which works on implicit sets, resulting in complexity $O(nLP(n,m))$. If the set is free-disposal, we can also use WF, but we need to find an explicit characterization of the feasible set. In most cases, finding an explicit characterization of the feasible set can be done in polynomial time. To see that, consider again the example from Figure 3. The slack variables represent rates of different paths, whereas we are interested only in the end-to-end rates. Finding a set of feasible end-to-end rates is equivalent to a well known problem of finding maximum flows in a network [25] (see [15] for an example in the networking context). As shown in [25], this is a problem of a polynomial complexity. Note that it might be possible to construct an implicitly defined feasible set that cannot be converted to an explicit form in a polynomial time. However, we are not aware of any existing example of such a set. A further analysis is out of the scope of our paper.

Once we have an explicit characterization, the remaining complexity of WF is still $O(mn)$. In practical applications, we are likely to be interested in explicitly finding the values of the slack variables at the max-min fair vector. Finding these values is a linear program. Here, it is sufficient to make the set explicit only once for a given problem. We conclude that in many practical problems, it is likely to be faster to make the set of constraints explicit and use WF rather than MP.

4 Example Scenarios

In this section we provide examples that arise in a networking context, which were not previously studied, and to which our theory applies. The first example deals with maximizing the life-time of certain sensor network. The feasible set does have the free-disposal property, hence both WF and MP work. However, the goal of this example is to elucidate how to convert an implicit feasible set to an explicit one, which is illustrated here with an example.

The second and the third examples are taken from problems that occur in P2P and wireless sensor networks, respectively. We show that in these two scenarios the feasible sets do not have the free-disposal property. We illustrate on simple but detailed numerical examples that WF does not work, whereas MP gives a correct result.

4.1 Minimum Energy Wireless Network

Our first novel example is the lifetime of a sensor network. Assume that we have a wireless sensor network, where an arbitrary set of source nodes collect and communicate information to corresponding destination nodes by relaying over intermediate nodes. Assume that each node has an initial energy, which decreases each time it transmits a message. Once a node has deplated its power, it cannot send anything any more. Rather than analyzing the dynamics of the traffic, we consider the total amount of information that can be conveyed over the network during its lifetime.

Consider a network with $n = \{1 \cdots N\}$ nodes, and $l = \{1 \cdots L\}$ links. The initial energy of node n is E_n. The energy needed to send one bit over link l is e_l. There is a set of $s = \{1 \cdots S\}$ source destination pairs that communicate data. We denote with X_s the total amount of bits transferred over the network during its lifetime by source-destination pair s. Those bits may be transferred over one of $p = \{1 \cdots P\}$ different paths, and the number of bits transferred over path p is denoted with Y_p. We further set $a_{n,l}$ to 1 if a link l contains node n, otherwise to 0, and $b_{l,p}$ to 1 if path p contains link l, otherwise to 0.

In general, a feasible information allocation can be described by the following constraint set

$$\mathcal{X} = \{X_s \,:\, E_n \geq \sum_l a_{n,l} e_l \sum_p b_{l,p} Y_p, X_s = \sum_p d_{s,p} Y_p\},$$

Since $\mathcal{X}$ is convex, it is also max-min achievable. It also has the free-disposal property so a max-min fair allocation can be found by either WF or MP since $\mathcal{X}$ is not given explicitly.

A simple four node example is given in Figure 6. Source S_1 sends data to destination D_1, and source S_2 sends data to destination D_2 (which happens to be the same node as D_1). Source S_1 sends over two paths, a direct path Y_1, and path Y_3 over node 3. Source S_2 sends over one path Y_3, over node 3. The initial energy of a node n is E_n, and the cost of sending each bit over link l is e_l.

For example, let us denote with Y_2 and Y_1 the total number of bits node 1 transmits over links 1 and 2 during its lifetime, respectively. For each bit sent over link 1 it will spend e_1 joules of energy, and for each bit sent over link 2 it will spend e_2 joules of energy. Hence, node 1 cannot violate the following energy constraint $E_1 \geq e_2 Y_1 + e_1 Y_2$. Similarly, we can write constraints for other nodes. We also have constraints on total traffic over a given link. Since flows Y_1 and Y_2 pass over link 1, total traffic on both flows cannot be larger than X_1. Putting all of them together, we obtain the following set of constraints

$$E_1 \geq e_2 Y_1 + e_1 Y_2, \tag{9}$$
$$E_2 \geq e_3 Y_3, \tag{10}$$
$$E_3 \geq e_4 (Y_2 + Y_3), \tag{11}$$

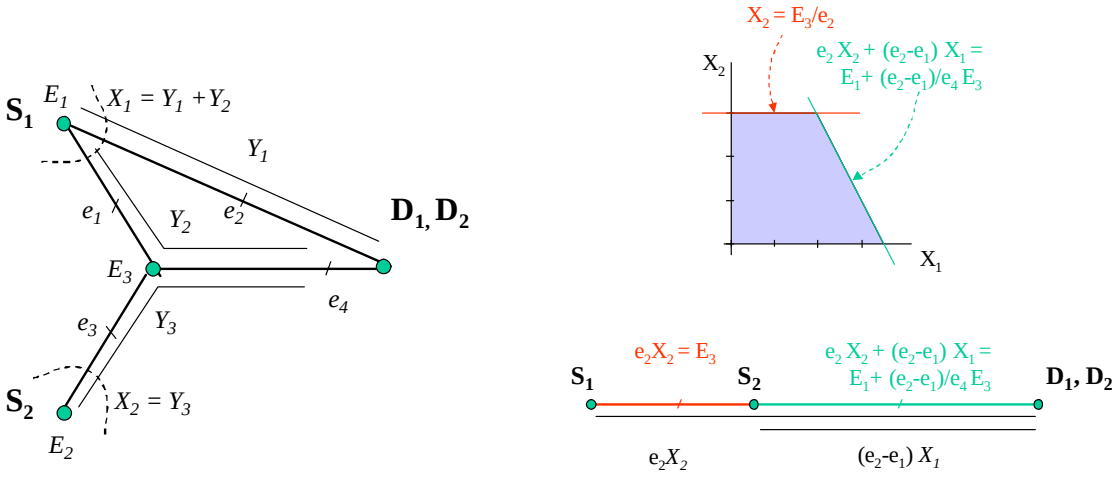

Figure 6: Minimum energy networks: Left: a four node example, with two sources and two destinations. Upper right: feasible set for the example on the left. Lower right: Single path routing problem with the equivalent feasible set.

$$X_1 = Y_1 + Y_2, \tag{12}$$
$$X_2 = Y_3, \tag{13}$$

where X_i is the total amount of information transferred from S_i to D_i. We are looking for a max-min fair allocation of information transferred.

To decrease the complexity of WF, we have to find an explicit characterization of the feasible set, hence we need to get rid of the slack variables and define constraints solely on X_1, X_2. To do this, one can solve the max-flow min-cut problem, using e.g. augmenting path method [25], only for a flow from S_1 to D_1, only for a flow from S_2 to D_2, and jointly for both flows from S_1 to D_1 and from S_2 and D_2. In the specific example from Figure 6, assuming that $E_1 > E_3$ and $e_1 > e_2$, we can transform the feasible set (9) - (13) into

$$X_2 \leq E_3/e_2, \tag{14}$$
$$e_2 X_1 + (e_1 - e_2) X_2 \leq E_1 + E_3(e_2 - e_1)/e_4. \tag{15}$$

as depicted in the upper right corner of Figure 6, and we can apply WF to find the max-min fair allocation. The same approach can be extended straightforward to an arbitrary network topology and performed in polynomial time. We refer to section 3.3 for discussion. It is interesting to notice that there exists a single-path routing problem with the same feasible set, shown in the lower right corner of Figure 6, with link capacities that correspond to inequalities (14) and (15). Therefore we can still identify bottlenecks in the classical sense, although now they do not have a clear physical interpretation i.e., they correspond to a virtual notion of maximum cuts in the max-flow min-cut algorithm.

The above model is an extension of the model presented in [14]. In [14] they maximize only the shortest lifetime, hence solving only MP^1. This is not useful in the case of large networks, where a death of one node cannot be treated as a death of the network itself. Also, the authors of [14] optimize the lifetime given the traffic matrix, whereas we allow an arbitrary traffic matrix and we optimize the amount of information transferred. Thus, [14] is a special case of our model presented here.

Our analysis is focused only on properties of the performance metric, and not on a specific realization of an algorithm. Unlike [14], where a fully distributed algorithm is given, we only give a centralized solution, and an implementation of a distributed protocol is out of scope and remains as a future work.

4.2 Load Distribution In P2P Systems

Let us consider a peer-to-peer network, where several servers can supply a single user with parts of a single data stream (e.g. by using Tornado codes [11]). There is a minimal rate a user needs to achieve, and there is an upper bound on each flow given by a network topology and link capacities.

Let $\vec{x}$ be the total loads on the servers, $\vec{y}$ the flows from the servers to clients, $\vec{z}$ the total traffic received by clients, $\vec{c}$ the capacities of links and $\vec{m}$ the minimum required rates of the flows. We can then represent the feasible rate set as

$$\mathcal{X} = \{\vec{x} : (\exists \vec{y}, \vec{z})\ A\vec{y} \leq \vec{c}, B\vec{y} = \vec{x}, C\vec{y} = \vec{z}, \vec{z} \geq \vec{m}\}, \quad A, B, C \geq 0, \tag{16}$$

where A, B and C are arbitrary matrices defined by network topology and routing.

A simple example depicted in Figure 4. Client D receives data from both servers S_1 and S_2 and it wants minimal guaranteed rate m. There is flow y_1 going from S_1 to D over links 1 and 3, and flow y_2 going from S_2 to D over links 2 and 3. We have that the total egress traffic of S_1 is $x_1 = y_1$, and of S_2 is $x_2 = y_2$. The total ingress traffic of D is $z_1 = y_1 + y_2$. We thus have the following matrices

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

that define the constraint set, visualized in Figure 4.

In a peer-to-peer scenario, each server is interested in minimizing its own load, hence it is natural to look for the min-max fair vector on set $\mathcal{X}$, which minimizes loads on highly loaded servers.

Since set $\mathcal{X}$ is convex, it is min-max achievable. Since it does not have the free-disposal property in general, WF is not applicable. This is shown in Section 3.1.2 on a simple example. Min-max fair allocation can be found by means of the MP algorithm. This is illustrated on the example in Section 3.1.2.

Note that this form of a feasible set is unique in that it introduces both upper and lower bounds on a sum of components of $\vec{x}$ and, as such, is more general than the feasible sets in the above presented examples, such as [15].

4.3 Maximum Lifetime Sensor Networks

In this section we consider an example motivated by [14, 23]. This example is somewhat similar to the example in Section 4.1. However, Section 4.1 considers wireless networks where the underlying physical interface eliminates intra-node interference, which holds only for a small class of physical and MAC layers (i.e. a model of 802.11 where the interference is eliminated by exclusions). In an improved, more realistic physical layer model, a transmission on one link provokes interference, which decreases the rates of other links in a region. In this example we assume that the network is built on the top of the ultra-wide band physical model described in [26].

Consider a set of $n = \{1 \cdots N\}$ nodes, some of which are sensors and some are sinks. We assume sensors feed data to sinks over the network, and can do so by sending directly, or relaying over other sensors or sinks. When node s sends data to node d, it does so using some transmission power P_s. The signal attenuates while propagating through space, and is received at d with power $P_s h_{sd}$, where h_{sd} is an arbitrary positive number, referred to as the attenuation between s and d.

Receiver d tries to decode the information sent by s in presence of noise and interference. If N denotes the white background noise, than the total interference experienced by D is $I = N + \sum_{i \neq s} P_i h_{id}$. The maximum rate of information d can achieve is then [26]

$$x_{sd} = K \frac{P_s h_{sd}}{N + \sum_{i \neq s} P_i h_{id}}.$$

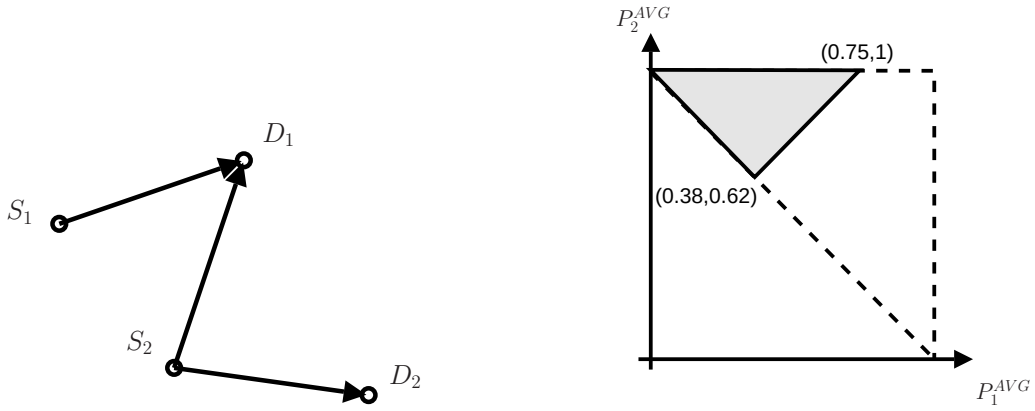

Figure 7: Sensor example: On the left an example of a network with 2 sensors and 2 sinks is given. We let $P^{MAX} = N = 1$, and $h_{S_1D_1} = h_{S_1D_2} = 1, h_{S_2D_1} = 10, h_{S_2D_2} = 0.7$, and the lower bounds on rates are $M1 = 0.6, M2 = 0.4$. On the right, the set of feasible average power dissipations is given.

We also assume that a node can only send to or receive from one node at a time.

In addition, nodes can change their transmission power over time. We assume a slotted protocol, where in every slot t, every node s can choose an arbitrary transmission power $P_s(t)$. If s chooses not to transmit, it sets $P_s(t) = 0$. A succession of slots in time is called a schedule. Link sd achieves rate $x_{sd}(t)$ where the rate depends on allocated powers, as explained above. We denote with x_{sd}^{AVG} the average rate of link sd throughout a schedule. Let $\mathbf{x}^{AVG}$ be the vector of all $\{x_{sd}^{AVG}\}_{1\leq s,d\leq N}$. We denote by $\mathcal{X}$ a set of feasible $\mathbf{x}^{AVG}$, that is such that there exists a schedule and power allocations that achieve those rates. Similarly to the average rate, we can calculate the average power dissipated by a node during a schedule, which we denote by P_s^{AVG}. We denote by $\mathcal{P}(\mathbf{x}^{AVG})$ a set of possible average power dissipations that achieve average rate $\mathbf{x}^{AVG}$. Refer to [21, 20] for a more detailed explanation of the model.

From the application point of view, we assume sensors measure the same type of information. Each of the several sinks needs to receive a certain rate of the information, regardless from what sensor it comes. Let us denote with R_d the total rate of information received by sink d. We then have a constraint $R_d \leq M_d$.

In order to define routing, let us further introduce a concept of paths. Path $p = \{1 \cdots P\}$ is a set of links. We say $A_{l,p} = 1$ if link $l = (s, d)$, for some s, d, belongs to path p. Otherwise, $a_{p,l} = 0$. We also say $B_{s,p} = 1$ and $C_{s,p} = 1$ if node s is the starting or the finishing point of the path p, respectively. Let y_p be the average rate on path p.

The goal is to minimize the average power dissipations, under the above constraints. The set of feasible average power dissipations can be formally described as

$$\mathcal{P} = \{\mathbf{p}^{AVG} \,|\, (\exists \mathbf{x}^{AVG} \in \mathcal{X})\, \mathbf{p}^{AVG} \in \mathcal{P}(\mathbf{x}^{AVG}), A\mathbf{y} \leq \mathbf{x}^{AVG}, \mathbf{R} = C\mathbf{y} \leq \mathbf{M}\}$$

We are interested in finding the min-max average power allocation over set $\mathcal{P}$.

This is a difficult optimization problem that has not been fully solved, and we do not intend to solve it here in its general form. Instead, we want to illustrate in a simple example from Figure 7, that the feasible set does not always have the free-disposal property, and furthermore that WF, as such, cannot be used.

In our simple example from Figure 7, we consider two sensors, S_1 and S_2, and two sinks, D_1 and D_2. We have three links, (S_1, D_1), (S_2, D_1), (S_2, D_2), and three paths that coincide with each link (we assume other links cannot be established due to for example a presence of a wall).

It is shown in [20] that in this type of network any average rate allocation can be achieved by using the following simple power allocation policy: when a node is transmitting, it does so with maximum

power; otherwise it is silent. It follows that any possible schedule in the network can have four possible slots:

Slot 1 of duration α_1: Only sensor S_1 sends to sink D_1 with full power P^{MAX} and S_2 is silent.

Slot 2 of duration α_2: Sensor S_1 sends to D_1 while S_2 sends to D_2.

Slot 3 of duration α_3: Only S_2 sends to D_1.

Slot 4 of duration α_4: Only S_2 sends to D_2.

If we normalize the duration of the schedule, we have $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$.

Under the above scheduling, we have the following average rates and average dissipated powers

$$R_1 = \alpha_1 \frac{P^{MAX} h_{S_1D_1}}{N} + \alpha_2 \frac{P^{MAX} h_{S_1D_1}}{N + P^{MAX} h_{S_2D_1}} + \alpha_3 \frac{P^{MAX} h_{S_2D_1}}{N}, \quad (17)$$

$$R_2 = \alpha_2 \frac{P^{MAX} h_{S_2D_2}}{N + P^{MAX} h_{S_1D_2}} + \alpha_4 \frac{P^{MAX} h_{S_2D_2}}{N}, \quad (18)$$

$$P_1^{AVG} = (\alpha_1 + \alpha_2 + \alpha_3) P^{MAX}, \quad (19)$$

$$P_2^{AVG} = (\alpha_2 + \alpha_4) P^{MAX}. \quad (20)$$

The set of feasible average powers is thus $\mathcal{X} = \{(P_1^{AVG}, P_2^{AVG}) \mid (\exists \alpha_{1\cdots 4}) \sum_{i=1}^{4} \alpha_i = 1, R_1 \geq M_1, R_2 \geq M_2\}$.

To obtain a numeric example, we set $K = P^{MAX} = N = 1, h_{S_1D_1} = h_{S_1D_2} = 1, h_{S_2D_1} = 10, h_{S_2D_2} = 0.7$, and $M1 = 0.6, M2 = 0.4$. Setting these values in (17)-(20) and simplifying the constraints, we achieve the following set of inequalities that defines set $\mathcal{X}$:

$$\begin{aligned} P_1^{AVG} + P_2^{AVG} &\geq 1, \\ P_1^{AVG} + \alpha_3 &\leq 1, \\ 7P_1^{AVG} + 14\alpha_3 + 1 &\leq 7P_2^{AVG}, \\ P_1^{AVG} + 110\alpha_3 - 3.4 &\geq 10P_2^{AVG}, \\ P_1^{AVG}, P_2^{AVG}, \alpha_3 &\in [0, 1]. \end{aligned}$$

The set $\mathcal{P}$ is depicted on the right of Figure 7. It is easy to verify that this set does not have the free-disposal property. We verify that the first step of WF algorithm has no solution, hence water filling does not give the min-max allocation. On the other hand, a single iteration of MP gives us the min-max allocation on the set $\mathcal{X}$ which in this case is $(0.38, 0.62)$. We underline again that only due to the simplicity of the example, WF fails at the first step, and MP solves the problem in one step. In a more complex example WF might fail on any step whereas MP will again solve the problem. However, due to the simplicity of the presentation we give here only a 4 node example.

5 Conclusion

We have given a general framework that unifies several results on max-min and min-max fairness encountered in networking examples. We have extended the framework to account for new examples arising in mobile and peer-to-peer scenarios. We have elucidated the role of bottleneck arguments in the water-filling algorithm, and explained the relation to the free-disposal property; we have shown that the bottleneck argument is not essential to the definition of max-min fairness, contrary to popular belief. However, when it holds, it allows us to use simpler algorithms. We have given a general

purpose algorithm (MP) for computing the max-min fair vector whenever it exists, and showed that it degenerates to the classical water-filling algorithm, when free disposal property holds. The existence of a max-min fair vector is not always guaranteed, even on compact sets. We have found a class of compact sets on which max-min fairness does exist. The extension of the class to other useful cases (such as discrete sets [24]) remains to be studied. Finally, we have focused on centralized algorithms for calculating max-min and min-max fair allocations. It will be interesting to explore their distributed counterparts.

References

[1] A. Charny. An algorithm for rate allocation in a packet-switched network with feedback. *M.S. thesis*, MIT, May 1994.

[2] A. Mas-Colell, M. Whinston, J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[3] ATM Forum Technical Committee. "Traffic Management Specification - Version 4.0". *ATM Forum/95-0013R13*, February 1996.

[4] W. Bossert and J.A. Weymark. Utility in social choice. In S. Barbera, P.J. Hammond, and C. Seidl, editors, *Handbook of Utility Theory*. Kluwer Academic Publishers, 2004.

[5] M.A. Chen. Individual monotonicity and the leximin solution. *Economic Theory*, 15:353–365, 2000.

[6] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[7] D. Rubenstein, J. Kurose, D. Towsley. "The Impact of Multicast Layering on Network Fairness". *IEEE/ACM Transactions on Networking*, 10(2):169–182, Apr. 2002.

[8] E. Hahne. "Round-Robin Scheduling for Max-Min Fairness in Data Networks". *IEEE Journal on Selected Areas in Communications*, 9(7):1024–1039, Sept. 1991.

[9] F. P. Kelly, A.K. Maulloo and D.K.H. Tan. "Rate control in communication networks: shadow prices, proportional fairness and stability". *Journal of the Operational Research Society*, 49:237–252, 1998.

[10] H. Tzeng and K. Siu. "On Max-Min Fair Congestion Control for Multicast ABR Service in ATM". *IEEE Journal on Selected Areas in Communications*, 15(3):545–556, April 1997.

[11] J. Byers, et al. "A Digital Fountain Approach to Reliable Distribution of Bulk Data". In *ACM SIGCOMM '98*, September 2-4 1998.

[12] J. Mo, J. Walrand. "Fair end-to-end window-based congestion control". *IEEE/ACM Transactions on Networking*, 8(5):556–567, Oct. 2000.

[13] J. Ros and W. Tsai. "A Theory of Convergence Order of Maxmin Rate Allocation and an Optimal Protocol". In *INFOCOM'01*, pages 717–726, 2001.

[14] J.H. Chang and L. Tassiulas. "Energy Conserving Routing in Wireless Ad-hoc Networks". In *INFOCOM'00*, pages 22–31, 2000.

[15] L. Georgadis, et al. "Lexicographically Optimal Balanced Networks". In *INFOCOM'01*, pages 689–698, 2001.

[16] L. Tassiulas and S. Sarkar. "Maxmin Fair Scheduling in Wireless Networks". In *INFOCOM'02*, pages ?–?, 2002.

[17] S. Low and D. Lapsley. Optimization flow control, I:basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, December 1999.

[18] A.L. McKellips and S. Verdu. Maximin performance of binary-input channels with uncertain noise distributions. *IEEE Transactions on Information Theory*, 44(3):947–972, May 1998.

[19] P. Marbach. "Priority Service and Max-Min Fairness". In *INFOCOM'02*, pages ?–?, 2002.

[20] B. Radunović and J.-Y. Le Boudec. When power adaptation is useless or harmful.

[21] B. Radunović and J. Y. Le Boudec. Optimal power control, scheduling and routing in UWB networks. *IEEE Journal on Selected Areas in Communications*, September 2004.

[22] J. Rawls. *A Theory of Justice*. Harvard University Press, 1971.

[23] V. Rodoplu and T.H. Meng. Minimum energy mobile wireless networks. *IEEE J. Select. Areas Commun.*, 17(8):1333 – 1344, August 1999.

[24] S. Sarkar and L. Tassiulas. "Fair Allocation of Discrete Bandwidth Layers in Multicast Networks". In *INFOCOM'00*, pages 1491–1500, 2000.

[25] J. Van Leeuwen. Graph algorithms. In J. Van Leeuwen, editor, *Algorithms and Complexity*. Elsevier, 1992.

[26] M. Win and R. Scholtz. Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications. *IEEE Transactions on Communications*, 48(4):679–691, April 2000.

[27] Xiao Long Huang, Brahim Bensaou. "On Max-min Fairness and Scheduling in Wireless Ad-Hoc Networks: Analytical Framework and Implementation". In *Proceedings MobiHoc'01*, Long Beach, California, October 2001.

[28] Y. Hou, H. Tzeng, S. Panwar. "A Generalized Max-Min Rate Allocation Policy and Its Distributed Implementation Using the ABR Flow Control Mechanism". In *INFOCOM'98*, pages 1366–1375, 1998.

[29] Z. Cao and E. Zegura. "Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme". In *INFOCOM'99*, pages 793–801, 1999.

A Proofs

A.1 Proof of Existence of MMF

We first give an intuition on how we shall prove the theorem. We consider vector $\vec{x}$ that is leximin maximal on the set $\phi(\mathcal{X})$, and we want to prove that this is at the same time the max-min fair vector. The proof is done by contradiction. We assume that there exists a vector $\vec{y}$ that violates the definition of max-min fairness of vector $\vec{x}$. We will then construct vector $\vec{z}$ from $\vec{x}$ and $\vec{y}$ such that $\vec{z}$ is leximin-larger than $\vec{x}$, which will lead to contradiction. Function $\phi()$ is strictly increasing, hence there exists and inverse $\phi^{-1}()$, which is also strictly increasing. Although set $\phi(\mathcal{X})$ is not convex, set $\mathcal{X}$ is convex. Therefore, we will chose α such that vector $\vec{z}$, constructed as $\phi^{-1}(\vec{z}) = \alpha\phi^{-1}(\vec{x}) + (1-\alpha)\phi^{-1}(\vec{y})$, is leximin larger than $\vec{x}$.

Proof of Theorem 1: Let $\vec{x} \in \phi(\mathcal{X})$ be a vector such that for all $\vec{y} \in \phi(\mathcal{X})$ we have $\mathcal{T}(\vec{x}) \stackrel{lex}{\geq} \mathcal{T}(\vec{y})$. Such a vector exists according to proposition 2, since set $\mathcal{X}$ is compact. In order to prove the theorem, we proceed by contradiction, assuming that there exist $\vec{y}$ and an index $s \in \{1, ..., N\}$ such that $y_s > x_s$ and for all $t \in \{1, ..., N\}$, $x_t \leq x_s$ we have $y_t \geq x_t$. We then define a permutation $\pi : \{1, ..., N\} \rightarrow \{1, ..., N\}$ such that for all $i < j$, $\vec{x}_{\pi(i)} \leq \vec{x}_{\pi(j)}$, and either $x_{\pi(l)} < x_{\pi(l+1)}$ or $l = N$, where $l = \pi^{-1}(s)$. The last part of the requirement is important if there are several components of the vector that are equal to x_s, hence there are several permutations that maintain non-decreasing ordering. We then want s to be mapped by π to the largest such index: if $l = \pi^{-1}(s)$ than either $x_s < x_{\pi(l+1)}$ or l is the last index ($l = N$).

Next, let us define vector

$$\vec{z}(\alpha) = \phi(\alpha\phi^{-1}(\vec{x}) + (1-\alpha)\phi^{-1}(\vec{y})). \tag{21}$$

Although we cannot make a convex combination of $\vec{x}$ and $\vec{y}$ since set $\phi(\mathcal{X})$ is not convex, we can make a convex combination of $\phi^{-1}(\vec{x})$ and $\phi^{-1}(\vec{y})$ in the set $\mathcal{X}$ which is convex.

For $\alpha \in (0,1)$, $\vec{z}(\alpha)$ belongs to $\phi(\mathcal{X})$ due to convexity of $\mathcal{X}$. From (21) we have for all $\alpha \in (0,1), i \in \{1, ..., N\}$, $\min(\phi^{-1}(x_i), \phi^{-1}(y_i)) < \phi^{-1}(\vec{z}(\alpha)_i) < \max(\phi^{-1}(x_i), \phi^{-1}(y_i))$, hence $\min(x_i, y_i) < \vec{z}(\alpha)_i < \max(x_i, y_i)$, due to strictly increasing properties of functions ϕ_i and ϕ_i^{-1}. Also, for all i let us pick an arbitrary α_i satisfying

$$\alpha_i \in \begin{cases} \left(\frac{\phi_i^{-1}(x_s) - \phi_i^{-1}(y_i)}{\phi_i^{-1}(x_i) - \phi_i^{-1}(y_i)}, 1\right), & x_s \in [y_i, x_i), \\ [0, 1), & \text{otherwise} \end{cases}$$

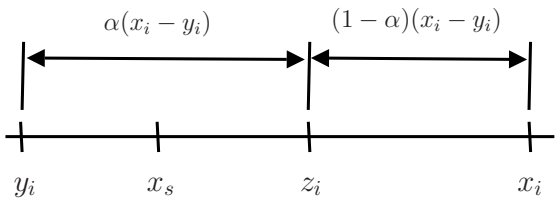

Figure 8: Choice of constants: suppose for simplicity that $\phi(x) = x$. We choose α such that: if for some i, $x_i > x_s$, then $z_i > x_s$ (as depicted on the figure), if $x_i \leq x_s$, then $y_i \geq z_i \geq x_i$, or else for any α, $z_i > x_s$.

and we call $\alpha_m = \max_i(\alpha_i)$ and $\vec{z} = \vec{z}(\alpha_m) \in \phi(\mathcal{X})$ (since $\alpha_m \in [0, 1)$). Intuitively, if for some i, $y_i < x_s$, we want to have $z_i > x_s$. If $x_i \leq x_s$ (including when $i = s$) we than by assumption have $y_i \geq x_i$, and we choose α such that we get $z_i > x_i$. Finally, if both $y_i > x_s, x_i > x_s$, than we can select any α and we will have $z_i > x_s$. A choice of constant is depicted in Figure 8.

We have chosen the highest of α_i, hence we now have that if $x_i \leq x_s$, than $z_i \geq x_i$, otherwise $z_i \geq x_s$. We also have $z_s > x_s$. From this, we derive the following property of the sorted vectors

$$\begin{array}{rcll} z_{\pi(i)} & \geq & x_{\pi(i)}, & \text{for } i < l, \\ z_{\pi(i)} & > & x_{\pi(l)}, & \text{for } i \geq l. \end{array}$$

We first notice that for all i, $z_{\pi(i)} \geq x_{\pi(1)}$, and as $\mathcal{T}(\vec{x}) \overset{lex}{\geq} \mathcal{T}(\vec{z})$ we conclude that $z_{(1)} = z_{\pi(1)} = x_{\pi(1)}$. Next, assuming that for some $i < l$ and for all $j < i$ we have $z_{(j)} = z_{\pi(j)} = x_{\pi(j)}$, then again as for all $j \geq i$, $z_{\pi(j)} \geq x_{\pi(i)}$, and $\mathcal{T}(\vec{x}) \overset{lex}{\geq} \mathcal{T}(\vec{z})$ we conclude that $z_{(i)} = z_{\pi(i)} = x_{\pi(i)}$. Hence, by induction we have proved that for all $i < l$ we have $z_{(i)} = z_{\pi(i)} = x_{\pi(i)}$. Finally, since for all $i \geq l$ we have $z_{\pi(i)} > x_{\pi(l)}$, hence $z_{(i)} > x_{\pi(l)}$ we necessarily have that $\mathcal{T}(\vec{z}) \overset{lex}{>} \mathcal{T}(\vec{x})$, which brings us to the contradiction.

Therefore, we conclude that a leximin maximal vector on a set $\mathcal{X}$ is also a max-min fair vector, and set $\mathcal{X}$ is max-min achievable.

q.e.d.

A.2 Proof of Correctness of MP

The idea of the proof is the following. We first want to show that in every step we decrease the size of S^n, that is $S^n \subset S^{n-1}$. From this we will conclude that the algorithm finishes in at most N steps. We then show that what remains in the set R^n once the algorithm stops (that is $S^n = \emptyset$), is the max-min fair allocation.

Before starting, we us define the following, as in Section 3.1:

$$\mathcal{X}^n = \{\vec{x} \in \mathcal{X}^{n-1} \,|\, (\forall i \in S^{n-1})\, x_i \geq T^n, (\exists i \in S^{n-1})\, x_i > T^n\} \quad (22)$$

$$\mathcal{R}^n = \{\vec{x} \in \mathcal{X}^{n-1} \,|\, (\forall i \in S^{n-1})\, x_i \geq T^n\} \quad (23)$$

$$S^n = \{i \in \{1, ..., N\} \,|\, (\forall \vec{x} \in \mathcal{X}^n)\, x_i > T^n\} \quad (24)$$

$$T^n = \max\{T \,|\, (\exists \vec{x} \in \mathcal{X}^{n-1})(\forall i \in S^{n-1}) x_i \geq T\}. \quad (25)$$

We also introduce several lemmas before proving the main theorem.

We first prove a lemma that illustrates the main idea of the algorithm, that in each steps we fix one by one the smallest coordinates of vectors to corresponding T values.

Lemma 1 *For all n where T^n exists, for all $x \in \mathcal{X}^n$, and for all $i \in S^{n-1} \setminus S^n$, we have $x_i = T^n$. Furthermore, if for all $m < n$ and for all $i \in S^{m-1} \setminus S^m$ we have $x_i = T^m$, for all $i \in S^n$ we have $x_i \geq T^n$ and for some $i \in S^m$ we have $x_i > T^m$, then $\vec{x} \in \mathcal{X}^n$.*

Proof: For $n = 0$ we have $S^0 = \{1, ..., N\}$ and the result is trivial. Let us select arbitrary $\vec{x} \in \mathcal{X}^n$, $n > 0$, and $i \in S^n \setminus S^{n+1}$. From (22) we have for all $i \in S^{n-1}$, $x_i \geq T^n$, and from (24) we have for all $i \notin S^n$, $x_i \leq T^n$. Hence we have $x_i = T^n$.

For the second part, we also proceed by induction. Obviously $\vec{x} \in \mathcal{X}^0$. Suppose, for some $m < n$, $\vec{x} \in \mathcal{X}^{m-1}$. Then it is easy to verify $\vec{x}$ satisfies (22) for $\mathcal{X}^m$, hence $\vec{x} \in \mathcal{X}^m$. By induction, we verify that also $\vec{x} \in \mathcal{X}^n$.

q.e.d.

Set $\mathcal{X}^n$ is not compact by definition and we do not know if the maximum in (25) exists. The following lemma is rather technical, and it proves the maximum always exists.

Lemma 2 *If set $\mathcal{X}$ is compact, then the maximum in (25) exists for all n.*

Proof: Since $\mathcal{X}^0 = \mathcal{X}$ is compact, the maximum exists for $n = 0$. Suppose $n > 0$. From (25) we have that

$$T^n = \max_{\vec{x} \in \mathcal{X}^{n-1}} \min_{i \in S^{n-1}} x_i.$$

Let us denote with $T' = \sup_{\vec{x} \in \mathcal{X}^{n-1}} \min_{i \in S^{n-1}} x_i$. T' always exists and $T' > T^{n-1}$. We proceed by contradiction. Suppose that the maximum does not exist hence $T' \notin \mathcal{X}^n$. By definition of T', for every integer $k > 0$ there exists $\vec{x}^k \in \mathcal{X}^{n-1}$ such that $T' - \min_{i \in S^{n-1}} x_i^k < 1/k$.

We next want to select a subsequence of sequence $\{\vec{x}^k\}$ such that for each member of the subsequence, the minimal component always has the same index, denoted by l. More formally, since S^{n-1} is a finite set, we can select $l \in S^{n-1}$ such that there is an infinite subsequence $\{\vec{x}^{k(l)}\} \in \mathcal{X}^{n-1}$ of sequence $\{\vec{x}^k\}$ where for all $k(l)$ we have $\arg\min_{i \in S^{n-1}} x_i^{k(l)} = l$.

This subsequence converges to $\vec{x}' = \lim_{k(l) \to \infty} \vec{x}^{k(l)}$. We have that $\vec{x}' \in \mathcal{X}$ due to compactness of $\mathcal{X}$. By construction, we also have for all $i \in S^{n-1}$, $x'_i \geq x'_l = T' > T^{n-1}$. By lemma 1 we have that for all $i \notin S^{n-1}, k_1(l), k_2(l)$, $\vec{x}_i^{k_1(l)} = \vec{x}_i^{k_2(l)} = \vec{x}'_i$, hence $\vec{x}' \in \mathcal{X}^{n-1}$, again by lemma 1. We see that vector $\vec{x}'$ satisfies all the conditions of (22), hence it belongs to $\mathcal{X}^n$ which leads to a contradiction.

q.e.d.

We next show another property of the coordinates of vectors in $\mathcal{X}^n$

Lemma 3 *For all n, $\vec{x}, \vec{y} \in \mathcal{X}^n$ and $t \in \{1, ..., N\}$ such that $x_t \leq T^n$, we have $y_t \geq x_t$.*

Proof: We prove lemma by induction over n. If $n = 1$, we have for all t, $x_t \geq T^1$ and $y_t \geq T^1$, hence for $x_t = T^1$, we have $y_t \geq x_t$. Next assume the above is true for $n - 1$. Suppose $x_t < T^n$. We then also have $x_t \leq T^{n-1}$, hence by the induction assumption we have $y_t \geq x_t$. Finally, if for some t, $x_t = T^n$ then $y_t \geq T^n$ or else we have a contradiction with the definition of T^n.

q.e.d.

Finally, we show that in each step we keep the max-min fair vector in $\mathcal{X}^n$ in order to show that in the last step, when we have a single point remaining, this point will indeed be the max-min fair one.

Lemma 4 *If $\vec{x}$ is max-min fair vector on $\mathcal{X}$ then for all n such that $\mathcal{X}^n \neq \emptyset$, $\vec{x} \in \mathcal{X}^n$. The same holds for $\mathcal{R}^n$.*

Proof: We prove lemma by induction. If $\vec{x} \notin \mathcal{X}^1$ then $\vec{x}$ is not leximin maximal, hence the contradiction. Let us next assume $\vec{x} \in \mathcal{X}^{n-1}$ and $\vec{x} \notin \mathcal{X}^n$, where $\mathcal{X}^n \neq \emptyset$. Then there exists $\vec{y} \in \mathcal{X}^n$ and $s \in S^n$ such that $y_s > x_s$. Also, by lemma 3, for all $t \in \{1, ..., N\}$ such that $x_t \leq T^n$, we have $y_t \geq x_t$. This contradicts the assumption that $\vec{x}$ is max-min fair which proves the lemma. Since $\mathcal{X}^n \subseteq \mathcal{R}^n$, we have the second claim.

q.e.d.

Now we are ready to prove the main theorem.

Proof of theorem 3: Let us call $\vec{x}$ max-min fair vector on $\mathcal{X}$. From lemma 2 we know that the minimum in (25) is achieved. Therefore, there exist $i^* \in S^{n-1}, \vec{x}^* \in \mathcal{X}^{n-1}$ such that $x^*_{i^*} = T^n$, and we have $i^* \notin S^n$, thus we proved $S^n \subset S^{n-1}$. We conclude that sequence $|S^n|$ decreases and we will have $S^n = \emptyset$ in at most N steps.

We also notice that for every $i \in \{1, ..., N\}$ there exists m such that $i \in S^{m-1}$ and $i \notin S^m$. From $i \in S^{m-1}$ we have that for all $\vec{x} \in \mathcal{X}^m$, $x_i \leq T^m$. From $i \in S^m$ we have that for all $x \in \mathcal{X}^{m-1}$ we have $x_i \geq T^m$ in the constraints for MP^m. Now as for all n, $\mathcal{X}^n \subseteq \mathcal{X}^{n-1}$ we have that for all $n \geq m$ and $\vec{x} \in \mathcal{X}^n$ we have $x_i = T^m$. Once we have $S^n = \emptyset$ it means that all components of vectors in $\mathcal{R}^n$ are fixed hence $|\mathcal{R}^n| = 1$. According to lemma 4, this single vector in $\mathcal{R}^n$ is also max-min fair on $\mathcal{X}$.

q.e.d.

A.3 Proof of Equality of MP and WF

Proof of theorem 4: Let us call T^1_{MP} the solution to the MP^1 and T^1_{WF} the solution to the WF^1. T^1_{WF} is obviously achievable in MP^1 so we have $T^1_{MP} \geq T^1_{WF}$. Suppose that $T^1_{MP} > T^1_{WF}$. This implies that for all $s \in \{1, ..., N\}$ we have $(\vec{x}^1_{MP})_s \geq T^1_{MP}$. Due to the free-disposal property, we can successively decrease each of the components of $\vec{x}$ larger than corresponding lower bound in $\vec{m}$, until arriving to a vector $\vec{y}$, $y_i = \max(T^1_{MP}, m_i)$. This vector is feasible, which contradicts the optimality of T^1_{WF}. The same reasoning can be applied to the successive algorithm steps, by decreasing the dimension of the feasible set.

q.e.d.

A.4 Proof of Convergence of Utility Fairness to MMF

Proof of theorem 2: We first prove that if $\vec{x}^*$ is an accumulation point for the set of vectors $\vec{x}^m$, then $\vec{x}^*$ is the max-min fair vector, and to prove that we assume the contrary, that exists a vector $\vec{y}$ and a corresponding index $s = s(y)$ such that $y_s > x_s$ and for all $t \in S$ $x_t \leq x_s \Rightarrow x_t \leq y_t$. Let us call $T_s(x) = \{t \in \{1, ..., N\} \,|\, x_t > x_s\}$. If set $T_s(x)$ is empty, we pick an arbitrary $\epsilon > 0$, else we call $t = \arg\min_{t \in T_s(x)} x^*_t$ and $\epsilon = x^*_t - x^*_s$.

Next, similarly as in proof of theorem 1, we pick an arbitrary $\alpha < \epsilon/(y_s - x^*_s + x^*_t - y_t)$, and we call $\vec{z} = (1 - \alpha)\vec{x} + \alpha\vec{y}$. If $\epsilon_1 = z_s - x^*_s$ and $\epsilon_2 = x^*_t - z_t$, it is easy to verify that $\epsilon > \epsilon_1 + \epsilon_2$. Also, $\vec{z} \in \mathcal{X}$ due to convexity of $\mathcal{X}$.

Finally we pick an arbitrary $\delta > 0$, $\Delta > 0$ such that $\delta + \Delta < \epsilon_1$, and some n_0 such that for all $n \geq n_0$ and for all $u \in \{1, ..., N\}$ we have $|f_{m_n}(x^*_u) - f_{m_n}(x^{m_n}_u)| < \delta$. The choice of the above constants is depicted in Figure 9.

Now we show that for n large enough we have a contradiction with optimality of $\vec{x}^{m_n}$. Consider the expression A defined by

$$A = \sum_{t \in \{1, ..., N\}} \left(f_{m_n}(z_t) - f_{m_n}(x^{m_n}_t)\right).$$

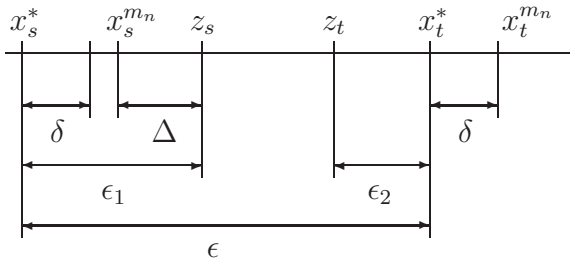

Figure 9: Choice of constants

From the optimality of $\vec{x}^{m_n}$ we have $A \leq 0$. Also

$$A \geq f_{m_n}(z_s) - f_{m_n}(x_s^{m_n}) + \sum_{t \in T_s(x)} \left(f_{m_n}(z_t) - f_{m_n}(x_t^{m_n})\right). \tag{26}$$

From the theorem of intermediate values, there exist numbers c_s^n and c_t^n for all $t \in T_s(x)$ such that

$$\begin{aligned} &x_s^{m_n} \leq c_s^n \leq z_s, \\ &f_{m_n}(z_s) - f_{m_n}(x_s^{m_n}) = f'_{m_n}(c_s^n)(z_s - x_s^{m_n}). \end{aligned}$$

and

$$\begin{aligned} &z_t \leq c_t^n \leq x_t^{m_n}, \\ &f_{m_n}(x_t^{m_n}) - f_{m_n}(z_t) = f'_{m_n}(c_t^n)(x_t^{m_n} - z_t). \end{aligned}$$

where f'_{m_n} is the right derivative of f_{m_n}. From the above we have

$$\begin{aligned} f_{m_n}(z_s) - f_{m_n}(x_s^{m_n}) &> \Delta f'_{m_n}(z_s), \\ f_{m_n}(x_t^{m_n}) - f_{m_n}(z_t) &< (\delta + \epsilon_2) f'_{m_n}(z_t). \end{aligned}$$

thus from eq. (26)

$$\begin{aligned} A &> \Delta f'_{m_n}(z_s) - (\delta + \epsilon_2) M f'_{m_n}(z_t) \\ &> f'_{m_n}(z_s) \left(\Delta - (\delta + \epsilon_2) M \frac{f'_{m_n}(z_t)}{f'_{m_n}(z_s)}\right) \end{aligned}$$

where M is the cardinality of set $T_s(x)$. Now from eq. (5), the last term in the above equation tends to Δ as n tends to infinity. Now $f'_{m_n} > 0$ from our assumption thus, for n large enough, we have $A > 0$, which is the required contradiction.

The set of vectors $\vec{x}^m$ is in a compact set $\mathcal{X}$ hence it has at least once accumulation point. Since each accumulation point is also max-min fair, and from property 1 we have the uniqueness of a max-min fair vector we conclude that the set of vectors $\vec{x}^m$ has a unique accumulation point hence

$$\lim_{m \to +\infty} \vec{x}^m = \vec{x}^*.$$

q.e.d.