Fairness in Capacitated Networks: a Polyhedral Approach

Gábor Rétvári, József J. Bíró, Tibor Cinkler

High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
H-1117, Magyar Tudósok körútja 2., Budapest, Hungary
E-mail: {retvari, biro, cinkler}@tmit.bme.hu

***Abstract*—The problem of fair and feasible allocation of user throughputs in capacitated networks is investigated. The main contribution of the paper is a novel geometric approach, which facilitates to generalize several throughput allocation strategies, most importantly max-min fairness, from the traditional "fixed-path" model to a more versatile, routing-independent model. We show that the set of throughput configurations realizable in a capacitated network makes up a polyhedron, which gives rise to a max-min fair allocation completely analogous to the conventional one. An algorithm to compute this polyhedron is also presented, whose viability is demonstrated by comprehensive evaluation studies.**

I. Introduction

In this paper we address the problem of allocating scarce resources in a network so that every user gets a fair share, for some reasonable definition of fairness. For example, a fair allocation would be such that every user gets the same share, and the allocation is maximal in the sense that there does not exist any larger, even and feasible allocation. We shall focus on the fair allocation problem that arises most often in networking: compute a fair rate at which users can send data in a telecommunications network, whose links are of limited capacity.

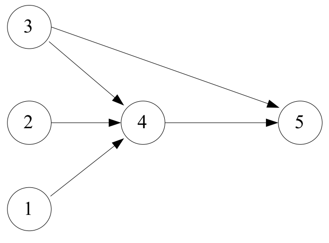
Perhaps the most practical way to understand the context of this paper is through an example. Consider the simple directed network of Fig. 1a, and suppose that there are 3 source-destination pairs (or users or commodities): $(1, 5)$, $(2, 5)$ and $(3, 5)$ (see Fig. 1b). All the edge capacities are uniformly 1. Now, the task is to compute a transmission rate (or throughput, for short) for each user that is on the one hand feasible (so it can be routed in the network without violating the edge capacities) and, on the other hand, satisfies some fairness criteria. For example, according to the above naive interpretation of fairness, we would allocate $\frac{1}{2}$ amount of throughput for each user. This allocation is certainly feasible and gives even share to each user, and it is also maximal in this regard.

Amongst the many different definitions of fairness perhaps the most prevailing one is max-min fairness. A max-min fair allocation is, roughly speaking, such that we cannot increase the throughput of any of the users without decreasing the throughput of some other user, which is already smaller [1]. Max-min fairness is a simple yet powerful fairness criterion, and consequently it has grown to be an essential ingredient in diverse fields of networking, like flow control protocols [2], bandwidth sharing in ATM networks [3], etc. For further analysis of the related extensive literature, the reader is referred to [4] and [5]. For some economical aspects, see [6].

Max-min fairness is most easily described in a network model, where a single path is assigned to each user and this path remains fixed during the lifetime of the communication. Here, the task is to compute a rate at which users can send data to their path, so that the allocation is max-min fair and neither of the edges gets overloaded. A very useful tool to solve this problem is the notion of bottlenecks [4]. A bottleneck edge, with respect to a certain user, is an edge with the properties that *(i)* it is filled to capacity and *(ii)* the user has the maximum throughput amongst the users whose path traverses the edge. Bottlenecks are very tightly coupled with max-min fairness, for it can be shown that an allocation of throughputs is max-min fair over some fixed single-path routing, if and only if all the users have a bottleneck edge.

From the practical standpoint, the importance of this *bottleneck argumentation* is multi-faceted. First, as the name suggests, bottlenecks point to certain shortages of resources in the network that, given the selected set of paths, constrain the fair allocation. Additionally, bottlenecks substantiate a fast algorithm, the so called *water-filling algorithm*, to find a max-min fair allocation [4]: we increase the throughput of the users at the same pace until an edge gets saturated. Then we fix the throughput of the users whose path passes through the saturated edge and keep on increasing others. The procedure is repeated until eventually a bottleneck is found for each user, and the allocation obtained is guaranteed to be the max-min fair allocation.

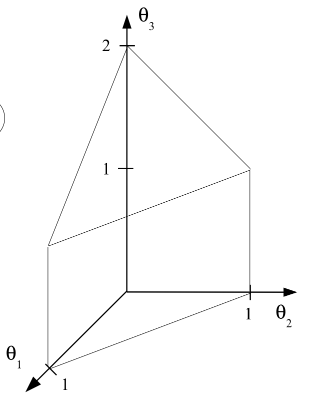
Assume that, in the sample network of Fig. 1, path $1 \rightarrow 4 \rightarrow 5$ is assigned to user $(1, 5)$, path $2 \rightarrow 4 \rightarrow 5$ to user $(2, 5)$, and the direct path $3 \rightarrow 5$ to user $(3, 5)$, respectively. Then, the edge that first becomes saturated as the water-filling algorithm proceeds is edge $(4, 5)$, which becomes the bottleneck edge for users $(1, 5)$ and $(2, 5)$. So the throughput of both of these users is fixed at $\frac{1}{2}$, and only the throughput of user $(3, 5)$ is increased any more. This, in turn, gets saturated at a throughput of 1 unit.



(a) topology

Users:
$(s_1, d_1) = (1, 5)$
$(s_2, d_2) = (2, 5)$
$(s_3, d_3) = (3, 5)$

(b) parameters



(c) feasible allocations

Figure 1: A sample network and the set of throughputs realizable in it. All edge capacities are equal to 1. There are 3 source-destination pairs $(1, 5)$, $(2, 5)$ and $(3, 5)$, whose throughput is denoted by θ_1, θ_2 and θ_3, respectively.

The final max-min fair allocation is represented by the vector $[\frac{1}{2}, \frac{1}{2}, 1]$, using the order of users set out above.

Highlighting its usefulness, several extensions and ramifications of max-min fairness have come to existence throughout the years (min-max fairness [7], weighted max-min fairness [4], max-min utility fairness [8] and various combinations of these). Since all of these concepts can be traced back to the unweighted case [7] and a respective bottleneck argumentation, analogous to the one above, can always be made, we shall not address these concepts hereafter. Furthermore, the limitation that each user employs one single, fixed path can also be weakened somewhat without invalidating the bottleneck argumentation: it is permitted to assign several paths to each of the users, provided that the splitting ratios at branching nodes, and the paths themselves, remain fixed.

Curiously, the actual selection of paths influences the emergent max-min fair allocation to a great extent. For example, if the path of user $(3, 5)$ is changed to $3 \rightarrow 4 \rightarrow 5$, then the max-min fair throughput vector turns to $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$. If we assign both paths $3 \rightarrow 5$ and $3 \rightarrow 4 \rightarrow 5$ to user $(3, 5)$ with the restriction that traffic must be split equally between the two paths, then the max-min fair allocation ends up to be $[\frac{2}{5}, \frac{2}{5}, \frac{2}{5}]$. Apparently, different routings give rise to different max-min fair allocations, which is somewhat unnatural since, after all, it is the network that determines feasible allocations. Accordingly, we should first compute a max-min fair allocation that is only dependent on the network itself, and only after this we should pick a routing that realizes it. Below, we shall refer to this problem as *the general max-min fair allocation problem*, and all the former incarnations will be called *fixed-path max-min fairness problems*.

Recently, several attempts have been made to address the general max-min fair allocation problem using lexicographical optimization [9], [7]. These works are based on the observation that a max-min fair allocation is lexicographically maximal above the set of all feasible routings, so successive linear programming can be invoked to obtain it. The approach taken in [7] is, however, more general: it not only states the existence and the uniqueness of a max-min fair allocation over *any* compact and convex set, which the set of all possible routings certainly is, but it also gives an algorithm, called Max-min Programming, to compute it over any such set. While these excellent works provide adequate quantitative treatment, an in-depth qualitative analysis, which would reveal the intricate relationship between the specifics of a network and the emergent max-min fair allocation, is still absent in the literature. For instance, it is still not clear whether or not the bottleneck argumentation and, consequently, the water-filling algorithm generalize from the fixed-path model to the routing-independent, generic model, and if yes, then in what particular form bottlenecks arise. These questions have gone mostly unresolved so far, albeit their relevance has been very clearly pointed out [7, Section "When bottleneck and water-filling become less obvious"].

In this paper we offer affirmative answers to these important questions. After a quick roundup on the notation in Section II, we shall introduce a novel polyhedral description of the throughput allocations realizable in a network (see Section III). This polyhedral description is so concise that for simple networks we can as well easily visualize it (see Fig. 1c) and it allows us to gain interesting new insights into the nature of capacitated networks and the throughput allocation strategies they give rise to. In particular, in Section IV we study non-dominated and Pareto-efficient allocations, basic concepts of which more sophisticated ones can be constructed. Then, we turn to max-min fair allocations and we reveal how the bottleneck argumentation extends to the routing-independent case. Finally, some related algorithmic questions are discussed and then Section V concludes the paper.

Reading this paper requires a minimal understanding of the theory of network flows and linear programming. To make it more accessible even to the less mathematically inclined, all the proofs are deferred to the Appendix. For a good introductory material on polyhedra, the reader is referred to [10].

II. Preliminaries

In this section, we present the most important notations and conventions we shall use throughout this paper. A vector will be denoted by a lowercase letter. Most of the time, the ith coordinate of a vector v will be referred to as v_i, but in some cases, to stress that we are dealing with a specific coordinate, we shall use the notation $(v)_i$. What now follows is a list of the notation we shall use in the sequel:

- $G(V, E)$: a directed graph, with the set of nodes V ($|V| = n$) and the set of directed edges E ($|E| = m$).
- u: the column m-vector of edge capacities.
- $(s_k, d_k) : k \in \mathcal{K}$, $\mathcal{K} = \{1, \ldots, K\}$: the set of source-destination pairs (users or commodities).

Note that the graph $G(V, E)$, the edge capacities u and the set of source-destination pairs $(s_k, d_k) : k \in \mathcal{K}$ together describe a *network*, which will be referred to as G_u for brevity. Hereafter, we shall only deal with networks that satisfy certain, rather mild, regularity conditions:

Definition 1: A network G_u is *regular*, if

- a path exists in G_u from s_k to d_k for each $k \in \mathcal{K}$ and
- all edge capacities are finite and strictly positive.

It is easy to see that any network can be reduced to a collection of regular networks by eliminating edges with zero capacity and fixing the throughput of the un-connected users at zero. The further notation goes on as follows:

- e_i: the canonical unit vector (of proper size implied by the context) with 1 in the position corresponding to the ith coordinate and all zero otherwise.
- $\mathbf{1}$: an all-one vector of proper size.
- $\mathcal{P}_k$: the set of all directed paths from s_k to d_k in $G(V, E)$ for some $k \in \mathcal{K}$.
- Δ_k: an $m \times |\mathcal{P}_k|$ matrix. The column corresponding to path $P \in \mathcal{P}_k$ holds the path-arc incidence vector of P.
- f_k: a column vector of path-flows, whose coordinate corresponding to path $P \in \mathcal{P}_k$ denotes the amount of flow sent by user k to path P.
- f: a column-vector of f_ks: $f = [f_1, f_2, \dots, f_K]$. In fact, f represents a *routing* in G_u.
- θ_k: the throughput of some user $k \in \mathcal{K}$, that is, the aggregate flow that flows from s_k to d_k. The vector of throughputs is a column K-vector θ.
- $\beta\theta \le b$: an inequality constraining the set of throughputs, where β is a row K-vector and b is a scalar. An inequality $\beta\theta \le b$ is *valid* for some set T, if $\forall\theta \in T : \beta\theta \le b$.
- $T(G_u)$: the set of throughputs realizable in the network G_u, subject to edge capacity constraints.
- $\mathcal{S}$: a separating edge set, that is, a set of edges $\mathcal{S} \subseteq E$ whose removal from the network would destroy all the directed s_k to d_k paths for at least one user $k \in \mathcal{K}$.
- $\mathcal{K}_\mathcal{S}$: the set of users disconnected by some separating edge set $\mathcal{S}$.

III. The Throughput Polytope

The central problem we investigate in this paper is to determine a feasible and fair allocation of user throughputs in a capacitated network, independently of paths fixed beforehand in any ways. A plausible way to attack this problem would be to describe the entire set of possible flow routings f and throughput allocations θ as a giant set and then search for the fair allocation in this very set. Consider the following formulation:

$$M(G_u) = \{[f, \theta] : \quad \textstyle\sum_{k\in\mathcal{K}} \Delta_k f_k \le u \tag{1}$$
$$\mathbf{1} f_k = \theta_k \qquad \forall k \in \mathcal{K} \tag{2}$$
$$f_k \ge 0, \theta_k \ge 0 \qquad \forall k \in \mathcal{K}\ \} \tag{3}$$

Readers more proficient in network flow theory might find this formulation familiar, since $M(G_u)$ is in fact the set of feasible solutions of the family of *multicommodity flow problems*. Here (1) requires that, for all edges, the sum of all path-flows routed to the edge does not exceed the capacity of that edge; (2) produces the throughput for each user by summing up the flow traveling along each of its paths; and (3) requires the flows and throughputs to be non-negative.

While using $M(G_u)$ to deduce a routing-independent fair allocation is clearly viable (see e.g. [9]), it is unfortunate in many regards, most notably because $M(G_u)$ usually has a plethora of problem variables, the majority of which completely redundant. This is because, at the moment, we are only interested in a fair throughput allocation but not in the way this allocation is accommodated in the network, apart from the requirement that the allocation must be realizable by some legitimate routing. Therefore, instead of studying the full-fledged set $M(G_u)$ one might rather choose to eliminate the path-flow variables f all together from the description. The emergent set, denoted hereafter by $T(G_u)$, contains all the possible throughput allocations feasible in G_u:

Definition 2: $T(G_u) = \{\theta : \exists f$ so that $[f, \theta] \in M(G_u)\}$.

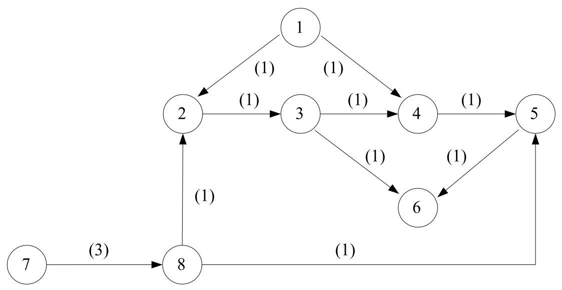
For the sample network of Fig. 1a, the corresponding set of feasible throughput allocations is depicted in Fig. 1c. To obtain it, we reason as follows. Let θ_1 denote the throughput of user $(1, 5)$, θ_2 of user $(2, 5)$ and θ_3 of user $(3, 5)$, respectively. Since we can not push more flow than 1 via the edge $(4, 5)$, which is traversed by all the potential paths of user $(1, 5)$ and $(2, 5)$, we have that $\theta_1 + \theta_2 \le 1$. Furthermore, after routing 1 unit of flow of user $(3, 5)$ along the edge $(3, 5)$, every additional ϵ units of flow of this user have to traverse edge $(4, 5)$, decreasing the aggregate throughput remaining available to user $(1, 5)$ and $(2, 5)$ by exactly ϵ units. So, $\theta_1 + \theta_2 + \theta_3 \le 2$. It can be shown that these inequalities, together with the restriction that the throughputs are non-negative, give rise to a complete and irredundant description of the set of throughputs realizable in the network of Fig. 1a:

$$T(G_u) = \{[\theta_1, \theta_2, \theta_3] : \quad \theta_1 + \theta_2 \le 1 \tag{4}$$
$$\theta_1 + \theta_2 + \theta_3 \le 2 \tag{5}$$
$$\theta_1, \theta_2, \theta_3 \ge 0 \quad \} \tag{6}$$

Observe how all the constraints turned out to be linear. Sets of similar kind are called polyhedra, which might be familiar as these are exactly the geometric objects that underlie linear programming [10]. A polyhedron is basically an intersection of finitely many halfspaces, and as such, closed and convex. Additionally, a bounded polyhedron is called a *polytope*. The result below reveals that the set $T(G_u)$ is not coincidentally polyhedral in our example.

Proposition 1: $T(G_u)$ is a polyhedron. Provided that G_u is regular, $T(G_u)$ is a polytope.

Henceforward, we shall only deal with regular networks, and so we shall refer to $T(G_u)$ as the *throughput polytope*. But not just that $T(G_u)$ is a polytope with "nice" properties like convexity and compactness, it has yet another interesting quality that makes it even more attractive to work with: observe that in the formulation (4)–(6), every coefficient and also the right-hand-side of all the inequalities are non-negative. This, as the next result claims, is again not coincidental, but instead a very important general property of throughput polytopes, one that we shall exploit in the sequel to study fair

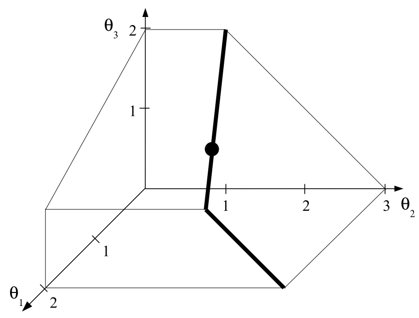


(a) topology (edge capacity is marked in parentheses)

Users:
$(s_1, d_1) = (1, 6)$
$(s_2, d_2) = (7, 8)$
$(s_3, d_3) = (7, 5)$

Max-min fair allocation:
$[\theta_1, \theta_2, \theta_3] = [\frac{4}{3}, \frac{5}{3}, \frac{4}{3}]$

(b) parameters



(c) $T(G_u)$

Figure 2: Another sample network and the associated set of feasible throughputs. All edge capacities are equal to 1 except for edge $(7, 8)$, whose capacity is 3. Pareto-efficient allocations are marked by bold lines, while the max-min fair point is denoted by a small circle.

allocations arising in $T(G_u)$.

Proposition 2: For a regular network G_u, the corresponding throughput polytope can always be transformed to the following standard form:

$$T(G_u) = \{\theta \geq 0 : \beta_i \theta \leq b_i, \quad \forall i \in \mathcal{I}\} ,$$

where $\mathcal{I}$ is a (finite) index set and for each $i \in \mathcal{I}$ it holds that $\beta_i \geq 0$ and b_i is a positive scalar.

Since the network of Fig. 1a is not complex enough to demonstrate anything but the most basic ideas, next we introduce a bit more complicated sample network (see Fig. 2). The users are as follows: let $(s_1, d_1) = (1, 6)$, $(s_2, d_2) = (7, 8)$ and $(s_3, d_3) = (7, 5)$. Computing the throughput polytope by hand, while doable, is a bit more involving for such nontrivial networks, therefore we shall discuss an algorithm to automate this process in Section IV-D. Running this algorithm on the network of Fig. 2a yields the standard form of its throughput polytope:

$$\begin{aligned} T(G_u) = \{[\theta_1, \theta_2, \theta_3] : \qquad & \theta_1 \leq 2 && (7)\\ & \theta_2 + \theta_3 \leq 3 && (8)\\ & \theta_1 + 2\theta_3 \leq 4 && (9)\\ & \theta_1, \theta_2, \theta_3 \geq 0 \quad \} && (10)\end{aligned}$$

IV. Throughput Allocation Strategies

In the previous section, we introduced the throughput polytope as the lower-dimensional projection of the set of all feasible routings and throughput allocations, with the path-flow variables eliminated. In this section, we shall study efficient and fair allocation strategies arising in a network by means of the corresponding throughput polytope.

A. Efficient throughput allocations

When deciding which particular throughput allocation to offer for the users, the first requirement one has to consider is that the allocation must be *feasible*. Feasibility is, however, easy to assure in our model: one might choose whatever $\theta \in T(G_u)$ and the construct then automatically assures that this θ will be realizable by some legitimate routing. The second requirement is that θ must be *efficient*. By efficiency we mean that the throughput allocation, or more precisely the routing that realizes it, must utilize the valuable network resources in an effective way. While there are many different notions of efficiency, below we concentrate only on the two most prevalent ones: non-dominated and Pareto-efficient allocations.

A user, say, k, is non-dominated at some allocation $\theta_0 \in T(G_u)$, if either it is not possible to increase its throughput from $(\theta_0)_k$ without violating edge capacities, or increasing it is only possible at the cost of decreasing the throughput of some other user [6]. More formally:

Definition 3: A user $k \in K$ is dominated at $\theta_0 \in T(G_u)$, if $\exists \epsilon > 0$ so that $\theta_0 + \epsilon e_k \in T(G_u)$. Otherwise, k is non-dominated at θ_0. Furthermore, an allocation of throughputs θ_0 is non-dominated, if at least one user is non-dominated at θ_0. Otherwise, θ_0 is dominated.

In the network of Fig. 2, the point $\theta_0 = [2, 0, 1]$ is for instance non-dominated, since user $(1, 6)$ is at its maximum flow. In other words, as we try to increase the throughput of $(1, 6)$ from θ_0, a constraint in $T(G_u)$ becomes active (in this case this constraint is $\theta_1 \leq 2$), which inhibits any further increase. Besides user $(1, 6)$, $(7, 5)$ is also non-dominated at θ_0, since its throughput can only be increased at the cost of decreasing the throughput of $(1, 6)$. Here, a constraint causing non-dominatedness is for example $\theta_1 + 2\theta_3 \leq 4$. Finally, we see that user $(7, 8)$ is dominated at θ_0, since we can by no means find a valid inequality that would block it at θ_0. This suggests that θ_0 is only partially efficient, since there exist other feasible allocations in the network delivering strictly higher throughput to at least one user and, at the same time, not decreasing the throughput received by anyone else. Such a more efficient allocation is for instance $\theta = [2, 2, 1]$

The above reasoning helps identify where exactly dominated and non-dominated allocations are located in the throughput polytope. Since at least one constraint is binding at any non-dominated point, these allocations are exactly those residing at any one of the *facets* of $T(G_u)$ (note that, for simplicity, we do not count the constraints $\theta_k \geq 0$ as facets here). In contrast, dominated throughput vectors lie in the *interior* of $T(G_u)$.

The following important result, which we shall often put to use in the sequel, relates a non-dominated allocation to a very special inequality (and a dominated one to the lack thereof).

Theorem 1: Let $\theta_0 \in T(G_u)$ be non-dominated. Now, some set of users $\mathcal{N} \subseteq \mathcal{K}$ is non-dominated, and $\mathcal{K} \setminus \mathcal{N}$ is dominated at θ_0, if and only if there exists an inequality $\beta\theta \le b$ so that:

i) $\beta\theta \le b$ is valid for $T(G_u)$ and $\beta \ge 0$
ii) $\beta\theta_0 = b$
iii) $(\beta)_k > 0$ if and only if $k \in \mathcal{N}$

The proof of the theorem (see the Appendix) is based on the observation that if one sums up all the constraints of $T(G_u)$ that are binding at θ_0, then a valid inequality is obtained that satisfies all the requirements of the theorem. In the case of the non-dominated point $\theta_0 = [2, 0, 1]$, this valid inequality would be the sum of the two binding constraints (7) and (9): $\theta_1 + \theta_3 \le 3$.

In their own right, non-dominated allocations do not make really much sense. Why we still discuss them is because non-dominatedness is exactly the essential building block of which we can construct more complex and more efficient allocation strategies. The first such concept we discuss here is Pareto-efficiency.

A Pareto-efficient allocation is such that "there is no way to make any person better off without hurting anybody else" [6]. That is, for a Pareto-efficient allocation of throughputs it holds that any user is either at its maximum flow, so the throughput can not be increased at all, or otherwise increasing it is only possible at the expense of decreasing the throughput of some other user. It is then easy to dissect Pareto-efficiency to non-dominatedness.

Definition 4: An allocation of throughputs $\theta_0 \in T(G_u)$ is (strictly) Pareto-efficient if all the users are non-dominated at θ_0.

To identify the faces of $T(G_u)$ that contain Pareto-efficient allocations, we reason as follows. Being non-dominated, a Pareto-efficient vector must reside at some facet of $T(G_u)$. In fact, we expect it to reside at certain intersections of the facets, since Pareto-efficiency is generally a stronger property than non-dominatedness. And indeed, applying directly Theorem 1 yields the following characterization:

Corollary 1: Some $\theta_0 \in T(G_u)$ is Pareto-efficient, if and only if there exists an inequality $\beta\theta \le b$, so that:

i) $\beta\theta \le b$ is valid for $T(G_u)$ and $\beta \ge 0$
ii) $\beta\theta_0 = b$
iii) $\forall k \in \mathcal{K} : (\beta)_k > 0$

Corollary 1 suggests a simple algorithm to search for a Pareto-efficient allocation in $T(G_u)$: taking some random ordering of the users, increase the throughput of the users one by one as long as it is possible. Eventually, all the users will be blocked so a valid inequality $\beta\theta \le b$ with all strictly positive $(\beta)_k$ coordinates must be binding at the resultant point. In the case of the network of Fig. 2a, the two line segments joining the points $[2, 3, 0]$ and $[2, 2, 1]$, respectively $[2, 2, 1]$ and $[0, 1, 2]$, contain all the Pareto-efficient points (see the bold line segments in Fig. 2c).

B. Max-min fair allocations

Besides ensuring feasibility and efficiency, a throughput allocation strategy must also guarantee that network resources are arbitrated between the users in an equitable and rightful manner or, in other words, it must be *fair*. From the aspect of fairness, Pareto-efficiency is a somewhat weak, though clearly desirable criterion. Desirable, because it avoids the wastage of resources non-dominatedness generally allows for. Weak, because Pareto-efficiency permits allocations where one user gets everything, which is not really fair (observe for instance that the allocation $[0, 0, 2]$ is Pareto-efficient in the network of Fig. 1). The concept of max-min fairness is based on the idea to pick the "fairest" Pareto-efficient allocation. Here, fairness means that any throughput increase must be at the cost of a decrease of some already smaller throughput. Formally:

Definition 5: Some $\theta_0 \in T(G_u)$ is max-min fair, if $\forall\theta \in T(G_u) : (\theta)_k > (\theta_0)_k \Rightarrow \exists l \in \mathcal{K} \setminus \{k\}$, so that $(\theta)_l < (\theta_0)_l$ and $(\theta_0)_l \le (\theta_0)_k$.

It is by far not evident whether or not this definition makes sense in the case of $T(G_u)$ or, in fact, how many max-min fair allocations it yields. Though, the following claim states that the notion of max-min fairness over $T(G_u)$ is well-defined:

Proposition 3: Let G_u be a regular network. Then there exists a max-min fair allocation over $T(G_u)$, and it is unique.

Being now safe that the general max-min fair allocation problem is soluble, we now move on to investigate how to actually compute that solution. In the case of non-dominated and Pareto-efficient allocations, it turned out really beneficial to characterize the conforming vectors in terms of valid inequalities. This characterization helped localize non-dominated and Pareto-efficient allocations in $T(G_u)$, and also suggested a simple algorithm to compute one. Below we again take this route, however our characterization now involves not just one but exactly K valid inequalities (one for each user).

Corollary 2: Some $\theta_0 \in T(G_u)$ is max-min fair, if and only if for each $k \in \mathcal{K}$ there exists an inequality $\beta\theta \le b$, the so called *bottleneck inequality*, such that:

i) $\beta\theta \le b$ is valid for $T(G_u)$ and $\beta \ge 0$
ii) $\beta\theta_0 = b$
iii) $\forall l \in \mathcal{K} : (\beta)_l > 0$ if and only if $(\theta_0)_l \le (\theta_0)_k$

Again, consult the Appendix for the proof.

What is remarkable in this result is that bottleneck inequalities work very much like bottleneck edges in the fixed-path model (hence the name). With this analogy in mind we could rephrase Corollary 2 as: *an allocation of throughputs is max-min fair in the generic sense, if and only if all users have a bottleneck (inequality).* This formulation is exactly the same as the one given for the fixed-path model, only the definition of bottlenecks differs somewhat. Interestingly, the analogy goes even further, since not just bottlenecks but the water-filling algorithm too extends to the general max-min fair allocation problem. Recall that the water-filling algorithm is based on the idea to generate a bottleneck for at least one user in every iteration, no matter in which form bottlenecks are defined. Provided that the bottlenecks arise in the form of a bottleneck inequality, Corollary 2 guarantees that what we eventually obtain by running the water-filling algorithm on the throughput polytope is exactly the max-min fair allocation. Thus, the second important consequence of this theorem is that *the water-filling algorithm is correct to search for a max-min fair allocation over $T(G_u)$*.

Consider the network of Fig. 2 and execute the water-filling algorithm. As the first step, increase the throughput of all the users at the same pace. This amounts to, starting from the origin, moving along the direction $[1, 1, 1]$ as long as some of the users gets blocked. This occurs at the point $[\frac{4}{3}, \frac{4}{3}, \frac{4}{3}]$, where the constraint $\theta_1 + 2\theta_3 \leq 4$ becomes active. As a matter of fact, this constraint will be the bottleneck inequality for users $(1, 6)$ and $(7, 5)$. The only user that remains dominated at this point is $(7, 8)$, whose throughput can be increased to $\frac{5}{3}$. The resultant allocation, $\theta_0 = [\frac{4}{3}, \frac{5}{3}, \frac{4}{3}]$ is max-min fair. To obtain the bottleneck for the last user, $(7, 8)$, sum up the two constraints binding at θ_0, which yields $\theta_1 + \theta_2 + 3\theta_3 \leq 7$.

The final question that remained to be answered is that, once we computed the max-min fair allocation θ_0, how to obtain a routing that realizes it. That is, we need to find path-flows $f : [\theta_0, f] \in M(G_u)$. This amounts to solving a multicommodity flow problem over the constraint set (1)–(3) with the throughput variables fixed at θ_0, which can be done in polynomial time [11]. The computed path-flows will then supply a set of forwarding paths and a rate at which users have to distribute their traffic to those paths. In the case of Fig. 2, user $(1, 6)$ must split its traffic evenly between the paths $1 \to 2 \to 3 \to 6$ and $1 \to 4 \to 5 \to 6$; user $(7, 8)$ must transmit over the direct link; while user $(7, 5)$ has to realize a traffic splitting ratio of $1 : 3$ between the paths $7 \to 8 \to 2 \to 3 \to 4 \to 5$ and $7 \to 8 \to 5$. This routing, once established in the network, will automatically realize the max-min fair throughput allocation θ_0 using the exact same distributed flow control and queuing techniques as in the fixed-path model [5].

C. A bottleneck argumentation

So far, we have shown how the concept of bottlenecks extend from the fixed-path max-min fairness problem to the generic case. Analogously to the traditional model, we could obtain an "if and only if" relation between the existence of bottlenecks for each user and max-min fairness, which also guaranteed the correctness of the water-filling algorithm. Quite regrettably, however, our bottlenecks are currently defined in terms of valid inequalities, which, being more of a polyhedral concept than a network theoretical one, is not really descriptive. In this section, we translate this bottleneck argumentation to the more palpable concept of separating edge sets, whose properties show remarkable similarity to the properties of "bottleneck edges" in the fixed-path model.

In the heart of the fixed-path model there lies the notion of bottleneck edges. A bottleneck edge is one that blocks any increase in the throughput of the user it belongs to. This is because *(i)* it is filled up to capacity when we realize the max-min fair allocation, and *(ii)* the corresponding user has the maximum throughput amongst the users that might want to use that edge. This conventional interpretation fails in the generic model, since neither the set of paths nor the users of a particular edge are fixed.

A bottleneck edge blocks one particular, fixed path of some user. To block *all* paths we have to treat an entire set of edges, a so called separating edge set, which, when removed from the network, destroys all directed paths connecting the source to the destination node. This suggests the idea to search for the generalization of bottleneck edges in the form of *bottleneck separating edge sets*. What remained to be done is to translate the defining properties of bottleneck edges to separating edge sets.

Let θ_0 be max-min fair in a regular network G_u and choose some user $k \in \mathcal{K}$. Additionally, suppose that we have somehow found the corresponding bottleneck separating edge set $\mathcal{S}_k$ and let $\mathcal{K}_{\mathcal{S}_k} \subseteq \mathcal{K}$ denote the set of users, whose source node is separated away from the respective destination node by $\mathcal{S}_k$. First, we reformulate the following property of bottleneck edges: a user's throughput is maximal at its bottleneck edge amongst the ones that utilize that edge. But "utilizers" of separating edge sets are exactly the users that are separated away by it, so for $\mathcal{S}_k$ it must hold that:

Property 1: $l \in \mathcal{K}_{\mathcal{S}_k} \Leftrightarrow (\theta_0)_l \leq (\theta_0)_k$.

The second defining property of bottleneck edges is that they are always filled to capacity when we realize the max-min fair allocation and, furthermore, in the fixed-path model there is no way for the traffic of the blocked users to circumvent this bottleneck. We translate this property to separating edge sets as follows:

Property 2: For any routing f that realizes θ_0, it holds that

$$\forall\ (i, j) \in \mathcal{S}_k : \sum_{l \in \mathcal{K}_{\mathcal{S}_k}} \sum_{P \in \mathcal{P}_l : (i,j) \in P} (f_l)_P = u_{ij}$$

In words, Property 2 insists that a bottleneck separating edge set is always saturated by the flow of the users separated away by it, no matter how we route the max-min fair allocation in the network. Therefore, any increase in the throughput of some user would decrease the throughput of some other user that utilizes the same (bottleneck) separating edge set, and consequently, whose throughput is already smaller (by Property 1), and this property is independent of the actual routing. Interestingly, these properties give rise to a bottleneck argumentation completely analogous to the conventional one:

Theorem 2: An allocation of throughputs $\theta_0 \in T(G_u)$ is max-min fair, if and only if each user has a bottleneck separating edge set exhibiting both Property 1 and Property 2.

For brevity, we do not detail here how to derive the bottleneck separating edge set from the bottleneck inequalities (the precise relation is revealed in the proof of Theorem 2 in the Appendix). It suffices to know that with users $(1, 6)$ and $(7, 5)$ we can associate the edge set $\mathcal{S}_1 = \mathcal{S}_3 = \{(2, 3), (4, 5), (8, 5)\}$ as a bottleneck. This edge set remarkably demonstrates the essence of our bottleneck argumentation: it separates away exactly users $(1, 6)$ and $(7, 5)$, and it is always saturated by the flow of these users no matter how we accommodate the max-min fair allocation in the network (in fact, there is only one option to choose from). Finally, the bottleneck separating edge set of user $(7,8)$ is $\mathcal{S}_2 = \{(7,8)\} \cup \mathcal{S}_3$. We kindly encourage the reader to verify that both Property 1 and Property 2 hold true for this edge set.

As a final remark, we note that the above bottleneck argumentation is valid for any arbitrary regular network, not just the simple and, coincidentally, acyclic ones we cited as examples. Additionally, it is noteworthy to mention that our bottleneck argumentation contains the conventional one as a special case. To see this, it is enough to restrict each user to one single path and observe that bottleneck separating edge sets degrade to the conventional bottleneck edges in this case.

D. An algorithm to compute the throughput polytope

As it turns out, the throughput polytope is a remarkably useful tool in solving the general max-min fair allocation problem. It can be used for computing the proportional fair or the utility fair allocations as well, but we might need to deploy non-linear programming [5]. In this case, its low-dimensionality and "nice" properties make the use of the throughput polytope appealing. Unfortunately, no viable algorithm to compute this construct is known for the authors at the moment. Moreover, we are not aware of any reasonable upper bound on the number of constraints that define $T(G_u)$ in the generic case. Therefore, the problem appears difficult since there is no real hope for a polynomially sized description of the output. The following algorithm implements the most plausible way to attack this problem: generate *all* the valid inequalities one by one until we finally obtain $T(G_u)$.

1) Initialize the algorithm with an empty set $\mathcal{X}$ that will hold the valid inequalities we find. Let $j = 1$ and $\mathcal{D} = \{0,1\}^K$.
2) For all $\beta \in \mathcal{D}$ whose elements are relative primes, generate a valid inequality $\beta\theta \leq b$. For a particular β this is done by solving the linear program:
$$\min\{wu : w\Delta_k \geq \mathbf{1}(\beta)_k \quad \forall k \in \mathcal{K}, w \geq 0\ \}$$
Add the resultant inequalities $\beta\theta \leq wu$ to $\mathcal{X}$.
3) After eliminating redundancy from $\mathcal{X}$, our current estimation of the throughput polytope is
$$\hat{T}(G_u) = \{\theta \geq 0 : \beta_i\theta \leq w_i u \quad [\beta_i, w_i] \in \mathcal{X}\}\ .$$
Now, we check if $\hat{T}(G_u) = T(G_u)$. In fact, it is enough to check $\hat{T}(G_u) \subseteq T(G_u)$, as $\hat{T}(G_u) \subset T(G_u)$ is impossible because all the inequalities in $\mathcal{X}$ are valid. So we verify whether all extreme points of $\hat{T}(G_u)$ are feasible in G_u by using the double-description method [12] to compute the set of extreme points $V(\hat{T}(G_u))$ and then checking:
$$\forall\theta_e \in V(\hat{T}(G_u)) : \exists f_e \text{ so that } [f_e, \theta_e] \in M(G_u)\ .$$
This amounts to solving a linear program for each extreme point. If every extreme point is feasible, then terminate the algorithm since $\hat{T}(G_u) = T(G_u)$.
4) Extend $\mathcal{D}$: let $j \leftarrow j + 1$ and $\mathcal{D} \leftarrow \{0,\ldots,j\}^K \setminus \mathcal{D}$. Make a new iteration by proceeding with Step 2.

Theorem 3: The above algorithm is guaranteed to terminate in finite, albeit exponentially many, steps.

Unfortunately, the theoretical upper bound on the number of inequalities this brute-force algorithm needs to generate can be astronomically large, especially as the number of users (K) increases. Quite amazingly, however, in the course of our evaluation studies the algorithm generally terminated in much less iterations than its theoretical properties would suggest.

We implemented the algorithm as a bunch of Perl scripts gluing together the GNU Linear Programming Kit, the `LEMON` graph library and `cddlib` [12], a library for polyhedral computation[1]. First, we systematically constructed networks for which the algorithm displays the worst case behavior. Therefore, we generated a sequence of increasing sized, directed complete graphs and, for a complete graph G_N of size N, we set exactly N users: $(1,2),(2,3),\ldots,(N,1)$. The number of facets of the resultant throughput polytope $T(G_N)$ as the function of N is given in Fig. 3. The consequence is that one can easily construct graphs for which the size of the corresponding throughput polytope quickly grows intractable. However, in real network scenarios the situation seems a bit more promising, as demonstrated by the second round of our evaluation studies. We took a real-life network topology, placed an increasing number of users in it selecting the source and destination nodes randomly, fed the resultant network to the algorithm to compute the corresponding throughput polytope, repeated this process 30 times, and averaged the results. The network topologies were the 28 node European and the German reference networks used extensively in recent EU projects [13] and the ubiquitous US NSF network topology [14]. The capacity of the links was uniformly 10 units in the first 2 cases and 100 units in the third. Fig. 4 shows the maximum and average number of the facets of the throughput polytope. The level of significance is 95%. These results suggest that, despite of the intractable growth observed in artificial networks, in real networks a reasonable sized description of the throughput polytope and modest computational requirements of the algorithm are expectable as long as K does not grow beyond about a dozen. We found that the algorithm performed relatively few iterations (see Fig. 5): as a rule of thumb, $2 \leq B \leq 4$ seems plausible for the number of necessary iterations B, which is still quite large but much less than the theoretical limit, and looks tolerable when K remains low.

Our experiences showed that the difficulty in computing the throughput polytope stems from the sensitivity to the number of users, and to a lesser degree from the raw size of the network. For smaller networks, up to about a dozen users, even our brute-force algorithm looks viable and it is our belief that an improved algorithm could scale to the range of a few dozen users. An appealing optimization would be for instance to exploit the algorithm's inherent aptitude to parallel execution.

[1]See the `Math::GLPK` and `Lemon::Graph` project pages at http://qosip.tmit.bme.hu/~retvari.

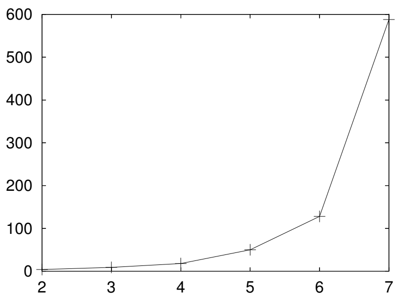


Figure 3: Number of facets of throughput polytope in complete graphs, as the function of the number of nodes.

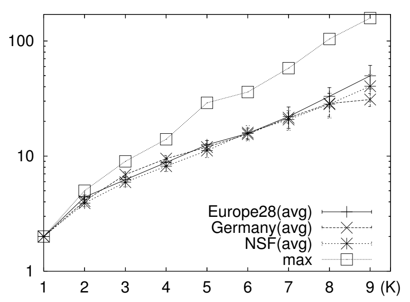


Figure 4: Average and maximal number of facets of $T(G_u)$ as the function of the number of users.

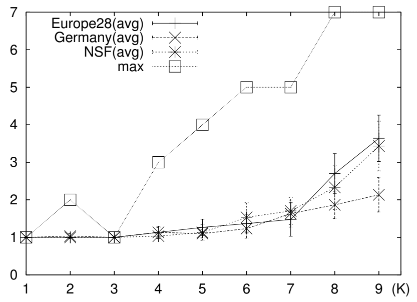


Figure 5: The average and the maximal number of iterations of running the algorithm on real networks.

V. Conclusions

Traditionally, fair allocation of user throughputs has been considered in the case when the path of the users is fixed for the lifetime of the communication. In this model, users get whatever "fair" share of network resources the actual routing allows them to receive. However, a user might ask rightfully: "Why has exactly *this* routing been implemented in the network instead of another one, which would be more beneficial for me within the current throughput allocation strategy?" This argumentation holds some merit, because in the fixed-path model the throughput allocated to a user depends quite heavily on the route taken by the traffic of that user, which, within the network architectures of our days, the user is not quite empowered to affect. In this paper we argued that it is much more natural to make throughput allocation strategies independent of routing, and we have extended the most commonly used fairness criterion, max-min fairness, to this generic case. If the throughput was determined independently of the actual routing, then no one would have the right to complain since it was the network, a given entity, that decided which particular share of network resources a user gets.

Our solution of the general, routing-independent throughput allocation problem was based on a polyhedral description of the range of throughput configurations realizable in a capacitated network. This throughput polytope is notable, not only because it helped us to characterize max-min fair allocations in the generic, routing-independent model, but also because it can easily help to do the same with other notions of fairness, like proportional fairness or utility fairness. Additionally, our polyhedral approach allowed us to extend the conventional bottleneck argumentation and the water-filling algorithm to the generalized setting in an illustrative manner. Unfortunately, the evaluations showed that the throughput polytope might be notoriously hard to calculate depending on the specifics of the network at hand. In such cases, Max-min Programming seems a better fit to obtain the max-min fair allocation.

Acknowledgements

The authors wish to thank the anonymous reviewers for their careful reading of the manuscript, for their criticism and suggestions. The first author would like to thank I. K. for the inspiration. This work was partially supported by the European IP NOBEL Project (www.ist-nobel.org).

References

[1] J. M. Jaffe, "Bottleneck flow control," *IEEE Transactions on Communications*, vol. 29, pp. 954–962, July 1981.

[2] E. L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE Journal on Selected Areas of Communication*, vol. 9, pp. 1024–1039, Sept. 1991.

[3] A. F. T. Committee, "Traffic Management Specification - Version 4.0." ATM Forum/95-0013R13, Feb 1996.

[4] D. P. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[5] J. Le Boudec, "Rate adaptation, congestion control and fairness: a tutorial." available online: http://ica1www.epfl.ch/PS_files/LEB3132.pdf, Feb 2005.

[6] R. Denda, "The fairness challenge in computer networks," Tech. Rep. TR-00-006, Department for Mathematics and Computer Science, University of Mannheim, 2000.

[7] J. Le Boudec and B. Radunovic, "A unified framework for max-min and min-max fairness with applications," in *Proceedings of 40th Annual Allerton Conference on Communication, Control, and Computing*, Oct 2002.

[8] Z. Cao and E. W. Zegura, "Utility max-min: an application-oriented bandwidth allocation scheme," in *Proceedings of INFOCOM 1999*, vol. 2, pp. 793–801, March 1999.

[9] D. Nace and L. Doan, "A polynomial approach to the fair multi-flow problem." Tech. Rep., Heudiasyc, UTC, available online: http://www.hds.utc.fr/~dnace/recherche/Publication/TR-MMF.pdf, 2002.

[10] G. Ziegler, *Lectures on Polytopes*, vol. 152 of *Graduate Texts in Mathematics*. New York: Springer, 1998.

[11] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. New York: John Wiley & Sons, second ed., 1990.

[12] K. Fukuda and A. Prodon, "Double description method revisited," in *Combinatorics and Computer Science, 8th Franco-Japanese and 4th Franco-Chinese Conference, Brest, France, July 3-5, 1995, Selected Papers* (M. Deza, R. Euler, and Y. Manoussakis, eds.), vol. 1120 of *Lecture Notes in Computer Science*, pp. 91–111, Springer-Verlag, Berlin, 1996.

[13] M. L. Garcia-Osma, "TID scenarios for advanced resilience." Tech. Rep., The NOBEL Project, Work Package 2, Activity A.2.1, Advanced Resilience Study Group, Sep 2005.

[14] B. Chinoy and H. W. Braun, "The national science foundation network." Tech. Rep., CAIDA, available online: http://www.caida.org/outreach/papers/1992/nsfn/nsfnet-t1-technology.pdf, Sep 1992.

[15] M. Iri, "On an extension of the maximum-flow minimum-cut theorem to multicommodity flows," *Journal of the Operations Research Society of Japan*, vol. 13, no. 3, pp. 129–135, 1971.

[16] K. Onaga and O. Kakusho, "On feasibility conditions of multicommodity flows in networks," *IEEE Transactions on Circuit Theory*, vol. 18, no. 4, pp. 425–429, 1971.

Appendix

Proof of Proposition 1: $M(G_u)$ is, by definition (1)–(3), an intersection of finitely many halfspaces, so it is a polyhedron. By Definition 2, $T(G_u)$ is the orthogonal projection of $M(G_u)$ to the space spanned by θ, therefore it is itself too a polyhedron [10]. Finally, compactness for regular networks is straightforward. ■

Proof of Proposition 2: Applying Černikov's block-elimination method [10] to $M(G_u)$, we have that row K-vectors β and row m-vectors w lying in the projection cone

$$W(G_u) = \{[\beta, w] : \quad \textstyle\sum_{(i,j)\in P} w_{ij} \geq \beta_k \quad \forall k \in \mathcal{K}, \forall P \in \mathcal{P}_k \\ w \geq 0 \ \}$$

generate *all* the inequalities of $T(G_u)$: $T(G_u) = \{\theta \geq 0 : \beta\theta \leq wu, \ \forall[\beta, w] \in W(G_u)\}$. In fact, it is enough to take the inequalities generated by the *extreme rays* of $W(G_u)$, so $\mathcal{I}$ is finite. Observe that here, vectors w can be thought of as non-negative *edge weights*, while the kth coordinate of β, β_k, is less than, or equal to the length of the shortest path from s_k to d_k over the edge weights w. (Note that in all practically important cases $(\beta)_k$ in fact attains the length of the shortest path.) Hence, β is non-negative and, for a regular G_u, wu is strictly positive. See alternative proofs in [15] and [16] (the Japanese Theorem). ■

Proof of Theorem 1: Let $T(G_u) = \{\theta \geq 0 : \beta_i\theta \leq b_i, \ i \in \mathcal{I}\}$ and let $\mathcal{B}$ be the set of constraints binding at θ_0: $\mathcal{B} = \{i \in \mathcal{I} : \beta_i\theta_0 = b_i\} \neq \emptyset$. Let $\beta = \sum_{i\in\mathcal{B}} \beta_i$ and $b = \sum_{i\in\mathcal{B}} b_i$. Obviously, $\beta\theta \leq b$ is valid for $T(G_u)$, $\beta \geq 0$ and $\beta\theta_0 = b$, so the first two claims immediately apply. To see that the last one also does, it is enough to show that $(\beta)_k = 0$ if and only if k is dominated at θ_0. Since $\beta_i \geq 0$ for each $i \in \mathcal{I}$, so $(\beta)_k = 0$ if and only if $(\beta_i)_k = 0$ for all constraints binding at θ_0. This means that $\exists\epsilon > 0$ and small enough, so that $\theta_0 + \epsilon e_k \in T(G_u)$, so k is dominated. The reverse direction of the proof comes similarly. ■

Proof of Proposition 3: Since $T(G_u)$ is a polytope, it is, by nature, convex and compact. Then the existence and uniqueness of the max-min fair vector is guaranteed by [7, Theorem 1]. ■

Proof of Corollary 2: For each $k \in \mathcal{K}$ construct the vector θ', whose coordinates are defined as

$$(\theta')_l = \begin{cases} (\theta_0)_k & \text{if } (\theta_0)_l > (\theta_0)_k \\ (\theta_0)_l & \text{otherwise} \end{cases}$$

Observe that, by definition, exactly those users l are non-dominated at θ' for which $(\theta_0)_l \leq (\theta_0)_k$. All the other users are dominated. Now, simply apply Theorem 1 to θ' to prove the Corollary. ■

Proof of Theorem 2: We have already seen that some $\theta_0 \in T(G_u)$ is max-min fair, if and only if each $k \in \mathcal{K}$ has a bottleneck inequality $\beta\theta \leq b = wu$ conforming to *(i)–(iii)* in Corollary 2. Here, w can be thought of as edge weights and $(\beta)_k$ as the length of the shortest path from s_k to d_k over the weights w. Now, define the corresponding bottleneck separating edge set as

$$\mathcal{S}_k = \{(i,j) \in E : w_{ij} > 0\} \ . \tag{11}$$

This implies that $\mathcal{K}_{\mathcal{S}_k} = \{l \in \mathcal{K} : (\beta)_l > 0\} = \{l \in \mathcal{K} : (\theta_0)_l \leq (\theta_0)_k\}$, using *(iii)* in Corollary 2. So $\mathcal{S}_k$ as defined by (11) immediately satisfies Property 1. To prove the theorem, we only need to show that it fulfills Property 2 too. For this, first we observe that vector $[\beta, w]$ taken from the bottleneck inequality of k solve the following linear program defined over the separating edge set $\mathcal{S}_k$ as defined by (11), with optimal objective function value zero:

$$\begin{aligned} 0 = \min wu - \beta\theta_0 & \\ w\Delta_l \geq \mathbf{1}(\beta)_l & \qquad \forall l \in \mathcal{K} \\ \beta_l \geq 1 & \qquad \forall l \in \mathcal{K}_{\mathcal{S}_k} \\ \beta_l = 0 & \qquad \forall l \in \mathcal{K} \setminus \mathcal{K}_{\mathcal{S}_k} \\ w_{ij} \geq 1 & \qquad \forall (i,j) \in \mathcal{S}_k \\ w_{ij} \geq 0 & \qquad \forall (i,j) \in E \setminus \mathcal{S}_k \end{aligned}$$

Now, by the strong duality theorem of linear programming, the dual linear program below is also soluble and the optimal objective function value is zero:

$$\begin{aligned} 0 = \max \textstyle\sum_{(i,j)\in\mathcal{S}_k} \lambda_{ij} + \sum_{l\in\mathcal{K}_{\mathcal{S}_k}} \mu_l & \\ \textstyle\sum_{l\in\mathcal{K}} \Delta_l f_l + \lambda = u & \\ \mathbf{1} f_l - \mu = (\theta_0)_l & \qquad \forall l \in \mathcal{K} \\ \mu_l \geq 0 & \qquad \forall l \in \mathcal{K}_{\mathcal{S}_k} \\ f_l \geq 0, \lambda \geq 0 & \qquad \forall l \in \mathcal{K} \end{aligned}$$

Let $[f, \lambda, \mu]$ be *any* optimal feasible solution. Now, the objective function value is zero, if and only if $\forall l \in \mathcal{K}_{\mathcal{S}_k} : \mu_l = 0$ and $\forall (i,j) \in \mathcal{S}_k : \lambda_{ij} = 0$, which exactly reproduces Property 2 and the proof is complete. ■

Remark: in the network of Fig. 2, we associated the inequality $\theta_1 + 2\theta_3 \leq 4$ with users $(1,6)$ and $(7,5)$ as bottleneck. By the Japanese Theorem, this inequality is generated by the shortest path lengths $[\beta_1, \beta_2, \beta_3] = [1, 0, 2]$ and the edge weights $w_{(2,3)} = w_{(4,5)} = 1$, $w_{(8,5)} = 2$ and all zero otherwise. So, according to (11), the edges of nonzero weight make up the bottleneck separating edge set for these two users: $\mathcal{S}_1 = \mathcal{S}_3 = \{(2,3), (4,5), (8,5)\}$. Finally, the bottleneck inequality of user $(7,8)$ is $\theta_1 + \theta_2 + 3\theta_3 \leq 7$, which is generated by the shortest path lengths $[\beta_1, \beta_2, \beta_3] = [1, 1, 3]$ and edge weights $w_{(2,3)} = w_{(4,5)} = w_{(7,8)} = 1$, $w_{(8,5)} = 2$ and all zero otherwise, so the corresponding bottleneck separating edge set is $\mathcal{S}_2 = \{(2,3), (4,5), (7,8), (8,5)\}$.

Theorem 3: Iri showed that edge weights no greater than K^m are enough to obtain a complete description of $T(G_u)$ [15]. Since $(\beta)_k$ is the length of the shortest $s_k \rightarrow d_k$ path for user k over those edge weights, we have that it is enough to go up to at most $(\beta)_k \leq mK^m$, which is an upper bound on the elements in $\mathcal{D}$ and hence on the number of iterations the algorithm performs. ■