Using Approximate Majorization to Characterize Protocol Fairness

Ashish Goel [*] Adam Meyerson [†] Rishi Bhargava [‡]

Univ. of Southern Calif. Stanford University. ONI Systems

February 25, 2002

Abstract

A good measure of fairness is an essential prerequisite for a systematic development of fair resource allocation protocols as well as for a systematic evaluation of the fairness of existing protocols. We propose the use of approximate majorization as a framework for quantifying the fairness of a resource allocation scheme. We demonstrate how approximate majorization subsumes and generalizes several natural measures of fairness. We then relate majorization to revenue maximization, and sketch an efficient algorithm to compute the fairness of a given allocation as well to find the fairest allocation. Our framework is quite general and can be applied to several routing, bandwidth allocation, load balancing, and clustering problems.

To illustrate the framework in a concrete setting, we perform a preliminary case study of the fairness of TCP as a bandwidth allocation protocol in communication networks. We demonstrate that the fairness of both TCP Reno and TCP Vegas is incomparable to the fairness obtained by max-min fair allocation. Finally, we explore the connections between approximate majorization and the notion of proportional fairness developed by Kelly, Maulloo, and Tan.

1 Introduction

Consider the problem of designing a protocol to *fairly* allocate a fixed resource among n individuals given some constraints. This general scenario can be specialized to several routing [19, 10], bandwidth allocation [10, 1, 2], clustering [21], and load balancing [11] problems. In order to evaluate the fairness of any

[*] Ashish Goel is at the Department of Computer Science, University of Southern California. Email: agoel@cs.usc.edu

[†] Adam Meyerson is at the Department of Computer Science, Stanford University. Email: awm@stanford.edu

[‡] Rishi Bhargava is at the Department of Computer Science, University of Southern California. Email: RBhargava@oni.com

proposed solutions for these problems, we must first decide on a good definition of fairness. This is a hard problem, as can be seen by the large number of (often disparate) fairness measures proposed by both system builders [16, 6, 8, 1, 2, 20] and theoreticians (see chapter 13 of [24]). Some of the more widely discussed measures are max-min fairness [15, 4, 1, 2], variance related measures [16], and average utility. This paper presents a framework to unify and generalize all the above measures. We will explain this theory largely in the context of networking problems, and perform a simple case study of the fairness of TCP. However, we believe that the underlying ideas are general and could find applications in several domains. It is our hope that this theory will aid in the evaluation of the fairness of existing protocols, and also serve as one of the criteria that inform the design of future protocols. In this paper, we will not discuss whether fairness is desirable and important; these considerations are best left to protocol designers in their respective application domains.

Let $s_i(X)$ denote the amount of utility that we provide to the i-th poorest individual if we follow allocation policy X. Note that different individuals may be the i-th poorest in different allocation policies. Let "prefix" $P_i(X)$ be the total amount of utility provided to the i poorest individuals. The central theme of this paper is that a "fair" allocation policy is one that simultaneously gives a high value for all the prefixes $P_1(X), P_2(X), \ldots, P_n(X)$. Thus, we would like to be fair on the average (i.e. get a high total utility $P_n(X)$), fair to the poorest individual (high $P_1(X)$) as well as fair to all the intermediate prefixes. For example, an allocation which gives a utility of $2, 3, 3$ to three individuals is deemed fairer than one that gives a utility of $2, 2, 4$ even though they both have the same minimum and average utilities. An elegant theory of majorization was developed by Hardy, Littlewood, and Polya in the 1920s which formalizes the above theme [13, 14]. Using their terminology, given two allocations X and Y, we say that X is majorized by Y if $P_i(X) \geq P_i(Y)$ for all i, and $P_n(X) = P_n(Y)$. Even before the notion of majorization was formally introduced, Lorenz [22] postulated that if X is majorized by Y, then X is "fairer than" Y. The rationale behind this postulate is that under any natural measure of fairness, we would expect the allocation (x_2, x_1) to be as fair as the allocation (x_1, x_2) and less fair than $(\frac{x_1+x_2}{2}, \frac{x_1+x_2}{2})$. This translates into saying that maximizing fairness should be equivalent to maximizing some symmetric[1] concave function $f(X)$. But Hardy, Littlewood, and Polya proved that if X is majorized by Y then $f(X)$ is larger than $f(Y)$ for *all* such functions. Thus if we could find a most majorized allocation, it would be the fairest under all "natural" measures of fairness. This

[1] In this paper, a symmetric multivariate function is one that is symmetric in all its arguments ie. exchanging the arguments does not change the value of the function.

is a very strong definition of fairness, and not surprisingly, a most majorized allocation may not exist for a large class of problems. This motivates our definition of *approximate majorization*[2] as detailed in section 2.

In this paper, we first define (section 2) the majorization index of a resource allocation protocol and explain why it is a good measure of fairness. We also show how the framework of majorization unifies all the commonly studied measures of fairness. We then explain how an approximately majorized allocation as defined in this paper is approximately fair for a large class of fairness functions (theorem 2.3). We then sketch algorithms (section 3) to compute the majorization index of an allocation, and present a bound on the majorization index in terms of the "inherent imbalance" of the problem.

We omit detailed algorithms and proofs from section 3; they can be found in a more formal companion paper [**?**] and are reproduced in appendix A. Instead, we illustrate our framework by a preliminary case study (section 4) of the fairness of TCP as a bandwidth allocation protocol in data networks. A lot of attention has been paid recently to fair allocation of bandwidths in data networks. RED [9], WFQ [7, 28], CHOKe [27], and CSFQ [30] can all be looked upon as bandwidth allocation/regulation schemes that promote fairness. The problem is also important in wireless communications [20]. One traditional definition of fairness in computer networks has been max-min fairness [15, 4, 1, 2]. We will use our framework to *quantitatively* compare the fairness of TCP with that of the max-min fair allocation. We demonstrate that the fairness of both TCP Reno and TCP Vegas is incomparable to the fairness obtained by max-min fair allocation[3]. Interestingly, for networks where a most majorized allocation of bandwidth exists, TCP Vegas attempts to find such a vector. In this section, we employ analytical models of TCP Reno and Vegas to prove our results [26, 17, 23].

Finally, in section 5, we compare our framework to the proportional fairness framework proposed by Kelly, Maulloo, and Tan [18], exploring the connections as well as the differences between the two models. We also give a simple example where our framework can be extended to non-symmetric fairness functions. Sections 4 and 5 can mostly be read independently of section 3. We summarize our results in section 6 and also present two challenging open problems.

[2]This definition is similar to the definition of prefix-competitivity proposed earlier by Kleinberg, Rabani, and Tardos [19] and Goel, Meyerson, and Plotkin [10] in the context of online algorithms.

[3]TCP Reno is a popular version of TCP; TCP Vegas is a recent version which is less widely deployed but has interesting properties.

2 A Majorization Based Fairness Framework

Consider a fixed resource or a set of resources which needs to be *fairly* allocated among n individuals given some constraints. This general scenario can be specialized to several routing [19, 10], load balancing [11, 19], clustering [21], and bandwidth allocation problems [1, 2, 20, 8, 6]. To evaluate a solution to these problems, we must decide on a good definition of fairness.

2.1 Using majorization to characterize fairness:

Let $X = \langle x_1, x_2, \ldots, x_n \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ be two feasible vectors (i.e. satisfying all the constraints) denoting resource allocation to n individuals. Let $s_i(X)$ denote the i-th smallest component of X. Informally, $s_i(X)$ is the amount of resource allocated to the i-th poorest individual. Then X is said to be majorized [13, 14, 24] by Y, denoted $X \prec Y$, if the following two conditions are satisfied:

1. $(\forall k, 1 \leq k \leq n)$, we have $\sum_{i=1}^{k} s_i(X) \geq \sum_{i=1}^{k} s_i(Y)$, and
2. $\sum_{i=1}^{n} s_i(X) = \sum_{i=1}^{n} s_i(Y)$

This concept was introduced by Hardy, Littlewood, and Polya in the 1920s to study properties of inequalities. Even before that, Lorenz [22] informally proposed that the above definition implies that X is fairer than Y. Figure 1(a) gives a pictorial representation of majorization for additional intuition. Intuitively $X \prec Y$ means that poorest individual in X is at least as rich as the poorest individual in Y, the two poorest individuals in X are together at least as rich as the two poorest individuals in Y and so on.

If $X \prec Y$ for all feasible Y, then X is said to be the *most majorized* vector. Suppose $f(X) = \sum_i g(x_i)$ captures the "fairness" of a solution. So the goal is to maximize $f(X)$. It seems natural that allocation $(\frac{x_1+x_2}{2}, \frac{x_1+x_2}{2})$ is fairer than (x_1, x_2); this is certainly true for *all* fairness measures we found in literature, some of which we discuss later [16, 6, 8, 20]. This natural condition translates to the requirement that $g(x)$ is concave[4]. The following theorem relates majorization to any natural notion of fairness as defined above:

Theorem 2.1 *[13, 24] If X is the most majorized allocation then for any concave function g, X maximizes the value $\sum_i g(x_i)$.*

[4] If one is trying to minimize the "unfairness index" then it makes sense for g to be convex by an analogous argument.

Now consider the case where the function f is not decomposable into $\sum_i g(x_i)$. It seems natural to require the allocation (x_2, x_1) to be as fair as the allocation (x_1, x_2) and less fair than $(\frac{x_1+x_2}{2}, \frac{x_1+x_2}{2})$. This translates into saying that maximizing fairness should be equivalent to maximizing some symmetric concave function $f(X)$. Theorem 2.1 continues to hold for such functions.

Theorem 2.2 *[13, 24] If X is the most majorized allocation then for any symmetric concave function f, X maximizes the value $f(X)$.*

Motivated by the above discussion and theorems 2.1 and 2.2, we will also use the term "globally fair" to describe a most majorized vector.

2.2 Majorization and other fairness criteria

In this subsection we quote several fairness criteria proposed by researchers and relate them to majorization:

- Max-min fairness [15, 4, 1, 2]
- Minimize the Variance [16]
- Minimize the Coefficient of Variation [16]
- Maximize the min-max ratio [16]: $f(X) = \frac{\min_i x_i}{\max_i x_i}$
- Maximize the power function [8]: $f(X) = \prod_{i=1}^{n} x_i$
- Maximize Jain's fairness index [16]: $f(X) = \frac{[\sum_{i=1}^{n} x_i]^2}{\sum_{i=1}^{n} x_i^2}$
- Kullback-Leibler fairness index [20]: This is not a fairness index in itself but is a measure of distance between two probability distributions. If p and q are two probability distributions then the Kullback-Leibler distance (or *relative entropy*) between them is denoted by $D(p \parallel q)$. Suppose the fairest allocation is represented by $\tilde{\Gamma} = (\beta_1, \beta_2, \beta_3, \ldots)$ and let the allocation to be analyzed be $\Gamma = (\gamma_1, \gamma_2, \gamma_3, \ldots)$. Then,
$$D(\Gamma \parallel \tilde{\Gamma}) = \left(\sum_{i=1}^{N} \gamma_i \log_2 \gamma_i\right) - \left(\sum_{i=1}^{N} \beta_i \log_2 \beta_i\right)$$

It is easy to prove using theorems 2.1 and 2.2 that if X is globally fair as defined earlier, then X is max-min fair, has the least possible variance, has the maximum min-max ratio possible, maximizes the power function, maximizes Jain's fairness index and subsumes the Kullback-Leibler index. As an illustration we

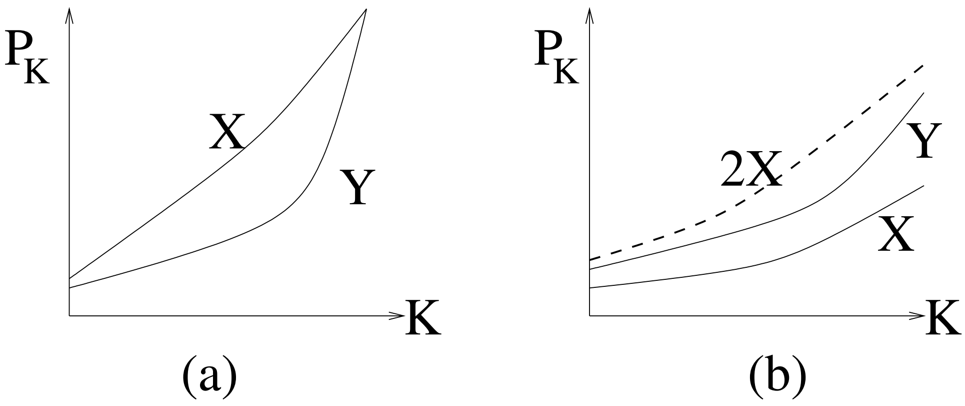

Figure 1: Here P_K refers to the sum of the resources allocated to the K poorest individuals. In figure 1(a), X is majorized by Y (i.e. fairer than Y) since all the prefixes in X are larger than the corresponding prefixes in Y. In figure 1(b), if we multiply the allocation X by a factor 2, then the resulting allocation $2X$ has larger prefixes than Y. Hence, X is 2-supermajorized by Y.

will prove that our fairness criterion subsumes the power function fairness index. The proof for the rest of the indices is similar and is omitted. Let $f(X) = \prod_{i=1}^{n} x_i$. This is not a symmetric concave function. However, taking the logarithm on both sides, we obtain $\log f(X) = \sum_{i=1}^{n} \log x_i$. But $\log x$ is a concave function, therefore theorem 2.1 implies that $\log f(X)$ is maximized by the most majorized allocation.

We do not claim that the above described measures are not good measures of fairness. In fact each measure described above is quite reasonable for its application domain. For instance, fairness in each measure corresponds to maximizing some symmetric concave function, which is consistent with our intuitive notion of fairness. We merely wish to establish that majorization subsumes and generalizes all the above commonly used measures and merits further study. If a most majorized allocation were to exist for a given problem, then all the above measures of fairness would agree that this allocation is the fairest.

There is also an interesting connection between majorization and revenue maximization, stemming from the fact that revenue functions typically follow the law of diminishing returns and are therefore typically concave. Suppose we are given a concave increasing revenue function f with $f(0) = 0$. Then $\sum_i f(x_i)$ is maximized for the most majorized vector[5]. Thus fairness as defined above is also beneficial to the "service provider".

[5] Again, this result holds for the more general class of symmetric concave revenue functions.

The terms "most majorized" and "globally fair" will be used interchangeably in this paper. Having adopted the above definition of fairness, our goal is to find the most majorized vector given a set of constraints. Unfortunately, the most majorized vector may not exist even for very simple constraints. For a simple real life example, consider a network which is a two-hop path and carries three connections. Connection 1 uses both the links; connection 2 uses only the first link; and connection 3 uses only the second link. Let x_1, x_2, x_3 be the bandwidth allocated to the three connections by some bandwidth allocation protocol. Let each link have unit capacity. The corresponding constraints are $\{x_1 + x_2 \le 1; x_1 + x_3 \le 1; x_1, x_2, x_3 \ge 0\}$. The solution that maximizes the prefix $x_1 + x_2 + x_3$ must assign zero bandwidth to the first connection and unit bandwidth to the other two connections. The solution that maximizes the bandwidth for the most starved connection must assign a bandwidth of half a unit to each connection. Thus, for this simple bandwidth allocation problem, there is no globally fair solution. We will take a closer look at this problem later in our case study.

To address this problem, we will now define the notion of approximate majorization to quantify fairness. Most of the desirable properties of majorization carry over to approximate majorization in a weaker form.

2.3 Approximate majorization and approximate fairness:

Given feasible vectors X and Y, X is said to be α-supermajorized by Y (α-fairer than Y), denoted $X \prec^{\alpha} Y$, if

$$(\forall k, 1 \le k \le n) \quad \alpha \sum_{i=1}^{k} s_i(X) \ge \sum_{i=1}^{k} s_i(Y)$$

A feasible vector X is said to be globally α-fair if $X \prec^{\alpha} Y$ for all feasible vectors Y and α is called the majorization index of X.

A globally α-fair vector can be thought as being "approximately most majorized". This is a natural and very strong definition of approximate fairness. Intuitively, this says that given any other allocation and any number k, the k poorest individuals in X together get at least $1/\alpha$ times as much resource as the k poorest individuals in Y. Globally α-fair vectors retain the nice properties of most-majorized vectors, but in an approximate sense as explained later. Further, as we show in section 3, globally α-fair vectors exist (and can be efficiently found) for small values of α for a large class of problems. Figure 1(b) shows an example where X is 2-majorized by Y.

Let P_j^* be the maximum total resource that can be allocated to the j poorest individuals by any allocation. We will call P_j^* the optimal j^{th} prefix.

Given any allocation X, $P_j(X)$ is the total resource allocated to the j poorest individuals in allocation X, i.e. $P_j(X) = \sum_{i=1}^{j} s_i(X)$. Now, $\alpha(X) = \max_j \frac{P_j(X)}{P_j^*}$ is the majorization index of X. Let α^* be the smallest possible value of $\alpha(X)$ for any feasible allocation X.

The following theorem presents strong quantitative evidence that approximate majorization is a good framework for studying fairness and that α is a good measure of fairness.

Theorem 2.3 *An allocation X is globally α-fair iff $\alpha f(X) \geq f(Y)$ for all feasible allocations Y and all symmetric concave functions f such that $f(0) = 0$ and f is non-decreasing in each argument.*

Proof Sketch: If X is globally α-fair then theorem 2.2 and the fact that f is a non-decreasing function together imply that $f(\alpha X) \geq f(Y)$ for any feasible vector Y. Here αX refers to the allocation where each component of X has been multiplied by α. Notice that the allocation αX need not be feasible. However, since $f(0) = 0$ and f is concave, we can conclude that $f(\alpha x) \leq \alpha f(x)$ for all $x \geq 0$. Hence, $f(X) \geq f(\alpha X)/\alpha \geq f(Y)/\alpha$ for *all feasible* Y. This proves the forward direction of the theorem.

For the reverse direction, we observe that for any k, the prefix $P_k(X)$ can be looked upon as a function of X. This function happens to be symmetric, concave, non-decreasing, and $P_k(0) = 0$. If $\alpha f(X) \geq f(Y)$ for all f that satisfy the conditions of the theorem, then $\alpha P_k(X) \geq P_k^*$, which implies that X is globally α-fair. ∎

The above theorem is slightly weaker than theorem 2.2, but it still holds for a large class of fairness functions; certainly all the fairness measures that we found in the literature can be transformed into the above form. Armed with the strong "if and only if" connection between approximate majorization and approximate fairness that the above theorem provides us, we will now use the *majorization index* of a resource allocation protocol to quantify its fairness.

3 Computing Optimal Prefixes and Optimal Majorization Index

For the approximate majorization framework to be useful, it is important to be able to compute the optimal prefixes given a set of constraints. Also, to evaluate how far a given protocol is from the optimum, we need to compute the best possible α such that a globally α-fair solution exists. Further, this framework would be interesting only if globally α-fair vectors exist for small values of α. All these issues are addressed in this section.

We are going to consider problems where the set of constraints is a linear program. This includes a large range of optimization problems. We will specifically mention the bandwidth allocation problem; this problem arises in connection with our chosen case study of the fairness of TCP.

Recall that a globally α-fair solution is α-supermajorized by every other feasible solution. Of course, a solution with $\alpha = 1$ may not exist. There are two distinct parts to a discussion of finding the fairest possible solution.

1. Finding the optimum prefixes P_j^* and the smallest α for which a globally α-fair solution exists. This smallest α will be denoted as α^*.

2. Proving that there always exists a solution with a small value of α^*.

3.1 Finding P_j^* and α^*

Suppose we are given a set of linear constraints on the variables $x_1, x_2 \ldots x_n$. A solution is said to be feasible if it satisfies all constraints. First, we will determine the value of each P_j^*. Then we will find α^* and the corresponding feasible solution x^*. We will present linear programs for each of these steps. The sizes of these linear programs are exponential but they can be solved in polynomial time using the ellipsoid method [12]. We will then describe ways to modify these programs in order to speed up the running time for the fair bandwidth allocation problem, which is an important component of our chosen case study about the fairness of network protocols.

Feasibility requires that $A\vec{x} \le \vec{b}$ for some matrix A, constant vector $\vec{b}$, and variable vector $\vec{x}$. We will add a number of new constraints to the linear program to produce the following.

- Maximize p_j subject to:
- $A\vec{x} \le b$
- $\sum_{i \in S} x_i \ge p_j$ for all $S \subseteq \{1, \cdots, n\}$ with $|S| = j$

This linear program finds the largest possible value for the jth prefix. The linear program has exponentially many equations, because there will be an exponential number of subsets S. Such linear programs can be solved using the ellipsoid method [12] if we are given an efficient separation oracle[6]. In this case, the separation oracle simply sums the j smallest x_i; if their sum is at least p_j then our solution is feasible, if not

[6] A separation oracle looks at a candidate solution and either guarantees that this solution satisfies all given linear constraints, or produces a constraint that is violated.

then the equation indicating that that particular subset S of the x_i sums to at least p_j is violated. The ellipsoid method is polynomial time (although inefficient in practice), and we can thus produce the optimum p_j in polynomial time. Repeating this n times (once for each j) provides the values P_j^* for each j.

We now need to produce the most-fair vector $\vec{x}^*$ along with the optimum value α^*. This can be done by solving the following linear program. The solution vector to this linear program is $\vec{x}^*$, with $\alpha^* = 1/\gamma$. The last set of constraints say that for all k and for any set of k individuals, the total "utility" for these individuals should be at least $\gamma P_k^* = P_k^*/\alpha^*$.

- Maximize γ subject to:
- $A\vec{x} \leq b$
- $\sum_{i \in S} x_i \geq \gamma P_{|S|}^*$ for all $S \subseteq \{1, \cdots, n\}$

Again the linear program has exponentially many constraints, but again we can solve it using the ellipsoid method. The separation oracle simply checks the values of all prefixes for our current vector $\vec{x}$ and compares them to γP_j^*. The solution is feasible if and only if all of these equations are satisfied. This yields a polynomial-time algorithm for computing the minimum possible α^* and P_j^*.

It is possible to reduce the size of the linear program so that the number of constraints become polynomial. However that discussion is beyond the scope of this paper and is detailed in [**?**] and appendix A. We now sketch the algorithm for computing P_j^* and α^* very efficiently for the bandwidth allocation problem; again, a detailed explanation and proofs can be found in appendix A.

The bandwidth allocation problem: We are given a network with m links $l_1, \ldots, l_m$, with C_i being the capacity of l_i. We are also given n connections with routes $R_1, \ldots, R_n$. We have to assign bandwidths $x_1, \ldots, x_n$ such that the total bandwidth allocated to all the connections that use link i is at most C_i, for each i. Our goal is to be as "fair" as possible to each connection. To evaluate the fairness of a bandwidth allocation protocol (such as TCP), we would like to first find the best prefixes P_j^* and the value α^*. Let A be the $n \times m$ route-link incidence matrix such that $A_{ik} = 1$ if link i is used by connection k and 0 otherwise. Also, let C be the column vector representing the capacities of all the links. We will now fix the value of P_j to be 1 and then try to find the smallest fraction λ_j of the capacity needed to provide this value. P_j^* will now be $1/\lambda_j$. This approach is captured by the following linear program for finding the prefix P_j^*:

- Minimize λ_j subject to:

- $A\vec{x} \leq \lambda_j C$
- $\vec{x} \geq 0$
- $\sum_{i \in S} x_i \geq 1$ for all $S \subseteq \{1, \cdots, n\}$ with $|S| = j$

This linear program is a special case of a *fractional packing problem*, and can be solved efficiently using the algorithm of Plotkin, Shmoys, and Tardos [29]. The value of α^* can be calculated using a similar approach.

3.2 Existence of small α^*

We will just state the theorems in this section; their proof can be found in appendix A. In general, the minimum α such that there exists a globally α-vector could be very large. We characterize this value in terms of the "inherent imbalance" of a given problem.

Definition 1 *The* **inherent imbalance** *of a problem is $\frac{P_n^*}{nP_1^*}$, where n is the dimension of the vector over which we want the fairest solution.*

In other words, the inherent imbalance of a problem is the ratio of the maximum value that can be achieved for the average utility (P_n^*/n) to the maximum utility that can be achieved for the poorest individual (P_1^*). Intuitively, the larger the inherent imbalance, the less likely it will be that there exists a most-fair vector with a small α. The following theorem (proved in A) states that α^* is roughly logarithmic in the inherent imbalance of the problem.

Theorem 3.1 *For any linear program, $\alpha^* \leq 2 + 2\log\frac{P_n^*}{nP_1^*}$.*

The two interesting features of this result are:

1. The inherent imbalance is defined not as the ratio of the maximum utility for the richest individual to the maximum utility for the poorest, but as the ratio of the maximum *average* to the poorest.

2. Even if the inherent imbalance grows linearly in the number of individuals, the optimum majorization index grows only as the logarithm of the number of individuals.

Using the above theorem, it follows that for the bandwidth allocation problem[7], $\alpha^* \leq 2+2\log n = O(\log n)$. This result can be improved to $O(\min\{\log n, \log m\})$ where m is the number of links in the network.

Of course, not all optimization problems of interest can be formulated as linear programs. For these problems, the two questions posed here need to be studied using domain specific and problem specific tools [19, 10, 21, 11, **?**].

4 Case Study: Comparing TCP to Max-Min Fairness

In this section we compare the fairness of TCP Reno and TCP Vegas to that of max-min fairness. We conclude that the fairness of both versions of TCP (as measured by the majorization index) is incomparable to that of max-min fairness. In other words, there are simple families of networks for which TCP has a much smaller majorization index than max-min, as well as other simple families for which TCP has a much larger majorization index. This is surprising, since max-min fairness consciously tries to promote fairness, whereas TCP is primarily geared towards other systems issues such as preventing congestion. TCP fairness has previously been studied formally, for example by [31, 5]. However, to the best of our knowledge, ours is the first quantitative argument that max-min fair allocations and TCP are incomparable in terms of their fairness properties.

Interestingly, for networks where a most majorized allocation of bandwidth exists, TCP Vegas attempts to find such a vector. In this section, we employ analytical models of TCP Reno and Vegas to prove our results [26, 17, 23].

4.1 TCP Reno and max-min fairness

We consider two families of networks, shown in figures 2 and 3. Both families have the same underlying topology, which is a line with $n+1$ nodes and n links. The $n+1$ nodes are $v_0, v_1, \ldots, v_n$. There is a link between nodes v_{i-1} and v_i for all $1 \leq i \leq n$. We will assume that all these links are identical and have unit capacity. This is a nice canonical topology for fairness evaluation [16, 8].

We are going to use the TCP equation due to Padhye *et al.* [26] to get approximations of the behavior of TCP Reno on these families. This equation relates the throughput λ of a TCP connection with the round trip delay (RTT) and the drop probability (p):

[7] Removing one unit of bandwidth from one connection can result in at most one extra unit of bandwidth for any other connection; this implies that the inherent imbalance of the bandwidth allocation problem is at most n.

$$\lambda \approx \frac{C}{\text{RTT} \cdot \sqrt{p}} \quad \text{, where } C \text{ is a constant.} \tag{1}$$

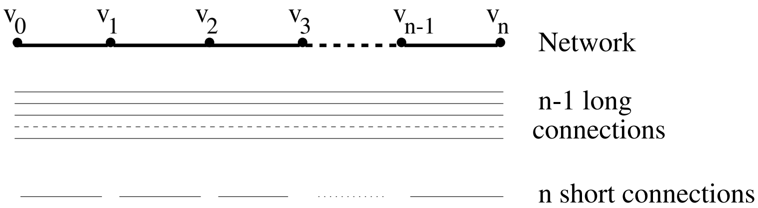

Figure 2: The first family

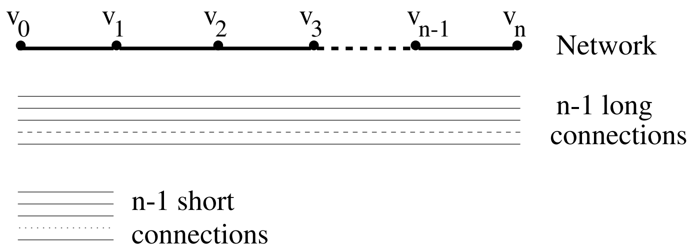

Figure 3: The second family

The first family: TCP Reno is fairer In the first family (figure 2) there are $n-1$ "long" connections which have their source at node v_0 and their destination at node v_n. Further, there are n "short" connections; the i-th short connection has its source at node v_{i-1} and its destination at node v_i. All these connections are bulk connections ie. we assume they would like to transfer infinite amounts of data, and desire as much bandwidth as possible.

For this family, $P_1^* = 1/n$ and $P_{2n-1}^* = n$. The former is achieved by assigning equal bandwidth to all connections. The latter is achieved by assigning zero bandwidth to all the long connections and unit bandwidth to all the short ones. This family does not admit a most-majorized allocation.

The max-min fair solution would allocate equal bandwidths to all flows in this family. Thus, each flow would get bandwidth $1/n$, and the total bandwidth allocated would be $(2n-1)/n$. For this family $P_{2n-1}^* = n$. But this implies that the ratio of the maximum total bandwidth possible and total allocated bandwidth for the max-min fair solution is $n^2/(2n-1) = \Theta(n)$. It is not hard to see that this is indeed the worst ratio of P_k^*/P_k

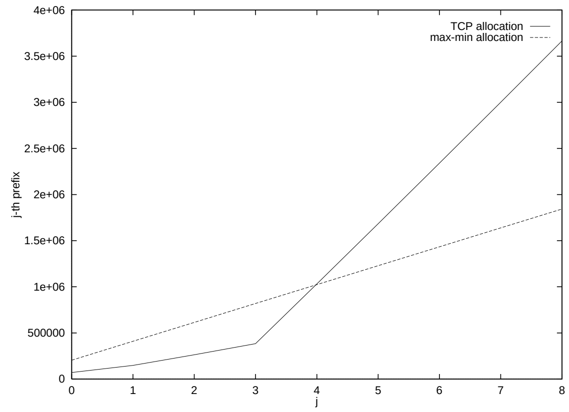

Figure 4: Prefix vector plot for TCP Reno and max-min. The X-axis represents j and Y-axis represents P_j.

for this family given max-min fair allocations, and hence, is the majorization index.

For TCP Reno, we assume that all links exhibit the same drop probability p_0, that p_0 is small, and that the packet drop events on different links are independent[8]. Then the drop probability seen by a "short" connection is merely p_0, whereas that seen by a long connection is approximately np_0. Also, the RTT observed by a long flow is n times that of a short flow. Consequently, the ratio of the bandwidth received by a long connection to that received by a short connection would be $n^{-3/2}$, from the TCP equation. Since a short connection can not get more than unit bandwidth, the long connections would get bandwidth $\Theta(n^{-3/2})$ each. But P_1^* for this family is $1/n$, which implies that the majorization index of TCP for this family is $\Theta(\sqrt{n})$.

As n grows larger, the majorization index for max-min fairness grows linearly, whereas that of TCP grows as $\sqrt{n}$. Clearly, TCP performs better for this family of networks.

For gaining further insight look at the plot of the prefixes shown in figure 4. These prefixes are obtained by running the network simulator ns-2 [25] to obtain TCP throughputs[9], with $n = 5$. As shown, TCP does badly in the "beginning" and much better at the "end" compared to max-min. This is consistent with our understanding of TCP. The majorization indices for different values for n for the first family are reported in table 1; again the TCP majorization index is computed using results from an actual ns simulation. These results are consistent with the conclusions of the approximate analysis performed above that TCP is fairer than max-min fair allocations as n grows large. In all these simulations, each link had a capacity of 1Mbps, each buffer-capacity was 50 packets, and all the queues were drop-tail.

[8] We do not claim these assumptions result in very accurate results. However, in this analysis, we are only interested in rough, ball-park bounds.

[9] We scale up the throughput obtained by TCP connections to saturate at least one link in the network.

n	α_{TCP}	$\alpha_{max-min}$
5	2.890	2.778
6	3.559	3.273
7	2.137	3.5
8	2.061	3.764
9	2.304	4.764
10	2.525	5.263

Table 1: Majorization index for the first family with varying number of nodes.

The second family: Max-min fair allocations are fairer The second family has $n-1$ long connections as before, and n short connections. Again, all these connections are bulk connections. However, in this family, all the short connections are from node v_0 to node v_1. This family admits a most majorized allocation – just divide the bandwidth equally between all the connections. This corresponds to max-min fairness, and hence the max-min fair solution has a majorization index of 1. In this scenario, all TCP connections see the same drop probability, but the RTT experienced by the long connections is roughly n times that of a short connection. Hence, using the TCP equation, the long connections will obtain bandwidths that would be roughly $1/n$ times those obtained by short connections, resulting in a majorization index of $\Theta(n)$ for TCP. Clearly, TCP allocations are not as fair as max-min in this family. Simulations confirm this conclusion; the details of the simulations are omitted.

4.2 The fairness of TCP Vegas

For TCP Vegas, we will use the work of and Low, Peterson, and Wang [23]. They demonstrate that TCP Vegas attempts to maximize the function

$$U(x) = \sum_i \log x_i$$

where $x = \langle x_1, x_2, \ldots, x_n \rangle$ is the vector of bandwidths received by the n TCP Vegas connections[10]. This expression is the culmination of a series of results on the utility function for TCP variants, starting with Kelly's work [17].

The set of constraints for the bandwidth allocation problem is linear, and hence the set of feasible solutions is a convex set. Further, the function $U(x)$ is symmetric and strictly concave in each argument. The following lemma follows easily from the above two statements; the proof is omitted.

[10] There is another interpretation of the TCP Vegas protocol which results in a different utility function; see [23] for details.

Lemma 4.1 *There is a unique vector x that maximizes $U(x)$ for the bandwidth allocation problem.*

By theorem 2.1, if a most majorized vector existed for a given instance of the bandwidth allocation problem, then this vector would also maximize $U(x)$. Combining theorem 2.1 with lemma 4.1, the following theorem is immediate:

Theorem 4.1 *If a most majorized allocation exists for a given instance of the bandwidth allocation problem, then TCP Vegas attempts to achieve this allocation, assuming that the TCP Vegas utility function is $U(x) = \sum_i \log x_i$.*

This property would hold for any variant of TCP that has a symmetric concave utility function and a unique optimum solution. In fact, this property also holds for max-min fair allocations. While theorem 4.1 is interesting, we still need to evaluate TCP Vegas in terms of its majorization index when a most majorized vector does not exist.

On the first family(figure 2), let y be the bandwidth allocated by TCP Vegas to each short connection. Then by symmetry, each long connection gets a bandwidth of $(1-y)/(n-1)$ and $U(x) = n \log y + (n-1) \log \frac{1-y}{n-1}$. This is maximized when $n(1-y) = (n-1)y$, or $y \approx 1/2$. Thus the short connections get roughly half a unit of bandwidth and the long connections get roughly $1/(2n)$. The majorization index of this allocation is approximately 2 ie. $\Theta(1)$, which is much better than $\Theta(\sqrt{n})$ for TCP-Reno or $\Theta(n)$ for max-min fair allocations.

For the second family (figure 3), theorem 4.1 implies that the majorization index for Vegas is $\Theta(1)$, which is the same as max-min fair allocations, but much better than that of Reno ($\Theta(n)$).

Now consider a third family of networks. Again the underlying topology is the same. There are n short connections, one on each link, and 1 long connection. The majorization index of the max-min fair allocations on this family of networks is roughly 2. Using simple calculus like we did for the first family, it is straightforward to prove that the long connection receives bandwidth $1/(n+1)$ using Vegas, resulting in a majorization index of $\Theta(n)$. Hence, TCP Vegas and max-min fairness are incomparable.

Open problem: Approximately majorized TCP? From the above examples, it is clear that each of max-min fairness, TCP Reno, and TCP Vegas can have a majorization index as large as $\Theta(n)$, albeit in different networks. This is an unsatisfactory state of affairs [11] and motivates the following ambitious research prob-

[11] At least from a theoretical point of view; TCP Reno works quite well in most practical settings as demonstrated by the immense success of the Internet.

lem: *Design a version of TCP that is guaranteed to have a logarithmic majorization index.*

5 Connections to Proportional Fairness

Kelly, Maulloo, and Tan [18] studied the following optimization problem:

$$\text{Maximize } \textstyle\sum_i U_i(x_i) \text{ subject to:}$$
$$Ax \leq C$$
$$x \geq 0$$

Here x is interpreted as a vector of bandwidth allocations to connections. $Ax \leq C$ represent the capacity constraints. The i-th user is assumed to have a (potentially different) concave utility function U_i. They present an elegant distributed paradigm for solving this optimization problem. The resulting optimum solution is *weighted proportionally fair* with respect to some shadow weights computed during the distributed optimization; a formal definition of proportional fairness requires careful notation, and is beyond the scope of this paper. The proportional fairness framework is quite different from the approximate majorization framework presented in this paper. The former results in *exact* optimization of a *known* utility function. The latter is more suited to the case where the function to be optimized is not known, as in the case of fairness, since it simultaneously approximates a very large class of functions. Nonetheless, there are some interesting connections between the two models.

If the utility functions U_i are all identical, then the joint utility is a symmetric concave function. In this case, a most-majorized solution would also be proportionally fair. An α-fair solution would yield an α-approximation for the optimum objective value without needing to know the utility function; also $\alpha^* = O(\log n)$.

One major disadvantage of the approximate majorization framework presented in this paper (as compared to the proportional fairness work) is the lack of efficient distributed algorithms for finding globally α-fair vectors; all the algorithms presented in this paper are centralized. Finding such distributed schemes is a challenging open problem. Our approximation majorization framework can handle multivariate functions, whereas the framework of Kelly *et al.* can only handle sums of univariate functions. On the other hand, the proportional fairness framework allows optimization even when the U_i's are different ie. the joint utility function is not symmetric.

We now present a simple example where the approximate majorization framework can be applied even though the joint utility function is not symmetric.

5.1 Weighted bandwidth allocation

Consider the problem where bandwidth needs to be allocated to n connections, and the utility for the i-th connection is $w_i x_i$, where x_i is the bandwidth allocated to the i-th connection, and w_i is the *weight* of this connection. The joint utility function $\sum_i w_i x_i$ is not symmetric in the variables x_i. To remedy the situation, we can rewrite the problem in terms of the variables y_i, where $y_i = w_i x_i$. The joint utility function becomes symmetric in y_i now. However, the structure of the constraints changes; in particular, the inherent imbalance (see section 3 for a definition of inherent imbalance) of the problem might be different when the problem is restated in terms of the variables y_i. The following lemma states that the inherent imbalance of the problem can not increase too much; we omit the proof of this lemma.

Lemma 5.1 *Let w_{max} and w_{min} denote the largest and the smallest weights for any connection. Further, let I_y denote the inherent imbalance of the problem when written in terms of the variables y_i, and let I_x denote the inherent imbalance of the problem when written as a function of the variables x_i. Then, $I_y \leq I_x \cdot \frac{w_{max}}{w_{min}}$.*

Recall that the inherent imbalance for the unweighted bandwidth allocation problem is $O(n)$. Using theorem 3.1, we can claim the following:

Theorem 5.1 *For the weighted bandwidth allocation problem, there exists a solution that is $O(\log n + \log \frac{w_{max}}{w_{min}})$-fair.*

6 Conclusion and Open Problems

A good measure of fairness is an essential prerequisite for a systematic development of fair resource allocation protocols as well as for a systematic evaluation of the fairness of existing protocols. We proposed a majorization-based framework to study protocol fairness. We observed that majorization subsumes all the measures of fairness that we know of, and a most majorized allocation (if it exists) would be fair under all these measures. We gave a simple example to show that a most majorized vector need not always exist, and presented a definition of approximate majorization. We showed that a globally α-fair allocation under this definition is at least $1/\alpha$ as fair as any other allocation for a very general class of definitions of fairness. We explained how the value α can be computed for a linear program and related the smallest possible α to the inherent imbalance of the problem.

Finally, we performed a preliminary case study of the fairness of TCP as a bandwidth allocation protocol to illustrate our framework. We discovered that TCP is incomparable to max-min fairness in terms of the majorization index. We also compared our framework to that of proportional fairness and discovered it to be more general in some respects and less general in others. The proportional fairness framework provides a distributed protocol for optimizing; such a distributed protocol is missing from our framework. This motivates the following two open problems:

1. Design efficient distributed algorithms for finding $O(\log n)$-fair solutions to the bandwidth allocation problem.
2. Design new versions of TCP that are guaranteed to be logarithmically-fair.

References

[1] Y. Afek, Y. Mansour, and Z. Ostfeld. Convergence complexity of optimistic rate based flow control algorithms. *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 89–98, 1996.

[2] Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: A simple and effective flow control scheme. *ACM SIGCOMM*, pages 169–182, 1996.

[3] Majorized programs, approximate fairness, and distributional optimization. *Manuscript (author list omitted to preserve anonymity)*, Oct 2001.

[4] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1992.

[5] P. Brown. Resource sharing of TCP connections with different round-trip times. *IEEE Infocom*, March 2000.

[6] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[7] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26, 1990.

[8] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic. *Computer Communications Review*, 21(5):30–47, October 1991.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, August 1993.

[10] A. Goel, A. Meyerson, and S. Plotkin. Combining fairness with throughput: Online routing with multiple objectives. *32nd ACM Symposium on Theory of Computing*, 2000.

[11] A. Goel, A. Meyerson, and S. Plotkin. Approximate majorization and fair online load balancing. *To appear in ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[12] M. Grotschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, 1987.

[13] G.H. Hardy, J.E. Littlewood, and G. Polya. Some simple inequalities satisfied by convex functions. *Messenger Math.*, 58:145–152, 1929.

[14] G.H. Hardy, J.E. Littlewood, and G. Polya. *Inequalities*. 1st ed., 2nd ed. Cambridge University Press, London and New York., 1934, 1952.

[15] J.M. Jaffe. Bottleneck flow control. *IEEE Trans. on Communications*, COM-29(7):954–962, July 1981.

[16] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *DEC Research Report TR-301*, September 1984.

[17] F. Kelly. Mathematical modeling of the internet. *Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, pages 105–116, July 1999.

[18] 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.

[19] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 1999.

[20] C. E. Koksal, H. I. Kassab, and H. Balakrishnan. An analysis of short-term fairness in wireless media access protocols. *ACM SIGMETRIC*, June 2000.

[21] A. Kumar and J. Kleinberg. Fairness measures for resource allocation. *Proceedings of 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[22] M.O. Lorenz. Methods of measuring concentrations of wealth. *J. Amer. Statist. Assoc.*, 9:209–219, 1905.

[23] S. Low, L. Peterson, and L. Wang. Understanding TCP Vegas: a duality model. *Proceedings of ACM Sigmetrics*, 2001.

[24] A.W. Marshall and I. Olkin. *Inequalities: theory of majorization and its applications*. Academic Press (Volume 143 of Mathematics in Science and Engineering), 1979.

[25] ns: UCB/LBNL/VINT network simulator. *http://www-mash.cs.berkeley.edu/ns/*.

[26] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, Vancouver, CA, 1998.

[27] R. Pan, B. Prabhakar, and P. Psounis. Choke, a stateless active queue management scheme for approximating fair bandwidth allocation. *IEEE INFOCOM*, 2000.

[28] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks - the single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[29] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math of Oper. Research*, pages 257–301, 1994.

[30] I. Stoica and H. Zhang. Providing guaranteed services with no per flow management. *ACM SIGCOMM*, 1999.

[31] M. Vojnovic, J.-Y. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. *IEEE Infocom*, March 2000.

A Detailed Version of Section 3 with Proofs

Many optimization problems can be reduced to linear or integer programs. One example of such a problem is the bandwidth allocation problem discussed in this paper. Another is multi-commodity flow. We will present a general framework for finding approximately-fair solutions to sets of linear constraints. Recall that a globally α-fair solution is α-supermajorized by every other feasible solution. Of course, a solution with $\alpha = 1$ may not exist. In the case of linear programs, we can always find the solution which minimizes α exactly in polynomial time. In the case of integer programs, it may sometimes be possible to round the globally α-fair fractional solution while maintaining fairness.

A.1 Finding P_j^* and α^*

Suppose we are given a set of linear constraints. Instead of being asked to produce a feasible solution which maximizes (or minimizes) some function, we would like to find a globally α-fair $\vec{x}$ for the smallest possible value of α.

Define P_j^* to be the maximum possible value for the jth prefix of $\vec{x}$. Thus $P_j^* = \max_{\vec{x} \in X} \sum_{i=1}^{j} x_{(i)}$. The property we need to guarantee is exactly $\alpha \sum_{i=1}^{j} x^*_{(i)} \geq P_j^*$ by the definition of majorization. This observation results in a two-step algorithm to find $\vec{x}^*$. First, we will determine the optimum value for each prefix. Then we will impose the constraint that each prefix of $\vec{x}^*$ is approximately its optimum value, and attempt to minimize α. Each of these steps can be performed by solving linear programs. We will present exponential-sized linear programs which can be solved using the ellipsoid method. We will then describe ways to modify these programs in order to speed up the running time; the general process of solving for the prefixes and then computing $\vec{x}^*$ will remain the same.

Feasibility requires that $A\vec{z} \leq \vec{b}$ for some matrix A, constant vector $\vec{b}$, and variable vector $\vec{z}$ (which includes the x_i but may also include other additional variables). We will add a number of new constraints to the linear program to produce the following.

Maximize P_j subject to:

$$A\vec{z} \leq b$$

$$\sum_{i \in S} x_i \geq P_j \text{ for all } S \subseteq \{1, \cdots, n\} \text{ with } |S| = j$$

This linear program finds the largest possible value for the jth prefix. The linear program has exponentially many equations, because there will be an exponential number of subsets S. Such linear programs can be solved using the ellipsoid method [12] provided we can produce a separation oracle. In this case, the separation oracle simply sums the j smallest x_i; if their sum is at least P_j then our solution is feasible, if not then the equation indicating that that particular subset S of the x_i sums to at least P_j is violated. The ellipsoid method is polynomial time (although inefficient in practice), and we can thus produce P_j^* in polynomial time. Repeating this n times (once for each j) provides P_j^* for every prefix.

We now need to produce the globally α-fair vector $\vec{x}^*$ along with the optimum value α^*. This can be done by solving the following linear program.

$$\text{Maximize } \gamma \text{ subject to:}$$
$$A\vec{z} \leq b$$
$$\textstyle\sum_{i \in S} x_i \geq \gamma P^*_{|S|} \text{ for all } S \subseteq \{1, \cdots, n\}$$

The solution vector to this linear program is $\vec{x}^*$, with $\alpha = 1/\gamma$. Again the linear program has exponentially many constraints, but again we can solve it using the ellipsoid method. The separation oracle simply checks the values of all prefixes for our current vector $\vec{x}$ and compares them to γP_j^*. The solution is feasible if and only if all of these equations are satisfied. This yields a polynomial-time algorithm for computing the minimum possible α^* and the corresponding vector of variables. If we wish to optimize something else while maintaining approximate fairness as a constraint, we observe that once α^* is known, we can add a constraint specifying $\gamma \geq 1/\alpha^*$ and change our objective function (the same trick also works for the more efficient algorithms detailed below).

A.1.1 Reducing the LP size

Linear programming has been studied extensively, and a number of fast algorithms are known for solving (or approximately solving) linear programs. Practically speaking, the ellipsoid method is not usually very fast; if we could reduce the number of constraints to polynomial, then we could solve our linear programs much more efficiently.

Consider the following linear program for finding P_j^*:

$$\text{Maximize } \left(\textstyle\sum_{i=1}^{n} x'_i\right) - (n-j)U \text{ subject to:}$$
$$A\vec{z} \leq b$$
$$x'_i \leq x_i \text{ for all } i \in \{1, \cdots, n\}$$
$$x'_i \leq U \text{ for all } i \in \{1, \cdots, n\}$$

The above linear program is polynomial-sized; in fact it adds only $2n$ constraints and $n+1$ variables, so it's essentially the same size as the original linear program and can be solved efficiently.

Lemma A.1 *The above linear program produces P_j^*.*

Proof: Consider the optimum $\vec{x}$. We immediately observe that $x_{(i)} = x_{(i+1)}$ for all $i \geq j$. Suppose for some $i < j$ we have $x_{(i)} > 0$ and $x_{(i)} < x_{(i+1)}$. If we increase $x_{(i)}$ by a small ϵ and decrease $x_{(i')}$ by $\epsilon/(j-i)$ for all $i' > i$ then we will remain in the polytope (since the order of the x_i and the sum of the jth prefix are unchanged). On the other hand, if we decrease $x_{(i)}$ by the same ϵ and increase $x_{(i')}$ by $\epsilon/(j-i)$ for all $i' > i$ then we again remain in the polytope. Since one of these two directions does not make the solution worse, we can conclude that for every i, either $x_{(i)} = 0$ or $x_{(i)} = x_{(i+1)}$. In order to be in the polytope, if the smallest i coordinates of $\vec{x}$ are zero, the remaining coordinates will all have to be at least $1/(j-i)$. We can sort the x_i in decreasing order of their cost c_i; since swapping the values of the coordinates of $\vec{x}$ cannot cause it to leave the polytope, it is clear that the largest c_i will correspond to the smallest x_i and so forth. Thus there are only j possible vectors $\vec{x} \in P$ which could minimize $\vec{c} \bullet \vec{x}$ and we can produce them all (and check their costs) in polynomial time. ■

We still need to produce the optimum α and the vector $\vec{x}^*$. For that, we write the following linear program.

Maximize γ subject to:

$$A\vec{z} \leq b$$

$$x_i^j \leq x_i \text{ for all } i \text{ and } j$$

$$x_i^j \leq U^j \text{ for all } i \text{ and } j$$

$$\left(\sum_{i=1}^n x_i^j\right) - (n-j)U^j \geq \gamma P_j^* \text{ for all } j$$

This linear program adds $n^2 + n$ new variables and $2n^2 + n$ constraints. This may make it larger than the original LP, but it is still polynomial in size.

Lemma A.2 *The above linear program produces a globally α-fair solution for the minimum value of $\alpha = \alpha^*$.*

Proof: If we set $\vec{x} = \vec{x}^*$ along with $U^j = x_{(j)}$, then arguments similar to those of lemma A.1 show that the optimum γ is feasible. It follows that $\vec{x}^*$ along with $\gamma = 1/\alpha^*$ is a feasible solution.

On the other hand, consider any feasible solution. We observe that $\sum_{i=1}^n x_i^j - (n-j)U^j$ is at most the jth prefix of our current vector $\vec{x}$. It follows that $\vec{x}$ must have prefix j at least equal to γP_j^*. Any feasible solution immediately produces a globally $1/\gamma$-fair vector. It follows that the optimum solution (which maximizes γ) must minimize α while providing a globally α-fair solution. ■

A.1.2 Fast Fairness Approximations for Packing Problems

Packing constraints are a set of linear constraints $A\vec{z} \leq \vec{b}$ with the additional properties that matrix A has nonnegative entries and vector $\vec{z}$ is constrained to have nonnegative coordinates. Optimization problems on packing constraints can be approximated very efficiently [29]. We would like to be able to apply fast approximation techniques for packing problems to the problem of finding a globally α-fair feasible solution on a set of packing constraints. Unfortunately, the linear programs described in the previous sections are *not* packing programs. Even if the original matrix A is nonnegative, the constraints added to compute fairness do not have the packing property. We can rewrite the linear program we need to solve in order to find prefix P_j^* as follows:

$$\text{Minimize } \lambda_j \text{ subject to:}$$
$$A\vec{z} \leq \lambda_j b$$
$$\textstyle\sum_{i \in S} x_i \geq 1 \text{ for all } S \subseteq \{1, \cdots, n\} \text{ with } |S| = j$$

Here $P_j^* = 1/\lambda_j$. We will represent the set of equations which insures each subset of j variables sums to at least one by the polytope $\mathcal{P}_j$. Provided we can, for any $\vec{c} \geq 0$, produce the $\vec{x} \in \mathcal{P}_j$ which minimizes $\vec{c} \bullet \vec{x}$, we can run the algorithm of Plotkin, Shmoys, and Tardos [29] to produce a fast approximation to λ_j. The existence of such a polynomial time *minimization oracle* is the subject of the following theorem.

Theorem A.1 *We can produce $\vec{x} \in \mathcal{P}_j$ to minimize $\vec{c} \bullet \vec{x}$ in time $O(n \log n)$.*

The crucial step is to prove that there exists an optimum solution such that, for some threshold value c_T, all x_i with $c_i > c_T$ are zero and all other x_i are identical. Then we can sort the x_i and check each of the n possible values of c_T to find the optimum solution. Here is the detailed proof: **Proof:** Consider the optimum $\vec{x}$. We immediately observe that $x_{(i)} = x_{(i+1)}$ for all $i \geq j$. Suppose for some $i < j$ we have $x_{(i)} > 0$ and $x_{(i)} < x_{(i+1)}$. If we increase $x_{(i)}$ by a small ϵ and decrease $x_{(i')}$ by $\epsilon/(j-i)$ for all $i' > i$ then we will remain in the polytope (since the order of the x_i and the sum of the jth prefix are unchanged). On the other hand, if we decrease $x_{(i)}$ by the same ϵ and increase $x_{(i')}$ by $\epsilon/(j-i)$ for all $i' > i$ then we again remain in the polytope. Since one of these two directions does not make the solution worse, we can conclude that for every i, either $x_{(i)} = 0$ or $x_{(i)} = x_{(i+1)}$. In order to be in the polytope, if the smallest i coordinates of $\vec{x}$ are zero, the remaining coordinates will all have to be at least $1/(j-i)$. We can sort the x_i in decreasing order of their cost c_i; since swapping the values of the coordinates of $\vec{x}$ cannot cause it to leave the polytope, it is clear that the largest c_i will correspond to the smallest x_i and so forth. Thus there are only j possible vectors $\vec{x} \in P$ which could minimize $\vec{c} \bullet \vec{x}$ and we can produce them all (and check their costs) in polynomial time.

■

This gives a fast algorithm to produce approximately optimum prefixes for each j. We still need to explain how to produce an approximately-fair vector. We can rewrite the linear program to find α^* as follows:

Minimize α subject to:

$$A\vec{z} \le \alpha b$$

$$\textstyle\sum_{i \in S} x_i \ge P^*_{|S|} \text{ for all } S \subseteq \{1, \cdots, n\}$$

We will need a minimization oracle which can, given a vector $\vec{c}$, produce the vector $\vec{x} \in P$ which minimizes $\vec{c} \bullet \vec{x}$. In this case our polytope is the set of vectors $\vec{x}$ which have prefix j at least equal to P^*_j for every j.

Theorem A.2 *There exists a minimization oracle for the above linear program that runs in time $O(n \log n)$.*

The minimizing vector $\vec{x}$ has $x_{(1)} = P^*_1$ and $x_{(j+1)} = P^*_{j+1} - P^*_j$. This can be computed in $O(n \log n)$ time by observing that $c_i > c_j \Rightarrow x_i \le x_j$. Thus if the original linear program constraints were packing constraints, we can produce fast globally α-fair vector solutions while guaranteeing $\alpha \le \alpha^*(1+\epsilon)$. This is particularly useful for the bandwidth allocation problem with fixed routes.

A.2 Existence of small α^*

The previous section described methods for finding a globally α-fair vector for a linear program. In general, the minimum α such that there exists a vector which α-supermajorizes all others could be very large. We will describe several upper bounds on α^* in the worst case. We assume only that all feasible vectors $\vec{x}$ have nonnegative coordinates, and that for any feasible $\vec{x_1}, \vec{x_2}$ the weighted average $\beta\vec{x_1} + (1-\beta)\vec{x_2}$ is feasible for $0 \le \beta \le 1$. We will use the term "nonnegative convex program" to apply to such a set of feasible points.

Lemma A.3 *For any nonnegative convex program, $\alpha^* \le n$ where n is the dimension of $\vec{x}$.*

Proof: Let $\vec{x_j}$ be the solution which optimizes prefix j. We define $P_j(\vec{x})$ to be the jth prefix of vector $\vec{x}$; thus we know that $P_j(\vec{x_j}) = P^*_j$ where P^*_j is the optimum prefix. Since any weighted average of solutions is feasible, one possible solution to the convex program is $\vec{x} = \sum_{i=1}^{n} \frac{1}{n}\vec{x_i}$. We observe that $P_j(\vec{x}) \geq \frac{1}{n}\sum_{i=1}^{n} P_j(\vec{x_i}) \geq \frac{1}{n}P_j(\vec{x_j}) \geq \frac{1}{n}P_j^*$. It follows that $\vec{x}$ is globally n-fair. ■

This result is tight, in that we can construct examples where $\alpha^* = n$. However, in many cases the optimum prefixes are fairly similar in value, and the optimum α^* is much smaller than n. We will formalize this below:

Definition 2 *The* **inherent imbalance** *of a problem is* $\frac{P_n^*}{nP_1^*}$.

The inherent imbalance is the ratio of the maximum average utility that can be achieved in any allocation to the maximum utility for the poorest individual that can be achieved in any allocation. Intuitively, the larger the inherent imbalance, the less likely it will be that there exists a most-fair vector with a small α.

Theorem A.3 *For any set of convex constraints,* $\alpha^* \leq 2 + 2\log\frac{P_n^*}{nP_1^*}$.

Proof: We observe that for any vector $\vec{x}$, $P_n(\vec{x}) \geq \frac{n}{j}P_j(\vec{x})$. This holds because the $n - j$ terms not included in prefix j are all larger than the average term (or any term) which is included. Let $y_1 = 1$. We define y_i to be the smallest number such that $P_{y_i}^*/y_i \geq 2P_{y_{i-1}}^*/y_{i-1}$. Let y_k be the largest of the y_i. We define vector $\vec{x} = \sum_{i=1}^{k}\frac{1}{k}\vec{x_{y_i}}$. Consider prefix j of this vector. By the definition of y_i, we know there exists some i such that $y_i \leq j$ and $P_j^*/j \leq 2P_{y_i}^*/y_i$. It follows that $P_j(\vec{x_{y_i}}) \geq \frac{j}{y_i}P_{y_i}(\vec{x_{y_i}})$ from which we can conclude that $P_j(\vec{x_{y_i}})/j \geq P_{y_i}^*/y_i \geq P_j^*/2j$. This means that $P_j(\vec{x}) \geq P_j^*/2k$ and $\vec{x}$ is globally $2k$-fair. But what is k? We know that P_j^*/j form a nondecreasing sequence from P_1^* to P_n^*/n. It follows that since each y_i increases the value in this sequence by a factor of 2, $k \leq 1 + \log(\frac{P_n^*}{nP_1^*})$. This proves the theorem. ■

Here n is the dimension of the vector over which we want the fairest solution. In the case of multi-commodity flow, n would represent the number of commodities (*not* the size of the graph).