No Justified Complaints: On Fair Sharing of Multiple Resources

Danny Dolev
Dept. Computer Science
The Hebrew University
Jerusalem, Israel
dolev@cs.huji.ac.il

Dror G. Feitelson
Dept. Computer Science
The Hebrew University
Jerusalem, Israel
feit@cs.huji.ac.il

Joseph Y. Halpern*
Computer Science Dept.
Cornell University
Ithaca, NY
halpern@cs.cornell.edu

Raz Kupferman
Institute of Mathematics
The Hebrew University
Jerusalem, Israel
raz@math.huji.ac.il

Nathan Linial
Dept. Computer Science
The Hebrew University
Jerusalem, Israel
nati@cs.huji.ac.il

ABSTRACT

Fair allocation has been studied intensively in both economics and computer science. This subject has aroused renewed interest with the advent of virtualization and cloud computing. Prior work has typically focused on mechanisms for fair sharing of a single resource. We consider a variant where each user is entitled to a certain fraction of the system's resources, and has a fixed usage profile describing how much he would want from each resource. We provide a new definition for the simultaneous fair allocation of multiple continuously-divisible resources that we call *bottleneck-based fairness* (BBF). Roughly speaking, an allocation of resources is considered fair if every user either gets all the resources he wishes for, or else gets at least his entitlement on some *bottleneck resource*, and therefore cannot complain about not receiving more. We show that BBF has several desirable properties such as providing an incentive for sharing, and also promotes high overall utilization of resources; we also compare BBF carefully to another notion of fairness proposed recently, *dominant resource fairness*.

Our main technical result is that a fair allocation can be found for every combination of user requests and entitlements. The allocation profile of each user is proportionate to the user's profile of requests. The main problem is that the bottleneck resources are not known in advance, and indeed one can find instances that allow different solutions with different sets of bottlenecks. Therefore known techniques such as linear programming do not seem to work. Our proof uses tools from the theory of ordinary differential equations, showing the existence of a sequence of points that converge upon a solution. It is constructive and provides a practical method to compute the allocations numerically.

*Much of this work was done while the author was on sabbatical leave at Hebrew University.

ITCS'12 January 06-08 2012, Cambridge, MA, USA

Categories and Subject Descriptors

D.4.1 [**OPERATING SYSTEMS**]: Process Management—*Scheduling*; K.6.2 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Installation Management—*Pricing and resource allocation*

General Terms

Management, performance

Keywords

Resource allocation, fair share, bottleneck

1. INTRODUCTION

Fair allocation has been widely studied both in economics and computer science (See [4, 11, 19] for a sample of the wide-ranging work in this area.) Generally speaking, the notion of fairness may pertain to mechanisms like bargaining and their relationship to ethical issues (e.g. [18]). We assume fairness to mean that each user has a certain level of resources to which he is entitled, and take an allocation to be fair if each user indeed gets at least this level. But how exactly should the entitlements be interpreted? Specifically, what does it mean that a user is "entitled to 20% of the system"? Is this a guarantee for 20% of the CPU cycles? Or maybe 20% of each and every resource? And what should we do if the user requires, say, only 3% of the CPU, but over 70% of the network bandwidth? Reserving 20% of the CPU will cause obvious waste, while curbing the network usage would be unreasonable if no other user can take up the slack.

Our goal in this paper is to define a notion of fair allocation that applies when multiple, continuously-divisible resources are to be allocated. In a nutshell, we observe that allocations need only focus on contended resources. Our scheme, which we call *bottleneck-based fairness* (BBF), therefore requires that each user receives his entitlement on at least one bottleneck resource. We claim that the user can then not justify complaining about not getting more. We then show that a BBF allocation is guaranteed to exist.

2. CONTEXT AND PREVIOUS WORK

This work concerns mostly a *collaborative* environment. The different users may represent, e.g., different activities of one orga-

nization or even a set of computational systems that are all owned by the same entity. Another possible scenario is an installation put together by a group of mutually trusting partners. In this context a user's entitlement may represent that user's share in the investment that created the installation and the shared resources.

Simple and direct approaches for scheduling according to entitlements include lottery scheduling [17] and economic models [16], where each process's relative share of the resources is expressed by its share of lottery tickets or capital. Another popular approach is based on *virtual time* [7, 14]. The idea is that time is simply counted at a different rate for different processes, based on their relative allocations. In networking research the most common approach is max-min fairness, where the goal is to maximize the minimal allocation to any user [15]. Using weights this can be adjusted to support diverse entitlements.

The main drawback of the approaches above is that they focus on one resource—the CPU or the bandwidth of a link—irrespective of contention. This may be inappropriate when the goal is to achieve a predefined allocation of the resources. For example, by trying to promote an I/O-bound process (because it deserves more of the CPU than it is using), we might turn the disk into a bottleneck, and inadvertently allow the internal scheduling of the disk controller to dictate the use of the whole system.

In order to avoid such problems, it has recently been suggested that fair-share scheduling be done in two steps [2, 8]: first, identify the resource that is the system bottleneck, and then enforce the desired relative allocations on this resource. This approach is in line with basic results in performance evaluation, as it is well known that the bottleneck device constrains system performance (this is, after all, the definition of a bottleneck) [13]. An important manifestation of this result is that, in a queueing network, most of the clients will always be concentrated in the queue of the bottleneck device. This implies that scheduling the bottleneck device is the only important activity, and moreover, that judicious scheduling can be used to control relative resource allocations. The fair usage of the bottleneck resource induces some level of usage of other resources as well, but this need not be controlled, because there is sufficient capacity on those resources for all contending processes.

The question is what to do if two or more resources become bottlenecks. This may easily happen when different processes predominantly use distinct resources. For example, consider a situation where one process makes heavy use of the CPU, a second is I/O-bound, while a third process uses both CPU and I/O, making both bottlenecks.

There have been a number of approaches suggested for fairly allocating multiple resources. Most relevant to our work is the recently proposed notion of *dominant resource fairness* (DRF) [10]. DRF does not explicitly consider bottlenecks, but rather focuses on each user's maximal usage of any single resource. We describe this in more detail and compare it with our definition in Section 4.

3. BOTTLENECK BASED FAIRNESS WITH MULTIPLE BOTTLENECKS

Consider a setting with N users and m resources (e.g. CPU and network and disk bandwidth). Without loss of generality we assume that there is exactly one unit available of each resource. We assume that each user i is entitled to a fixed percentage e_i of the full capacity, and hence of each resource, where $\sum_i e_i = 1$. Alternatively, the actual number of users may be $M \gg N$, where all these M users are treated equally (so each user's entitlement is $\frac{1}{M}$). However, there are only N *types* of users, of which Me_i are of type i. It is not hard to verify that there is no loss of generality in handling all users of the same type equally. In this case we are led to the problem formulation as described next.

Each user i requests a fraction r_{ij} of resource j. Obviously the interesting situation is when for each i there exists a j such that $r_{ij} > e_i$, and for every j, $\sum_i r_{ij} > 1$. Our goal is to find a set of allocations that allow us to exploit complementary usage profiles to achieve high utilization, but at the same time respect the different entitlements. By respecting the entitlements, the allocations can be claimed to be fair.

An important attribute of our user model is that the request profile of each user is *fixed*. Thus the allocation to user i is characterized by a single factor x_i, rather than a separate factor x_{ij} for each resource j. The fraction allocated to i of each resource j will be $x_i r_{ij}$. This model reflects a situation where each user is engaged in a specific type of activity with a well-defined resource usage profile. For example, a user may be serving requests from clients over the Internet. Each request requires a certain amount of computation, a certain amount of network activity, and a certain amount of disk activity. If the rate of requests grows, all of these grow by the same factor. But if one resource is constrained, limiting the rate of serving requests, this induces a similar limit in the usage of all other resources. This is essentially the "knee model" of Etsion et al. [9], where I/O activity is shown to be linearly proportional to CPU allocation up to some maximal usage level. It also corresponds to the task model of Ghodsi et al. [10] when all tasks belonging to a user have identical resource requirements (which is indeed the specific model they use in their proofs). Note, however, that this is indeed a limiting assumption. Specifically, it excludes usage patterns where one resource is used to compensate for lack of another resource, as happens, for example, in paging, or when using compression to reduce bandwidth.

All the above leads to the following problem definition. We want to find $x_1, \ldots, x_N$ with $0 \leq x_i \leq 1$. Here x_i is the fraction of user i's request which will be granted. Feasibility of these x_i's means that our total consumption of each resource is at most one:

$$\forall j : \sum_i x_i r_{ij} \leq 1. \quad (1)$$

Those resources j for which equality holds in (1) are the *bottleneck resources*. These are important for our fairness condition, which we call the "No Justified Complaints" condition. The idea is that a user cannot justify complaining about his allocation if either he gets all he asked for, or else he gets his entitlement on some bottleneck, so giving him more would come at the expense of other users who have their own entitlements. This is formally expressed as:

$$\forall i : [x_i = 1] \vee [\exists j^* : (\textstyle\sum_k x_k r_{kj^*} = 1) \wedge (x_i r_{ij^*} \geq e_i)]. \quad (2)$$

In the sequel, we call this requirement *bottleneck based fairness* (BBF).

Note that it may happen that a user receives less than his entitlement on *other* resources, including other bottleneck resources, where the entitlement would seem to indicate that a larger allocation is mandated. This is where the fixed request profile assumption comes in. Recall that the factor x_i is common to all resources. Thus, giving a user a higher allocation on any resource implies that his allocation must grow on *all* resources. The original bottleneck resource j^* thus constrains all allocations, even on other bottleneck resources or resources that are not themselves contended.

Showing that a fair allocation according to this definition exists turns out to be surprisingly nontrivial. As far as we know, all obvious approaches (e.g., Linear Programming) seem to fail. We prove the existence of fair allocations in Section 5.

4. PROPERTIES OF BOTTLENECK BASED FAIRNESS

In this section, we discuss properties of BBF, and compare it to DRF. A user's dominant resource is the one where the user requires the largest fraction, i.e. $\text{argmax}_j\, r_{ij}$. Given the fixed request profile assumption, in any allocation the user's maximal usage of any resource will be his usage of the dominant resource. DRF fairness is then defined as equalizing these maximal usage levels across users, or more generally, making them proportional to the entitlements [10]. In the special case where each user can use the full capacity of some resource (so the dominant resource j satisfies $r_{ij} = 1$) this is equivalent (in our notation) to requiring that $x_i \propto e_i$.

An important difference between DRF and BBF is that DRF allocations can be found using an incremental algorithm [10]. Finding BBF allocations is harder, because we do not know in advance which resources will be the bottlenecks. But interestingly, the trajectory argument used in the proof that a BBF allocation exists is actually somewhat similar to the way that allocations are constructed for DRF.

4.1 Defining Fairness

BBF and DRF both define a notion of fairness across multiple resources. At a very basic level, the notion of fairness depends on perception of utility. In the context of allocating resources on computer systems, the utility is typically unknown. Consequently the notion of fairness is ill-defined.

To better understand the difference between utility and allocation, we recount an example used by Yaari and Bar-Hillel [18]. Jones and Smith are to share a certain number of grapefruit and avocados to obtain certain vitamins they need. They have different physiological abilities to extract these vitamins from the different fruit. The overwhelming majority (82%) of people polled agreed that the most fair division is one that gives them equal shares of extracted vitamins, despite being quite far from being equal numbers of actual fruit. Thus respondents clearly favored equal utility as the criterion for fairness. But such considerations would be impossible if you do not know their specific ability to extract vitamins, and that they actually only eat fruit for their vitamins.

When allocating resources we do not know the real utility of these resources for the users. We are therefore forced to just count the amount of resources being allocated. The difference between definitions of fairness is in how this counting is done. A simple counting rule is *asset fairness* [10] , where the fractions of all resources used are summed up. Thus the total allocation to user i is $\sum_j x_i r_{ij}$, and these allocations should be equalized across users. In DRF, only the largest fraction is considered. To be fair, all users should receive the same fractions of their respective dominant resources. In BBF we take a system-wide view, and only count the usage of bottleneck resources. Thus a user may receive more than his entitlement of non-bottleneck resources, but this is considered immaterial because there is no contention for those resources.

Interestingly, Ghodsi et al. prove that under DRF each user will actually be constrained by some resource that is a bottleneck [10]. However, their fairness criterion does not depend on this bottleneck, while ours does. As a result DRF may constrain the allocations of a non-bottleneck resource, and use this as an argument for being fair to the user. There seems to be no criterion by which to say that either DRF or BBF is fairer than the other. It may well be that a user derives much benefit from using the non-bottleneck resource, and therefore cutting him back on other resources is perfectly justified. But given that we do not *know* that this is the case, we suggest that it is safer to focus exclusively on the bottleneck resources.

To further support the focus on bottlenecks, we note that Yaari and Bar-Hillel extended the Jones and Smith example to a situation where Smith's ability to extract vitamins from fruit is extremely low. In this scenario, a large number of respondents no longer tolerated his inefficiency, and broke from the goal of achieving equal utility. An additional consideration that was not checked in the study was contention for limited resources. We conjecture that if a minimal level of vitamin was given as a requirement, especially if there were many potential beneficiaries rather than just two, respondents would be even less tolerant of inefficiency, and opt for fair division of the resources (or fruit).

In fact, Ghodsi et al. also mention bottleneck fairness in their description of DRF, but only as a secondary criterion. They define bottleneck fairness only when all users have the same dominant resource, essentially reducing the scope to the single bottleneck case. Our work is the first to extend this with a meaningful definition of fairness for multiple bottlenecks, and when the dominant resources are different.

4.2 Game-Theoretic Considerations

Ghodsi et al. [10] show that DRF has four desirable attributes, under the assumption that all tasks belonging to a user have identical resource requirements (in which case their model reduces to ours). We now show that BBF also has two of them, and explain the tradeoff regarding the other two.

The first requirement is what Ghodsi et al. call *sharing incentive*: each user i should be better off than he would be if he could work with only his entitlement e_i of each resource. Due to the fixed request profile assumption, if user i gets a fraction e_i of each resource, much of this capacity may remain unused. Indeed, user i is no better off from his point of view than if he got a fraction z of his requests, where $z = e_i / \max_j\{r_{ij}\}$. In a BBF allocation user i gets a fraction x_i of his requests, where $x_i r_{ij} \geq e_i$ for some bottleneck resource j. Thus $x_i \geq z$, which means that BBF provides an incentive for sharing resources, and thus allows the system to exploit situations where users have complementary requirements.

Another attribute is *Pareto efficiency*. This means that increasing the allocation to one user must come at the expense of another. This follows immediately from doing allocations based on bottlenecks.

The two properties that are more problematic are strategyproofness and envy-freedom. Being *strategyproof* means that users won't benefit from lying about their resource needs. Being *envy free* means that users don't prefer another user's allocation. Ghodsi et al. provide one approach to obtain a DRF allocation, and show that their approach is strategyproof, and its outcome satisfies envy freedom. As we show in Section 6 BBF may allow multiple solutions. This provides flexibility in the sense that secondary objectives may be used to select among the options. But it may also be susceptible to manipulations by users who try to influence the decision in their favor. But note that this effect is limited to the choice among alternative fair allocations. However, as Ian Kash [private communication, 2011] has shown, even for an instance of the problem (i.e., choice of $e_1, \ldots, e_n$ and r_{ij} for $1 \leq i \leq n$ and $1 \leq j \leq m$) that has a unique BBF allocation, strategyproofness does not hold.

4.3 Utilization Considerations

We now turn to a discussion of how fairness definitions may affect system utilization. First, we observe that if all users have the same dominant resource, DRF and BBF are equivalent. This follows since the common dominant resource is the only bottleneck. Thus the resulting utilization is the same. But in other cases there may be differences. In fact, it is easy to find examples where BBF

leads to higher overall utilization than DRF, and counterexamples where the opposite is true.

However, the following claim indicates that BBF may actually have an edge over DRF. Assume that every user has at least one resource that he can use to capacity (i.e. for every user i there is a resource j such that $r_{ij} = 1$). Consider those problems that have a full-utilization solution, meaning that we can find $x_1, \ldots x_N$ such that $x_i \geq e_i$ and, for every resource j, we have $\sum x_i r_{ij} = 1$. We claim that all such cases are BBF solutions, but there exist such cases where the DRF solution exhibits very low utilization.

These assumptions are not as restrictive as they may seem. The requirement that each user has a resource he can use to capacity just means that users are greedy and want lots of power. Moreover, every DRF and BBF solution must be such that $x_i \geq e_i$ (in the case of BBF, this is because $x_i = 1$ or $x_i r_{ij} \geq e_i$ for some j and $r_{ij} \leq 1$). Since, in a full-utilization solution, all resources are bottlenecks, and each user has a resource where $x_i r_{ij} \geq e_i$ (namely, the resource j such that $r_{ij} = 1$), it is easy to see that a full-utilization solution satisfies BBF. But consider the following specific example where DRF does badly. Assume Kn users want resource 1 at full capacity. An additional n users want only a very small ϵ of resource 1, and $1/n$ of all the other resources. DRF will seek to give each user $\sim 1/Kn$ of its dominant resource, so these last n users will get $\sim 1/K$ of what they want. But BBF can opt to give the last n users their full request, at very small cost to the others. With small K (e.g. $K = 2$), DRF gives a third of the population just half of what they could get without really benefiting the others. With small n (e.g. $n = 1$) it reduces the utilization of all resources except the first to $1/K$.

This example in itself does not prove that BBF is superior to DRF. It might be the case that other examples will show large utilization differences in the other direction. We are currently attempting to achieve a more complete characterization of the relative utilization implications of BBF and DRF.

5. EXISTENCE OF A FAIR ALLOCATION

In this section, we prove that an allocation satisfying (1) and (2) always exists. Note, that (2) deals separately with the case where $x_i = 1$ and user i's request is respected in full, and where $x_i < 1$ and we need to at least give i his entitlement on some bottleneck resource. Consider the following simplification of (2), that leaves out the first disjunct:

$$\forall i \; \exists j^* : (\textstyle\sum_k x_k r_{kj^*} = 1) \wedge (x_i r_{ij^*} \geq e_i). \qquad (2')$$

We claim that, given a problem X, we can convert it to a problem X' such that an allocation $(x_1, \ldots, x_n)$ for X satisfies (1) and (2) iff $(x_1, \ldots, x_n)$ satisfies (1) and (2′) for X'. To convert X to X', we simply add N new dummy resources $r'_1, \ldots, r'_N$ such that $r'_{ij} = 1$ if $j = i$ and 0 otherwise. In light of this, the following theorem establishes that there always exists a solution that satisfies BBF.

Theorem 1. *Given*

- *entitlements $e_1, \ldots, e_N$ such that $e_i \geq 0$ for $i = 1, \ldots, N$ and $e_1 + \cdots + e_N = 1$, and*
- *resource requirements r_{ij} such that $0 \leq r_{ij} \leq 1$ and $r_{1j} + \cdots + r_{Nj} \geq 1$ for $i = 1, \ldots, N$ and $j = 1, \ldots, m$,*

there exists an allocation $x_1, \ldots, x_N$, where $0 \leq x_i \leq 1$ for $i = 1, \ldots, N$, such that (1) and (2′) hold.

5.1 A Few Simplifying Assumptions

Before proving the theorem, we make three simplifying assumptions, all without loss of generality.

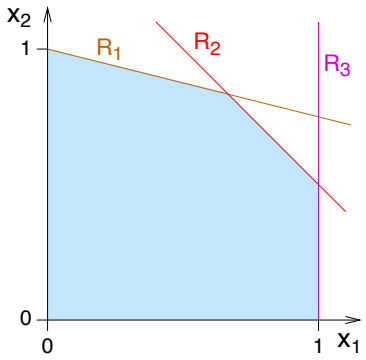

Figure 1: *Depiction of bounds on x_i values due to capacity constraints of resources, for $N = 2$ and $m = 3$.*

We can and will assume that $\max_j r_{ij} \geq e_i$, for each user i. Otherwise, we give user i everything he asked for, remove his requests, renormalize the entitlements of the remaining users so that they still sum to 1, renormalize the remaining capacity of the different resources so that it is still 1, and renormalize the remaining requests by the same factors. Again, it's not hard to see that this can be done without changing the problem and the possible outcomes.

Also, say that resource j is *dominated* if the inequality $x_1 r_{1j} + \cdots + x_N r_{Nj} \leq 1$ is a consequence of all other inequalities $\{x_1 r_{1s} + \cdots + x_N r_{Ns} \leq 1 | s \neq j\}$. Clearly, the existence of such an inequality can be efficiently detected by standard linear programming methods. Again, dominated resources can be eliminated from the problem without any change.

We turn to prove the existence of a solution $x_1 \ldots x_N$ satisfying Theorem 1 under these simplifying assumptions. As mentioned, this is done without loss of generality, and a solution that is found under the simplifying assumptions can be easily turned into a solution for the original formulation of the problem.

5.2 Proof Structure

We first establish some notation. The set of all feasible solutions is the polytope $\mathcal{D} \subseteq (\mathbb{R}^+)^N$, where

$$\mathcal{D} = \{ (x_1, \ldots, x_N) : 0 \leq x_i \leq 1, \; \forall i \quad \text{and} \quad x_1 r_{1j} + \cdots + x_N r_{Nj} \leq 1, \; \forall j \}.$$

This is illustrated in Fig. 1 for $N = 2$.

For $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathcal{D}$, the set of bottleneck resources is

$$J(\boldsymbol{x}) = \{j : \; 1 \leq j \leq m, \quad x_1 r_{1j} + \cdots + x_N r_{Nj} = 1\}.$$

The solution $\boldsymbol{x}$ that we seek must clearly reside on the boundary of $\mathcal{D}$, for $J(\boldsymbol{x})$ is empty when $\boldsymbol{x}$ is in $\mathcal{D}$'s interior. So, paraphrasing (2′), our goal is to find an allocation $\boldsymbol{x} = (x_1, \ldots, x_N)$, such that

$$\forall i \; \exists j^* \in J(\boldsymbol{x}) : x_i r_{ij^*} \geq e_i. \qquad (3)$$

This is exactly the source of our difficulty. Given the set of bottleneck resources, the problem of finding $\boldsymbol{x}$ is just a linear program. Specifically, given an arbitrary subset $I \subseteq \{1, \ldots, m\}$, the following decision problem is an LP: Is there an $\boldsymbol{x} \in \mathcal{D}$ for which $J(\boldsymbol{x}) = I$ such that condition (3) holds?

How can we overcome the difficulty involved in satisfying condition (3) without prior knowledge of the set $J(\boldsymbol{x})$? As a first step, we approximate the polytope $\mathcal{D}$ by a subset $\mathcal{Q} \subseteq \mathcal{D}$ that is convex and has a smooth boundary. Intuitively, $\mathcal{Q}$ "rounds off" the corners of $\mathcal{D}$ (see below for further discussion). Such a set $\mathcal{Q}$ is defined by *infinitely many* linear inequalities: For every hyperplane H that is

tangent to $\mathcal{Q}$ we write a linear inequality that states that $\boldsymbol{x}$ must reside "below" H. It would seem that this only complicates matters, replacing the finitely defined $\mathcal{D}$ by $\mathcal{Q}$. However, the problematic condition (3) takes on a much nicer form when applied to $\mathcal{Q}$, and becomes a very simple relation involving the contact point of H and $\mathcal{Q}$, the normal to H, and the vector e (see Equation (7) below). Moreover, using standard tools from the theory of ordinary differential equations, we can find a point on the boundary of $\mathcal{Q}$ where this relation holds.

To find the solution, we do not consider a single smooth $\mathcal{Q}$, but rather a whole parametric family $\mathcal{Q}_t$. This family has the properties that (a) the sets $\mathcal{Q}_t$ grow as the parameter t increases; (b) they are all contained in $\mathcal{D}$; and (c) as $t \to \infty$ the sets $\mathcal{Q}_t$ converge to $\mathcal{D}$. In the language of the description below, $\mathcal{Q}_t$ is defined as the set of those $\boldsymbol{x} \in \mathcal{D}$ for which $f(\boldsymbol{x}) \le t$. For every $t > 0$, we find a point $\boldsymbol{x}^{(t)}$ on the boundary of $\mathcal{Q}_t$ such that $\boldsymbol{x}^{(t)}$ satisfies the analogue of condition (3). As $t \to \infty$, the points $\boldsymbol{x}^{(t)}$ tend to the boundary of $\mathcal{D}$. We argue that there always exists a convergent subsequence of the points $\boldsymbol{x}^{(t)}$, and show that the limit point of this subsequence solves our original problem.

The procedure above hinges on our ability to define the appropriate points $\boldsymbol{x}^{(t)}$ that satisfy the required condition. This is based on considering the tangent to the surface of $\mathcal{Q}_t$. Note that the only essential difference between $\mathcal{D}$ and $\mathcal{Q}$ is that the latter is defined by an infinite family of defining linear inequalities, namely, one for each hyperplane H that is tangent to $\mathcal{Q}$. Keeping this perspective in mind, let us apply the original problem definition to a point $\boldsymbol{x} \in \mathcal{Q}$. If $\boldsymbol{x}$ lies in the interior of $\mathcal{Q}$, then none of $\mathcal{Q}$'s defining inequalities holds with equality. Thus, as before, $J(\boldsymbol{x})$ is empty for any $\boldsymbol{x}$ in the interior of the domain $\mathcal{Q}$. We therefore consider $\boldsymbol{x}$ that lies on the boundary of $\mathcal{Q}$. In this case, the set $J(\boldsymbol{x})$ is a singleton, the only member of which is the inequality corresponding to the hyperplane H that is tangent to $\mathcal{Q}$ and touches it at the point $\boldsymbol{x}$. The equation of the tangent hyperplane H can be written as $\sum \nu_i x_i = 1$, where the vector $(\nu_1, \ldots, \nu_n)$ is normal to H. Now condition (3) becomes

$$\forall i \quad \nu_i x_i \ge e_i. \tag{4}$$

When we sum over all i this becomes $\sum \nu_i x_i \ge \sum e_i = 1$. But $\boldsymbol{x}$ lies on H, so that $\sum \nu_i x_i = 1$. It follows that all inequalities in Eq. (4) hold with equality. But we also have, from the definition of the bottlenecks, that $\sum r_{ij} x_i = 1$. Thus, the normal is simply defined by the requirements vectors. Moreover, we can use this as a condition on the gradients of the surfaces of $\mathcal{Q}_t$ for successive t's, and follow a trajectory that leads to a solution on the boundary of $\mathcal{D}$. This is then the desired constructive proof: it both shows that a solution exists, and provides a mechanism for finding it. In the next subsection we formalize this argument.

5.3 Proof of Theorem 1

Construction 1. *To every allocation $\boldsymbol{x}$ in the interior of the domain $\mathcal{D}$, we assign a value*

$$f(\boldsymbol{x}) = -\sum_{j=1}^{m} \log\left(1 - \sum_{k=1}^{N} x_k r_{kj}\right). \tag{5}$$

Remark 1. *The function f is positive in the interior of $\mathcal{D}$, diverging to infinity as $\boldsymbol{x}$ tends to the boundary of $\mathcal{D}$.*

Remark 2. *Clearly, there are other choices of f that satisfy these desired properties. This choice seems like the simplest one for our purposes.*

Definition 1. *To every number $t > 0$, there corresponds a level set of f, namely,*

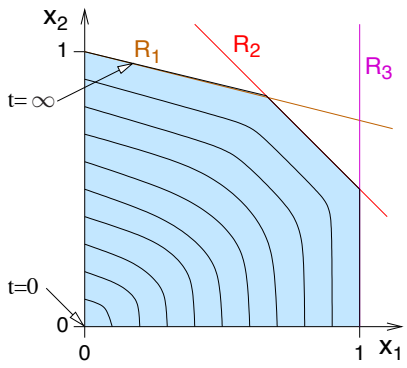

Figure 2: *Illustration of level-sets of f from $t = 0$ to $t = \infty$.*

$$\Gamma_t = \{\boldsymbol{x} \in \mathcal{D} : \ f(\boldsymbol{x}) = t\},$$

Remark 3. *This is an $(N-1)$-dimensional hypersurface. (Fig. 2 illustrates this for $N = 2$.)*

Definition 2. *To every point $\boldsymbol{x} \in \mathcal{D}$, there corresponds a unique unit vector $\boldsymbol{\nu}(\boldsymbol{x}) = (\nu_1(\boldsymbol{x}), \ldots, \nu_N(\boldsymbol{x}))$, normal to the level set of f at $\boldsymbol{x}$.*

The unit normal $\boldsymbol{\nu}(\boldsymbol{x})$ is proportional to the gradient of f at $\boldsymbol{x}$, implying that

$$\nu_i(\boldsymbol{x}) = \tilde{c}\,\frac{\partial f}{\partial x_i}(\boldsymbol{x}) = \tilde{c}\sum_{j=1}^{m} \frac{r_{ij}}{1 - \sum_{k=1}^{N} x_k r_{kj}}, \quad \forall i = 1, \ldots, N, \tag{6}$$

where the normalization constant $\tilde{c}$ is chosen so as to guarantee that $\boldsymbol{\nu}$ is a unit vector, that is, $\nu_1^2 + \cdots + \nu_N^2 = 1$.

Construction 2. *We now construct a vector-valued function*

$$\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t)), \qquad t \ge 0,$$

satisfying the following properties:

1. *$\boldsymbol{x}(t)$ lies on the level set Γ_t for all $t \ge 0$ (and, in particular, remains in $\mathcal{D}$).*
2. *For all $t > 0$, there exists a t-dependent normalization factor $c(t)$, such that for every $i = 1, \ldots, N$,*

$$x_i(t)\,\nu_i(\boldsymbol{x}(t)) = \tilde{c}\,c(t) e_i. \tag{7}$$

Remark 4. *Note that since $f(\boldsymbol{x}(0)) = 0$ it follows that $\boldsymbol{x}(0) = 0$, that is, the vector-valued function $\boldsymbol{x}(t)$ "starts" at the origin.*

Remark 5. *Substituting* (6) *into* (7) *and summing over the index i determines $\tilde{c}\,c(t)$. After simple algebraic manipulations, we get*

$$\sum_{j=1}^{m} \frac{x_i(t) r_{ij} - (\sum_{k=1}^{N} x_k(t) r_{kj}) e_i}{1 - \sum_{k=1}^{N} x_k(t) r_{kj}} = 0, \ \forall i = 1, \ldots, N, \ \forall t > 0. \tag{8}$$

Intuitively, $\boldsymbol{x}(t)$ is a "trajectory" that takes us from the origin $\boldsymbol{x} = 0$ to a point on the boundary of $\mathcal{D}$ as t grows from 0 to ∞[1].

[1] In networking, allocations to flows traversing multiple links are also viewed as using multiple resources, where again the constraints stem from links that become bottlenecks. In this context max-min fairness can be characterized based on a geometrical representation that is very similar to ours [15]. However, the requirements from all the resources (links) are equal, making the search for a solution easier. Specifically, it is often possible to move in a straight line from the origin to the boundary, in a direction based on the desired relative allocations, rather than using a more complicated trajectory as we do.

The formal proof now follows from the following sequence of three lemmas, proved below. First, we show that a trajectory with the required properties exists (Lemma 4). Given such a trajectory, we show that a subsequence of this trajectory converges to a point on the boundary of $\mathcal{D}$ (Lemma 2). Finally, this accumulation point is shown to be a solution to our allocation problem (Lemma 3).

It is convenient to postpone the discussion of whether there indeed exists a trajectory $\boldsymbol{x}(t)$ satisfying the required properties, and consider convergence first.

Lemma 2. *Let $0 < t_1 < t_2 < \cdots$ be a sequence tending to infinity. Let $x(t)$ be a vector-valued function as defined in Construction 2. Then, the sequence $x(t_i)$ has a subsequence that converges to an allocation $\boldsymbol{x}^*$ on the boundary of $\mathcal{D}$.*

PROOF. Consider what happens as $t \to \infty$. Since $\boldsymbol{x}(t) \in \Gamma_t$, it follows that $\boldsymbol{x}(t)$ approaches the boundary of $\mathcal{D}$. However, the function $\boldsymbol{x}(t)$ may not tend to a limit as $t \to \infty$. Nevertheless, since $\mathcal{D}$ is a compact domain, $\boldsymbol{x}(t)$ has a convergent subsequence. That is, there exists an allocation $\boldsymbol{x}^* = (x_1^*, \ldots, x_N^*)$ on the boundary of $\mathcal{D}$ and a subsequence $t_{n_1} < t_{n_2} < \ldots$ such that

$$\lim_{k\to\infty} \boldsymbol{x}(t_{n_k}) = \boldsymbol{x}^*.$$

□

The next lemma shows that this accumulation point is a solution to the fair allocation problem.

Lemma 3. *An allocation $\boldsymbol{x}^*$ as resulting from Lemma 2 is a fair allocation according to our definition.*

PROOF. Since $\boldsymbol{x}^*$ is on the boundary of $\mathcal{D}$, it has a non-empty set $J(\boldsymbol{x}^*)$ of bottleneck resources such that

$$x_1^* r_{1j} + \cdots + x_N^* r_{Nj} = 1 \qquad \forall j \in J(\boldsymbol{x}^*) \neq \emptyset.$$

We then rewrite (8) by splitting the resources j into bottleneck resources and non-bottleneck resources, and setting $t = t_n$:

$$\begin{aligned} &\sum_{j\notin J(\boldsymbol{x}^*)} \frac{x_i(t_n) r_{ij} - (\sum_{k=1}^N x_k(t_n) r_{kj}) e_i}{1 - \sum_{k=1}^N x_k(t_n) r_{kj}} + \\ &\sum_{j\in J(\boldsymbol{x}^*)} \frac{x_i(t_n) r_{ij} - (\sum_{k=1}^N x_k(t_n) r_{kj}) e_i}{1 - \sum_{k=1}^N x_k(t_n) r_{kj}} = 0. \end{aligned} \tag{9}$$

The two summations behave very differently as $n \to \infty$. For a non-bottleneck resource j, $\sum_{k=1}^N x_k^* r_{kj} < 1$, so the summation over the non-bottleneck resources tends to a limit obtained by letting $\boldsymbol{x}(t_n) \to \boldsymbol{x}^*$ term-by-term:

$$\begin{aligned} &\lim_{n\to\infty} \sum_{j\notin J(\boldsymbol{x}^*)} \frac{x_i(t_n) r_{ij} - (\sum_{k=1}^N x_k(t_n) r_{kj}) e_i}{1 - \sum_{k=1}^N x_k(t_n) r_{kj}} \\ &= \sum_{j\notin J(\boldsymbol{x}^*)} \frac{x_i^* r_{ij} - (\sum_{k=1}^N x_k^* r_{kj}) e_i}{1 - \sum_{k=1}^N x_k^* r_{kj}}. \end{aligned} \tag{10}$$

For a bottleneck resource j, the denominator $1 - \sum_{k=1}^N x_k r_{kj}$ tends to zero as $x \to x^*$, so the limit exists only if the numerator vanishes as well. But if it were the case that, for a given user i,

$$x_i^* r_{ij} < e_i \qquad \text{for all } j \in J(\boldsymbol{x}^*),$$

then

$$\lim_{n\to\infty} \sum_{j\in J(\boldsymbol{x}^*)} \frac{x_i(t_n) r_{ij} - (\sum_{k=1}^N x_k(t_n) r_{kj}) e_i}{1 - \sum_{k=1}^N x_k(t_n) r_{kj}} = -\infty.$$

This is a contradiction to the fact that, by (9), the limit should be the negative of the right-hand side of (10). Hence we conclude that $\boldsymbol{x}^*$ has the property that for all users i, there exists a bottleneck resource j such that $x_i^* r_{ij} \geq e_i$. Thus, $\boldsymbol{x}^*$ is a fair allocation. □

It remains to show that the trajectory $\boldsymbol{x}(t)$ is indeed well-defined for all system parameters e_i and r_{ij}. This is handled by the following lemma.

Lemma 4. *There exists a function $\boldsymbol{x}(t)$ with the properties specified in Construction 2.*

PROOF. To prove this we show that we can find points satisfying property 1 that also satisfy property 2. Since $\boldsymbol{x}(t) \in \Gamma_t$, we have $f(x(t)) = t$, that is,

$$-\sum_{j=1}^m \log\left(1 - \sum_{k=1}^N x_k(t) r_{kj}\right) = t. \tag{11}$$

By (7),

$$\sum_{j=1}^m \frac{x_i(t) r_{ij}}{1 - \sum_{k=1}^N x_k(t) r_{kj}} = c(t) e_i, \qquad \forall i = 1, \ldots, N. \tag{12}$$

Differentiating both equations with respect to t, we obtain a linear system of equations for the derivative $d\boldsymbol{x}/dt$. Differentiating (11), we get

$$\frac{\sum_{k=1}^N \frac{dx_k}{dt} r_{kj}}{1 - \sum_{k=1}^N x_k(t) r_{kj}} = 1.$$

Differentiating (12), we get

$$\sum_{j=1}^m \frac{\frac{dx_i}{dt} r_{ij}}{1 - \sum_{k=1}^N x_k r_{kj}} + \sum_{j=1}^m \frac{x_i r_{ij} \sum_{k=1}^N \frac{dx_k}{dt} r_{kj}}{(1 - \sum_{k=1}^N x_k r_{kj})^2} = \frac{dc}{dt} e_i. \tag{13}$$

Observe that, without loss of generality, we can set $dc/dt = 1$, compute the resulting vector of derivatives $d\boldsymbol{x}/dt$, and then multiply it by a constant for the normalization condition to hold. Thus, it remains only to show that (13) has a unique solution when $dc/dt = 1$. To do so, we define an $\boldsymbol{x}$-dependent matrix with entries

$$b_{ij} = \frac{r_{ij}}{1 - \sum_{k=1}^N x_k r_{kj}}, \qquad i = 1, \ldots, N, \qquad j = 1, \ldots, m\ .$$

These entries are non-negative for $\boldsymbol{x} \in \mathcal{D}$. We now rewrite (13) in a more compact form,

$$\sum_{k=1}^m \frac{dx_k}{dt} \left(\sum_{j=1}^m b_{ij}\delta_{ik} + \sum_{j=1}^N x_i b_{ij} b_{kj}\right) = e_i.$$

The term inside the brackets is the (k, i) entry of a symmetric positive-definite $N \times N$ matrix, which immediately implies that there exists a unique solution $d\boldsymbol{x}/dt$. Since the dependence of $d\boldsymbol{x}/dt$ on $\boldsymbol{x}$ is continuous, the existence and uniqueness of $\boldsymbol{x}(t)$ follows from the Fundamental Theorem of Ordinary Differential Equations [5]. (More precisely, the fundamental theorem of ODEs guarantees only the existence and uniqueness of a solution for some small t; global existence follows from the boundedness of the domain $\mathcal{D}$.) □

This completes the proof of Theorem 1.

We note that our proof that a fair allocation exists is almost constructive. The trajectories $\boldsymbol{x}(t)$ can easily be computed numerically using standard ODE integrators (for example, Matlab's `ode45` function). If $\boldsymbol{x}(t)$ is found to tend to a limit for large t, then this limit is a fair allocation. The only reservation is that numerical integration only provides approximate solutions (however,

with a controllable error), and can only be carried out over a finite t interval.

Following our work, Gutman and Nisan [12] have provided a polynomial-time solution to the problem of finding a BBF allocation. They did so by first providing a characterization of BBF. To explain their results, we require some definitions, which use the notation of Section 3.[2] Define an *allocation for player i* to be a tuple $\mathbf{y}_i = (y_{i1}, \ldots, y_{im})$ such that $y_{ij} \geq 0$; intuitively, y_{ij} is the amount of resource j that player i gets. We can associate with an allocation $\mathbf{y}_i$ its utility. To capture BBF, we take $u_i(\mathbf{y}_i) = \min_j(y_{ij}/r_{ij})$. An allocation $\mathbf{y}_i$ for player i is *parsimonious* if there is no allocation $\mathbf{y}'_i < \mathbf{y}_i$ such that $u_i(\mathbf{y}'_i) = u_i(\mathbf{y}_i)$, where $\mathbf{y}'_i < \mathbf{y}_i$ if $y'_{ij} \leq y_{ij}$ for all j and $y'_{ij} < y_{ij}$ for some j. It is easy to see that $\mathbf{y}_i$ is parsimonious iff there exists x_i such that $y_{ij} = x_i r_{ij}$ for all j. An *allocation* $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ is *feasible* if $\sum_i y_{ij} \leq 1$ for all j. In our setting, a *Fisher market equilibrium* consists of an allocation $\mathbf{Y}$ and a vector $\pi = (\pi_1, \ldots, \pi_m)$ with $\pi_j \geq 0$ for $j = 1, \ldots, m$ (which can be thought of as a price vector, where π_j is the "price" of resource j) such that

1. for $i = 1, \ldots, n$, the vector $\mathbf{y}_i$ maximizes $u_i(\mathbf{y}_i)$, under the constraint $\sum_j \pi_j y_{ij} \leq e_i$;
2. $\sum_i y_{ij} = 1$.

Note that although we took e_i to be user i's allocation, in the context of Fisher market equilibrium it can be thought of as user i's budget. The constraint $\sum_i y_{ij} = 1$ says that each resource is "used up".

Gutman and Nisan show that, given an instance of our problem (defined by the allocations $e_1, \ldots, e_n$ and the requests r_{ij}), by adding a new resource for each player, we can easily convert this instance to a new instance of the problem such that (1) if $(\mathbf{Y}, \pi)$ is a Fisher market equilibrium corresponding to the new instance, and $\mathbf{X}$ is a parsimonious allocation such that, for all users i, (a) $\mathbf{X}_i \leq \mathbf{Y}_i$, and (b) $u_i(\mathbf{X}_i) = u_i(\mathbf{Y}_i)$, then $\mathbf{X}$ is a BBF allocation; (2) the allocation to the original resources in $\mathbf{X}$ gives a BBF solution for the original instance of the problem. By the results of [6], a Fisher market equilibrium can be computed in polynomial time. It easily follows that a BBF allocation can be computed in polynomial time. Moreover, it follows from the results of [6] that the feasible solution to the constraints (1) that maximizes $\prod_{i=1}^n x_i$ is a BBF allocation. We note that this statement can be verified directly, using convex programming duality.

6. UNIQUENESS AND OPEN QUESTIONS

Generally speaking, the BBF allocation problem does not have a unique solution; moreover, different solutions may depend on different sets of bottlenecks. Consider the following example, with four users and four resources ($N = m = 4$). Assume all users have the same entitlements, that is $e_i = 0.25$ for $i = 1, \ldots, 4$. Arrange the users and resources in a circle, and make each user request the full capacity of its resource and those of its neighbors. Thus the requirements matrix becomes

$$r = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

This instance is completely symmetric, and the obvious solution is a symmetric allocation where $x_i = \frac{1}{3}$ for $i = 1, \ldots, 4$. In this solution, all 4 resources are bottlenecks, and all users get more than their entitlements on all the resources they use. But there are 6 additional solutions. Pick any two users i and j, and set $x_i = x_j = 0.25$. Let k and l be the other two users, and set $x_k = x_l = 0.375$. Now two resources are bottlenecks ($0.25 + 0.375 + 0.375 = 1$) but the other two are not ($0.25 + 0.25 + 0.375 = 0.875$). Which resources become bottlenecks depends on the choice of k and l. This example can easily be generalized as follows. Consider an example with $N = m > 2$. The requirements are $r_{ii} = 0$ for all i and $r_{ij} = 1$ when $i \neq j$. All $e_i = 1/N$. For every $N - 1 \geq t \geq 1$ we select an arbitrary set of t users and let the corresponding x_i be $1/n$. For all other $N - t$ users we let $x_i = \frac{N-t+1}{N(N-t)}$. It is easily verified that in this case there are exponentially many solutions.

[2]We remark that we simplify the definitions of [12].

m	N	Solutions
4	20	5.4±1.9
5	25	14.6±3.8
6	30	34.8±8.4
7	35	77±20
8	40	186±31
9	45	401±63

Table 1: *Average number of solutions ± standard deviation for randomized resource request matrices of different sizes.*

To get some idea of more general results, we provide a brief report about some preliminary numerical experiments that we have carried out. We start by observing that for a given $N \times m$ matrix R, it is possible to discover all the BBF solutions of the corresponding problem by solving $2^m - 1$ linear programs. This is done as follows: Fix a nonempty set of resources J. We can test, using an LP solver, whether it is possible to satisfy condition (1) with equality iff $j \in J$ along with condition (2) (with respect to the same set of bottlenecks J).

Our experiment runs as follows: We repeatedly (100 times) sample a random $N \times m$ matrix R whose entries r_{ij} are drawn independently from the uniform distribution on $[0, 1]$. For the resulting matrix we solve the $2^m - 1$ linear programs described above and list all the resulting BBF solutions. Needless to say, we had to limit our search to relatively small values of m. Our preliminary results suggest that, for a fixed m, the number of solutions grows very substantially when the number of players N increases. In fact, the number of solutions seems to be growing exponentially with m (for N large) (Table 1).

These preliminary observations are very intriguing and these issues call for a more thorough investigation. Several questions suggest themselves: Of the many available BBF solutions, which should be preferred? Can the "better" solutions be found efficiently? We note in particular that the Gutman-Nisan algorithm yields only one solution, and tells us nothing about the rest of them.

The solution concept we develop here has many properties that are desirable as well in an environment where different users compete for resources. However, we still do not know much about possible manipulations in this context, and how they affect the outcome.

The suggested approach is off-line, and requires full data about requirements to be available in order to compute a solution. Another interesting question is how to formulate an on-line algorithm that schedules tasks in a way that will lead to the desired allocations.

7. CONCLUSIONS

To summarize, our main contribution is the definition of what it means to make a fair allocation of multiple continuously-divisible

resources when users have different requirements for the resources, and a proof that such an allocation is in fact achievable. The definition is based on the identification of bottleneck resources, and the allocation guarantees that each user either receives all he wishes for, or else gets at least his entitlement on some bottleneck resource. The proof is constructive in the sense that it describes a method to find such a solution numerically. The method has in fact been programmed in Matlab, and was used in our exploration of various scenarios.

Note that, in the context of on-line scheduling, we may not need to find an explicit solution in advance. Consider for example the RSVT scheduler described by Ben-Nun et al. [2]. This is a fair share scheduler that bases scheduling decisions on the gap between what each user has consumed and what he was entitled to receive. To do so, the system keeps a global view of resource usage by the different users. If there is only one bottleneck in the system, this would be applied to the bottleneck resource. The question is what to do if there are multiple bottlenecks. Our results indicate that the correct course of action is to prioritize each process based on the *minimal* gap on any of the bottleneck devices, because this is where it is easiest to close the gap and achieve the desired entitlement. Once the user achieves his target allocation on any of the bottleneck devices, he should not be promoted further. This contradicts the intuition that when a user uses multiple resources, his global priority should be determined by the one where he is farthest behind. Note that such an algorithm is similar to the construction used for DRF, and also works if users have different tasks with different resource requirement vectors.

It should also be noted that our proposal pertains to the policy level, and only suggests the considerations that should be applied when fair allocations are desired. It can in principle be used with any available mechanism for actually controlling resource allocation, for example, resource containers [1].

A possible direction for additional work is to extend the model. In particular, an interesting question is what to do when the relative usage of different resources is not linearly related. In such a case, we need to replace the user-based factors x_i by specific factors x_{ij} for each user and resource. This also opens the door for a game where users adjust their usage profile in response to system allocations — for example, substituting computation for bandwidth by using compression — and the use of machine learning to predict performance and make optimizations [3]. Finally, we might consider approaches where users have specific utilities associated with each resource.

Acknowledgments

Danny Dolev is Incumbent of the Berthold Badler Chair in Computer Science, and was supported in part by the Google Inter-university center for Electronic Markets and Auctions. Dror Feitelson was supported by the Israel Science Foundation (grant no. 28/09), and by an IBM faculty award. Joseph Halpern was supported in part by NSF grants ITR-0325453, IIS-0534064, IIS-0812045, and IIS-0911036, by AFOSR grants FA9550-08-1-0438 and FA9550-09-1-0266, ARO grant W911NF-09-1-0281, and a Fulbright Fellowship.

8. REFERENCES

[1] G. Banga, P. Druschel, and J. C. Mogul, "*Resource containers: A new facility for resource management in server systems*". In 3rd *Symp. Operating Systems Design & Implementation*, pp. 45–58, Feb 1999.

[2] T. Ben-Nun, Y. Etsion, and D. G. Feitelson, "*Design and implementation of a generic resource sharing virtual time dispatcher*". In 3rd *Ann. Haifa Experimental Syst. Conf.*, May 2010.

[3] R. Bitirgen, E. İpek, and J. F. Martínez, "*Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach*". In 41st *Intl. Symp. Microarchitecture*, pp. 318–329, Nov 2008.

[4] S. J. Brams and A. D. Taylor, *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambidge University Press, Cambridge, U.K., 1996.

[5] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*. Krieger Pub. Co., 1984.

[6] B. Codenotti and K. Varadarajan, "*Efficient computation of equilibrium prices for markets with Leontief utilities*". In 31st *Intl. Colloq. Automata, Lang., & Prog.*, pp. 257–287, Springer-Verlag, Jul 2004. Lect. Notes Comput. Sci. vol. 3142.

[7] K. J. Duda and D. R. Cheriton, "*Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler*". In 17th *Symp. Operating Systems Principles*, pp. 261–276, Dec 1999.

[8] Y. Etsion, T. Ben-Nun, and D. G. Feitelson, "*A global scheduling framework for virtualization environments*". In 5th *Intl. Workshop System Management Techniques, Processes, and Services*, May 2009.

[9] Y. Etsion, D. Tsafrir, and D. G. Feitelson, "*Process prioritization using output production: scheduling for multimedia*". *ACM Trans. Multimedia Comput., Commun. & App.* **2(4)**, pp. 318–342, Nov 2006.

[10] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "*Dominant resource fairness: Fair allocation of multiple resource types*". In 8th *Networked Systems Design & Implementation*, pp. 323–336, Mar 2011.

[11] A. V. Goldberg and J. Hartline, "*Envy-free auctions for digital goods*". In 4th *ACM Conf. Electronic Commerce*, pp. 29–335, 2003.

[12] A. Gutman and N. Nisan, "*Fair allocation without trade*", Oct 2011. Working paper.

[13] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[14] J. Nieh, C. Vaill, and H. Zhong, "*Virtual-Time Round Robin: An O(1) proportional share scheduler*". In *USENIX Ann. Technical Conf.*, pp. 245–259, Jun 2001.

[15] B. Radunović and J.-Y. Le Boudec, "*A unified framework for max-min and min-max fairness with applications*". *IEEE/ACM Trans. Networking* **15(5)**, pp. 1073–1083, Oct 2007.

[16] I. Stoica, H. Abdel-Wahab, and A. Pothen, "*A microeconomic scheduler for parallel computers*". In *Job Scheduling Strategies for Parallel Processing*, pp. 200–218, Springer-Verlag, 1995. Lect. Notes Comput. Sci. vol. 949.

[17] C. A. Waldspurger and W. E. Weihl, "*Lottery scheduling: Flexible proportional-share resource management*". In 1st *Symp. Operating Systems Design & Implementation*, pp. 1–11, USENIX, Nov 1994.

[18] M. E. Yaari and M. Bar-Hillel, "*On dividing justly*". *Social Choice and Welfare* **1(1)**, pp. 1–24, May 1984.

[19] H. P. Young (ed.), *Fair Allocation*. Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1985.