Controlled Dynamic Fair Division

Eric J. Friedman [*] Christos-Alexandros Psomas [†] Shai Vardi[‡]

July 24, 2017

Abstract

In the single-resource dynamic fair division framework there is a homogeneous resource that is shared between agents dynamically arriving and departing over time. When n agents are present, there is only one truly "fair" allocation: each agent receives $1/n$ of the resource. Implementing this static solution in the dynamic world is notoriously impractical; there are too many disruptions to existing allocations: for a new agent to get her fair share, all other agents must give up a small piece.

A natural remedy is simply to restrict the number of allowed disruptions when a new agent arrives. [16] considered this setting, and introduced a natural benchmark - the *fairness ratio* - the ratio of the minimal share to the ideal share ($1/k$ when there are k agents in the system). They described an algorithm that obtains the optimal fairness ratio when $d \geq 1$ disruptions are allowed per arriving agent. However, in systems with high arrival rates even one disruption per arrival can be too costly. We consider the scenario when fewer than one disruption per arrival is allowed. We show that we can maintain high levels of fairness even with significantly fewer than one disruption per arrival. In particular, we present an instance-optimal algorithm (the input to the algorithm is a vector of allowed disruptions) and show that the fairness ratio of this algorithm decays logarithmically with c, where c is the longest number of consecutive time steps in which we are not allowed any disruptions.

We then consider dynamic fair division with multiple, heterogeneous resources. In this model, agents demand the resources in fixed proportions, known in economics as Leontief preferences. We show that the general problem is NP-hard, even if the resource demands are binary and known in advance. We study the case where the fairness criterion is *Dominant Resource Fairness* (DRF), and the demand vectors are binary. We design a generic algorithm for this setting using a reduction to the single-resource case. To prove an impossibility result, we take an integer program for the problem and analyze an algorithm for constructing dual solutions to a "residual" linear program; this approach may be of independent interest.

1 Introduction

Fair division has been a central topic in economics and mathematics (e.g., [2–4, 6, 13, 35, 42, 43]). More recently, it has received more attention in computer science due to its applications to resource sharing in data centers and the cloud (e.g., [5, 12, 18, 37, 46]). Traditionally, research on fair division has focused on static allocations; contemporary resource allocation protocols, however, need to be dynamic in nature. This has led to more research on fairness in the dynamic setting (e.g., [1, 16,

[*] ICSI and UC Berkeley

[†] UC Berkeley.

[‡] California Institute of Technology. E-mail: svardi@caltech.edu.

29, 45]): There is some finite amount of resource(s), agents arrive and depart, and the goal is to constantly maintain allocations that are "fair". There are several accepted notions of fairness in the literature, for example, *envy-freeness* (no agent would like to exchange shares with any other agent) and *equitability* (every agent has the same utility). Arguably the most widely accepted notion is *proportionality*: if there are n agents in the system, each agent is allocated at least a $1/n$ fraction of what she would receive if she were allocated all of the resource. If there are multiple resources and agents have heterogeneous demands, the notions of fairness become more complex; two of the most common notions in this setting are *Competitive Equilibrium from Equal Incomes* (CEEI) and *Dominant Resource Fairness* (DRF). We elaborate more on these later on.

A major difficulty in maintaining fairness in dynamic settings comes from the price of reallocating resources. Even if there exist good solutions for k agents and for $k+1$ agents, these solutions do not typically include instructions for transitioning from one solution to the other without reclaiming and reallocating all the resources. As an example, consider the case of a single homogeneous resource. If reallocating resources is cheap and efficient, it is trivial to satisfy any and all of the above fairness notions in the dynamic setting: when there are k agents, allocate each one $1/k$ of the resource. When a new agent arrives, reallocate the resource evenly once more; note that this necessitates reducing the allocation of *all* agents, whenever a new agent arrives. If there are many successive arrivals, much of the time could be spent on reallocation, instead of resource consumption. This is an important issue in practice (e.g., [26, 31, 44]).

> "The offline scheduler is not applicable . . . because, if we simply called it each time a resource freed up, we might have to reallocate a large number of machines to obtain the configuration it returns. " [19].

Motivated by these issues, we study dynamic fair division problems where the amount of disruption is a hard constraint.

1.1 The Dynamic Fair Division Model

We first consider the case of a single homogeneous resource.

Dynamic fair division of a single resource. One unit of a homogeneous resource is shared among agents that arrive and depart over time. An allocation for t agents is denoted by a vector $\mathrm{X}^t \in [0,1]^t$. The utility of agent j at step t is proportional to $\mathrm{X}^t(j)$. An allocation is *feasible* if $\mathrm{S}^t = \sum_{j=1}^{t} \mathrm{X}^t(j) \leq 1$. X and S are usually defined with respect to a resource allocation algorithm; this is omitted from the notation when the algorithm is obvious from context. An allocation algorithm is feasible if it always outputs a feasible allocation. An allocation is *Pareto optimal* if $\mathrm{S}^t = 1$. An allocation algorithm is Pareto optimal if it always outputs Pareto optimal allocations. An allocation for t agents is σ_t*-fair* if $\min_{j=1,\ldots,t}\{\mathrm{X}^t(j)\} \geq \frac{\sigma}{t}$. An allocation algorithm A is σ-fair if it outputs a σ_t*-fair* allocation in the presence of t agents, and $\sigma \leq \min_t\{\sigma_t\}$. σ_t is called A's *fairness ratio*.

We restrict our algorithms to disrupt a small number of agents when a new agent arrives: An allocation algorithm is d-disruptive if it is allowed to decrease the allocation of at most d agents at every arrival. Upon departure, the algorithm is not allowed to reduce the allocation of any agent except the departing agent (this is known as *population monotonicity*). Augmenting the resource of any agent is always allowed (for both arrivals and departures).

Example 1.1. *Assume we would like to design a Pareto optimal* 1*-disruptive algorithm for a system with a capacity for* 3 *agents. The naïve algorithm divides the largest available share equally at each arrival: When the first agent arrives, she is allocated the entire resource;* $\sigma_1 = 1$*. When the second agent arrives, she is given half of the resource, and the first agent's resource is halved:* $\sigma_2 = \frac{\min_i\{\mathrm{X}^2(i)\}}{1/2} = 1$*. When the third agent arrives, one agent is allocated half of the resource and the two other agents are allocated one quarter of the resource each:* $\sigma_3 = \frac{1/4}{1/3} = \frac{3}{4}$*. The fairness ratio of this algorithm is* $\sigma = \min_i\{\sigma_i\} = \frac{3}{4}$*. It is easy to verify that there is no algorithm that guarantees perfect fairness (*$\sigma = 1$*), even in this simple scenario.*

The optimal algorithm is the following [16]: when the second agent arrives, she is given $3/7$ *of the resource;* $\sigma_2 = \frac{3/7}{1/2} = \frac{6}{7}$*. When the third agent arrives, the first agent's share is split evenly between the first and third agents, giving the allocation* $(2/7, 3/7, 2/7)$*;* $\sigma_3 = \frac{2/7}{1/3} = \frac{6}{7}$*. This fairness ratio of this algorithm is* $\frac{6}{7}$*.*

Define $\sigma^*(d)$ to be the optimal fairness ratio of any d-disruptive mechanism (with an unbounded number of agents). [16] show that $\sigma^*(d) = \left((d+1)\ln\left(\frac{d+1}{d}\right)\right)^{-1}$, for all (integer) $d \geq 1$, and describe an algorithm that achieves this fairness ratio. They also give a $\frac{3(d+1)^2}{2d^2}$ bound on the *envy ratio* of this algorithm, the ratio of the maximum and minimum allocations. In addition, the algorithm accommodates arbitrary arrivals and departures of agents, and it satisfies population monotonicity as well as Pareto optimality. Their analysis relies on a characterization of the optimal allocation as an LP, reducing its feasible region to a certain class of allocations, and then bounding the fairness ratio for this class.

At first glance, one might think that a single disruption per arrival is the best we can hope for; this is true if, for example, we require Pareto optimality (Lemma 1.2). However, if n agents arrive, this still mandates n disruptions overall! We consider what can be done with even fewer disruptions. In particular, we study the problem of maximizing the fairness ratio, when fewer than one disruption is allowed per arrival.

Fewer than one disruption per arrival. Consider the following scenario: an algorithm designer is given a list of constraints: one constraint per arrival, denoting the number of disruptions allowed. The algorithm designer's goal is to design an algorithm that maximizes the fairness ratio subject to these constraints. For now, assume that there are only arrivals - agents arrive and stay in the system indefinitely; we will remove this restriction later on. Ideally, the algorithm designer should design an *instance-optimal* algorithm: one that takes the list of constraints as an input and outputs a set of allocations that maximizes the fairness ratio for that list. We consider the case that fewer than one disruption is allowed per arrival, and model it as follows: the algorithm takes as input a vector ψ (which we call a *control vector*), and is allowed to use d_i donors when the i^{th} agent arrives if $\psi[i] = d_i$. We call this problem *Controlled Dynamic Fair Division (CDFD)*. For simplicity, we only consider *binary* control vectors (i.e., at each time step, the algorithm is either not allowed to use a donor or allowed to use exactly one); it is straightforward to extend the results to non-binary vectors. If ψ has at most c consecutive zeros, we call it a c-control vector. To align with the notation of [16], we say that a c-control vector implies a $\frac{1}{c+1}$-disruptive algorithm. Notice that the setting of [16] is a special case; for d donors per arrival, ψ has "d" at every coordinate.

Let $\sigma^*(\psi)$ be the optimal fairness ratio for a given control vector ψ. We overload the notation and define $\sigma^*\left((c+1)^{-1}\right)$ to be the optimal fairness ratio for all c-control vectors ψ, i.e.,

$\sigma^*\left((c+1)^{-1}\right) = min_{\psi}\{\sigma^*\left(\psi\right)\}$. Note that, with fewer than one disruption per arrival, Pareto optimality is impossible.

Lemma 1.2. *For any $c > 0$, there is no Pareto optimal algorithm for $CDFD$ that guarantees a fairness ratio of $\sigma((c+1)^{-1}) > 0$.*

Proof. Assume that such an algorithm exists. Let i be the first coordinate for which $\psi[i] = 0$. As the algorithm is Pareto optimal, there is no available resource, and agent i receives nothing. □

One can also consider a more adversarial online setting, in which the control vector is revealed online. If there are no restrictions on the adversary, it is easy to see that no algorithm can guarantee a good fairness ratio: the adversary can simply give the algorithm an arbitrarily long all-zero control vector. We therefore restrict our attention to adversaries limited to control vectors that have bounded length and/or at most c consecutive zeros, for some constant c.

Multi-resource fairness. Consider the case of multiple heterogeneous resources. Let r be the number of resources in the system and, without loss of generality, assume there is one unit of each resource available. As in the single resource case, define $\mathrm{X}^t(j)$ to be the allocation of the j-th agent at step t. Notice that the allocation of agent j this time is a vector, not a number.

Every agent i has a demand vector $\mathcal{D}_j \in [0,1]^r$ over the resources. Let $\mathcal{D}^t$ (or simply $\mathcal{D}$) be the t by r matrix of demands for t agents. For example, if 3 agents with demands $\mathcal{D}_1 = [1, 1/2]$, $\mathcal{D}_2 = [0,1]$, and $\mathcal{D}_3 = [1,1]$ are present $\mathcal{D} = \begin{bmatrix} 1 & 1/2 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$.

Assume that the agents demand the resources in fixed proportions, known in economics as Leontief preferences, and have binary demand vectors. Let $u_j(\mathrm{X}^t(j))$ be the utility of agent j for an allocation $\mathrm{X}^t(j)$. It is convenient to think of $\mathcal{D}_{j,l}$ as the amount of resource l agent j needs to execute a task. For example, if $r = 3$, agent j with demand vector $\mathcal{D}_j = [1,0,1]$, needs one unit of resource 1 and one unit of resource 3 to execute a task. In this example, Leontief preferences imply that given the resource vector $\mathrm{X}^t(j) = (\frac{1}{2}, 0, \frac{1}{2})$, the agent would have utility $u_j(\mathrm{X}^t(j)) = \frac{1}{2}$. The agent would have the same utility for resource vector $(1, 0, \frac{1}{2})$.

In the case of multiple resources, the definition of fairness is not as straightforward. One could define fairness in a very general way: at every step t the minimum utility of an agent should be $L(t)$, for some non increasing function $L(t)$, and the maximum utility should be $U(t)$. $L(t)$ dictates the lower bound on the satisfaction of each agent, maintaining fairness in terms of proportionality. $U(t)$ maintains fairness in the sense of envy-freeness: $\min_t \left\{\frac{U(t)}{L(t)}\right\}$ is an upper bound on the envy of such an allocation. (Intuitively, an allocation of $\left(\frac{1}{3}, \frac{1}{4}, \frac{1}{4}\right)$ is more fair than an allocation of $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$.) Define *General Dynamic Fair Division* ($GDFD$) to be this problem; the inputs are agents' demands $\mathcal{D}$, the number of resources r, and vectors L and U, and the algorithm is asked to decide whether there exist feasible allocations (i.e., ones that satisfy all the constraints).

Dominant Resource Fairness. We show in Section E that $GDFD$ is computationally intractable (assuming $\mathsf{P} \neq \mathsf{NP}$), even in simple settings and clairvoyant allocation algorithms (ones that know in advance the number of agents that will arrive and their demands). We focus on a less general, well-studied notion of fairness: *Dominant Resource Fairness* (DRF), introduced by [18].

The *dominant resource* of an agent is the resource for which the agent's task requires the largest fraction of total availability. The *dominant share* of an agent is the fraction of her dominant resource that she receives. The DRF algorithm seeks to maximize the minimum dominant share. DRF can also be interpreted as the leximin mechanism, i.e., it maximizes the minimum utility, and subject to that, maximizes the second minimum utility, and so on, when applied to Leontief utilities [30]. DRF has multiple advantages: it is Pareto optimal, strategy-proof, envy-free, and proportional: it guarantees to each agent a utility of

$$DRF_{\mathcal{D}^t} = \left(\max_{l=1,\ldots,r} \sum_{j=1}^{t} \mathcal{D}^t_{j,l}\right)^{-1}.$$

The notion of DRF is naturally extendable to the dynamic case:

Dynamic fairness for multiple resources. An allocation is σ_t-DRF fair at step t, if

$$\min_{j=1,\ldots,t}\{u_j(\mathrm{X}^t(j))\} \geq \sigma_t DRF_{\mathcal{D}^t}.$$

An allocation algorithm is σ-DRF fair if outputs a σ_t-DRF fair allocation when t agents are present, and $\sigma \geq \min_t\{\sigma_t\}$. Let $\sigma^*(d, r)$ be the optimal fairness ratio in the case of $r > 1$ resources, when d donors are allowed at every arrival, over all possible agent demands $\mathcal{D}$. We allow d to be less than 1; we discuss this in more detail later on. Define *Dynamic Dominant Resource Fairness* (DDRF) to be the problem of maximizing $\sigma^*(d, r)$.

1.2 Main Results and Techniques

Controlled dynamic fair division. Given a binary control vector ψ as an input, the goal is to output a set of allocations that gives the best possible fairness ratio. We describe an instance-optimal algorithm for this case, which we call SKIP. SKIP has two stages; in the first (which we sometimes refer to as the *preprocessing* stage), it computes the optimal fairness ratio $\sigma^*(\psi)$; in the second, it allocates the resource frugally, giving each agent the least amount of resource possible in order to maintain $\sigma^*(\psi)$.

Theorem 1.3. SKIP *is an instance-optimal algorithm for controlled dynamic fair division.*

To prove Theorem 1.3, we show that for any allocation algorithm $\mathcal{A}$ and any control vector ψ, if the allocations produced by $\mathcal{A}$ for ψ give a fairness ratio of σ, then the allocations produced by SKIP do too. Although, by Lemma 1.2, SKIP cannot be Pareto optimal, it can accommodate departures, and is population-monotone - it does not disrupt any agent when there is a departure (except the departing agent); it is only allowed to re-allocate the departing agent's share.

We would like to provide a lower bound[1] σ on the fairness ratio of SKIP. This is particularly important for unbounded systems; while SKIP is optimal for any finite control vector ψ,[2] it is not defined on infinite vectors, which are necessary for systems that can accommodate an arbitrary number of agents. In such a case, we skip the preprocessing stage of SKIP, and simply give SKIP σ as an auxiliary input. As long as σ is upper bounded by the worst fairness ratio possible for any number of agents, the allocations produced by SKIP will be feasible. We bound $\sigma^*\left((c+1)^{-1}\right)$, the optimal

[1] Note that upper bounds on the fairness ratio are negative results, while lower bounds are positive results.

[2] We do not explicitly compute the running times of the algorithms in this paper, but all are easily implemented in time linear in the size of the input.

fairness ratio possible for all control vectors ψ with at most c consecutive zeros (in particular, including infinite vectors such as $(1,0,0,0,1,0,0,0,\ldots)$). This also gives us a bound on the performance of SKIP in the adversarial online setting, when the adversary is limited to selecting vectors with at most c consecutive zeros. (In the adversarial online an adversary chooses the control vector and it is revealed online - when the i^{th} agent arrives, the i^{th} coordinate of the control vector is revealed.)

Showing bounds for $\sigma^*\left((c+1)^{-1}\right)$ is much more complicated than for $\sigma^*(d)$, $d \geq 1$: there are infinitely many possible control vectors (as opposed to exactly one for constant $d \geq 1$), and the allocations SKIP produced for each of these are different; furthermore, they are not as "well behaved" as the allocations for $d \geq 1$. Compare Figures 1 and 2.[3]. Figure 1 shows the allocations created by the optimal algorithm when 1 donor is allowed at every step (for an unbounded number of agents, truncated at 100). Figure 2 shows the allocations created for 4 different infinite control vectors (truncated at 30 agents). The difficulty in the second setting comes from several facts: the allocations are not monotone, they are not pointwise comparable, and they are not simple transformations one of the other. Furthermore, the allocations do not necessarily take their maxima at the limit. Still, in both cases, the total allocation converges to a limit as the number of agents grows (in the second case, *all* infinite allocations that obey certain natural requirements converge to the *same* limit (Theorem 1.6)). The horizontal line in both figures denotes this limit. It is easy to see (this is formally proved for the two cases in [16] and Section 2 respectively), that the first set of allocations takes its maximum at the limit, while the second case does not.[4]

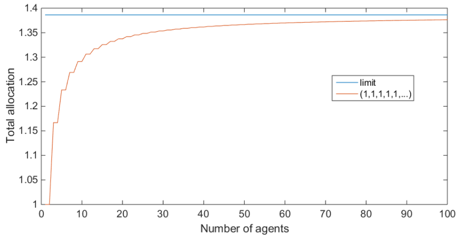

Figure 1: Optimal values of S for $d = 1$.

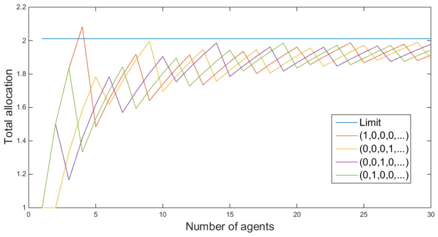

Figure 2: Optimal values of S for some 2-control vectors.

We show almost matching upper and lower bounds for the optimal fairness ratio attainable for any (possibly infinite) c-control vector. Let $H_n = \sum_{i=1}^{n} \frac{1}{i}$ be the n-th harmonic number.

Theorem 1.4. *The optimal fairness ratio of $CDFD$ for any c-control vector is bounded by*

$$(H_{c+1})^{-1} \geq \sigma^*\left((c+1)^{-1}\right) \geq \left(H_{2c+3} - \tfrac{1}{2}\right)^{-1}.$$

In order to prove Theorem 1.4, we need to be able to argue about allocations of optimal algorithms for *any* ψ. For this, we define *basic* control vectors - vectors that always have exactly 1 donor every

[3] The graphs are normalized by artificially setting $\sigma = 1$, and then running the respective optimal algorithms without any constraints on the amount of available resource. While this obviously leads to the total allocation being greater than 1, it is still instructive, as the optimal fairness ratio of these algorithms can be easily be deduced from these plots - it is simply the inverse of the maximal total allocation created

[4] It might be tempting to think that, as in Figure 2, there always exists *some* basic c-control vector (see Section 2 for a definition of basic c-control vectors) that takes its maximum at the limit, and this is indeed the case for $c < 8$ (we do not prove this, but it is easy to verify). However, for $c \geq 8$, *every* basic c-control vector has its maximum at a finite number of agents (see Figure 3 in Appendix I for a pictoral example.)

$c+1$ agents, and show that for any c-control vector, there exists some basic c-control vector whose allocations are pointwise greater. For any c, there are exactly $c+1$ basic control vectors, therefore it remains to reason about them. For the cases $c=1$ and $c=2$, it turns out that the worst allocations are always at the limit. In order to show this, we consider allocations (of the basic control vectors) that are a certain distance apart (specifically $(c+1)(c+2)$), and show that these special allocations are monotone increasing, and each is greater than the $(c+1)(c+2)$ allocations that came before it. This gives exact bounds:

Theorem 1.5. *(Appendix C) The optimal fairness ratio of $CDFD$ for any c control vector for $c=1$ and $c=2$ are*

$$\sigma^*\left((2)^{-1}\right) = 2(3\ln 3)^{-1} \quad \textit{and} \quad \sigma^*\left((3)^{-1}\right) = 3(4\ln 4)^{-1}.$$

For $c>2$, we consider a set of allocations that are pointwise greater than the allocations for all basic vectors simultaneously, and bound them, to obtain Theorem 1.4. In addition, we show that as the number of agents grows, the optimal allocations for all basic c-control vectors converge to the same value:

Theorem 1.6. *There exists a number n_0 such that, for all $c>2$, all c-control vectors, and present agents $t>n_0$, the allocation of* Skip *is σ_t-fair, where $\sigma_t = (c+1)\left((c+2)\ln(c+2)\right)^{-1}$.*

We show (Appendix D) that Skip can accommodate departures, and therefore all the bounds on $\sigma^*\left((c+1)^{-1}\right)$ hold even when we allow agents to depart. We summarize our main results for $\sigma^*\left((c+1)^{-1}\right)$ in Table 1.

Disruptions	**Bound on the fairness ratio**
$d \geq 1$	$\left((d+1)\ln\left(\frac{d+1}{d}\right)\right)^{-1}$ (tight) [16]
2^{-1}	$2\left(3\ln 3\right)^{-1}$ (tight)
3^{-1}	$3\left(4\ln 4\right)^{-1}$ (tight)
$(c+1)^{-1}, c>2$	$(H_{c+1})^{-1} \geq \sigma^*\left((c+1)^{-1}\right) \geq \left(H_{2c+3} - \frac{1}{2}\right)^{-1}$
	$(c+1)\left((c+2)\ln(c+2)\right)^{-1}$ (asymptotic bound, tight)

Table 1: Results for a single resource

The adversarial online case. In the adversarial online case, if there are no restrictions on the adversary, there is no lower bound on the fairness ratio, as the adversary can give an infinite vector of zeros. If, however, there is an a-priori upper bound n on the number of agents that can arrive (with no restriction on the number of consecutive zeros), Skip obtains a fairness ratio of $\sigma = \frac{1}{H_n}$.

Multiple resource dynamic fair division. In the multiple resource case, we consider the *uncontrolled* setting. The input to the algorithm is a demand matrix $\mathcal{D}$ and the number of disruptions allowed per arrival, d. We consider two scenarios: in the first, the *clairvoyant* setting, $\mathcal{D}$ is known in advance. In the *non-clairvoyant* (or online) setting, row i of $\mathcal{D}$ is revealed upon arrival of agent i. We focus on two notions of fairness, $GDFD$ and $DDRF$.

Our first result is that $GDFD$ is NP-hard (Appendix E):

Theorem 1.7. *$GDFD$ is NP-hard, even for 2 resources, 1 donor, binary demand vectors, and clairvoyant algorithms.*

We do not know if the same is true for less general notions of fairness.

Open Problem 1. *Is there a polynomial time algorithm for computing $\sigma^*(d,r)$ for $DDRF$?*

Open Problem 2. *Is there a polynomial time algorithm for computing $\sigma^*(d,r)$ when the notion of fairness is $CEEI$?*

In Section 3 we consider $DDRF$, for binary demand vectors. To see why the techniques of the single resource case do not extend to the multi-resource case, consider the following simple example for $d=1$: Three agents arrive; the first has a demand vector $[1,0]$, the second $[0,1]$, and the third $[1,1]$. The following algorithm gives the optimal allocation of $2/3$ (see Section 4 for the upper bound): the first two agents receive $2/3$ of their respective resource (we do not use a donor at the second arrival), and at the third arrival, one of the two agents has her allocation reduced by half, and the arriving agent is given $1/3$ of that resource. $1/3$ of the other resource is taken from the ”bank” - the resource that was left unallocated. It is easy to see that a non-clairvoyant algorithm can never do as well as a clairvoyant one (in contrast to the single resource case): in the non-clairvoyant case, if an agent with demand $[0,1]$ arrives, how much do we allocate her? Even if we know three agents will arrive in total, and the second one has demand $[1,0]$, if the third agent has demand vector $[1,0]$, we “should have” given the first agent all of the first resource, to get a fairness ratio of 1, but we have to allow for the possibility of the third agent's demand being $[1,1]$. To complicate matters further, it is possible to find examples where solutions that seem intuitively correct are wrong; consider the following set of demands:

$$[0,1],[1,0],[0,1],[1,0],\mathbf{[1,0]},[1,1],[1,1].$$

Surprisingly, the only optimal algorithm uses an agent with demand $[0,1]$ as a donor when the fifth agent (who has demand $[1,0]$) arrives![5]

We prove upper and lower bounds on $\sigma^*(d,r)$ for $DDRF$ for the non-clairvoyant setting for three cases: (1) $d=kr$ for integer k, (2) $d=1$ and (3) $d=(c+1)^{-1}$ for $c>0$. For the lower bounds (positive results), we use the single resource algorithms as subroutines. The challenge is choosing which donor to use when only one donor is allowed. Our negative results use reductions to single resource upper bounds.

Theorem 1.8. *The optimal fairness ratio of $DDRF$ for $d=k\cdot r$ donors and r resources is bounded by*

$$\left((kr+1)\ln\left(\frac{kr+1}{kr}\right)\right)^{-1}\geq\sigma^*(kr,r)\geq\left((k+1)\ln\left(\frac{k+1}{k}\right)\right)^{-1}.$$

Theorem 1.9. *The optimal fairness ratio of $DDRF$ for $d=1$ donors and $r\geq 2$ resources is bounded by*

- $\sigma^*(1,2)\geq 2(3\ln 3)^{-1}\approx 0.6068$
- $\sigma^*(1,3)\geq 3(4\ln 4)^{-1}$

[5] This can be verified by optimizing the fairness for each choice of donors - when the donors are fixed, finding the optimal fairness can be easily computed by a linear program, as we show in Section 4.

- $\sigma^* (1, r) \geq \left(H_{2r+1} - \frac{1}{2} \right)^{-1}, r > 3$.

Theorem 1.10. *The optimal fairness ratio of $DDRF$ for r resources and $(c+1)^{-1}$ donors (1 donor allowed per $c+1$ arrivals) is bounded by*

$$(H_{c+1})^{-1} \geq \sigma^* \left((c+1)^{-1}, r \right) \geq \left(H_{2r(c+1)+1} - \frac{1}{2} \right)^{-1}.$$

The gap between the upper and lower bounds for $DDRF$ is far from tight. Even for $d = 1$, $r = 2$, a seemingly simple case, the best lower bound we can show is is $2(3 \ln 3)^{-1} \approx 0.6068$, by using the single resource algorithm with 1-control vectors ($c = 1$). An immediate upper bound is $(\ln 4)^{-1} \approx 0.7213$: the fairness ratio $\sigma^* (d, r)$ is non-increasing in the number of resources r, and we have tight bounds for $\sigma^* (d, 1)$ from [16] (a subtle but necessary condition for this argument to go through is that DRF reduces to proportionality when there is only one resource). In Section 4 we improve this upper bound to the following.

Theorem 1.11. *The optimal fairness ratio of $DDRF$ for $d = 1$ and $r \geq 2$ resources is bounded by*

$$\sigma^* (1, r) \leq 0.6318 \ldots$$

Our proof technique for Theorem 1.11 may be of independent interest. We first identify a set of bad inputs (demand matrices $\mathcal{D}$): $n\,[0, 1]$'s followed by $n\,[1, 0]$'s, followed by a series of $2n\,[1, 1]$'s. We describe the optimal mixed integer program for maximizing the fairness ratio. Let Z be the set of all integer variables. Given an algorithm for $DDRF$ we can simulate its execution on the bad input and fix these integer variables. Fixing Z allows us to use LP duality! We take the dual with respect to the remaining (non-integer) variables, and give a procedure for constructing feasible dual solutions, for any choice of Z, and therefore any algorithm for $DDRF$.

Notice that the bound is independent of r. For $r > 2$, the same technique does not seem to provide better results, at least not for choices of $\mathcal{D}$ that are the most natural; for example, for $r = 3$, one would expect that the worst inputs are $n\,[0, 0, 1]$'s, followed by $n\,[0, 1, 0]$'s, followed by $n\,[1, 0, 0]$'s, followed by a series of $3n\,[1, 1, 1]$'s. Surprisingly, the fairness ratio for these inputs is an increasing function of n, at least for n small enough for us to verify computationally.

An interesting open problem is finding tight bounds for $\sigma^* (d, r)$. Our algorithms so far rely on a reduction to $CDFD$. We conjecture that this is not necessary. Constructing algorithms that accommodate departures is also an intriguing open direction.

Open Problem 3. *Compute the exact value of $\sigma^* (d, r)$ for DDRF, for all $d > 0$, $r > 1$.*

Open Problem 4. *Compute the exact value of $\sigma^* (d, r)$ for DDRF, for all $d > 0$, $r > 1$, when allowing departures.*

Note that our results for multiple resources only apply to binary demand vectors.

Open Problem 5. *Compute the value of $\sigma^* (d, r)$ for DDRF, for all $d > 0$, $r > 1$ for arbitrary demand vectors.*

Our main results for the optimal fairness ratio of $DDRF$ are summarized in the following table.

Resources	Disruptions	Bound on the fairness ratio
2	1	$0.6318\ldots \geq \sigma^*(1,2) \geq 2(3\ln 3)^{-1} \approx 0.6068$
3	1	$0.6318\ldots \geq \sigma^*(1,3) \geq 3(4\ln 4)^{-1}$
$r > 3$	1	$0.6318\ldots \geq \sigma^*(1,r) \geq \left(H_{2r+1} - \frac{1}{2}\right)^{-1}$
r	kr	$\left((kr+1)\ln\left(\frac{kr+1}{kr}\right)\right)^{-1} \geq \sigma^*(kr,r) \geq \left((k+1)\ln\left(\frac{k+1}{k}\right)\right)^{-1}$
r	$(c+1)^{-1}$	$(H_{c+1})^{-1} \geq \sigma^*\left((c+1)^{-1}, r\right) \geq \left(H_{2r(c+1)+1} - \frac{1}{2}\right)^{-1}$

Table 2: Results for many resources and DRF

1.3 Related Work

Competitive Equilibrium from Equal Incomes (CEEI) was introduced by [32] and has been studied extensively in the economics and theoretical computer science literature (for some recent results see [11], [7], [33], [10], [8]). Dominant Resource Fairness was proposed by [18]. DRF attracted significant attention from the computer systems community [17,25,27,41], as well as the theoretical computer science community [15,24,29,34]. It is interesting to note that DRF can also be interpreted as the Kalai-Smorodinsky bargaining solution [28].

[45] was the first to study the problem of online fair cake cutting when agents arrive, receive a piece and depart. He showed how several well-known fair division solutions (cut-and-choose, Dubins-Spanier, etc) can be adapted to satisfy desirable properties in an online setting with a single (heterogeneous) divisible cake. More recently, and closer to our problem, [29] introduced a model of dynamic allocations. However, their model only considers arrivals and their main algorithm reserves resources for future arrivals; it does not allow the reallocation of resources, or agents to depart. This leads to allocations that satisfy neither fairness nor Pareto efficiency, as resources are left idle.

[21] study the problem of repeatedly allocating a single item between competing agents, and give allocation algorithms that don't allow monetary transfers with good competitive ratios with respect to optimal allocation algorithms with payments. [39] studied the problem of re-dividing a two-dimensional resource, subject to fairness and ”geometric” constraints on the allocations.

The idea of making a small number of alterations to maintain a good solution in online settings had been studied for other problems. In online scheduling, one of the earliest works is by [36], who show that linear preemption in the number of tasks yields a competitive ratio of $O(\log n)$ on makespan, where n is the number of tasks. Preemption in their context means reassigning jobs to a different machine. [38] study online scheduling when the migration is bounded by a constant, and [14] who study a similar problem with bounded migrations, where the capacity of change is based on the size of the new input. [23] consider online matching, online flow and online scheduling, and the number of changes they allow is an amortized constant per step. For scheduling they show a $O(\log\log(nm))$ approximation to the makespan, where n is the number of tasks and m is the number of machines. Other examples include [20] and [22] who show how to maintain an online Steiner tree, vertices arriving online, where the algorithm can change an edge at every arrival; [9] and [40] who consider reoptimization with costs.

2 Single Resource

In this section we study CDFD. The algorithm is given as input a binary c-control vector ψ (Definition 2.1). A solution consists of a fairness ratio σ and a set of allocations $A_1, \dots, A_{|\psi|}$, one for each number of agents present t, such that: (1) for every $1 \leq t \leq |\psi|$, the fairness ratio σ_t is at least σ, and (2) some agent's resource is reduced from step t to step $t+1$ if and only if $\psi[t+1] = 1$. Note that (as we only consider binary vectors) only one agent can have their resource reduced at any time. In some cases (for example, when the control vectors are unbounded), we allow the algorithm to receive σ as an auxiliary input, in which case a possible output is "infeasible", if no such set of allocations exists. For now, we assume that agents only arrive, and do not depart; we can use t both for the time and the number of agents present. In Subsection D we show how to augment SKIP to accommodate departures.

Definition 2.1. *[Control vector] Let ψ^N be a binary vector of length N. $\psi^N[i] = 1$ means that we use a donor when agent i arrives, and a $\psi^N[i] = 0$ means we do not. If the maximal number of consecutive 0s is c, we call this a* c-control vector.

We define a *basic c-control vector* to be one in which there is exactly one donor every $c+1$ arrivals; otherwise the control vector is non-basic. There are $c+1$ possible basic control vectors.

Example 2.2. *For $c = 2$, the three possible basic control vectors are:*

1. *(0,1,0,0,1,0,0,1,0,…), denoted $(0, 1, 0)^\infty$,*
2. *(0,0,1,0,0,1,0,0,1,…), denoted $(0, 0, 1)^\infty$, or $(0^2, 1)^\infty$.*
3. *(1,0,0,1,0,0,1,0,0,…), denoted $(1, 0, 0)^\infty$, or $(1, 0^2)^\infty$.*

We also use the following definition.

Definition 2.3. *Let $A = (a_1, \dots, a_t)$ and be $B = (b_1, \dots, b_t)$ be two sorted vectors. We say that A* dominates *B, denoted $A \succeq B$, if $\forall i \in [1, t]$, $a_i \geq b_i$.*

2.1 SKIP - an Instance-Optimal Algorithm

Definition 2.4 (Algorithm SKIP)**.** *Algorithm* SKIP *receives as an input a control vector ψ. Set $n = |\psi|$.* SKIP *has a preprocessing stage (which we describe after the main algorithm description), in which it computes the optimal σ. When agent $1 \leq i \leq n$ arrives, allocate her $\frac{\sigma}{i}$ of the resource. If $\psi[i] = 1$, take the agent with the most resource to be the donor, and reduce her allocation to $\frac{\sigma}{i}$ as well.*

The preprocessing stage is the following: simulate the arrivals of the agents $1, \dots, n$, with $\sigma = 1$, and compute the sum of allocations at each step, S_i. Set the optimal fairness ratio to be $\sigma = (\max_{1 \leq i \leq n}\{\mathrm{S}_i\})^{-1}$.

First, note that the allocations created by SKIP are always feasible, by the choice of σ. We prove SKIP is optimal among all feasible allocation algorithms, for any control vector.

Theorem 1.3. SKIP *is an instance-optimal algorithm for controlled dynamic fair division.*

Proof. Let ψ be an input for any allocation algorithm $\mathcal{A}$ and Skip. Let σ be the maximum fairness ratio achievable by $\mathcal{A}$ and σ' the maximum fairness ratio achievable by Skip. We consider Skip when σ' is replaced by σ during its execution. We show that Skip is feasible in this case, therefore $\sigma' \geq \sigma$. We assume w.l.o.g. that $\mathcal{A}$ never increases the allocation of any agent (except the arriving agents).

Let $\mathcal{A}^t$ and $\textsc{Skip}^t$ be the sorted allocations of $\mathcal{A}$ and Skip, respectively, for t agents in the system. It suffices to show that $A^t \succeq \textsc{Skip}^t$ for all $t \leq |\psi|$. The proof is by induction on the number of agents in the system, t.

The base case: By the definition of Skip, $\textsc{Skip}^1(1) = \sigma$. Because $\mathcal{A}$'s fairness ratio is σ, it must hold that $\mathcal{A}^1(1) \geq \sigma$.

The inductive step: Assume the statement holds for $t-1$ agents. We show it holds for t. There are two cases: $\psi[t] = 0$ and $\psi[t] = 1$.

Case $\psi[t] = 0$ **:** Let the allocation of Skip at time $t-1$ be $(\textsc{Skip}_1^{t-1}, \ldots, \textsc{Skip}_{t-1}^{t-1})$. Then $\textsc{Skip}^t = (\textsc{Skip}_1^{t-1}, \ldots, \textsc{Skip}_{t-1}^{t-1}, \frac{\sigma}{t})$. Algorithm $\mathcal{A}$ will allocate an amount of resource $x \geq \frac{\sigma}{t}$ to the incoming agent. Let the allocation of $\mathcal{A}$ at time $t-1$ be $(\mathcal{A}_1^{t-1}, \mathcal{A}_2^{t-1}, \ldots, \mathcal{A}_{t-1}^{t-1})$. Then

$$\mathcal{A}^t = (\mathcal{A}_1^{t-1}, \mathcal{A}_2^{t-1}, \ldots, \mathcal{A}_j^{t-1}, x, \mathcal{A}_{j+1}^{t-1}, \ldots, \mathcal{A}_{t-1}^{t-1}),$$

for some $0 \leq j \leq t-1$. It is easy to see that $\mathcal{A}^t \succeq \left(\mathcal{A}_1^{t-1}, \mathcal{A}_2^{t-1}, \ldots, \mathcal{A}_{t-1}^{t-1}, \frac{\sigma}{t}\right)$ for any such value of j. Furthermore, by the induction hypothesis,

$$\left(\mathcal{A}_1^{t-1}, \mathcal{A}_2^{t-1}, \ldots, \mathcal{A}_{t-1}^{t-1}, \frac{\sigma}{t}\right) \succeq (\textsc{Skip}_1^{t-1}, \ldots, \textsc{Skip}_{t-1}^{t-1}, \frac{\sigma}{t}) = \textsc{Skip}^t.$$

This concludes the proof for $\psi[t] = 0$.

Case $\psi[t] = 1$ **:** Starting with the allocation vector at time $t-1$, $\mathcal{A}^{t-1}$, we break the allocation changes at time t into two steps:

1. Reduce the share of the donor, to get $\hat{\mathcal{A}}^{t-1}$.

2. Allocate x to the incoming agent, to obtain $\mathcal{A}^t$.

Denote $\mathcal{A}^{\min} = (\mathcal{A}_2^{t-1}, \ldots \mathcal{A}_{t-1}^{t-1}, \frac{\sigma}{t})$. We show that at the end of the first step, $\hat{\mathcal{A}}^{t-1} \succeq \mathcal{A}^{\min}$. The second step is identical to the $\psi[t] = 0$ case, hence we can combine the two results to conclude that $A^t \succeq \textsc{Skip}^t$. Assume algorithm $\mathcal{A}$ reduced the allocation of an agent from $\mathcal{A}_j^{t-1}$ to y, where $1 \leq j \leq t-1$ (possibly $\mathcal{A}_j^{t-1} = y$). There is some k, $j \leq k \leq t-1$ such that $\mathcal{A}_k^{t-1} \geq y \geq \mathcal{A}_{k+1}^{t-1}$. Then,

$$\hat{\mathcal{A}}^{t-1} = (\mathcal{A}_1^{t-1}, \mathcal{A}_2^{t-1}, \ldots, \mathcal{A}_{j-1}^{t-1}, \mathcal{A}_{j+1}^{t-1}, \mathcal{A}_{j+2}^{t-1}, \ldots, \mathcal{A}_k^{t-1}, y, \mathcal{A}_{k+1}^{t-1}, \ldots \mathcal{A}_{t-1}^{t-1}).$$

For $i \in [1, j-1]$, $\hat{\mathcal{A}}_i^{t-1} \geq \mathcal{A}_i^{\min}$. For $i \in [j, k]$, $\hat{\mathcal{A}}_i^{t-1} = \mathcal{A}_i^{\min}$. Then, by definition of y, $\hat{\mathcal{A}}_{k+1}^{t-1} = y \geq \mathcal{A}_{k+1}^{t-1} = A_{k+1}^{\min}$. Similarly, $\hat{A}_i^{t-1} \geq \mathcal{A}_i^{\min}$, for all $i \in [k+1, t-2]$. For the last term, we note that since $\mathcal{A}$ is σ-fair, $\mathcal{A}_{t-1}^{t-1} \geq \frac{\sigma}{t-1} \geq \frac{\sigma}{t}$ (possibly y is the last share, but then $y \geq \frac{\sigma}{t}$ as $\mathcal{A}$ is σ-fair). □

2.2 Reduction to Basic Control Vectors

We wish to compute a lower bound (positive result) on the fairness ratio of Skip for *all* control vectors, but the optimal fairness ratio for any control vector depends on the vector itself. We show that basic control vectors have the worst fairness ratio; hence in order to provide a lower bound on the fairness ratio, it suffices to analyze basic control vectors. We allow Skip to receive a fairness ratio, σ as an auxiliary input. Denote the (possibly infinite) set of allocations of SKIP when ψ^N is the control vector and σ is the fairness ratio by $\text{SKIP}(\psi^N, \sigma)$. It will be useful to think of the unallocated resource as a "bank": Let $\text{BANK}(\psi^N, t) = 1 - \text{S}(\psi^N, t)$.

We want to show that for any legal c-control vector ψ^N and real number σ, it holds that if $\text{SKIP}(\hat{\psi}^N, \sigma)$ is feasible, then $\text{SKIP}(\psi^N, \sigma)$ is feasible, where $\hat{\psi}^N$ is a basic c-control vector. To this end, we define a series of control vectors $\hat{\psi}^N = \psi_1^N, \psi_2^N, \ldots, \psi_k^N = \psi^N$ such that if $\text{SKIP}(\psi_i^N, \sigma)$ is feasible, then $\text{SKIP}(\psi_{i+1}^N, \sigma)$ is feasible, for all $1 \leq i < k$. We define these vectors inductively: Let t_i^* be the leftmost coordinate on which ψ_i^N and ψ^N differ. The first $t_i^* - 1$ entries of ψ_{i+1}^N are the same as ψ_i^N, the t_i^*-th entry becomes the same as $\psi^N[t_i^*]$, and the remainder continues as $(0^c 1)^\infty$.

Example 2.5. *Let $\psi^7 = (0, 1, 1, 1, 0, 1, 0, 0, 1)$. We derive the ψ_i^7s.*

$$\begin{aligned} \hat{\psi}^N = \psi_1^7 &= (0, 1), (0, 0, 1)^\infty, \\ \psi_2^7 &= (0, 1, 1), (0, 0, 1)^\infty, \\ \psi_3^7 &= (0, 1, 1, 1), (0, 0, 1)^\infty \\ \psi_4^7 &= (0, 1, 1, 1, 0, 1), (0, 0, 1)^\infty = \psi^7 (0, 0, 1)^\infty \end{aligned}$$

Lemma 2.6. *For every c-control vector ψ, there exists some basic c-control vector ψ' such that $\sigma^*(\psi) \geq \sigma^*(\psi')$.*

Proof. We prove that, if $\text{SKIP}(\psi_i^N, \sigma)$ is feasible, then $\text{SKIP}(\psi_{i+1}^N, \sigma)$ is feasible, for all steps t such that $t = t_i^* \pmod{c+1}$ in Lemma 2.7. We then show the same holds for steps $t \neq t_i^* \pmod{c+1}$ in Lemma 2.8. □

Notice that $\text{X}(\psi_i^N)$ and $\text{X}(\psi_{i+1}^N)$ are identical up to step $t_i^* - 1$. Then, on step t_i^*, necessarily $\psi_i^N[t_i^*] = 0$, $\psi_{i+1}^N[t_i^*] = 1$.

Lemma 2.7. $\text{X}(\psi_i^N, t) \succeq \text{X}(\psi_{i+1}^N, t)$ *for all $t = t_i^* \pmod{c+1}$.*

The proof of Lemma 2.7 is by simple induction and is deferred to Appendix B.1. Showing that if $\text{SKIP}(\psi_i^N, \sigma)$ is feasible, then $\text{SKIP}(\psi_{i+1}^N, \sigma)$ is feasible, for steps $t \neq t_i^* \pmod{c+1}$ is more involved. First observe that there exists some t_{max} such that, for all steps $t > t_{max}$, $\text{X}(\psi_{i+1}^N, t)$ is identical to the allocation of some basic control vector at step t (as the allocations produced by SKIP are memoryless). Hence, it suffices to show the result for all $t \leq t_{max}$. The value of t_{max} is computed as follows: Let p be number of 0s between t_i^* and the previous 1, and $p' = c + 1 - p$. In Example 2.5, for ψ_4^7, $p = 1$, $p' = 2$. Set $k = t_i^* - (c+2) + p'$. Once the largest share is $\frac{\sigma}{k}$ (and the second largest is strictly smaller), the allocation is the same as it would have been at time k for some basic control vector (in Example 2.5, this allocation is $\left(\frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{6}, \ldots\right)$). A straightforward calculation gives $t_{max} = t_i^* + (c+1)(k-1) = (c+2)t_i^* + (c+1)(p' - c - 3)$. Note that this is *exactly* the time at which the allocations converge.

Lemma 2.8. *For all $t \neq t_i^* \pmod{c+1}$, $t \in [t_i^*, t_{max}]$,* $\text{BANK}(\psi_{i+1}^N, t) \geq \sum_{j=1}^{c} \frac{\sigma}{t+j}$.

The proof for this Lemma is quite technical; as it mostly involves applications of known techniques, it is deferred to Appendix B.2.

2.3 Bounding the Fairness Ratio of Skip

The allocations created by Skip have a particular structure: they resemble a segment of the harmonic series, with some doubled entries (see Example 2.9 and Table 3). Even though we showed that in order to bound the fairness ratio, it is enough to consider only basic control vectors, each basic control vector has a different fairness ratio. Nevertheless, we would like to provide some upper bound on the fairness ratio of Skip, for each c. We give two types of bounds: an upper bound that applies to all control vectors for any number of agents, and an asymptotic bound. The asymptotic bound is particularly useful for systems that wish to be able to accommodate an unbounded number of agents, and in which the amount of time the system will have fewer than n_0 agents (for some constant n_0) is vanishingly small. In this case, one can set the fairness ratio to be the asymptotic fairness ratio, with an arbitrary "quick fix" heuristic for when there are fewer than n_0 agents in the system (for example, allowing slightly more disruptions to guarantee the asymptotic fairness ratio).

To better characterize these allocations, we need some notation. Elements of an allocation (an element is a real number[6]) that appear once are called *singletons*, and those that appear twice *doubles*. We use the following to make our notation more compact.

$\bowtie_c$ represents a double followed by c singletons, followed by a double, and so on, ending in c singletons. Let the share before $\bowtie_c$ be $\frac{\sigma}{i}$; the first double of $\bowtie_c$ is $\frac{\sigma}{i+1}, \frac{\sigma}{i+1}$. Let the share after $\bowtie_c$ be $\frac{\sigma}{j}$; the last singleton of $\bowtie_c$ is $\frac{\sigma}{j-1}$.

$\rightharpoonup_k$ represents $k-1$ more singletons continuing the series, $k \geq 1$. We use $k=0$ is used to denote no singletons, not even the one before $\rightharpoonup_k$. (This may seem strange at first, but will greatly simplify the notation.)

Example 2.9. *The following are possible allocations, written in full and abbreviated.*

1. $\left(\frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{5}, \frac{\sigma}{5}, \frac{\sigma}{6}, \frac{\sigma}{7}, \frac{\sigma}{8}, \frac{\sigma}{8}, \frac{\sigma}{9}, \frac{\sigma}{10}\right)$ *or* $\left(\frac{\sigma}{3} \rightharpoonup_2, \bowtie_2, \frac{\sigma}{8}, \frac{\sigma}{8}, \frac{\sigma}{9} \rightharpoonup_2\right)$.
2. $\left(\frac{\sigma}{2}, \frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{5}, \frac{\sigma}{5}, \frac{\sigma}{6}, \frac{\sigma}{7}, \frac{\sigma}{8}\right)$, *or* $\left(\frac{\sigma}{2} \rightharpoonup_3, \frac{\sigma}{5}, \frac{\sigma}{5}, \frac{\sigma}{6} \rightharpoonup_3\right)$.
3. $\left(\frac{\sigma}{4}, \frac{\sigma}{4}, \frac{\sigma}{5}, \frac{\sigma}{6}, \frac{\sigma}{7}, \frac{\sigma}{8}, \frac{\sigma}{8}, \frac{\sigma}{9}, \frac{\sigma}{10}, \frac{\sigma}{11}, \frac{\sigma}{12}, \frac{\sigma}{12}, \frac{\sigma}{13}, \frac{\sigma}{14}, \frac{\sigma}{15}\right)$ *or* $\left(\frac{\sigma}{3} \rightharpoonup_0, \bowtie_3, \frac{\sigma}{12}, \frac{\sigma}{12}, \frac{\sigma}{13} \rightharpoonup_3\right)$.

Instead of individually analyzing each basic control vector, for any $c > 0$, we define a set of allocations:

$$S^c = S^c_{1,1}, \dots S^c_{1,c+1}, S^c_{2,0}, S^c_{2,1} \dots, S^c_{2,c+1}, S_{3,0}, S^c_{3,1} \dots, S^c_{3,c+1}, \dots$$

that simultaneously upper bounds all allocations created by all basic control vectors, for a fixed c. (This series is not a series of valid allocations; it is only used for the analysis.) At a high level, $S_{t,i}$ is the allocation created by some basic control vector ψ, such that the largest share is $\frac{1}{t}$ (all of the allocations in Example 2.9 are such allocations; they correspond to $S^2_{3,2}$, $S^3_{2,3}$ and $S^3_{3,0}$ respectively). For ψ, the allocation at this time is necessarily greater than the previous c allocations (as there was no donor for c rounds). The following observation is a characterization of the allocations just before the round when a donor is used. Note that the control vector is characterized by the variable k, which does not appear in the expression; we only claim that for every k and every round t that is just before the donor, there is some i for which the expression holds.

[6] In all of the algorithms that we describe, shares are rational numbers.

Definition 2.10. *For $t = 1$ and $j \in \{1, \dots, c+1\}$,*

$$S^c_{t,j} = \sigma \rightharpoonup_{(j)} .$$

For $t > 1$ and $j \in \{0, 1, \dots, c+1\}$, $t'_{t,j} = (t-1)(c+2) + j - c$,

$$S^c_{t,j} = \frac{\sigma}{t} \rightharpoonup_{(j)}, \bowtie_c, \frac{\sigma}{t'}, \frac{\sigma}{t'}, \frac{\sigma}{t'+1} \rightharpoonup_{(c)} .$$

It is easy to verify that these are exact descriptions of all possible allocations of Skip in the round before a donor is used; we invite the reader to consult Example 2.9 in which the allocations correspond to $S^2_{3,2}$, $S^3_{2,3}$ and $S^3_{4,0}$ respectively.

Lemma 2.11. *The fairness ratio of* Skip *for any $t > 1$, j is at most*

$$\left(\arg\max_{t \in \mathbb{N}^+, j \in \{0, \dots, c+1\}} \sum_{i=t}^{(t-1)(c+2)+j} \frac{1}{i} + \sum_{i=0}^{t-2} \frac{1}{t + i(c+1) + j} \right)^{-1} .$$

Proof. The maximal allocation of S^c is an upper bound on the maximal allocation of Skip, with a basic c-control vector. The allocation of $S_{t,j}$ is

$$\sum_{i=t}^{(t-1)(c+2)+j} \frac{1}{i} + \sum_{i=0}^{t-2} \frac{1}{t + i(c+1) + j},$$

by straightforward summation over the allocation vector of Definition 2.10, where the first is a sum of all the values appearing in the allocation vector and the second part is a sum of the duplicates. □

The following is our bound for basic control vectors, for $c > 3$. For the cases of $c = 1$ and $c = 2$, we get a stronger bounds. We note that this discrepancy between $c = 1, 2$ and $c = 3$ is unavoidable, as the asymptotic bound holds for all time periods for $c = 1, 2$, but not for $c = 3$. The plot for $c = 3$ is slightly misleading, in that it appears that for some control vectors, the asymptotic bound holds. While this is true for all $c < 8$, it ceases to be true thereafter. The plot in Appendix I (taken with Lemma 2.6) shows that no control vector has fairness ratio at most the asymptotic bound. See Appendix C for the missing proofs and the analysis for $c = 1$ and $c = 2$.

Theorem 2.12. *For all $c > 2$, the fairness ratio, $\sigma^* \left((c+1)^{-1} \right)$ for all c-control vectors and all steps t satisfies:*

$$\frac{1}{H_{c+1}} \geq \sigma^* \left((c+1)^{-1} \right) \geq \frac{1}{H_{2c+3} - \frac{1}{2}}.$$

Proof. We consider two cases: $t = 1$ and $t > 1$. For the former, the allocation simply is $1 + \frac{1}{2} + \cdots + \frac{1}{j}$. For control vector $(10^c)^\infty$ this is the first $c+1$ elements of the harmonic progression. For $t > 1$, the total resource allocated is:

$$\sum_{i=t}^{(t-1)(c+2)+j} \frac{1}{i} + \sum_{i=0}^{t-2} \frac{1}{t + i(c+1) + j} \leq \sum_{i=t}^{(t-1)(c+2)+c+1} \frac{1}{i} + \sum_{i=0}^{t-2} \frac{1}{t + i(c+1)}.$$

We bound each term separately. For the term on the left, one can show that the expression is decreasing in t (Lemma B.2 in Appendix B), and therefore is upper bounded by $\sum_{i=2}^{2c+3} \frac{1}{i} = H_{2c+3} - 1$. The term on the right is upper bounded by $\frac{1}{2}$ (Lemma B.3 in Appendix B). Combined, we get that the total resource allocated is at most $H_{2c+3} - \frac{1}{2}$. $H_{2c+3} - \frac{1}{2} > H_{c+1}$, for all $c \geq 2$. Furthermore, the bound of the total resource for $t = 1$, is tight. □

We show a tight asymptotic bound on the fairness ratio of Skip as the number of agents tends to infinity.

Theorem 1.6. *There exists a number n_0 such that, for all $c > 2$, all c-control vectors, and present agents $t > n_0$, the allocation of* Skip *is σ_t-fair, where $\sigma_t = (c+1)\left((c+2)\ln(c+2)\right)^{-1}$.*

Proof. We use the following fact about harmonic sums (Appendix A): $\sum_{x=a}^{b} \frac{1}{x} \leq \ln\left(\frac{b}{a-1}\right)$.

We bound the total allocation at step t, as it appears in Lemma 2.11. For the first term:

$$\sum_{i=t}^{(t-1)(c+2)+j} \frac{1}{i} \leq \ln\left(\frac{(t-1)(c+2)+j}{t-1}\right) = \ln\left(c+2+\frac{j}{t-1}\right),$$

which approaches $\ln(c+2)$ as $t \to \infty$. For the second term:

$$\begin{aligned}\sum_{i=0}^{t-2} \frac{1}{t+i(c+1)+j} &\leq \sum_{i=0}^{t-2} \frac{1}{t+i(c+1)} = \frac{1}{c+1}\sum_{i=0}^{t-2} \frac{1}{\frac{t}{c+1}+i} = \frac{1}{c+1}\sum_{i=\frac{t}{c+1}}^{\frac{t}{c+1}+t-2} \frac{1}{i} \\ &\leq \frac{1}{c+1}\ln\left(\frac{\frac{t}{c+1}+t-2}{\frac{t}{c+1}-1}\right) = \frac{1}{c+1}\ln\left(\frac{t+(t-2)(c+1)}{t-c-1}\right),\end{aligned}$$

which approaches $\frac{1}{c+1}\ln(c+2)$ as $t \to \infty$. Combining gives a bound of $(c+1)^{-1}\left((c+2)\ln(c+2)\right)$ on S^t as t goes to infinity; the theorem follows. □

3 Multiple Resources

We show lower and upper bounds for the optimal fairness ratio of $DDRF$. In Section 4 we show a tighter upper bound for the case $d = 1$ and r resources. We describe algorithms for the following cases:

1. the number of donors allowed is a multiple of the number of resources ($d = kr$), for some integer $k \geq 1$ (Appendix F),
2. $d = 1$, (r is any integer),
3. one disruption is allowed for per $c + 1$ arrivals, for $c \geq 1$.

Our algorithms use solutions to the single resource algorithms as subroutines: For each resource ℓ, we run a copy of a single resource algorithm. Let $\mathcal{SR}$ be an algorithm for the single resource case (we expand shortly about the exact nature of $\mathcal{SR}$). When the t-th agent arrives with demand $\mathcal{D}_t = (\mathcal{D}_{t,1}, \mathcal{D}_{t,2}, \ldots, \mathcal{D}_{t,r})$, the ℓ-th copy of $\mathcal{SR}$ is given as an input the number $D_{t,\ell} \in \{0, 1\}$. If $D_{t,\ell} = 1$, $\mathcal{SR}$ behaves as if an agent arrived. If the number is $D_{t,\ell} = 0$, $\mathcal{SR}$ "ignores" it. The total amount of resource ℓ allocated to agent i is dictated by the ℓ-th copy. The following Lemma, whose proof is deferred to Appendix F, allows us to combine the single resource algorithms:

Lemma 3.1. *Let $\mathcal{MR}$ be a multiple resource algorithm that executes a single resource algorithm $\mathcal{SR}$ with fairness ratio σ^* for each resource. Then, $\mathcal{MR}$ is σ^*-DRF fair.*

Theorem 1.9. *The optimal fairness ratio of $DDRF$ for $d = 1$ donors and $r \geq 2$ resources is bounded by*

- $\sigma^*\left(1, 2\right) \geq 2\left(3 \ln 3\right)^{-1} \approx 0.6068$
- $\sigma^*\left(1, 3\right) \geq 3\left(4 \ln 4\right)^{-1}$
- $\sigma^*\left(1, r\right) \geq \left(H_{2r+1} - \dfrac{1}{2}\right)^{-1}, r > 3.$

Proof. Our algorithm for $d = 1$ will run r copies of the SKIP algorithm, one for each resource, where every copy can use a at least one donor for every r arrivals.

The algorithm maintains a priority list O_t over the copies of SKIP. Let $O_1 = (1, 2, \ldots, r)$: at step $t = 1$, the first copy has the highest priority, the second copy has the second highest, and so on. At the arrival of the t-th agent with demand $D_t = (D_{t,1}, D_{t,2}, \ldots, D_{t,r})$, the ℓ-th copy of the single resource algorithm will be given as an input the number $D_{t,\ell}$ if that number is non-zero (otherwise it receives no input). All copies that received a non-zero input request a donor. Copies with no input stay idle. If there are conflicts, (two or more copies request *different* donors), then the copy with the highest priority in O_t, among those that requested a donor, is the only one allowed to use a donor, and is moved to have the lowest priority in O_{t+1}. The total amount of resource ℓ allocated to agent i is dictated by the ℓ-th copy of SKIP.

Clearly, only one donor is used per step. Therefore we only need to show that the algorithm guarantees a $\sigma^*\left(r^{-1}\right)$ fraction of the DRF utility for each agent (recall that here $\sigma^*\left(r^{-1}\right)$ is the fairness ratio of SKIP when one donor every r arrivals is allowed). Notice that for each step that a copy requests a donor but is not allowed to use one, it moves up one spot in the priority list. A copy cannot be denied a donor more than $r - 1$ times consecutively. This implies that the ℓ-th copy, if seen independently, behaves identically to SKIP, and therefore, by Lemma 3.1 the algorithm is $\sigma^*\left(r^{-1}\right)$-$DRF$ fair. □

Using a similar approach we can show positive results for $d = (c + 1)^{-1}$. The main observation here is that each single resource subroutine cannot be denied a donor for more than $r(c+1) - 1$ steps. The upper bound on the fairness ratio is implied by the upper bound on the single resource case.

Theorem 1.10. *The optimal fairness ratio of $DDRF$ for r resources and $(c + 1)^{-1}$ donors (1 donor per $c + 1$ arrivals) is bounded by $\left(H_{c+1}\right)^{-1} \geq \sigma^*\left(\left(c + 1\right)^{-1}, r\right) \geq \left(H_{2r(c+1)+1} - \frac{1}{2}\right)^{-1}$.*

4 Impossibility Result for $\sigma^*(1,r)$

We prove the following theorem. The complete proof is deferred Appendix G.

Theorem 1.11. *The optimal fairness ratio of $DDRF$ for $d=1$ and $r\geq 2$ resources is bounded by $\sigma^*(1,r)\leq 0.6318\ldots$*

The high level of our approach to the proof of Theorem 1.11 is the following.

1. Write the optimal mixed integer program for a general number of donors d and fix an input $\mathcal{D}$ (to be described later).
2. Notice that the only integer variables are variables $Z_{t,i}\in\{0,1\}$, encoding whether the i-th agent is a donor at step t. If we fix these variables, the remaining program is an LP. Treat these variables as constants Z and consider the dual of this LP.
3. Show an algorithm that, given Z, outputs a feasible solution for the dual with a value of at most $\sigma^*\approx 0.6318$.

The main difficulty is showing that no matter what the adversary picks Z to be, a feasible dual solution can always be constructed, such that we get the desired bound. The optimal integer program for general d is:

$$\begin{aligned}
&\max \sigma\\
\text{subject to: } \forall i\leq t, t\in[N]: \quad & u_i^t \geq \frac{\sigma}{DRF_t}\\
\forall r\in[R], t\in[N]: \quad & \sum_{i=1}^{t} u_{ir}^t D_{i,r}\leq 1, \qquad\qquad \forall t\in[N]: \quad \sum_{i=1}^{t} Z_{t,i}\leq d\\
\forall i,t\in[N]: \quad & u_i^t - u_i^{t-1}\leq 1-Z_{t,i}, \qquad\qquad u_i^t-u_i^{t-1}\geq -z_i^t\\
& Z_{t,i}\in\{0,1\}
\end{aligned}$$

Fixing our integer variables $Z_{t,i}$ and taking the dual for the resulting linear program gives:

$$\begin{aligned}
\min \; & \sum_{t,r} f(t,r)+\sum_{t,i} x(t,i)\left(1-Z_{t,i}\right)+\sum_{t,i} y(t,i)Z_{t,i}\\
\text{subject to} \; & \sum_{t,i} l(t,i)\geq 1\\
\forall t\in[N-1], i\in[t]: \; & \sum_r D_{i,r} f(t,r)+x(t,i)-x(t+1,i)+y(t+1,i)-y(t,i)\geq \frac{1}{DRF_t} l(t,i)\\
\forall i\in[N]: \; & \sum_r D_{i,r} f(N,r)+x(N,i)-y(N,i)\geq \frac{1}{DRF_N} l(N,i)
\end{aligned}$$

Acknowledgements

We thank Ophir Friedler for helpful discussions and the anonymous referees for their useful comments.

References

[1] Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: analysing a food bank problem. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2540–2546. AAAI Press, 2015. 1

[2] Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987. 1

[3] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. *arXiv preprint arXiv:1604.03655*, 2016. 1

[4] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for four agents. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 454–464. ACM, 2016. 1

[5] Arka A. Bhattacharya, David Culler, Eric Friedman, Ali Ghodsi, Scott Shenker, and Ion Stoica. Hierarchical scheduling for diverse datacenter workloads. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 4:1–4:15, 2013. 1

[6] Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *American Mathematical Monthly*, pages 9–18, 1995. 1

[7] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011. 1.3

[8] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, Junxing Wang, et al. The unreasonable fairness of maximum nash welfare. Technical report, 2016. 1.3

[9] Edith Cohen, Graham Cormode, Nick G. Duffield, and Carsten Lund. On the tradeoff between stability and fit. *ACM Trans. Algorithms*, 13(1):7:1–7:24, 2016. 1.3

[10] Richard Cole and Vasilis Gkatzelis. Approximating the nash social welfare with indivisible items. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 371–380. ACM, 2015. 1.3

[11] Nikhil R Devanur, Christos H Papadimitriou, Amin Saberi, and Vijay V Vazirani. Market equilibrium via a primal–dual algorithm for a convex program. *Journal of the ACM (JACM)*, 55(5):22, 2008. 1.3

[12] Danny Dolev, Dror G. Feitelson, Joseph Y. Halpern, Raz Kupferman, and Nathan Linial. No justified complaints: On fair sharing of multiple resources. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 68–75, New York, NY, USA, 2012. ACM. 1

[13] Lester E Dubins and Edwin H Spanier. How to cut a cake fairly. *American mathematical monthly*, pages 1–17, 1961. 1

[14] Leah Epstein and Asaf Levin. Robust algorithms for preemptive scheduling. In *European Symposium on Algorithms*, pages 567–578. Springer, 2011. 1.3

[15] Eric Friedman, Ali Ghodsi, and Christos-Alexandros Psomas. Strategyproof allocation of discrete jobs on multiple machines. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 529–546. ACM, 2014. 1.3

[16] Eric J. Friedman, Christos-Alexandros Psomas, and Shai Vardi. Dynamic fair division with minimal disruptions. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, pages 697–713, 2015. (document), 1, 1.1, 1.1, 1.2, 1.2, 1.2, A.2, F.2

[17] Ali Ghodsi, Vyas Sekar, Matei Zaharia, and Ion Stoica. Multi-resource fair queueing for packet processing. *ACM SIGCOMM Computer Communication Review*, 42(4):1–12, 2012. 1.3

[18] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 24–24, 2011. 1, 1.1, 1.3

[19] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 365–378, 2013. 1

[20] Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: maintaining a constant-competitive steiner tree online. *SIAM Journal on Computing*, 45(1):1–28, 2016. 1.3

[21] Mingyu Guo, Vincent Conitzer, and Daniel M Reeves. Competitive repeated allocation without payments. In *International Workshop on Internet and Network Economics*, pages 244–255. Springer, 2009. 1.3

[22] Anupam Gupta and Amit Kumar. Online steiner tree with deletions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 455–467. Society for Industrial and Applied Mathematics, 2014. 1.3

[23] Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 468–479. Society for Industrial and Applied Mathematics, 2014. 1.3

[24] Avital Gutman and Noam Nisan. Fair allocation without trade. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 719–728. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 1.3

[25] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI*, volume 11, pages 22–22, 2011. 1.3

[26] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: fair scheduling for distributed computing clusters. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 261–276. ACM, 2009. 1

[27] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Transactions on Networking (TON)*, 21(6):1785–1798, 2013. 1.3

[28] Ehud Kalai and Meir Smorodinsky. Other solutions to nash's bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 513–518, 1975. 1.3

[29] Ian Kash, Ariel D Procaccia, and Nisarg Shah. No agent left behind: Dynamic fair division of multiple resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 351–358, 2013. 1, 1.3

[30] David Kurokawa, Ariel D Procaccia, and Nisarg Shah. Leximin allocations in the real world. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 345–362. ACM, 2015. 1.1

[31] Dejan S Milojičić, Fred Douglis, Yves Paindaveine, Richard Wheeler, and Songnian Zhou. Process migration. *ACM Computing Surveys (CSUR)*, 32(3):241–299, 2000. 1

[32] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950. 1.3

[33] Abraham Othman, Christos Papadimitriou, and Aviad Rubinstein. The complexity of fairness through equilibrium. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 209–226. ACM, 2014. 1.3

[34] David C. Parkes, Ariel D. Procaccia, and Nisarg Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 808–825, 2012. 1.3

[35] Elisha A Pazner and David Schmeidler. Egalitarian equivalent allocations: A new concept of economic equity. *The Quarterly Journal of Economics*, pages 671–687, 1978. 1

[36] Steven Phillips and Jeffery Westbrook. Online load balancing and network flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 402–411. ACM, 1993. 1.3

[37] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: sharing the network in cloud computing. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 187–198. ACM, 2012. 1

[38] Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009. 1.3

[39] Erel Segal-Halevi. How to re-divide a cake fairly. *arXiv preprint arXiv:1603.00286*, 2016. 1.3

[40] Hadas Shachnai, Gal Tamir, and Tami Tamir. A theory and algorithms for combinatorial reoptimization. In *LATIN 2012: Theoretical Informatics - 10th Latin American Symposium, Proceedings*, pages 618–630, 2012. 1.3

[41] David Shue, Michael J Freedman, and Anees Shaikh. Performance isolation and fairness for multi-tenant cloud storage. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 349–362, 2012. 1.3

[42] Hugo Steinhaus. The problem of fair division. *Econometrica*, 16(1), 1948. 1

[43] Walter Stromquist. How to cut a cake fairly. *American Mathematical Monthly*, pages 640–644, 1980. 1

[44] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, page 18. ACM, 2015. 1

[45] Toby Walsh. Online cake cutting. In *Algorithmic Decision Theory*, pages 292–305. Springer, 2011. 1, 1.3

[46] Wei Wang, Baochun Li, and Ben Liang. Dominant resource fairness in cloud computing systems with heterogeneous servers. In *INFOCOM, 2014 Proceedings IEEE*, pages 583–591. IEEE, 2014. 1

A Number Theory Facts

Fact A.1.

$$H_n \geq \ln n + \gamma', \tag{1}$$

$$H_n \leq \ln n + \gamma' + \frac{1}{2n}, \tag{2}$$

It is easy to derive the following:

Lemma A.2. *[16] For natural numbers $b > a > 1$,*

$$\ln(b) - \ln(a-1) - \frac{1}{2a-2} \leq \sum_{x=a}^{b} \frac{1}{x} \leq \ln(b) - \ln(a-1).$$

B Missing Proofs from Section 2

B.1 Proof of Lemma 2.7

Lemma 2.7. $\mathrm{X}(\psi_i^N, t) \succeq \mathrm{X}(\psi_{i+1}^N, t)$ *for all* $t = t_i^* \pmod{c+1}$.

Proof. If $t < t_i^*$, the Lemma holds trivially. We prove the case $t \geq t_i^*$ by induction on t. For the base case, denote $\mathrm{X}(\psi_i, t_i^* - 1) = \mathrm{X}(\psi_{i+1}, t_i^* - 1) = (a_1, a_2, \ldots, a_{t_i^*-1})$.

Then

$$\mathrm{X}(\psi_i^N, t_i^*) = (a_1, a_2, \ldots, a_{t_i^*-2}, a_{t_i^*-1}, \sigma/t_i^*),$$
$$\mathrm{X}(\psi_{i+1}^N, t_i^*) = (a_2, a_3, \ldots, a_{t_i^*-1}, \sigma/t_i^*, \sigma/t_i^*).$$

Noticing that $\mathrm{X}(\psi_i, t_i^* - 1)$ is sorted (hence $a_\ell \geq a_{\ell+1}$ for all ℓ), and $a_{t_i^*-1} = \frac{\sigma}{t_i^*-1} > \frac{\sigma}{t_i^*}$ completes the proof of the base case.

For the inductive step, let $\mathrm{X}(\psi_i^N, t) = (a_1, a_2, \ldots, a_t)$ and $\mathrm{X}(\psi_{i+1}^N, t) = (b_1, b_2, \ldots, b_t)$. Since $t = t_i^* \pmod{c+1}$, we know that $\psi_i^N[t+j] = 0$, for all $0 \leq j \leq c-1$, $\psi_i^N[t+c] = 1$ and

$\psi_{i+1}^N[t+j] = 0$ for $j \neq 1 \pmod{c+1}$. p is the distance of t_i^* to the previous 1. $p' = (c+1-p)$. (In Example 2.5, ψ_4^7, $p=2$.)

$$\begin{aligned}
\mathrm{X}(\psi_i^N, t) &= (a_1, \ldots, a_{t-1}, a_t), \\
\mathrm{X}(\psi_i^N, t+p) &= (a_2, \ldots, a_t, \frac{\sigma}{t+1} \rightharpoonup_{(p-1)} \frac{\sigma}{t+p}, \frac{\sigma}{t+p}), \\
\mathrm{X}(\psi_i^N, t+c+1) &= (a_2, \ldots, a_t, \frac{\sigma}{t+1} \rightharpoonup_{(p-1)} \frac{\sigma}{t+p}, \frac{\sigma}{t+p}, \frac{\sigma}{t+p+1} \rightharpoonup_{(c-p)}).
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{X}(\psi_{i+1}^N, t) &= (b_1, \ldots, b_{t-1}, b_t), \\
\mathrm{X}(\psi_{i+1}^N, t+c+1) &= (b_2, \ldots, b_t, \frac{\sigma}{t+1} \rightharpoonup_{(c-1)} \frac{\sigma}{t+c}, \frac{\sigma}{t+c+1}, \frac{\sigma}{t+c+1}).
\end{aligned}$$

The inductive step holds, as $a_\ell \geq b_\ell$ for all ℓ. □

B.2 Proof of Lemma 2.8

Lemma 2.8. *For all $t \neq t_i^* \pmod{c+1}$, $t_i^* \leq t \leq t_{max}$, it holds that* $\textsc{bank}(\psi_{i+1}^N, t) \geq \sum_{j=1}^{c} \frac{\sigma}{t+j}$.

Proof. Recall that p' is the distance from t to the next 1. It must be that $\textsc{bank}(\psi_i^N, t+p') \geq \sum_{j=1}^{c} \frac{\sigma}{t+j+p'}$, as the c entries following the next 1 in ψ_i^N are 0. Therefore it suffices to prove that

$$\begin{aligned}
\mathrm{S}(\psi_i^N, t+p') - \mathrm{S}(\psi_{i+1}^N, t) &\geq \sum_{j=1}^{c} \frac{\sigma}{t+j} - \sum_{j=1}^{c} \frac{\sigma}{t+j+p'} \\
&= \sum_{j=1}^{p'} \frac{\sigma}{t+j} - \sum_{j=c-p'+1}^{c} \frac{\sigma}{t+j+p'}.
\end{aligned}$$

Some simple arithmetic shows that

$$\begin{aligned}
\big(\mathrm{S}(\psi_i^N, t+p') - \mathrm{S}(\psi_{i+1}^N, t)\big) &= \sum_{j=1}^{p'} \frac{1}{t+j} \\
&+ \left(\frac{1}{t_i^*+p'} + \frac{1}{t_i^*+c+1+p'} + \cdots + \frac{1}{t+p'}\right) \\
&+ \left(\frac{1}{t_i^*} + \frac{1}{t_i^*+c+1} + \frac{1}{t_i^*+2(c+1)} + \cdots + \frac{1}{t}\right).
\end{aligned}$$

We show that this is at least $\displaystyle\sum_{j=1}^{p'} \frac{\sigma}{t+j} - \sum_{j=c-p'+1}^{c} \frac{\sigma}{t+j+p'}$, or equivalently, that

$$\sum_{j=c-p'+1}^{c} \frac{1}{t+j+p'} + \left(\frac{1}{t_i^*+p'} + \frac{1}{t_i^*+c+1+p'} + \cdots + \frac{1}{t+p'}\right)$$
$$-\left(\tfrac{1}{t_i^*} + \tfrac{1}{t_i^*+c+1} + \tfrac{1}{t_i^*+2(c+1)} + \cdots + \tfrac{1}{t}\right) \geq 0.$$

It is easily verifiable that the LHS is decreasing in t in this range ($t_i^* \leq t \leq t_{max}$), therefore its suffices to prove the inequality for $t = t_{max}$. Let

$$f(t_i^*) = \left(\frac{1}{t_{max}+c+1} + \frac{1}{t_{max}+c+2} + \cdots + \frac{1}{t_{max}+c+p'}\right)$$
$$+\left(\tfrac{1}{t_i^*+p'} + \tfrac{1}{t_i^*+c+1+p'} + \cdots + \tfrac{1}{t_{max}+p'}\right) + \left(\tfrac{1}{t_i^*} + \tfrac{1}{t_i^*+c+1} + \tfrac{1}{t_i^*+2(c+1)} + \cdots + \tfrac{1}{t_{max}}\right).$$

Consider $f(t_i^*+c+1) - f(t_i^*)$. We prove that $f(t_i^*+c+1) - f(t_i^*) \leq 0$, for all t_i^* and c in Lemma B.1. Given this inequality holds, it remains to show that $f(t_i^*)$ is non-negative in the limit. We can re-write the function as:

$$f(t_i^*) = \sum_{j=1}^{p'} \frac{1}{t_{max}+j} + \frac{1}{c+1} \sum_{j=0}^{t_i^*+p'-c-3} \frac{1}{\frac{t_i^*+p'}{c+1}+j} + \frac{1}{c+1} \sum_{j=0}^{t_i^*+p'-c-3} \frac{1}{\frac{t_i^*}{c+1}+j} =$$
$$= \tfrac{1}{c+1}\left(\textstyle\sum_{j=(t_i^*+p')/(c+1)}^{\frac{(c+2)t_i^*+(c+1)(p'-c-3)+p'}{c+1}} \tfrac{1}{j} - \textstyle\sum_{j=t_i^*/(c+1)}^{\frac{(c+2)t_i^*+(c+1)(p'-c-3)}{c+1}} \tfrac{1}{j}\right) + \sum_{j=(c+2)t_i^*+(c+1)(p'-c-3)+c+1}^{(c+2)t_i^*+(c+1)(p'-c-3)+c+p'} \frac{1}{j}$$

Applying the approximations for the Harmonic number from Appendix A and taking the limit as t_i^* goes to infinity completes the proof. □

Lemma B.1. *$f(t_i^*+c+1) - f(t_i^*) \leq 0$, for all t_i^* and c.*

Proof. Set $t_{max} = (c+2)t_i^* + (c+1)(p'-c-3)$ (this is the t_{max} w.r.t. t_i^*) and $t'_{max} = (c+2)(t_i^* + c+1) + (c+1)(p'-c-3) = t_{max} + (c+1)(c+2)$ (this is w.r.t. t_i^*+c+1). We have that:

$$\begin{aligned} f(t_i^*+c+1) - f(t_i^*) = &\left(\frac{1}{t_{max}+(c+1)(c+2)+c+1} + \cdots + \frac{1}{t_{max}+(c+1)(c+2)+c+p'}\right) \\ &+\left(\frac{1}{t_i^*+c+1+p'} + \frac{1}{t_i^*+2(c+1)+p'} + \cdots + \frac{1}{t_{max}+(c+1)(c+2)+p'}\right) \\ &-\left(\frac{1}{t_i^*+c+1} + \frac{1}{t_i^*+2(c+1)} + \cdots + \frac{1}{t_{max}+(c+1)(c+2)}\right) \\ &-\left(\frac{1}{t_{max}+c+1} + \frac{1}{t_{max}+c+2} + \cdots + \frac{1}{t_{max}+c+p'}\right) \\ &-\left(\frac{1}{t_i^*+p'} + \frac{1}{t_i^*+c+1+p'} + \cdots + \frac{1}{t_{max}+p'}\right) \\ &+\left(\frac{1}{t_i^*} + \frac{1}{t_i^*+c+1} + \frac{1}{t_i^*+2(c+1)} + \cdots + \frac{1}{t_{max}}\right). \end{aligned}$$

This can be re-written as:

$$\begin{aligned}
f(t_i^* + c + 1) - f(t_i^*) &= \sum_{j=1}^{p'} \left(\frac{1}{t_{max} + (c+1)(c+2) + c + j} - \frac{1}{t_{max} + c + j} \right) \\
&\quad + \left(\sum_{j=1}^{c+2} \frac{1}{t_{max} + (c+1)j + p'} \right) - \frac{1}{t_i^* + p'} \\
&\quad - \left(\sum_{j=1}^{c+2} \frac{1}{t_{max} + (c+1)j} \right) + \frac{1}{t_i^*} \\
&= \frac{p'}{t_i^*(t_i^* + p')} - \sum_{j=1}^{c+2} \frac{p'}{\left(t_{max} + (c+1)j + p'\right)\left(t_{max} + (c+1)j\right)} \\
&\quad - \sum_{j=1}^{p'} \frac{(c+1)(c+2)}{\left(t_{max} + (c+1)(c+2) + c + j\right)\left(t_{max} + c + j\right)} \\
&\le \frac{p'}{t_i^*(t_i^* + p')} - \frac{(c+2)p'}{\left(t_{max} + (c+1)(c+2) + p'\right)\left(t_{max} + (c+1)(c+2)\right)} \\
&\quad - \sum_{j=1}^{p'} \frac{(c+1)(c+2)}{\left(t_{max} + (c+1)(c+2) + c + j\right)\left(t_{max} + c + j\right)} \\
&\le \frac{p'}{t_i^*(t_i^* + p')} - \frac{(c+2)p'}{\left(t_{max} + (c+1)(c+2) + p'\right)\left(t_{max} + (c+1)(c+2)\right)} \\
&\quad - \frac{p'(c+1)(c+2)}{\left(t_{max} + (c+1)(c+2) + c + p'\right)\left(t_{max} + c + p'\right)}.
\end{aligned}$$

The expression above is equal to:

$$\begin{aligned}
f(t_i^* + c + 1) - f(t_i^*) \le \frac{1}{-1 + (2+c)p + (c+2)t} &+ \frac{1}{3 + c^2 - 2p - 2t - c(-3 + p + t)} \\
&+ \frac{((c+1)(1 - 3p - 4t + c(-1 + p + t)^2 + 2(p+t)^2))}{(t(t+p)(-1 + p + 2t + c(-1 + p + t))(-1 + 2p + 2t + c(-1 + p + t)))}.
\end{aligned}$$

This is easily verified by computer to be non-positive.

□

Lemma B.2. *The function*

$$f(t) = \sum_{i=t}^{(t-1)(c+2)+c+1} \frac{1}{i}$$

is monotone decreasing in t*, for integer values of* t *and integer* $c > 0$.

Proof.

$$\begin{aligned} f(t) - f(t+1) &= \sum_{i=t}^{(t-1)(c+2)+c+1} \frac{1}{i} - \sum_{i=t+1}^{t(c+2)+c+1} \frac{1}{i} \\ &= \frac{1}{t} - \sum_{i=t(c+2)}^{t(c+2)+c+1} \frac{1}{i} \\ &> \frac{1}{t} - (c+2)\frac{1}{t(c+2)} = 0 \end{aligned}$$

□

Lemma B.3. *For integer $c > 1$ and integer $t \geq 2$,*

$$\sum_{i=0}^{t-2} \frac{1}{t+i(c+1)} \leq 1/2.$$

Proof. It suffices to prove the Lemma for $c = 2$, as the sum decreases as c increases.

$$\begin{aligned} \sum_{i=0}^{t-2} \frac{1}{t+3i} &= \frac{1}{t} + \sum_{i=1}^{t-2} \frac{1}{t+3i} \\ &\leq \frac{1}{t} + \int_0^{t-2} \frac{1}{t+3x} dx \\ &= \frac{1}{t} + \frac{1}{3}\ln(4-6/t) \\ &\leq \frac{1}{t} + \frac{1}{3}\ln 4 \\ &\leq 1/2, \end{aligned}$$

for $t \geq 27$. It is easy to computationally verify the Lemma holds for smaller t. □

C The Fairness ratio of CDFD for $c = 1$ and $c = 2$

We prove the following.

Theorem 1.5. *The optimal fairness ratio of $CDFD$ for any c control vector for $c = 1$ and $c = 2$ are*

$$\begin{aligned} \sigma^*\left((2)^{-1}\right) &= 2\,(3\ln 3)^{-1} \text{ and} \\ \sigma^*\left((3)^{-1}\right) &= 3(4\ln 4)^{-1} \end{aligned}$$

respectively.

For $c = 1$, there are two basic control vectors: $(1, 0, 1, 0, \ldots)$ and $(0, 1, 0, 1, \ldots)$. For $c = 2$, there are three basic control vectors (Example 2.2). We prove the bound in Theorem1.5 for each basic control vector separately. The theorem then follows from Lemma 2.6. Here we only present the proof for one of the two control vectors. We omit the proof for the latter control vector as it is virtually identical. Furthermore, we only present the proof for $c = 1$; the proof for $c = 2$ is by similar (slightly more involved) case analysis.

step	allocation for $(0, 1, \ldots)$	sum	allocation for $(1, 0, \ldots)$	sum
1	σ	σ	σ	σ
2	$\frac{\sigma}{2}, \frac{\sigma}{2}$	σ	$\sigma, \frac{\sigma}{2}$	$\frac{90\sigma}{60}$
3	$\frac{\sigma}{2}, \frac{\sigma}{2}, \frac{\sigma}{3}$	$\frac{80\sigma}{60}$	$\frac{\sigma}{2}, \frac{\sigma}{3}, \frac{\sigma}{3}$	$\frac{70\sigma}{60}$
4	$\frac{\sigma}{2}, \frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{4}$	$\frac{80\sigma}{60}$	$\frac{\sigma}{2}, \frac{\sigma}{3}, \frac{\sigma}{3}, \frac{\sigma}{4}$	$\frac{85\sigma}{60}$
5	$\frac{\sigma}{2}, \frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{4}, \frac{\sigma}{5}$	$\frac{92\sigma}{60}$	$\frac{\sigma}{3}, \frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{5}, \frac{\sigma}{5}$	$\frac{79\sigma}{60}$
6	$\frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{4}, \frac{\sigma}{5}, \frac{\sigma}{6}, \frac{\sigma}{6}$	$\frac{82\sigma}{60}$	$\frac{\sigma}{3}, \frac{\sigma}{3}, \frac{\sigma}{4}, \frac{\sigma}{5}, \frac{\sigma}{5}, \frac{\sigma}{6}$	$\frac{89\sigma}{60}$

Table 3: Allocations for the first 6 steps of SKIP with basic 1-control vectors, with the sums of the allocations all given with the same denominator for easier comparison.

Lemma C.1. *The allocations created by* SKIP, *with basic control vector* $\psi^1 = (0, 1, \ldots)$ *(from step 6 onward) are*

1. *On steps* $t = 0 \pmod 6$*:* $\frac{\sigma}{(t/3)+1}, \bowtie_1, \frac{\sigma}{t}, \frac{\sigma}{t}$.

2. *On steps* $t = 2 \pmod 6$*:* $\left(\frac{\sigma}{((t+1)/3)+1}\right) \bowtie_1, \frac{\sigma}{t}, \frac{\sigma}{t}$.

3. *On steps* $t = 4 \pmod 6$*:* $\frac{\sigma}{((t+2)/3)}, \rightharpoonup_1, \bowtie_1, \frac{\sigma}{t}, \frac{\sigma}{t}$.

On odd steps, add $\frac{\sigma}{t}$ *to the previous step (note the denominators have "changed" relative to the new step).*

Proof. The proof is by induction on the round number. The base case (step 6) appears in Table 3. The move from even to odd steps is immediate, as there is no donor. Because only the agent with the highest utility has her allocation decreased, it is easy to verify the transition from odd to even steps, by renaming the denominators. □

In order to compute the optimal fairness ratio achieved by SKIP on ψ^1, we bound the sum of allocations at odd steps, as each odd step uses strictly more resources than the previous even step.

We bound the size of BANK(ψ^1, t), noting for all $t > 0$, BANK$(\psi^1, t) < 1$. On odd steps, we take $\frac{\sigma}{i}$ from the bank. On all even steps except the second and fourth, we return some resource to the bank. (Note that to be able to reach step 5, we need $\frac{23\sigma}{15}$ in the bank; this immediately implies that $\sigma \leq \frac{15}{23}$.) First, we show that the resource allocated monotonically increases, when we look at it through a slightly wider lens:

Lemma C.2. *The total resource allocated by* SKIP(ψ^1) *in steps* $t = 5 \pmod 6$ *is (strictly) monotone increasing.*

Proof. It is easy to verify the following, from Lemma C.1:

Fix some $\tau = 0 \pmod 6$, $\tau \geq 6$.

1. On step τ, we give $\sigma\left(\frac{\sigma}{\tau/3} - \frac{2}{\tau}\right) = \frac{\sigma}{\tau}$ to the bank.

2. On step $\tau + 2$, we give $\sigma\left(\frac{1}{\tau/3+1} - \frac{2}{\tau+2}\right) = \frac{\sigma\tau}{(\tau+2)(\tau+3)}$

3. On step $\tau + 4$, we return $\sigma\left(\frac{1}{\tau/3+2} - \frac{2}{\tau+4}\right) = \frac{\sigma\tau}{(\tau+4)(\tau+6)}$.

On steps $\tau + 1, \tau + 3, \tau + 5$, we take $\frac{\sigma}{\tau+1}, \frac{\sigma}{\tau+3}, \frac{\sigma}{\tau+5}$ respectively. It is easy to verify (by simple calculus), that for $\tau \geq 6$, the sum of what we take over these 6 steps is greater than the sum of what we give. The proof follows. □

We now complete the proof of Theorem 1.5 (for the control vector $(1, 0, 1, 0, \ldots)$).

Sketch of proof of Theorem 1.5. From Proposition C.2, the amount of resource allocated increases; therefore we only need to make sure the algorithm doesn't over allocate when t goes to infinity. We analyze the case when the allocation is larger than all previous allocations, $t = 5 \pmod 6$. At time $t = 0 \pmod 6$, the total resource allocated is

$$\sigma\left(2\sum_{i=1}^{t/3}\frac{1}{t/3+2i} + \sum_{i=1}^{t/3}\frac{1}{t/3+2i-1}\right) \approx \sigma\left(\frac{3}{2}\sum_{i=t/3}^{t}\frac{1}{i}\right) \approx \sigma\frac{3\ln 3}{2}.$$

It must hold that $\sigma\frac{3\ln 3}{2} \leq 1$, and the allocations at times $t = 0 \pmod 6$ and $t = 5 \pmod 6$ are asymptotically the same. The theorem follows. □

D Accommodating Departures

In Section 2 we described the input to $CDFD$ as a c-control vector ψ. This was sufficient in the "arrivals-only" model, but when allowing for departures of agents the situation can get more complex; the optimal fairness ratio could in theory depend on a number of parameters, for example who the departing agent is. In this subsection we prove that this is not the case.

The problem now is the following: we are given a c-control vector ψ. Agents arrive and depart arbitrarily. If $\psi[t] = 1$ the algorithm is allowed to use a donor when there are $t - 1$ agents in the system and a new one arrives. This could happen multiple times. Call this the *arrivals-departures model*. We prove that the fairness ratio of Skip for a given control vector ψ is the same as when agents are not allowed to depart.

Theorem D.1. *The optimal fairness ratio $\sigma^*(\psi)$ of* Skip *with input ψ is the same in the arrivals-departures model as in the arrivals-only model.*

Proof. Let X^t and X^{t+1} be the (sorted) allocations of Skip for a given vector ψ. It suffices to show that even when an arbitrary agent from X^{t+1} departs, it is possible to distribute her share in a way that the sorted allocation is X^t. If $\psi[t+1] = 0$, this is trivial; we focus on the case that $\psi[t+1] = 1$, i.e., the agent with the highest utility at step t was a donor. The two allocations we consider are $\mathrm{X}^t = (a_1, a_2, \ldots, a_t)$ and $\mathrm{X}^{t+1} = \left(a_2, a_3, \ldots, a_t, \frac{\sigma}{t+1}, \frac{\sigma}{t+1}\right)$.

Assume that one of the last two agents, with a $\frac{\sigma}{t+1}$ share, departs. Since both X^t and X^{t+1} are feasible, $a_1 - 2\frac{\sigma}{t+1}$ is equal to the difference $\textsc{bank}^{t+1} - \textsc{bank}^t$. Therefore, there must be a way to combine the departing agent's share $\frac{\sigma}{t+1}$, with the other share equal to $\frac{\sigma}{t+1}$, and the unallocated amount $\textsc{bank}^{t+1}$, to get a_1. Assume for contradiction this is not the case; then $a_1 > 2\frac{\sigma}{t+1} + \textsc{bank}^{t+1}$, which implies that $a_1 - 2\frac{\sigma}{t+1} > \textsc{bank}^{t+1}$. The LHS is equal to $\textsc{bank}^{t+1} - \textsc{bank}^t$. Combining gives $\textsc{bank}^t < 0$, a contradiction.

If the departing agent is some agent $j \in [2, t-1]$, we can do the following: allocate the difference $a_j - \frac{\sigma}{t+1}$ (which is positive since the allocation is sorted) to one of the last two agents. The amount left to distribute is exactly $\frac{\sigma}{t+1}$; we've already shown this is sufficient to increase the share of the other of the last two agents to a_1. □

E $GDFD$ is NP-hard

In this section we prove Theorem 1.7.

Theorem 1.7. *$GDFD$ is* NP*-hard, even for* 2 *resources,* 1 *donor, binary demand vectors, and clairvoyant algorithms.*

We show a reduction from the *Bounded Partition Problem (BPP)*: we are given a set $S = \{a_1, \ldots, a_{2n}\}$ of rational numbers, $a_i \in (\alpha, \beta)$. The goal is to partition S into two subsets of equal size, such that the sums of the numbers in each subset are equal. This problem is NP-hard for any $\beta > \alpha$. Assume w.l.o.g. that $a_1, \ldots, a_{2n}$ are non-decreasing.

In order to make the construction cleaner, and w.l.o.g., allow resources to have capacity more than 1. We reduce BPP to the following instance of $GDFD$ with 2 resources of capacity $\frac{3}{2}\sum_{i=1}^{2n} a_i$ each, 1 donor, and

- $4n$ agents: the first n have a demand vector $[0, 1]$, the next n have a demand vector $[1, 0]$, and the last $2n$ agents have a demand vector $[1, 1]$,
- $U = (1, \ldots, 1, a_{2n})$ ($4n - 1$ "1"'s, followed by a_{2n}), and
- $L = (1, \ldots, 1, a_{2n}, a_{2n-1}, \ldots, a_1)$ ($2n$ "1"'s, followed by $a_{2n}, \ldots, a_1$).

The key observation is that up to step $2n$, the first $2n$ agents must have utility 1, but at step $4n$ they all must have utility less than or equal to a_{2n}, and thus each one of them must be a donor exactly once. Therefore, the utility of an agent i, for $i > 2n$, never changes, is at least a_{4n-i+1}, and it contributes that amount to each of the two resources. Picking a $[1, 0]$ or a $[0, 1]$ agent as a donor for that step is equivalent to deciding in which partition the number a_{4n-i+1} belongs.

We need to show that there is a satisfying allocation if and only if there is a partition.

Observation E.1. *The following must hold:*

1. *The utilities of all agents up to time* $2n$ *must be exactly* 1.
2. *Since* $a_{2n} < 1$*, in order to meet the final upper bound* $U(4n) = a_{2n}$*, each of the first* $2n$ *agents must be a donor at least once. But since there are only* $2n$ *time steps, each of the first* $2n$ *agents must be a donor exactly once.*
3. *None of the last* $2n$ *agents can be donors.*

Let $x_1, \ldots, x_{2n}$ denote the final utilities of the first $2n$ agents, and let $z_1, \ldots, z_{2n}$ denote the final utilities of the last $2n$ agents (in reverse order). Consider how these utilities evolve, starting from step $2n$. Notice that, by Observation E.1, the values of the last $2n$ agents do not change once they are set, and the values of the first $2n$ agents are initially set to 1 (at time $2n$), and then change once.

Therefore, if the lower bounds L are satisfied, $z_i \geq a_i$ for all $i \in [2n]$, and there is some permutation π of $X = \{x_1, \dots, x_{2n}\}$ such that $\pi(x_i) \geq a_i$ for all $i \in 2n$. Note that each $[1,1]$ contributes twice its utility to the total resource allocated. Hence, the total allocated resource is:

$$\sum_{i=1}^{2n} x_i + 2z_i = \sum_{i=1}^{2n} \pi(x_i) + 2z_i \geq \sum_{i=1}^{2n} 3a_i.$$

As this is exactly the total available resource, equality must hold everywhere; that is, for all $i \in [2n]$, $z_i = \pi(x_i) = a_i$. For each of the two resources, the amount allocated to the last $2n$ agents is $\sum_{i=1}^{2n} a_i$, hence $\frac{1}{2}\sum a_i$ of each resource is allocated to the first $2n$ agents; hence there is a partition of the numbers $a_i, \dots a_{2n}$ into equal sized subsets such that the sum of elements of each subset is $\frac{1}{2}\sum a_i$, as required.

To verify that a partition implies a good allocation, we need to verify that

1. U and L are not violated.

2. The capacity of the resources is never exceeded.

The first requirement is immediate from the allocation process described above; as the final allocation satisfies the capacity bound, the following Lemma suffices to prove the second.

Lemma E.2. *The allocation of each resource is monotone non-decreasing.*

Proof. There are no donors in the first $2n$ rounds, therefore it suffices to look at the final $2n$ rounds (which we now label $1, \dots, 2n$). We have verified that in each of these rounds, an agent with demand $[1,1]$ arrives, and either $[1,0]$ or $[0,1]$ is the donor. An agent with demand $[1,1]$ arrives at round $i \in [2n]$. Assume w.l.o.g. that the donor is $[1,0]$. The amount allocated of the second resource increases by $a_i > 0$, and of the first decreases by $1 - a_i$, and increases by a_i, for a total increase of $2a_i - 1 > 0$, as $a_i > 0.5$. □

F Proofs missing from Section 3

F.1 Proof of Lemma 3.1

Proof. The DRF allocation at step t is: $DRF_{\mathcal{D}^t} = \left(\max\limits_{l=1,\dots,r}\left\{\sum_{j=1}^{t} \mathcal{D}^t_{j,l}\right\}\right)^{-1}$, where $\mathcal{D}_{j,l}$ is the demand for the l-th resource by agent j.

Notice that the amount of resource l agent i has at step t is at least $\sigma^* \frac{\mathcal{D}_{i,l}}{\sum_{j=1}^{t} \mathcal{D}_{j,l}} \geq \sigma^* \cdot \mathcal{D}_{i,l} \cdot DRF_t$, since the l-th copy of the single resource algorithm has received as input the numbers $\{\mathcal{D}_{1,l}, \dots, \mathcal{D}_{t,l}\}$ by step t, and their sum is exactly the number of agents that demand resource l. The utility of agent i for a vector of resources $v = (v_1, \dots, v_r)$ is $\min\limits_{l=1\dots r}\left\{\frac{v_l}{\mathcal{D}_{i,l}}\right\}$. Therefore, for the vector of resources

$$(\sigma_k^* \cdot \mathcal{D}_{i,1} \cdot DRF_t, \dots, \sigma_k^* \cdot \mathcal{D}_{i,r} \cdot DRF_t) = \sigma^* DRF_t\left(\mathcal{D}_{i,1}, \dots, \mathcal{D}_{i,r}\right).$$

The fairness ratio is therefore $\sigma^* \cdot DRF_t$. □

F.2 Theorem 1.8

Theorem 1.8. *The optimal fairness ratio of $DDRF$ for $d = kr$ donors and r resources is bounded by*

$$\left((kr+1)\ln\left(\frac{kr+1}{kr}\right)\right)^{-1} \geq \sigma^*(kr, r) \geq \left((k+1)\ln\left(\frac{k+1}{k}\right)\right)^{-1}$$

Proof. For the upper bound, notice that $\sigma^*(kr, r) \leq \sigma^*(kr, 1) \leq \left((kr+1)\ln\left(\frac{kr+1}{kr}\right)\right)^{-1}$. For the lower bound, our algorithm for $d = kr$ runs r copies of the optimal single resource algorithm from [16][7], one for each resource, where every copy is allowed to use at most k donors per arrival. The total number of donors is $d = kr$, and by Lemma 3.1 the algorithm guarantees a σ^*_k fraction of the DRF utility for each agent. □

G Proofs missing from Section 4

G.1 A Feasible Dual Solution

Let $\hat{Z}$ be an $4n$ by $4n$ lower triangular matrix such that $\hat{Z}_{t,i} = DRF_t$ if and only if $Z_{t,i} = 1$, and $Z_{t',i} = 0$ for all $t' > t$. In other words, $\hat{Z}_{t,i} = DRF_t$ if t is the last time agent i becomes a donor. If agent i is never a donor, then we let $\hat{Z}_{i,i} = DRF_i$.

Let s_1 be the sum of all elements in the first n columns of $\hat{Z}$, s_2 the sum of all elements in the next n columns of $\hat{Z}$, and s_3 the sum of all elements in the last $2n$ columns. We say that a column i is *valid* if $i > 2n$, or $i \in [1, n]$ and $s_1 > s_2$, or $i \in [n+1, 2n]$ and $s_2 > s_1$. Finally, let $\sigma = \frac{1}{\max\{s_1, s_2\} + s_3}$.

Our dual solution for a given choice of Z is as follows:

- $x(t, i) = 0, \forall t, i$, and $f(t, r) = 0, \forall t \leq N-1, r \in \{1, 2\}$.
- $f(N, 1) = \sigma$ if $s_1 \geq s_2$, and zero otherwise. $f(N, 2) = \sigma$ if $s_1 > s_2$, and zero otherwise.
- $y(t, i) = \sigma$, for all $t > t'$, where t' is such that $\hat{Z}_{t',i} \neq 0$, and i is *valid*. $y(t, i) = 0$ everywhere else. In other words, $y(t, i)$ is σ from the time after i was a donor for the last time, but only for i that is *valid*. Notice that this makes $y(t, i) \cdot Z_{t,i} = 0$ everywhere.
- $l(t, i) = \sigma \cdot DRF_t$ if and only if $\hat{Z}_{t,i} \neq 0$, and i is *valid*.

See Appendix H for an example.

Lemma G.1. *The solution described above is feasible for all Z and n, and the value of the objective is σ.*

Proof.

- **Objective:** Notice that $x(t, i) = y(t, i)Z_{t,i} = 0$, for all t, i, therefore the objective is simply $\sum_r f(N, r)$. At most one of $f(N, r)$ can be non-zero, with a value of σ.
- **First line of constraints:**

$$\sum_{t,i} l(t, i) = \sigma\left(\sum_{t,i \text{ valid}} \hat{Z}_{t,i}\right) = \sigma(\max\{s_1, s_2\} + s_3) = 1.$$

[7] Alternatively, we can use SKIP with the appropriate inputs.

- **Second line of constraints:**

  These constraints reduce to $y(t+1,i) - y(t,i) \geq \frac{1}{DRF_t} l(t,i)$. $l(t,i)$ is non zero only when $\hat{Z}_{t,i}$ is non-zero and i is valid, in which case, $y(t,i) = 0$, but $y(t+1,i) = \sigma$, therefore the constraint is satisfied. When $l(t,i)$ is zero, either both $y(t,i)$, $y(t+1,i)$ are zero, or both are equal to σ, therefore the constraint is again satisfied.

- **Third line of constraints:**

  These constraints reduce to $\sum_r f(N,r) - y(N,i) = \sigma - y(N,i) \geq \frac{1}{DRF_N} l(N,i)$. $l(N,i)$ is non-zero only when $\hat{Z}_{t,i}$ is non-zero and i is valid, in which case $y(N,i) = 0$, and therefore the constraint is satisfied. If $l(N,i) = 0$, then notice that the RHS is zero, and $y(N,i) \leq \sigma$, therefore the constraint is again satisfied.

□

Bounding σ

We now show that no matter how Z is chosen, σ is at most $0.6318\ldots$ Recall that $\sigma = \frac{1}{\max\{s_1,s_2\}+s_3}$. Therefore, in order to maximize σ, an adversary that picks Z needs to minimize $\max\{s_1, s_2\} + s_3$. Every choice of Z yields $\hat{Z}$ in a unique way, with all non zero $\hat{Z}_{t,i}$s taking certain values that depend on DRF.

Define $\mathcal{D}$ as follows: n $[0,1]$'s followed by n $[1,0]$'s, followed by a series of $2n$ $[1,1]$'s This input has: $DRF_t = \frac{1}{t}$ for $t = 1, \ldots, n$, $DRF_t = \frac{1}{n}$ for $t = n+1, \ldots, 2n$, and $DRF_t = \frac{1}{t-n}$ for $t = 2n+1, \ldots, 4n$. Instead of bounding the minimum $(\max\{s_1, s_2\} + s_3)$, we bound the minimum $\left(\frac{s_1+s_2}{2} + s_3\right)$. Notice that this value is only smaller, therefore our bound for σ is worse. Minimizing $\left(\frac{s_1+s_2}{2} + s_3\right)$ is equivalent to the following game:

We have an ordered set of $4n$ numbers A, where $A_t = DRF_t$ as described above. We also have a set of $4n-1$ numbers $B = A \setminus \{1\}$. Replace numbers in A with numbers in B, using each number in B at most once, to obtain $\hat{A}$. s_1 is the sum of the first n elements of $\hat{A}$, s_2 the sum of the next n elements, and s_3 the sum of the last $2n$ elements. Associate each number $A_i \in A$ with a weight w_i, which is $\frac{1}{2}$ if $i \leq 2n$, and 1 otherwise. Our objective is to minimize the weighted sum $\sum_{i=1}^{4n} w_i \hat{A}_i$. We show a strategy for this game that has value 1.58258. This gives $\sigma \approx 0.6318$.

Example G.2. *For $n = 2$, we have*

$$B = \{1/2, 1/2, 1/2, 1/3, 1/4, 1/5, 1/6\}.$$

A possible way to change A into $\hat{A}$ is the following (numbers that were not replaced are in bold):

w	1/2	1/2	1/2	1/2	1	1	1	1
A	1	1/2	1/2	1/2	1/3	1/4	1/5	1/6
$\hat{A}$	1/5	1/4	1/6	1/3	**1/3**	**1/4**	**1/5**	**1/6**

The sum of the first $n = 2$ elements of $\hat{A}$ is $\frac{1}{5} + \frac{1}{4} = 0.45$, the sum of the next n elements is $\frac{1}{3} + \frac{1}{6} = 0.5$, and the sum of the last $2n$ elements is $\frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} = 0.95$. The tight bound is $\max\{0.45, 0.5\} + 0.95 = 1.45$. *The value of the game, i.e., the weighted sum of $\hat{A}$, $\frac{s_1+s_2}{2} + s_3$, is* 1.425.

Observation G.3. *In the optimal solution, $\hat{A}_i \leq \frac{1}{n}$, for all i.*

Notice that if all the weights are the same, the problem is trivial: greedily replace the largest number in A with the smallest number in B, until the largest number in A is smaller than the smallest available number in B.

In order to minimize the weighted sum of A by replacing numbers in A with numbers in B, observe the following: for each pair of numbers $A_i, A_j \in A$, with $w_i < w_j$, we can find a number $x_{i,j}$, such that replacing A_i with a number $x \geq x_{i,j}$ gives a smaller or equal weighted sum than replacing A_j with $x_{i,j}$, and $x_{i,j}$ is the smallest such number. Therefore, $x_{i,j}$ is the solution to $A_i w_i + w_j x \geq w_j A_j + w_i x$, which is $x_{i,j} = \frac{w_j A_j - w_i A_i}{w_j - w_i}$. If $w_i = w_j$ then it is better to replace the smaller of the two.

The following Lemma is immediate:

Lemma G.4. *In the optimal solution: (1) all $i \geq 3n$ are never replaced, and (2) all $i \leq \frac{n}{2}$ are always replaced.*

Proof. We show that if for $i \geq 3n$, i.e., $A_i \leq \frac{1}{2n}$ and $w_i = 1$, then $x_{j,i} \leq 0$, for all $j \leq 2n$ ($w_j = 1/2$). This implies that for all positive numbers x, replacing A_j with x is better than replacing A_i with x. It suffices to consider the maximum reasonable value that A_j can take, which is $\frac{1}{n}$ (Observation G.3). By replacing $w_i = 1$, $A_j = \frac{1}{n}$ and $w_j = \frac{1}{2}$ we have: $x_{j,i} = \frac{w_i A_i - w_j A_j}{w_i - w_j} = 2A_i - \frac{1}{n}$, which less than zero, exactly when $A_i \leq \frac{1}{2n}$. The proof of the second part of the lemma is similar and is omitted. □

Even though, given a number $x \in B$, we can find the optimal number to replace, it is not generally true that the optimal algorithm considers all B_i in decreasing order. Regardless, we can show the following structure for the optimal strategy:

Lemma G.5. *In the optimal solution, if some $i > 2n$ is replaced by $b \in B$, then for all $b' \in B$ that replaced $j \in [0, 2n]$, we have $b \leq b'$.*

Proof. Assume this is not the case. This means that there is some $j \in [0, 2n]$ such that A_j was replaced by some $b' < b$. Let $S + \frac{b'}{2} + b$ be the value of the game in this strategy. Consider instead the following strategy: replace A_j with b and A_i with b'. The value now is $S + \frac{b}{2} + b' = S + \frac{b}{2} + \frac{b'}{2} + \frac{b'}{2} < S + b + \frac{b'}{2}$, which is the value of the optimal strategy; a contradiction. □

An immediate corollary is that the smallest numbers in B are used to replace the largest numbers A_i, for $i > 2n$. Combining with Lemma G.4, we get that in the optimal solution, a fraction f^* of the n smallest numbers in B is used to replace all $i \in [2n + 1, (2 + f^*)n]$, while the remaining numbers in B are used to replace $i \leq 2n$. There are $2n$ numbers in B that are smaller than $\frac{1}{n+1}$, therefore for $i \leq 2n$, a $1 - \frac{f^*}{2}$ fraction of them are replaced, while the $f^* n$ numbers remaining are equal to $\frac{1}{n}$. The value of the optimal strategy is:

$$
\begin{aligned}
V^* = \sum_{i=1}^{4n} w_i \hat{A}_i &= \frac{1}{2}\sum_{i=1}^{2n} \hat{A}_i + \sum_{i=2n+1}^{4n} \hat{A}_i \\
&= \frac{1}{2}\left(\sum_{i=f^*n+1}^{n} B_i + f^* n \frac{1}{n}\right) + \sum_{i=1}^{f^*n} B_i + \sum_{i=(2+f^*)n+1}^{4n} \hat{A}_i \\
&= \frac{1}{2}\left(\sum_{i=n+1}^{(3-f^*)n} \frac{1}{i} + f^*\right) + \sum_{i=(3-f^*)n}^{3n} \frac{1}{i} + \sum_{i=(1+f^*)n+1}^{3n} \frac{1}{i}.
\end{aligned}
$$

At the limit this converges to:

$$
\frac{f^*}{2} + \frac{1}{2}\ln(3-f^*) + \ln\left(\frac{3}{3-f^*}\right) + \ln\left(\frac{3}{1+f^*}\right).
$$

This function is convex for $f \in [0,1]$, with a minimum at $\frac{5-\sqrt{17}}{2} \approx 0.4384$, with a value of 1.5825. We conclude that σ is at most $\frac{1}{1.58258} \approx 0.6318$.

H Example of constructing a feasible dual solution

For a concrete example, let $n = 2$, with Z and $\hat{Z}$ as follows:

$$
Z = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
$$

$$\hat{Z} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ DRF_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & DRF_5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & DRF_6 & 0 & DRF_6 & 0 & 0 \\ 0 & DRF_7 & 0 & 0 & 0 & 0 & DRF_7 & 0 \\ 0 & 0 & 0 & 0 & DRF_8 & 0 & 0 & DRF_8 \end{bmatrix}$$

with $s_1 = DRF_4 + DRF_7$, $s_2 = DRF_5 + DRF_6$, $s_3 = 2 * DRF_8 + DRF_6 + DRF_7$. We have that $DRF_4 = \frac{1}{2}$, $DRF_5 = \frac{1}{3}$, $DRF_6 = \frac{1}{4}$, $DRF_7 = \frac{1}{5}$ and $DRF_8 = \frac{1}{6}$. Therefore, $s_1 = 0.7$, $s_2 = \frac{7}{12}$ and $s_3 = \frac{7}{12} + \frac{1}{5}$. Therefore, we have that

$$\sigma = \frac{1}{s_1 + s_3} = 0.6741.$$

This gives us:

$$y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma & 0 & 0 & 0 & 0 & \sigma & 0 & 0 \\ \sigma & \sigma & 0 & 0 & 0 & \sigma & \sigma & 0 \end{bmatrix}$$

$$l = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma DRF_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma DRF_6 & 0 & 0 \\ 0 & \sigma DRF_7 & 0 & 0 & 0 & 0 & \sigma DRF_7 & 0 \\ 0 & 0 & 0 & 0 & \sigma DRF_8 & 0 & 0 & \sigma DRF_8 \end{bmatrix}$$

and $f(N,1) = \sigma$, $f(N,2) = 0$.

I Figure

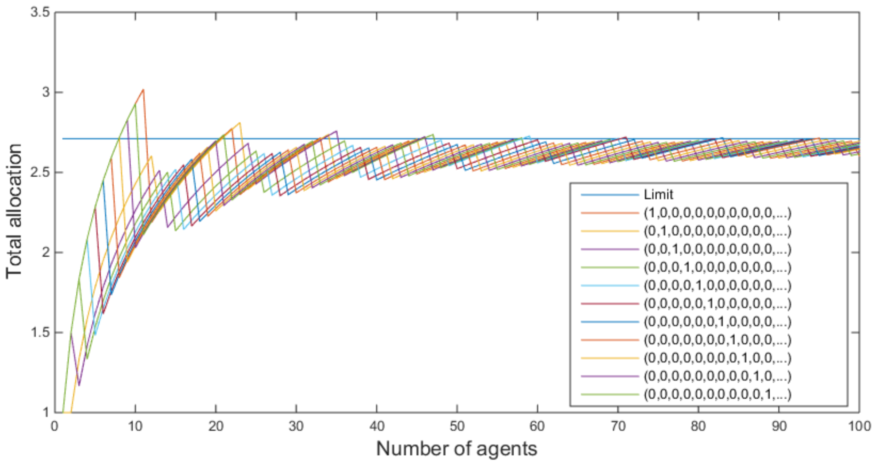

Figure 3: Optimal values of S for different control vectors when $c = 10$.