Fair and Efficient Memory Sharing: Confronting Free Riders

Eric J. Friedman
ICSI and UC Berkeley

Vasilis Gkatzelis
Drexel University

Christos-Alexandros Psomas
Carnegie Mellon University

Scott Shenker
ICSI and UC Berkeley

Abstract

A cache memory unit needs to be shared among n strategic agents. Each agent has different preferences over the files to be brought into memory. The goal is to design a mechanism that elicits these preferences in a truthful manner and outputs a fair and efficient memory allocation. A trivially truthful and fair solution would isolate each agent to a $1/n$ fraction of the memory. However, this could be very inefficient if the agents have similar preferences and, thus, there is room for cooperation. On the other hand, if the agents are not isolated, unless the mechanism is carefully designed, they have incentives to misreport their preferences and *free ride* on the files that others bring into memory. In this paper we explore the power and limitations of truthful mechanisms in this setting. We demonstrate that mechanisms *blocking* agents from accessing parts of the memory can achieve improved efficiency guarantees, despite the inherent inefficiencies of blocking.

1 Introduction

Resource allocation is the foundation of any computer system, and a particularly challenging problem in cloud environments. In shared computer systems, the users have access to the same (relatively small) cache memory unit, which they can use for high-speed access to data files. A naive memory management approach would be to partition this memory into as many blocks as the number of users and to isolate each user to her dedicated block. Traditional memory allocation policies, e.g., LRU (Least Recently Used) or MRU (Most Recently Used), could then be used to decide which files to keep in each block based on the frequency of the corresponding user's requests. This approach offers individual fairness guarantees to each user and users have no motive to misrepresent their preferences. However, this isolation can come at a great cost in terms of efficiency.

Although most of the files may be "private", i.e., only a single user wants each file, there may also exist a "public" file, e.g., a massive dataset that multiple users wish to access. Depending on how frequently each user is expected to access the public file, a much more efficient utilization of the memory may be achieved through sharing. For instance, several user blocks could be used to store as much of the public file as possible, and the users could then access all of that memory without isolation. But, mechanisms that strive for non-trivial efficiency guarantees through memory sharing are susceptible to unfairness and user manipulation. Our goal in this paper is to design novel memory sharing mechanisms that optimize *efficiency* while guaranteeing *fairness* and *truthfulness* (i.e., non-manipulability).

A fundamental notion of fairness, known as *sharing incentives* in systems or *proportionality* in AI, asks that every user should always be at least as happy as she would be if memory isolation was used instead. Many of today's multi-tenant environments, including cache systems for cloud serving and big data storage (e.g., (Redis 2009; Memcached 2003)), optimize efficiency by sharing all of the memory and using LRU or MRU to decide which files to keep in it. But, these policies are based only on the frequency of the requests and not on the identity of the requesting users, so they can inadvertently favor some users at the expense of others, and violate proportionality.

As fairness becomes more relevant in large computing systems (Verma et al. 2015; Ghodsi et al. 2013; 2011), users demand individual quality of service guarantees (QoS). A lot of recent work in AI focuses on building the foundations necessary to design new fair protocols for a variety of relevant settings (e.g., (Kurokawa, Procaccia, and Wang 2018; Fain, Munagala, and Shah 2018; Conitzer, Freeman, and Shah 2017; Kurokawa, Procaccia, and Wang 2016; Parkes, Procaccia, and Shah 2015; Goldman and Procaccia 2015; Dickerson et al. 2014; Friedman, Ghodsi, and Psomas 2014; Cole, Gkatzelis, and Goel 2013a; 2013b; Chen et al. 2013; Maya and Nisan 2012; Gutman and Nisan 2012)).

The other important concern is the manipulability of the mechanism. For instance, users can manipulate some memory allocation policies by artificially inflating the frequency of their requests for some files. (Pu et al. 2016) experimentally demonstrated that strategically making excessive access requests for files can improve a user's average response time for the files that she actually wants. Strategic user behavior has also been observed in multiple other real systems (e.g., (Verma et al. 2015; Ghodsi et al. 2011)). Two aspects of memory allocation that make the design of truthful mechanisms particularly challenging are i) the fact that memory is a *non-rival good*, i.e., multiple users can simultaneously benefit from its contents, and ii) the inability to use monetary payments to appropriately incentivize the users.

The fact that memory is a non-rival good permits a more effective utilization, but it can also lead to *free riding*, a phenomenon which is well documented in the economics literature (Groves and Ledyard 1977; Samuelson 1954). The main cause of free riding is the fact that users can often enjoy the benefits of non-rival goods without contributing to their value, knowing that other users will do so. In Section 2 we provide an example of how this may arise in our setting. Confronting the free riding motives without resorting to memory isolation requires a careful mechanism design process that yields the desired incentives. The long literature on mechanism design has supplied us with several widely applicable tools, but the vast majority of these tools leverages monetary payments. Since the use of such payments would be unreasonable or infeasible in a memory allocation setting, we resort to the alternative approach of *money burning*.

Broadly speaking, a money burning mechanism may intentionally underutilize some resource, effectively "penalizing" some users, and simulating the impact of a monetary payment without the use of money. In our setting, we achieve this effect through the use of *blocking*. A memory allocation mechanism controls not only which files are brought into memory, but also the access rights that each user has on each file. So, for example, a file j can be brought into memory and the system can give some user i_1 full access to j, completely prevent another user i_2 from accessing it, while allowing partial access to some third user i_3.

Recent work in systems has implemented memory sharing solutions attempting to address fairness (Kunjir et al. 2017) or incentive concerns (Yu et al. 2018; Pu et al. 2016). However, (Yu et al. 2018) argue that the system of (Pu et al. 2016) actually fails to prevent free-riding and, as it happens, the system proposed by (Yu et al. 2018) is not truthful either. The authors make the unusual assumption that it is acceptable for a user to lie, as long as this does not hurt the other users. However, basic game-theoretic arguments show that the resulting equilibria of the induced game among the users can actually end up hurting some of them.

Our Results

We study the extent to which we can optimize the *efficiency* of memory sharing mechanisms that are guaranteed to be *truthful* and *fair*. As a crucial tool in creating the right incentives for the users, we consider two types of blocking, which we refer to as *uniform* and *non-uniform*, with the latter being more powerful than the former (see Section 2 for formal definitions).

For instances with two users we show that uniform-blocking mechanisms cannot provide non-trivial efficiency guarantees, but using nonuniform blocking can surpass this obstacle. Specifically, we provide a nonuniform-blocking mechanism that is 83%-efficient, which is close to optimal.

For instances with multiple users we show that uniform-blocking mechanisms fail to achieve any constant approximation of efficiency. However, once again, we provide a nonuniform-blocking mechanism that performs much better; surprisingly, this mechanism can guarantee a $1 - \frac{1}{e} \approx 63\%$ approximation of efficiency for all instances, even with an arbitrary number of users.

2 Model

A unit of cache memory is shared among a set N of n agents who need high-speed access to large data files. We virtually partition this memory into n equal-sized *blocks*, one for each agent. Each file is a sequence of bits and its total size is at least one block, allowing us to treat the files as *divisible items*. Each agent has a favorite "private" file that no other agent is interested in, and there is also a "public" file which corresponds to a massive dataset that multiple agents may wish to access. Our goal is to design a mechanism that elicits each agent's interest in the public file relative to her private one and decide how much of the public file should be brought into the cache, which private files it should replace, and how much access to it should be granted to each agent.

We denote the public file as file 0 and use $j \in \{1, ...n\}$ to index the favorite private file of agent j. A memory allocation $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)$ specifies, for each file $j \in \{0, ..., n\}$, the amount, x_j, of this file that is in memory. For notational convenience we measure these amounts in blocks, i.e., $x_j = 1$ means that the amount of file j in memory is equal to one block (a $1/n$ fraction of the memory) and $x_j \in [0, n]$ for all j. Each agent attaches a value to each file, which is proportional to the frequency with which the agent needs to access this file. Without loss of generality[1], we normalize these values so that every agent's value for a block of her favorite private file is 1. We let $v_i \geq 0$ denote agent i's value for each block of the public file. If $v_i < 1$, then agent i has a higher value for her private file than she has for the public file. We use $\overline{v}_i = \max\{1, v_i\}$ to denote the maximum value of agent i between the public file and her private file.

Free Riding Example. In a setting involving two agents, if each of them used her block of memory for her private file, then they both receive a total value of 1, since they get no value from the other agent's private item. Now, assume that they both value the public item at 0.75 (less than their private item). If each agent used her half of the memory to store a distinct block of the public file instead, then they would each get a value of 0.75 from their own half, as well as an additional 0.75 from the other agent's half, leading to a total value of 1.5 for both; a considerably more efficient outcome. However, if the first agent brings a block of the public file in her own half and the second agent brings her private file instead, then the second agent's value becomes 1.75; she is effectively free-riding by taking advantage of the first agent's contribution without reciprocating.

Blocking Mechanisms

A memory allocation mechanism asks each agent i to report her value for the public file, it collects a bid b_i (possibly different than the true v_i), and it outputs an allocation $\boldsymbol{x}$ as well as memory access rights for the agents. Using these access rights, the mechanism can choose to block agents from parts of the memory. We consider three types of mechanisms, depending on the extent to which they employ blocking. The

[1] The efficiency and fairness notions considered in this paper are scale-independent, avoiding interpersonal comparisons of utility.

no-blocking mechanisms cannot block the agents from any part of the memory. The *uniform-blocking* mechanisms can block the agents from accessing some parts of the memory but, if they choose to block-off something, then *no agent* can access it. This is equivalent to keeping some part of the memory empty, i.e., $\sum_{j \in M} x_j < n$, where $M = \{0, ..., n\}$. Finally, the *nonuniform-blocking* mechanisms can block different agents from different parts of the memory. That is, instead of being able to access all of the x_j amount of file j that is in memory, agent i can access only an $f_{i,j}$ fraction, i.e., $f_{i,j} \leq x_j$.

Without loss of generality, we can restrict our attention to mechanisms that never block the agents from accessing any of the private files in memory,[2] i.e., $f_{i,j} = x_j$ for all $j \in [n]$, so the only relevant information is $f_{i,0}$, which we denote by just f_i for clarity. Therefore, given a vector of reported values $\boldsymbol{b} = (b_1, \ldots, b_n)$ a mechanism outputs a vector $\boldsymbol{x}(\boldsymbol{b}) = (x_0(\boldsymbol{b}), x_1(\boldsymbol{b}) \ldots, x_n(\boldsymbol{b}))$, along with a vector $\boldsymbol{f}(\boldsymbol{b}) = (f_1(\boldsymbol{b}), \ldots, f_n(\boldsymbol{b}))$ of values corresponding to the access to the public item allowed to each agent $i \in N$. A mechanism is *feasible* if $\sum_{j \in M} x_j(\boldsymbol{b}) \leq n$ and $\max_{i \in N}\{f_i(\boldsymbol{b})\} \leq x_0(\boldsymbol{b})$.

Preferences and Incentives

Each agent's preferences over the possible outcomes of a mechanism are defined via a linear utility function that depends on the agent's valuation. Formally, the utility of agent i for allocation $\boldsymbol{x}$, given blocking $\boldsymbol{f}$, is $u_i(\boldsymbol{x}, \boldsymbol{f}) = x_i + f_i v_i$. We also use $u_i(\boldsymbol{b})$ to denote the utility of agent i in the outcome that will arise if $\boldsymbol{b}$ is the set of bids reported. The linearity of the utility, which we discuss further in Section 6, is standard (Yu et al. 2018; Pu et al. 2016), and it corresponds to the delay that the agent avoids due to the blocks of the file that are already in memory.

A mechanism is *truthful* if no agent ever has an incentive to lie, i.e., if for every agent i, every possible reported bid $b_i \in \mathbb{R}$, every $b_{-i} \in \mathbb{R}^{(n-1)}$ (every possible report of all other agents), and every true valuation v_i, the mechanism satisfies $u_i(v_i, b_{-i}) \geq u_i(b_i, b_{-i})$. In other words, every agent's utility in the outcome that will arise if she is truthful is never less than her utility after any possible lie. Since our mechanisms are truthful, we often replace b_i with v_i.

Fairness and Efficiency

A mechanism is *anonymous* if its outcome does not depend on the "names" of the agents or the items and, hence, no permutation of these names would affect the outcome. A mechanism is *proportional* if the utility of agent i in every outcome of the mechanism is at least $\overline{v}_i$. These two properties ensure that each one of the participating agents receives individual QoS guarantees. That is, an agent cannot be discriminated against based on irrelevant information, and is guaranteed to receive her "fair-share" of the memory, i.e., the utility that she would receive if we isolated each agent to a $1/n$ fraction of the memory. We note that, although we use anonymity as an elementary notion of "symmetry" for our impossibility results, the mechanisms we provide in this paper actually satisfy much stronger symmetry properties.

A mechanism that yields a utility profile $\boldsymbol{u} = (u_i)_{i \in N}$ for the agents is Pareto efficient if there exists no other feasible utility profile $(u'_i)_{i \in N}$ such that, $u'_i \geq u_i$ for very $i \in N$ and $u'_k > u_k$ for some $k \in N$. In order to measure the efficiency of our mechanisms we use a common notion of approximate Pareto efficiency, initially defined in (Ruhe and Fruhwirth 1990). An outcome with utility profile $\boldsymbol{u}$ is ρ-Pareto efficient if there is no feasible utility profile in which, compared to $\boldsymbol{u}$, every agent is more than $1/\rho$ times better off. We also prove a negative result for a much more demanding variant of this measure: we say that a mechanism with utility profile $\boldsymbol{u}$ is ρ-*strongly* Pareto efficient if there is no feasible utility profile in which, compared to $\boldsymbol{u}$, no agent is worse off, and at least one agent is more than $1/\rho$ times better off.

An Example. Figure 1 displays a small 2-agent instance, where the public file is valued at $v_1 = 5/6$ and $v_2 = 2/3$. Then, $\overline{v}_1 = \overline{v}_2 = 1$. There are six possible (integral) allocations, which are represented as points in the utility profile space of the two agents (the points of the form (u_1, u_2)). E.g., the point $(2, 0)$ corresponds to filling all the cache with agent 1's private item, and $(5/6, 5/3)$ is from one block of the public item and one block of agent 2's private item. Note that no-blocking mechanisms are restricted to generating utility profiles that are convex combinations of these points, whereas blocking mechanisms can output any utility profile within the Pareto frontier. The point $(\overline{v}_1, \overline{v}_2)$ shows the minimum utility requirement that the proportionality constraint imposes. The dashed line in the figure defines the approximate Pareto region for an approximation factor of $\rho = 80\%$, i.e., all the feasible utility profiles that are at least 80%-Pareto efficient. Figure 1 also shows what the approximate strong Pareto region looks like. The shaded blue region corresponds to all outcomes where the utility of agent 1 (the x-axis agent) is a 80% of the best it could be without hurting agent 2. Similarly, the shaded red region serves the analogous purpose for agent 2, and the intersection of these two regions is the approximate strong Pareto region.

3 No-Cooperation Mechanism

Before delving into blocking mechanisms, and to get a sense of the performance guarantees that we can achieve with a very simple no-blocking mechanism, we first briefly define the *No-Cooperation* (No-Coop) mechanism, which we can use as a benchmark for the subsequent results. This mechanism virtually partitions the memory into n blocks, and for each agent i it uses all of her block to store that agent's favorite item, i.e., either her private item if $v_i < 1$ or, otherwise, the public one. As a result, if k agents prefer the public item, then $x_0 = k$. Note that this mechanism does not use any blocking, so all of the agents can access the portion of the public item that is in memory, but No-Coop does not attempt to identify room for cooperation among the agents

[2] This is because blocking agent i from some other agent's private item is pointless (i has no value for it anyway) and blocking i from her own private file means that we can instead replace that fraction of her item with a fraction of the public item and block everyone from accessing it, without affecting the outcome.

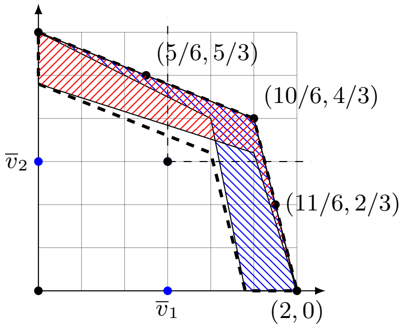

Figure 1: The dashed line defines the 80% approximate Pareto region. The 80% approximate strong Pareto region is the intersection of the two shaded regions.

that could yield higher efficiency outcomes. The following lemma, whose simple proof can be found in the full version of this paper, shows that No-Coop gives an approximation of exactly $1/n$.

Lemma 3.1. *The No-Coop mechanism is anonymous, proportional, truthful, and its approximation factor is exactly $1/n$ for both the Pareto and strong Pareto measures.*

4 Memory Sharing with Two Agents

In this section, we study two-agent instances. Due to space constraints, the proofs are deferred to the full version of this paper.

Uniform Blocking

We begin by restricting ourselves to uniform blocking mechanisms, i.e., mechanisms that either block none of the agents from accessing some part of the memory or all of the agents. We first show that no such mechanism can outperform the No-Coop mechanism in terms of efficiency.

Theorem 4.1. *There exists no anonymous, proportional, and truthful uniform blocking mechanism with a $(0.5 + \epsilon)$ Pareto approximation for a constant $\epsilon > 0$.*

Proof Sketch. Let $I = (v_1, v_2)$ denote an instance I where the value of the two agents is v_1 and v_2 respectively, and let δ be some constant such that $\frac{\delta}{1-\delta} < \epsilon$. Then consider the instances $I_1 = (1+\delta, 0)$, $I_2 = (1+\delta, 1-\delta)$, $I_3 = (0, 1+\delta)$, $I_4 = (1-\delta, 1+\delta)$, and $I_5 = (1-\delta, 1-\delta)$.

Using proportionality we can conclude that the outcome of I_1 needs to be $(1, 0, 1)$; this outcome and the truthfulness of agent 2 imply the outcome of I_2 is also $(1, 0, 1)$. Using a symmetric sequence of arguments, one can conclude that, facing problem instances I_3 and I_4, the mechanism would need to choose the allocation $(1, 1, 0)$.

Finally, using the truthfulness of agent 1 between instances I_2 and I_5 and the truthfulness of agent 2 between instances I_4 and I_5, we get a set of constraints in the allocation of I_5 which contradict the $(0.5 + \epsilon)$ Pareto approximation claim. See the full version of this paper for more details. □

Nonuniform Blocking: Lower Bounds

Having observed the limitations of uniform blocking mechanisms in (approximately) maximizing Pareto efficiency, we now turn to nonuniform blocking mechanisms. The main advantage of these mechanisms is the ability to penalize one agent without necessarily also penalizing the other. However, as we show in this section, despite the significant additional power that nonuniform-blocking mechanisms possess, they too fail to combine proportionality with 100% Pareto efficiency. In fact, no such mechanism can achieve more than a 91% approximation of Pareto.

Note that enabling nonuniform blocking enriches the design space for mechanisms and, as a result, proving the impossibility of an approximation factor becomes harder. In particular, the combinatorial arguments that we used in the proof of Theorem 4.1 become significantly more complicated. To overcome this obstacle, rather than using combinatorial arguments, we use duality theory.

Theorem 4.2. *There exists no proportional and truthful mechanism that can achieve a 91% Pareto approximation.*

Proof Sketch. We first attempt to write a linear program whose solution corresponds to the mechanism with the optimal approximation factor. After discretizing the valuation space of the agents and defining a variable for each f_i value of the mechanism, one can express (as a linear function) the utility $u_i\left(v_i, b_i, b_{-i}\right)$ of an agent i when the other agent reports valuation b_{-i}, agent i reports b_i, and her true valuation is v_i. Given these variables, $u_i\left(v_i, b_i, b_{-i}\right)$, we express truthfulness, proportionality, and feasibility as linear constraints.

The objective is to maximize ρ, the Pareto approximation factor, such that the outcome of the mechanism is guaranteed to be within the ρ-approximate Pareto region. But, we run into the following issue: *the intersection of the ρ-approximate Pareto region and the proportionality region in utility space is not always convex* (see Figure 2). This means that, without a creative change of variables, this region cannot be expressed using linear constraints. We therefore identify a set of reported value profiles of the two agents for which this region is convex (e.g., see Figure 3), and we drop the Pareto constraints for all other value profiles. The resulting linear program solves a relaxed, discretized, version of the initial mechanism design problem. However, using this formulation, we can now construct a dual program. A feasible solution to this dual program does, in fact, imply an impossibility result for nonuniform-blocking mechanisms. Using this technique, we construct a feasible (and quite complicated) solution for the dual which yields the inapproximability bound of Theorem 4.2; the proof as well as the primal and dual formulations can be found in the full version of this paper. □

Nonuniform-Blocking: Upper Bounds

We now provide a mechanism that achieves a 83% approximation of Pareto. Although we cannot use payments, we provide an interpretation of any instance in our setting as one where each agent has a "budget", corresponding to the amount of her private item in memory, that she can "spend"

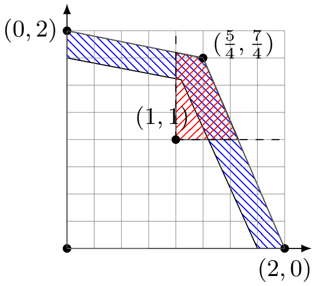

Figure 2: The intersection of the approximate Pareto region and the proportionality region is not convex.

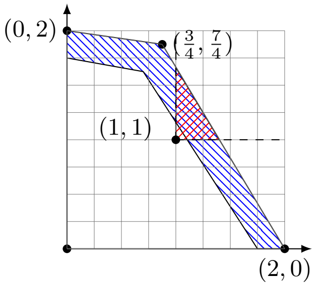

Figure 3: The intersection of the approximate Pareto region and the proportionality region is convex.

in order to bring more of the public item in memory. The memory allocation mechanism can then be interpreted as an auction that defines the "price" that each agent needs to pay for accessing the public item. This mapping is very useful, as it enables us to use well-known results from mechanism design with payments in designing our mechanism.

Note that, if $v_1 = v_2 = 0$, i.e., both agents have no value for the public item, then any proportional mechanism would have to output the allocation $(0, 1, 1)$, i.e., one block of each private item and none of the public item, otherwise one of the agents would get a value less than 1. We interpret this outcome as both agents having a budget of 1, which they are not willing to spend on the public item. Then, in instances where the value of v_i is higher, the mechanism may give each agent i the opportunity to spend some of this budget in order to increase x_0. Using this interpretation, when the two agents' reported v_i values are high, i.e., when there is room for cooperation among them, we wish to design a mechanism that provides them with the opportunity to "buy-into" the public item, at the appropriate price. As a result, a good mechanism needs to address the tension between efficiency, truthfulness, and proportionality, by carefully deciding the rate at which an agent can trade her private item blocks for a fraction of the public item.

The seminal work of Myerson (1981) dictates what the structure of a truthful mechanism needs to be in a setting with actual monetary payments. It restricts the allocation rule to being monotone and it also defines the form that the pricing rule should obey. Given the interpretation of the private item as a budget, we derive an analog of Myerson's characterization, which (1) restricts the function $x_0(v_1, v_2)$ implied by the mechanism to be (weakly) monotone, and (2) dictates the exact rate at which each agent i can exchange her private item for the public item as v_i changes, i.e., exactly how fast x_i needs to drop as a function of an x_0 increase. This allows us to only worry about choosing $x_0(v_1, v_2)$.

Lemma 4.3. *A mechanism is truthful for agent i if and only if for any fixed value of v_{-i} reported by the other agent, the following two properties are satisfied:*

1. *$x_0(v_i, v_{-i})$ is a monotone non-decreasing function of v_i.*
2. *For some function $c_i(.)$ that depends only on v_{-i},*

$$x_i(v_i, v_{-i}) = -v_i x_0(v_i, v_{-i}) + v_i \int_0^{v_i} x_0(z, v_{-i}) dz + x_i(0, v_{-i}) + c_i(v_{-i}).$$

We use this characterization to design the following mechanism, whose presentation will be in two parts. We first address what the mechanism does when neither of the agents prefers the public item to her private item, i.e., when $v_1 \le 1$ and $v_2 \le 1$. In this set of instances, our mechanism does not take advantage of its nonuniform blocking power and, instead, uses only uniform blocking. We then move on to describe the mechanism for the remaining instances, in which nonuniform blocking plays a crucial role.

Instances with $v_1 \le 1$ and $v_2 \le 1$. The mechanism that we propose, the *Buy-In* mechanism, decides how much of the public item to bring into memory as a function of the product $v_1 v_2$. If this product is at least $1/4$, then the amount of the public item that is brought into memory grows logarithmically as a function of $v_1 v_2$. It starts from a value of $x_0 = 0$ when $v_1 v_2 = 1/4$ and it goes up to $\frac{4}{3} \ln 4 \approx 1.84$ when $v_1 = v_2 = 1$. The product $v_1 v_2$ captures, in a very symmetric fashion, the extent to which there is room for cooperation. Our mechanism allows the agents to take advantage of that, as the product increases. On the other hand, if the product is less than $1/4$, the mechanism lets the agents keep their budgets unspent.

Given a function for $x_0(v_1, v_2)$, the form of the $x_1(v_1, v_2)$ and $x_2(v_1, v_2)$ is implied, up to a constant transformation, by Lemma 4.3, in order to ensure the truthfulness. Since $x_0(v_1, v_2)$ grows logarithmically with v_1, Lemma 4.3 implies that $x_1(v_1, v_2)$ should decrease linearly. However, the exact form of $x_1(v_1, v_2)$ and $x_2(v_1, v_2)$ that we have chosen below also ensures that the memory is never over-allocated, while providing good Pareto guarantees as well.

Algorithm 2: *

```
Buy-In mechanism for $v_1, v_2 \le 1$. **if** $v_1 v_2 \ge 1/4$ **then**
    $x_0(v_1, v_2) = \frac{4}{3} \ln(4 v_1 v_2)$
    $x_1(v_1, v_2) = \frac{3v_2 - 4v_1v_2 + 1}{3v_2}$
    $x_2(v_1, v_2) = \frac{3v_1 - 4v_1v_2 + 1}{3v_1}$
**else**
    $x_0(v_1, v_2) = 0$, $x_1(v_1, v_2) = 1$, $x_2(v_1, v_2) = 1$
```

Instances with $\max\{v_1, v_2\} > 1$. Since we have not used nonuniform blocking so far, Theorem 4.1 implies that, to achieve more than 50% approximation, we *must* use nonuniform blocking in the remaining instances. A closer look at the proof of Theorem 4.1 reveals that a crucial truthfulness constraint relates to a deviation from an instance with $v_1 > 1$ and $v_2 \leq 1$ to one with both $v_1 \leq 1$ and $v_2 \leq 1$. In other words, an agent who prefers the public item pretending to prefer her private one. For uniform-blocking mechanisms, this truthfulness constraint restricts the efficiency that can be guaranteed at the initial instance; nonuniform blocking alleviates these type of incentive issues. In particular, as the value of v_i increases, the moment it exceeds 1, we exchange all of agent i's private item with an equal fraction of the public item, *but block the other agent from accessing it*. Furthermore, in terms of the allocation of the other agent, whether i reports $v_i = 1$ or any $v_i > 1$ makes no difference. In Algorithm 3, note that the value of $x_0(1, v_2)$ when $v_2 \leq 1$ is borrowed from the Buy-In mechanism for both values are less or equal to 1, i.e., $x_0(1, v_2) = \frac{4}{3}\ln(4v_2)$, and similarly for all other allocations of $(1, v_2)$ or $(v_1, 1)$ when v_1 or $v_2 \leq 1$.

Algorithm 3: Buy-In mechanism for $\max\{v_1, v_2\} > 1$.

```
if $v_1 > 1$ and $v_2 \leq 1$ then
    $x_0(v_1, v_2) = x_0(1, v_2) + x_1(1, v_2)$
    $x_1(v_1, v_2) = 0$, and $f_1(v_1, v_2) = x_0(v_1, v_2)$
    $x_2(v_1, v_2) = x_2(1, v_2)$, and $f_2(v_1, v_2) = x_0(1, v_2)$
else if $v_2 > 1$ and $v_1 \leq 1$ then
    $x_0(v_1, v_2) = x_0(v_1, 1) + x_2(v_1, 1)$
    $x_1(v_1, v_2) = x_1(v_1, 1)$, and $f_1(v_1, v_2) = x_0(v_1, 1)$
    $x_2(v_1, v_2) = 0$, and $f_2(v_1, v_2) = x_0(v_1, v_2)$
else if $v_1 > 1$ and $v_2 > 1$ then
    $x_1(v_1, v_2) = x_2(v_1, v_2) = 0$, and
        $f_1(v_1, v_2) = f_2(v_1, v_2) = x_0(v_1, v_2) = \frac{4}{3}\ln(4)$
```

The non-trivial proof of the following result (see the full version of this paper) carefully analyzes the mechanism in terms of the tricky approximation measure based on Pareto efficiency.

Theorem 4.4. *The Buy-In mechanism is anonymous, proportional, truthful, and it guarantees a* 83% *approximation of Pareto.*

5 Instances with Multiple Agents

This section studies the power and limitations of anonymous, proportional and truthful memory sharing mechanisms for instances with multiple agents. Just like in two-agent instances, we show that uniform blocking mechanisms hit a significant road-block. The crucial information, which implies how much of the public item should be placed in memory, is how much the agents value the public item compared to their private item. If no agent values the public item then, clearly, none of it should be placed in memory. But, if enough agents have considerable value for this item, then they may benefit by sharing the public item using the memory that their private items would otherwise occupy. In this setting, even the task of figuring out the optimal outcome is not straightforward, let alone resolving the issue of agents misreporting their preferences.

We first prove that uniform blocking mechanisms fail to achieve *any* constant approximation of Pareto. We complement this result by providing a nonuniform-blocking mechanism that guarantees a $1 - \frac{1}{e} \approx 63\%$ approximation of Pareto. The proofs missing from this section can be found in the full version of this paper.

Uniform Blocking

For the special case of two-agent instances where the agents prefer their private item, our Buy-In mechanism achieved a good approximation using only uniform blocking (Section 4). Surprisingly, we prove that all attempts to extend this result to n-agent instances are hopeless: even if all the agents prefer their private items, the ideas used by the Buy-In and, in fact, all uniform blocking mechanisms fail to achieve a constant approximation. Intuitively, when many agents are present, it becomes much more tempting to free-ride on the public item.

Theorem 5.1. *There is no anonymous, proportional and truthful uniform blocking mechanism that can guarantee a constant approximation of Pareto for any number of agents.*

Proof Sketch. Since private items have value 1, if a sizeable group of agents report a value of $1/\sqrt{n}$ for the public item, any mechanism with good Pareto guarantees should bring in memory a lot of the public item. If a sequence of agents starts changing their valuation from $1/\sqrt{n}$ to 0, the proportionality property dictates that they should receive a block of their private file and Lemma 4.3 dictates that the deviating agents should "pay" something in exchange for any extra amount of this private file. The "payment" is the allocation of the public item. Since nonuniform blocking is not allowed, that payment has to affect all agents. This tension allows us to prove this very strong negative result. □

To gain a better understanding of the effect that proportionality plays in the lower bound of Theorem 5.1, we also provide a lower bound that does not leverage this property. The following theorem shows that, even if we were willing to relax the proportionality constraint, no truthful and anonymous mechanism can combine an $\omega(1/\sqrt{n})$ approximation of strong Pareto with a guarantee that every agent i gets a value of $\omega(\overline{v}_i/\sqrt{n})$. The proof uses similar ideas with the previous proof, but a longer sequence of deviations is required for the argument to go through.

Theorem 5.2. *No truthful and anonymous uniform blocking mechanism can simultaneously guarantee an* $\omega(1/\sqrt{n})$ *approximation of strong Pareto and an* $\omega(1/\sqrt{n})$ *approximation of proportionality.*

Nonuniform Blocking

Our final result is the *Opt-Out* mechanism, an anonymous, proportional and truthful mechanism for instances with multiple agents that uses nonuniform blocking and guarantees a $1 - \frac{1}{e} \approx 63\%$ approximation of Pareto. This mechanism begins with an allocation that places n blocks of the public item in memory. If there exists some agent i whose value for a block of her private item is greater than her value for n blocks of the public item, then a block of the public file is replaced by a block of file i, i.e., $x_i \leftarrow 1$ and $x_0 \leftarrow x_0 - 1$. This agent i is then blocked from accessing any remaining fractions of the public item ($f_i = 0$). Among the remaining agents, if there exists some other agent whose value for a block with her private item i is greater than her value for the remaining $n-1$ blocks of the public file, the same process is repeated, and this continues until either all agents switch to their private files or no other agent prefers that switch.

Algorithm 4: The Opt-Out mechanism.

```
Let $x_0 = n$ and $S = \{1, 2, \ldots n\}$.
while there exists an agent $i$ such that $v_i x_0 \leq 1$ do
    Remove $i$ from $S$
    Set $x_i \leftarrow 1$ and $x_0 \leftarrow x_0 - 1$
$f_i = 0$ for all $i \notin S$.
```

Theorem 5.3. *The Opt-Out mechanism is anonymous, proportional and truthful, and achieves a* $1 - \frac{1}{e} \approx 63\%$ *approximation of Pareto.*

Proof Sketch. Proving that Opt-Out is anonymous, proportional, and truthful is relatively straightforward. On the other hand, the proof of the Pareto approximation is quite involved. We start by identifying the worst case instances for Opt-Out. Specifically, we show that among all n-agent instances, the instances where agent i reports $v_i = 1/(n - i + 1)$ (we call these "harmonic" instances) are the ones where the approximation factor of Opt-Out is minimized. Notice that in a harmonic n-agent instance, all agents choose their private file instead of participating in sharing the public file, which we refer to as having *opted-out*. In order to show that harmonic instances are the worst case instances, we show that given a worst case instance I where $q < n$ agents opt-out, we can construct an instance I' where an additional agent opts-out and the Pareto approximation in I' is no better than the one in I. Therefore, we can focus on instances where all agents opt-out; among all such instances barely opting out is the worst case.

Having shown that harmonic instances are worst case instances for Opt-Out, in order to conclude the proof of Theorem 5.3 we need to also prove that the optimal allocation for an n-agent harmonic instance is not more than $\frac{e}{e-1}$ times better than Opt-Out. Our approach works as follows. First, we constrain the optimal solution such that exactly k out of n agents get a non-zero amount of their private item. Subject to this constraint, we can characterize the optimal solution and pinpoint the agent who improves the most by switching from the allocation of Opt-Out. Her improvement is a function $h(k, n)$ that we give a closed form for. We proceed to optimize $h(k, n)$. We show that the gap in performance between Opt-Out and the optimal allocation is maximized for n going to infinity, and some $k^* \in \Theta(n)$. The approximation ratio follows. More details are relegated to the full version of this paper. □

6 Discussion

We conclude with a brief discussion on some central aspects of the memory allocation setting. We first briefly discuss why linear valuations are very-well justified in memory allocation. Then, we address how bidding and manipulation may actually take place in a real system where valuations are not necessarily reported by the users in the form of a bid. Finally, we also provide some additional context regarding the fairness constraint that we enforced.

Linearity of Valuations. First, this assumption is quite realistic in memory sharing: in a real system an agent's utility is proportional to the time the agent saves due to the cache. Each agent requests access to a file and needs to retrieve all of its memory blocks; accessing a memory block from the main memory takes time t_1, while accessing a block from the cache takes much less time t_2. Every time an agent finds a requested block in cache implies $t_1 - t_2$ time saved. Item valuations are proportional to their request frequency, hence, each item's contribution to an agent's utility is linear in the number of blocks of the item in cache and the valuation of the agent for the item. Equivalently, an agent's utility is proportional to its expected cache hit-rate (also see (Yu et al. 2018; Pu et al. 2016), who make the same assumption). More importantly, the allocation outcome can be interpreted as time-sharing (the fractions correspond to how often a file is in memory) or a randomized choice of outcomes, in which case the linearity of expectation provides an alternative justification.

Direct Revelation Mechanisms. For simplicity's sake, we present our model as if the agents actually send a message to the mechanism, reporting their valuations. In reality, rather than eliciting bids, the system can just observe the agents' requests and regularly infer their valuations based on these requests. Due to locality of reference, memory access patterns are good predictors of future requests so there is no need for the agents to actually communicate their values. Analyzing direct revelation mechanisms, which are subsequently implemented as dynamic systems is very common (e.g., (Ghodsi et al. 2011; 2013)).

Manipulation. Even if the mechanism observes the *actual* requests of the agents, the agents can still manipulate the valuations that the mechanism infers. Specifically, they can generate excessive spurious requests for an item that they do not really need, thus increasing its inferred valuation. At first glance it seems like an agent can hence only increase and not reduce a valuation, but this is not the case! Note that both proportionality and (approximate) Pareto efficiency are scale-independent measures (they are unaffected by scaling all of an agent's valuations by some constant). Hence, an agent can decrease its (relative) valuation for an item by simply generating excessive requests for all other items.

Proportionality Constraint. Finally, the proportionality constraint that we enforce on our mechanisms is one of the most fundamental and widely used measures in fair division, and it also known as "sharing incentives", "stand-alone", or "isolation guarantee" in systems. By default, memory is divided into equal pieces, isolating each agent i, but guaranteeing them a value of $\overline{v}_i$. Proportionality requires that switching from this naive solution to another outcome should be a Pareto improvement: no participant should be worse-off than this "outside option" that the isolation offers. This is an analog of what is known as "individual rationality" in mechanism design: no agent should lose value for participating.

7 Acknowledgments

This work is supported by the National Science Foundation, under grants CCF-1755955, CCF-1216073, CNS-1161813, and CNS-1704941.

References

Chen, Y.; Lai, J. K.; Parkes, D. C.; and Procaccia, A. D. 2013. Truth, justice, and cake cutting. *Games and Economic Behavior* 77(1):284–297.

Cole, R.; Gkatzelis, V.; and Goel, G. 2013a. Mechanism design for fair division: allocating divisible items without payments. In *ACM Conference on Electronic Commerce, EC 2013*, 251–268.

Cole, R.; Gkatzelis, V.; and Goel, G. 2013b. Positive results for mechanism design without money. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, 1165–1166.

Conitzer, V.; Freeman, R.; and Shah, N. 2017. Fair public decision making. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, 629–646.

Dickerson, J. P.; Goldman, J. R.; Karp, J.; Procaccia, A. D.; and Sandholm, T. 2014. The computational rise and fall of fairness. In *AAAI*, volume 14, 1405–1411.

Fain, B.; Munagala, K.; and Shah, N. 2018. Fair allocation of indivisible public goods. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, EC '18, 575–592.

Friedman, E.; Ghodsi, A.; and Psomas, C.-A. 2014. Strategyproof allocation of discrete jobs on multiple machines. In *Proceedings of the fifteenth ACM conference on Economics and computation*, 529–546. ACM.

Ghodsi, A.; Zaharia, M.; Hindman, B.; Konwinski, A.; Shenker, S.; and Stoica, I. 2011. Dominant resource fairness: Fair allocation of multiple resource types. In *NSDI*, volume 11, 24–24.

Ghodsi, A.; Zaharia, M.; Shenker, S.; and Stoica, I. 2013. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *Proceedings of the 8th ACM European Conference on Computer Systems*, 365–378. ACM.

Goldman, J., and Procaccia, A. D. 2015. Spliddit: Unleashing fair division algorithms. *ACM SIGecom Exchanges* 13(2):41–46.

Groves, T., and Ledyard, J. 1977. Optimal allocation of public goods: A solution to the "free rider" problem. *Econometrica: Journal of the Econometric Society* 783–809.

Gutman, A., and Nisan, N. 2012. Fair allocation without trade. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 719–728. International Foundation for Autonomous Agents and Multiagent Systems.

Kunjir, M.; Fain, B.; Munagala, K.; and Babu, S. 2017. ROBUS: fair cache allocation for data-parallel workloads. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD*, 219–234.

Kurokawa, D.; Procaccia, A. D.; and Wang, J. 2016. When can the maximin share guarantee be guaranteed? In *AAAI*, volume 16, 523–529.

Kurokawa, D.; Procaccia, A. D.; and Wang, J. 2018. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)* 65(2):8.

Maya, A., and Nisan, N. 2012. Incentive compatible two player cake cutting. In *International Workshop on Internet and Network Economics*, 170–183. Springer.

Memcached. 2003. Memcached, a distributed memory object caching system. https://memcached.org/.

Myerson, R. B. 1981. Optimal auction design. *Mathematics of operations research* 6(1):58–73.

Parkes, D. C.; Procaccia, A. D.; and Shah, N. 2015. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. *ACM Transactions on Economics and Computation* 3(1):3.

Pu, Q.; Li, H.; Zaharia, M.; Ghodsi, A.; and Stoica, I. 2016. Fairride: Near-optimal, fair cache sharing. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, 393–406. USENIX Association.

Redis. 2009. Redis. https://redis.io/.

Ruhe, G., and Fruhwirth, B. 1990. ϵ-optimality for bicriteria programs and its application to minimum cost flows. *Computing* 44(1):21–34.

Samuelson, P. A. 1954. The pure theory of public expenditure. *The review of economics and statistics* 387–389.

Verma, A.; Pedrosa, L.; Korupolu, M.; Oppenheimer, D.; Tune, E.; and Wilkes, J. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems*, 18. ACM.

Yu, Y.; Wang, W.; Zhang, J.; Weng, Q.; and Letaief, K. B. 2018. Opus: Fair and efficient cache sharing for in-memory data analytics. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 154–164.