Fair Shares: Strategy-Proofness in Single-Resource Allocation for Dynamic Demands

Vaibhav Gupta
vaibhav7@illinois.edu

Ahsan Gilani
ahsang2@illinois.edu

Abstract

While a breadth of existing work studies resource allocation mechanisms within multiple contexts, published methods often evaluate allocation performance in non-adversarial settings that do not distinguish between honest and dishonest user demands. We consider the problem of evaluating strategy-proofness in single-resource allocation mechanisms within the context of dynamic user demands under real-world workloads. Specifically, we implement three distinct mechanisms based on prior work — a credit system that balances resources between borrowers and donors, a market-driven system that auctions resources, and a barter system that exchanges resources based on tickets and claims. We show that while these mechanisms may provide guarantees regarding Pareto efficiency, welfare, and fairness, none of them incentivize users to remain honest regarding their demand.

1 Introduction

Resource allocation is a fundamental concept in computer systems, involving a strategic distribution of limited resources among individual competing demands to achieve optimal performance across the system. Allocation is utilized in clusters, clouds, networks, etc and often prioritizes behaviors such as fairness, cost-efficiency, and data locality among others depending on the application. These systems must often provide a diverse set of resources (IaaS, PaaS, SaaS) to a diverse pool of clients with varying demands and usages [19, 23].

However, allocation depends on user behavior and is often susceptible to exploitation since users are incentivized to misrepresent their demand if it produces a more favorable allocation. We refer to *strategy-proofness* as the incentive for users to honestly represent their demand since this should produce the best possible allocation regardless of the demands of others. This is often desirable since it ensures resources are fully utilized while tending to reduce disparity and improve fairness [17]. As such, in order to provide the best possible service to each user, there is a need to understand strategy-proofness in resource allocation mechanisms within heterogeneous environments [10]. Although traditional single-resource approaches such as strict partitioning [1, 21] and max-min fairness [9, 14, 15] have been thoroughly studied, other complex mechanisms often claim significant performance gains under certain conditions without sufficient empirical evaluation in dynamic, adversarial settings. For instance, an approach may achieve fairness and Pareto efficiency in an environment with all honest users or static demands but fail to do so when there are dishonest users or demands are dynamic.

We introduce a standardized interface for evaluating single-resource allocation algorithms in multiple settings and present a comprehensive comparison of three such unique mechanisms. We implement mechanisms from (1) Karma, a credit-based system that considers past allocations for coordinating resources between donors and borrowers [20], (2) Ginseng, a market-driven system that incentivizes guests to bid their true valuation in resource auctions [2], and (3) Sharp, a barter-based system for exchanging resources based on probabilistic claims through tickets and claims [5].

Our evaluation considers desired notions of both short-term and long-term fairness, Pareto efficiency (not possible to increase a user's allocation without decreasing that of another), and strategy-proofness (robustness to unknown, selfish users who may collude and lie about their demands). Although some mechanisms were designed to address these properties to various degrees, we contribute the following:

- Standardized implementation of three single-resource allocation mechanisms and multiple metrics for evaluating Pareto efficiency, welfare, and fairness that is available here.
- Introduce a new method of computing user welfare for auction-based systems that considers a user's payment with respect to its valuation of its allocation.
- Implement an algorithm based off Sharp that performs a separate allocation for tickets that can be redeemed during resource allocation.
- Perform a comprehensive algorithmic analysis of these mechanisms under a real-world workload and show that none of them incentivize users to remain honest regarding their demands.

2 Related Work

We consider several systems that target single-resource allocations and are not intended for multi-resource allocations. Although these systems can be applied to any resource, we generally focus on memory within the scope of this paper.

Credit-based resource allocation. Karma is a credit-based resource allocation mechanism for dynamic user demands. Each user is guaranteed an $0 \leq \alpha \leq 1$ fraction of its fair share f, which they donate to a pool when its demand is less than its guaranteed share $g = \alpha * f$. This pool consists of resource slices that are either shared ($s = f - g$ per user) or donated and may be borrowed by users when its demand is higher than its guaranteed share. In each quantum, users can earn up to s credits for contributing its fair share to shared slices and one credit per donated slice, or lose one credit per borrowed slice.

Allocation is trivial when supply is equal to borrower demand — all slices in the pool can be allocated to borrowers and credits are updated accordingly. When supply is greater than the borrower demand, Karma prioritizes allocating donated slices from users with the lowest number of credits before shared slices so donors can earn credits while balancing credits between users. When supply is less than borrower demand, Karma prioritizes allocating slices to users with the highest number of credits to balance resources between users.

Although fairness decreases as α increases, Karma is fair long-term, Pareto efficient, and strategy-proof, incentivizing users to be truthful about their demands since both system-wide performance and user welfare improve as the number of conformant users increase [20].

Market-driven resource allocation. Ginseng is a market-driven system for efficiently allocating cloud memory to selfish clients. In particular, we focus on the novel Memory Progressive Second Price (MPSP) auction, where each guest has some permanent base memory and can bid for additional memory each auction round that they rent until the next round. Bids consist of a price per memory-time unit (unit-price) and a list of desired memory ranges. MPSP then uses a constrained divisible-good allocation algorithm by sorting guests by unit-price and current holdings to allocate each guest its maximal desired resource quantity. If some guest g would receive a quantity inside some forbidden range R (i.e. under its minimum request), the allocation is invalid and MPSP attempts to maximize social welfare by considering two separate constrained allocations where g receives quantity large enough to cover R

or small enough to not receive any quantity within R. This is compared against a valid allocation to determine the optimal allocation, with payments following the exclusion compensation principle where a winning guest pays for the damage its bid caused to the benefit of other guests.

Although guests implement an online strategy that may involve collusion, MPSP claims to always converge to steady state where the allocation optimizes social welfare, incentivizing guests to always bid their true valuation for a desired quantity [2, 6].

Barter-based resource allocation. Sharp is a framework for secure distributed resource management through resource peering, where partners may trade resources or contribute to a federated environment. Sites delegate agents to control possibly overlapping sections of the global resource pool, which handle exchanges between service managers and site authorities. Agents may function as brokers which exchange tickets between sites without contributing or requesting resources. Service managers can obtain resource claims as a ticket from an agent, which can be rejected or honored through a lease issued by a site authority. As such, tickets represent soft claims for resource ownership while leases represent hard claims until their expiration, maintaining local autonomy by allowing sites to consider current conditions when redeeming tickets. Tickets may be delegated to other principals but represent probabilistic claims as agents can oversubscribe its resources to maximize resource utilization based on ticket redemption rate. However, ticket delegation is accountable and can be efficiently checked for conflicts to identify the agent responsible and resolve conflicts accordingly [5].

Although Sharp emphasizes secure resource delegation and failure-tolerance in a decentralized manner, we look at a centralized approach that relaxes several of these conditions to focus primarily on performance with respect to the other mechanisms.

Other resource allocation schemes. Other than the systems described above, there are several other existing resource allocation mechanisms that offer specific advantages for certain applications or environments. While we cannot address each individual work here, we considered various other schemes that could be generalized to single resource allocation within the scope of this paper. For instance, many systems focus on low-latency scheduling or scheduling extremely short tasks, which often prioritize efficiency or interactivity for data processing jobs within datacenters [13,16]. Similarly, other systems are designed for cluster scheduling multiple resources [8] or GPUs, which either seek to optimize DRF [7] or rely on characteristics unique to ML workloads such as gang-scheduling and placement sensitivity [11]. More complex solutions include performance-aware systems that use online feedback to allocate resources based on real-world performance rather than the resources requested by a job [3]. Recent developments in machine learning have also motivated allocation strategies based on neural networks or reinforcement learning, which are often specialized for individual tasks in domains such as cloud and mobile edge computing [4, 12].

3 Preliminaries

Given the discrepancies in functionality and objectives between mechanisms, we establish several definitions and assumptions to ensure an equitable evaluation. For instance, welfare defined by one mechanism may not translate to another, i.e. auction-based welfare must consider the payment in addition to the allocation.

Host. We assume a single centralized host with complete information regarding each user at every quantum. Given a set of arbitrary user demands that update every quantum, each host algorithm returns a set of corresponding (re)allocations, where demands and allocations are integers representing fixed-size discrete blocks of some resource (i.e. memory).

The only information that the host provides to users is the *fair share*, which is the number of blocks that every user is theoretically entitled to at a given quantum. Note that the fair share is constant across all users but not necessarily throughout all quanta. For auction-based allocation, the host also provides *borderline bids*, which are the lowest accepted bid and highest rejected bid in the previous round.

Users. We assume both demands and allocations are unpredictable from the user's perspective; that is, a user does not alter its current demand based on information from past allocations or future demands. Each user has a valuation function $V(q)$ only for auction-based allocation that determines how much a user values some quantity q in terms of price per unit per quantum (unit-price).

Furthermore, each user is characterized as either non-greedy or *greedy*, with each implementing a separate strategy depending on the mechanism. Generally, we define a greedy user as one who attempts to gain some advantage in their allocation, i.e. is selfish and strategic (misreports demand to maximize own allocation) but not necessarily adversarial (misreports demand to minimize allocation of others). Note that for most cases these definitions are interchangeable but not always equivalent.

3.1 Metrics

We define the following metrics that are standardized across all mechanisms, where B is the number of available blocks and user k has demand q_k and allocation a_k at quantum t:

- **Utilization** ($0 \leq U_t \leq 1$): the fraction of total satisfied demands [1] over B at a given t:

$$U_t = \frac{\sum_k \min(a_k, q_k)}{B}$$

Note that the total utilization U across all t is equivalent to the average of all U_t since B remains constant.

- **Welfare** ($0 \leq W_k \leq 1$): the fraction of a user's total demands that are satisfied by its allocations over all t:

$$W_k = \frac{\sum_t \min(a_k, q_k)}{\sum_t q_k}$$

For MPSP, we introduce a new comparable method for determining auction welfare that weights the allocation with the payment p_k and valuation as follows:

$$W_k = \frac{\sum_t \min(\min(a_k, q_k)\frac{V(q_k)}{p_k}, q_k)}{\sum_t q_k}$$

While auction-based algorithms such as MPSP often aim to maximize social welfare, which is defined as the sum of all user valuations of their allocation at a given t, as far as we are aware there is no method for measuring an individual's user welfare over multiple t. For a full explanation and derivation, see 3.2.

- **Incentive** ($-1 \leq I \leq 1$): difference in average welfare between non-greedy and greedy users as a measure of strategy-proofness — positive values mean users are incentivized to be honest while negative values mean users are incentivized to be dishonest.

- **Fairness** ($0 \leq F \leq 1$): ratio of minimum to maximum user welfare as a measure of equity between users:

$$\frac{\min W_k}{\max W_k}$$

Short-term fairness f_t is computed based on the instantaneous welfare at a given t instead of over all t.

3.2 Auction-Based User Welfare

The objective of auction-based user welfare is to account for lower payments despite lower allocations

[1]We only consider useful allocations since strategies such as strict partitioning may provide an allocation that is partially unused.

since a user gains something by paying less than its valuation. If a user has demand q and pays p for allocation a during some auction round, it contributes $aV(a)$ to the *social welfare*, which is often maximized in auction-based allocations. However, this metric does not quantify individual user welfare in a way that can be directly compared to others.

Thus, we normalize a user's welfare against its expected contribution $qV(q)$ to the social welfare such that a user should achieve maximum welfare if all its demands are satisfied at its valuation. We derive welfare for the general case:

$$W = \frac{\min(u\frac{V(u)^2}{p}, qV(q))}{qV(q)} \tag{1}$$

Where $u = \min(a,q)$ discards over-allocations similar to the standard calculation for welfare and $\frac{V(u)^2}{p}$ scales this value based on how favorable the payment is with respect to the allocation's valuation. This is then capped at $qV(q)$ to normalize welfare between 0 and 1. However, when $V(q) = V(u)$ for all possible u, we can factor out $V(q)$ and reduce welfare to:

$$W = \frac{\min(u\frac{V(q)}{p}, q)}{q} \tag{2}$$

Finally, if $p \leq V(q)$, which can be guaranteed by never bidding higher than $V(q)$, $\frac{V(q)}{p} \geq 1$ and welfare can further be simplified to:

$$W = \frac{\min(a\frac{V(q)}{p}, q)}{q} \tag{3}$$

We use definition (2) since we let $V(q) = K$ for all q and greedy users may bid higher than $V(q)$. Note that this method assumes a user values the ratio of its allocation and payment equally, i.e. an allocation a at price p is equivalent to an allocation $\frac{a}{2}$ at price $\frac{p}{2}$ given $V(a) = V(\frac{a}{2})$.

Algorithm 1 : Karma resource allocation algorithm.

```
demand[u]: demand of user u in the current quantum
credits[u]: credits of user u in the current quantum
alloc[u]: allocation of user u in the current quantum
$f$: fair share
$\alpha$: guaranteed fraction of fair share

Every quantum do:
 1: shared_slices ← $n \cdot (1-\alpha) \cdot f$
 2: For each user u,
 3:       increment credits[u] by $(1-\alpha) \cdot f$
 4:       donated_slices[u] = max (0, $\alpha \cdot f -$ demand[u])
 5:       alloc[u] = min (demand[u], $\alpha \cdot f$)
 6: donors ← all users u with donated_slices[u] > 0
 7: borrowers ← all users u with
 8:         alloc[u]<demand[u] & credits[u]>0

 9: while borrowers ≠ $\phi$ and
10:         ($\sum_u$ donated_slices[u] > 0 or shared_slices > 0)
    do
11:     $b^*$ ← borrower with maximum credits
12:     if donors ≠ $\phi$ then
13:         $d^*$ ← donor with minimum credits
14:         Increment credits[$d^*$] by 1
15:         Decrement donated_slices[u] by 1
16:         Update the set of donors (line 6)
17:     else
18:         Decrement shared_slices by 1
19:     Increment alloc[$b^*$] by 1
20:     Decrement credits[$b^*$] by 1
21:     Update the set of borrowers (line 7)
```

Figure 1: Karma algorithm [20]. Refer to the paper for specific implementation details and discussion.

4 Implementation

We implement strict partitioning (static allocation) and max-min fairness as baseline algorithms along with Karma based on the open-source implementation provided by the authors in the Karma paper [20]. Given Ginseng and Sharp are not publicly available, we implement the underlying ideas based on published descriptions that are relevant for our evaluation. Specifically, we focus on the MPSP auction mechanism of Ginseng and introduce a new algorithm based on the concepts presented in Sharp. As such, we refer to the algorithm implemented from Ginseng as MPSP and our algorithm as Sharp in this paper.

Our implementation provides an interface for running simulations to measure algorithmic performance. We implement the mechanisms as detailed below.

Karma. We use the definition provided by Karma for selfish users, where a greedy user always demands the max of its fair share and true demand.

MPSP auction. For consistency, we assume users can only bid a single quantity q at unit-price p each round such that its bid range is $[0,q]$ with no forbidden ranges, i.e. they are willing to pay for any partial allocations. Similarly, each user shares the same constant valuation function $V(q) = K$ such that all quantities are valued equally at the same unit-price. With the same allocation rules and exclusion compensation payment [2], the MPSP auction performs similarly to a Vickrey–Clarke–Groves (VCG) auction [18]. For auction strategy, note that a greedy user is not incentivized to alter its bid quantity since it pays for each unit of allocation. Instead, users employ the following strategy:

- Non-greedy users always bid their valuation K regardless of the lowest accepted and highest rejected bid last round.
- Greedy users bid their valuation with some additional δ. If $\delta > 0$, a user bids higher than its valuation to increase the chance of its bid being accepted with the hope of a lower payment. If $\delta < 0$, a user bids lower than its valuation to lower its payment with the hope of its bid still being accepted.

Algorithm 2: MPSP auction algorithm

```
Data : user $k$ bids $b_k$ for quantity $q_k$ and pays
       $p_k$ for allocation $a_k$
welfare $\leftarrow 0$
bidders $\leftarrow$ users sorted by highest bid
while free_blocks $> 0$ do
    u $\leftarrow$ first user of bidders
    $a_u = \min(q_u,$ free_blocks$)$
    decrement $q_u$ by $a_u$
    decrement free_blocks by $a_u$
    increment welfare by $a_u \times b_u$
    if $b_u = 0$ then
        remove first user of bidders
end
/* Compute exclusion payment */
for each winner u do
    free_blocks $= a_u$
    w_old, w_new $=$ welfare $- a_u \times b_u$
    for each remaining user v in bidders
      do
        if free_blocks $= 0$ then
            break loop
        alloc $= \min(q_v,$ free_blocks$)$
        decrement free_blocks by alloc
        increment w_new by alloc $\times b_v$
    end
    $p_u =$ (w_new $-$ w_old) / $a_u$
end
```

4.1 Ticket-Based Allocation

Since we consider a centralized approach, we let the allocator act as both the sole agent and site authority such that it is responsible for granting and exchanging all tickets. As such, tickets and blocks are granted based on separate allocations over all user demands instead of in response to user requests on a first-come-first-serve basis. Note that two allocations occur each quantum — tickets are first allocated based on a user's ticket demand and the number of available tickets, then blocks are allocated based on a user's resource demand and the number of tickets it has.

- **Oversubscription.** The total number of available tickets is determined by the *oversubscription degree* ($OD \geq 1$), which allows more tickets to be in circulation than blocks.
- **Claims.** Each ticket represents a soft reservation of a single block for a single quantum (lease period) and are granted via *claims*, which represent a group of tickets granted to a user at a given quantum. Claims can be partially redeemed and are valid for a term

[2]Computed as $p_i = \frac{1}{q_i}\sum_{k\neq i} b_k(a'_k - a_k)$ for winner i, where user k bids b_k and is allocated a_k quantity at unit-price p_k while a'_k is the allocated quantity in an auction in which i does not participate. This scheme ensures the payment is never greater than the bid price ($p_i \leq b_i$).

P quanta after being granted, at which point they expire and any unredeemed tickets are returned to the pool of available tickets.

However, Sharp does not propose any algorithm for determining how to either distribute or redeem tickets since this is dependent on each site authority. Instead, we implement both mechanisms as detailed below.

Ticket allocation. Tickets are allocated using max-min fairness. Note that since ticket allocation is completely independent of block allocation, this is interchangeable with any other allocation mechanism.

Block allocation. In the trivial case when the total demand is less than the total number of blocks B, each user k is allocated r_k, the min of its demand and number of tickets c_k. Otherwise, we iterate over B, where a user whose current allocation $a_k < r_k$ can be allocated a given block with probability $\frac{c_k}{L}$, where $L = \sum_i c_i$ is the sum of all tickets from users whose $a_i < r_i$. This is equivalent to sampling a discrete distribution with replacement that is updated when a user satisfies $a_i = r_i$ since tickets are only updated after all allocations. This ensures that a user's chance of its demand being satisfied is proportional to its number of tickets regardless of its demand quantity.

Ticket redemption. After allocation, tickets are redeemed from claims with the shortest remaining term, each claim term is updated, and expired tickets are returned to the pool.

Strategy. Users employ a similar strategy to the baselines and Karma:

- A non-greedy user requests its true block demand for both ticket and block allocation. It does not reduce its request relative to the number of tickets it has since ticket redemption is probabilistic and not guaranteed.
- A greedy user requests the max of its fair share and true demand for both ticket and block allocation. Note that the fair share defined by ticket allocation is at max OD times the block fair share and fluctuates depending on the number of available tickets.

Algorithm 3: Sharp-based algorithm

```
Data : $N$ users, user $k$ has $q_k$ demand, $c_k$
       tickets, and is allocated $a_k$
/* total_redeem is sum of $\min(q_k, c_k)$
   over all users                              */
1  if total_redeem $\leq$ total_blocks then
2      for each user u do
3          $a_u = \min(q_u, c_u)$
4      end
5  else
6      weight $\leftarrow$ $[1...N]$ initialized to 0
7      for each user u do
8          if $q_u > 0$ then
9              weight[u] $= c_u$
10     end
11     $d \leftarrow$ discrete distribution of weight
12     for each block in total_blocks do
13         u $\leftarrow$ sample $d$
14         increment $a_u$ by 1
15         if $a_u = \min(q_u, c_u)$ then
16             set weight[u] $= 0$ and update $d$
17     end
18 end
```

5 Evaluation

We evaluate static, max-min, Karma, Ginseng, and Sharp algorithms with respect to utilization, welfare, and fairness over varying values of the proportion of selfish users $0 \leq \sigma \leq 1$. We also assume the state of users remain constant throughout all quanta (no user churn).

Experimental parameters. We simulate allocations for $N = 200$ users over $T = 86400$ quanta for $\sigma = 0, 0.2, 0.4, ..., 1$. We use $B = 200$ total blocks such that the fair share per user is $f = \frac{B}{N} = 1$.

- Karma is initialized with sensitivity parameter $\alpha = 1$ and $B * T$ initial credits such that each user is guaranteed its fair share and has sufficient credits to be allocated all B blocks each quantum.
- MPSP is assumed to start in steady-state and is initialized with no base blocks such that a user must bid for its full demand every t. Each user shares the same valuation function $V(q) = K = 100$ and the host always bids for all B blocks at $\frac{K}{2}$ to ensure a minimum bid price and non-zero payment. Finally, a greedy user always randomly bids uniformly between $\pm 10\%$ (δ is uniformly distributed between -10 and 10) of its valuation.
- Sharp is initialized with $OD = 2$ and claim term $P = 2$ such that there are $2B$ total tickets and claims are valid until the end of the round after which they were granted.

Workload. We use the public Snowflake dataset [22] that provides $\sim$ 70 million customer queries over a 14-day period on Snowflake's production cluster. We randomly select 200 users (out of $\sim$ 2600) from an arbitrary 1-day interval (03/01/2018) for a sample of $\sim 1,150,000$ queries (out of $\sim 5,250,000$). Each quantum has a length of one second such that $T = 86400$ spans the interval. We assume each query contributes two blocks of demand to all quanta spanning its duration.

We use this value such that the total demand at any quantum is normally distributed around ~ 193 and we can use $B = 200$ to ensure a divisible, non-zero fair share and balance the number of scenarios in which demand is less than or greater than supply. We randomly choose $\sigma * N$ users to be considered greedy that always misrepresent their demand based on their true demand according to the strategies outlined in 4.

Utilization. We observe that Karma and MPSP maintain consistently high ($\sim 84.7\%$) levels of utilization regardless of σ. We know that MPSP is

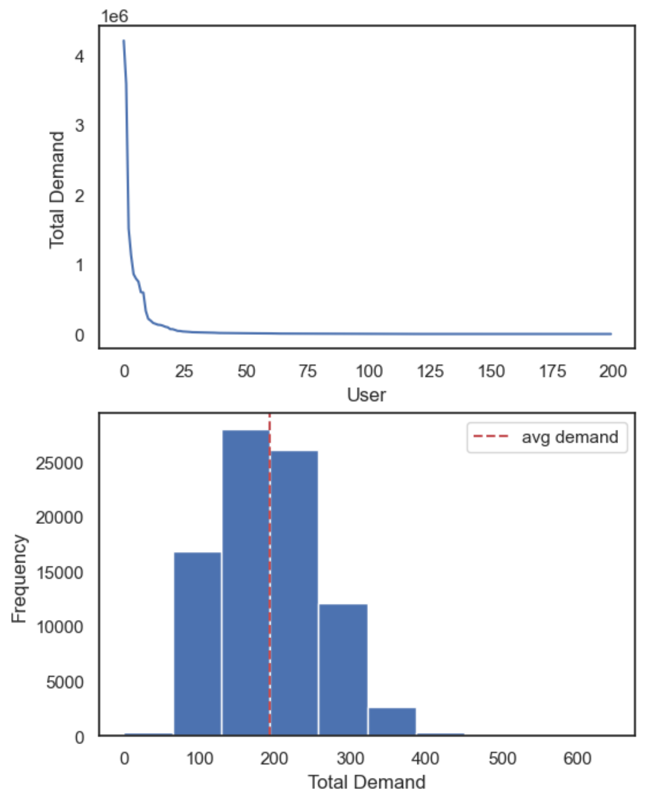

Figure 2: Distribution of total demand over all quanta by user and total demand per quantum. User demand is heavily skewed (extreme power-law distribution) while total demand is approximately normally distributed.

Pareto efficient since either all demands are satisfied or all blocks are allocated, where a user is never allocated more blocks than its true demand given that greedy users never misreport their bid quantity. Thus, we can infer that Karma is also Pareto efficient and near-optimal. Note that optimal utilization is $< 100\%$ since total demand is less than the available capacity for some quanta.

However, Sharp demonstrates a similar utilization to max-min fairness that decreases and reduces to static allocation ($\sim 19.6\%$) as σ increases. This may be attributed to Sharp using max-min fairness for ticket allocation, which attempts to balance non-greedy and greedy allocations despite the latter not being considered entirely useful. We also note that Sharp is not Pareto efficient since it is possible

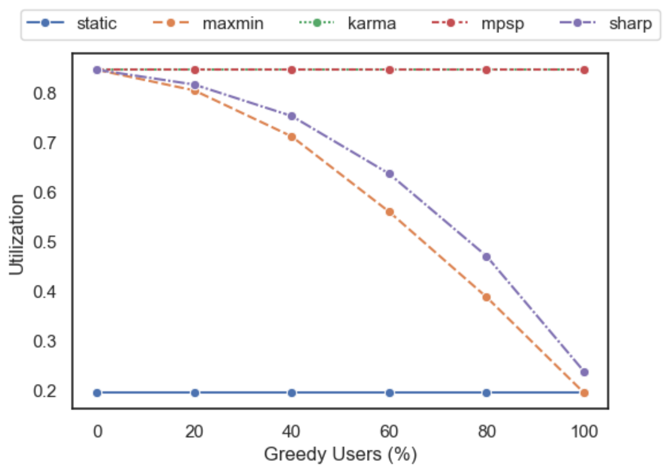

Figure 3: Utilization across percentage of greedy users. Note that Karma performance is nearly identically to MPSP.

that there is both unsatisfied demands and available blocks, which occurs when the total demand is less than the number of blocks but a user does not have sufficient tickets to redeem for its demand.

Welfare. We observe Karma outperforms max-min fairness and maintains consistently high welfare (> 0.99) regardless of σ. Sharp underperforms max-min fairness and reduces to static allocation (~ 0.82) more quickly as σ increases. However, MPSP demonstrates significantly poor performance ($\sim 0.20-0.22$) regardless of σ, suggesting that while welfare may be unaffected by the bidding behavior of the system, it is also not prioritized.

Incentive. This is the primary metric we use to evaluate strategy-proofness since it distinguishes between the average welfare of non-greedy and greedy users.

- Although Karma claims $1.17-1.6\times$ welfare improvement for non-conformant users that become conformant, we observe no such incentive similar to both baselines regardless of σ.
- Similarly, although Ginseng claims that MPSP incentivizes users to bid their true valuation, we observe that MPSP offers a slight incentive for users to be greedy (bid some δ from valuation) that decreases as σ increases, suggesting that greedy bidding is less effective the more prevalent it is.
- Sharp has a relatively significant incentive for users to be greedy with a welfare difference of $0.10-0.12$, which corresponds to a difference of $\sim 12-14\%$ between greedy and non-greedy users.

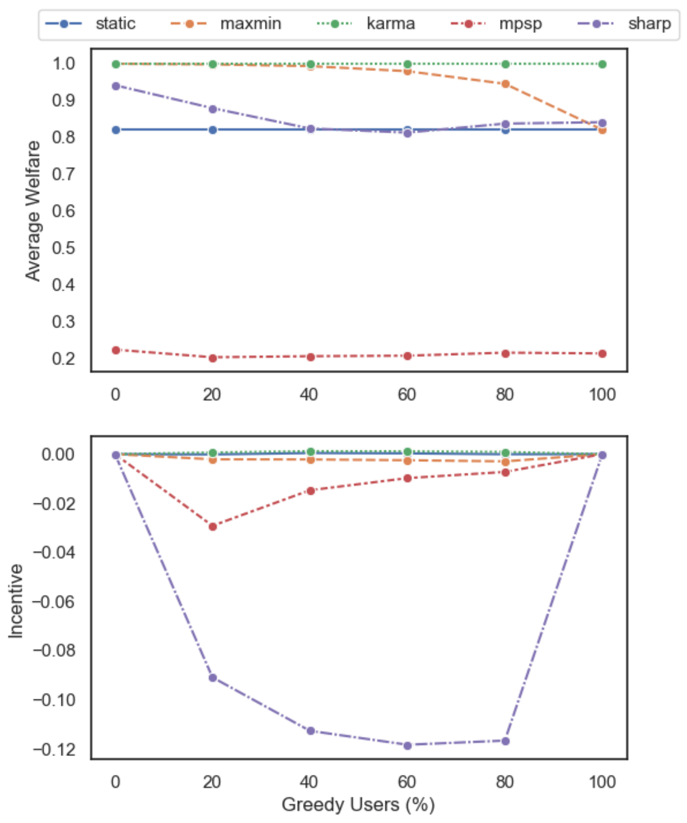

Figure 4: Average welfare and incentive across percentage of greedy users.

We find that none of the mechanisms offer incentive for users to be honest.

Long- and short-term fairness. We observe that Karma outperforms max-min fairness and achieves consistently high long-term fairness regardless of σ based on its high welfare across all users. Sharp outperforms static allocation but underperforms max-

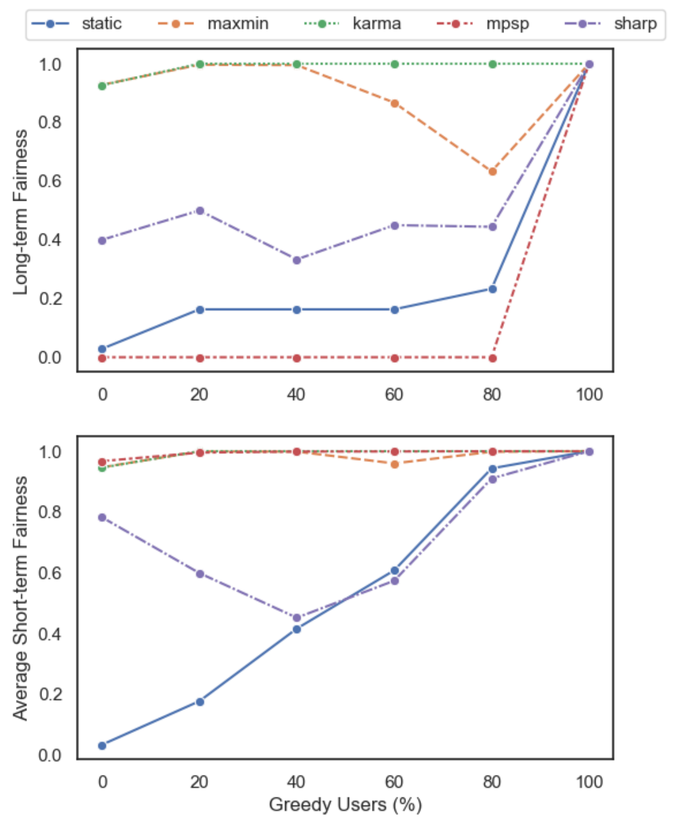

Figure 5: Long-term fairness and average short-term fairness across percentage of greedy users. We only consider fairness among non-greedy users since we expect, and perhaps seek, a welfare disparity between non-greedy and greedy users.

min fairness, fluctuating at ~ 0.4 before converging to 1 when $\sigma = 1$ (users receive same allocations when all users are greedy). However, MPSP consistently maintains zero fairness, which suggests that there may be at least one non-greedy user with extremely low demand who receives no allocations over T and has zero welfare.

On the other hand, Karma and MPSP demonstrate similar short-term fairness to max-min fairness. Since short-term fairness is based on instantaneous welfare (i.e. over a single t), Karma seems to offer consistently high welfare among all users regardless of the period. Similarly, this further suggests that MPSP still provides high fairness despite low welfare, with only a few quanta determining its long-term fairness. Sharp's short-term fairness decreases as σ increases and reduces to static allocation at $\sigma = 0.4 - 0.5$.

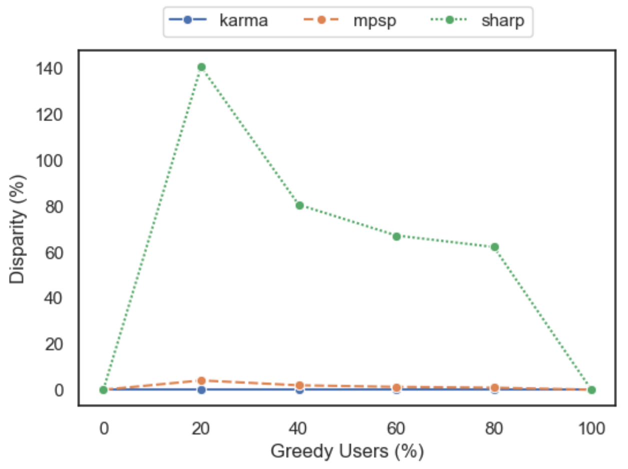

Figure 6: Disparity between non-greedy and greedy users via proxy metrics across percentage of greedy users.

Disparity. To better explain incentive, we look at the disparity in mechanism-specific metrics between greedy and non-greedy users, with higher values indicating a greater advantage for greedy users.

- For Karma, we look at the difference in average remaining credits at $t = T$ (after all allocations) between greedy and non-greedy users. Since Karma balances credits between users, we see that the disparity remains zero across all σ, which is similar to incentive.
- For MPSP, we look at the difference in average payments (unit-price) between greedy and non-greedy users when they win a bid. Greedy users receive payments that are at most 4% more favorable than non-greedy users, which may explain the slight greed incentive that we observe.
- For Sharp, we look at the difference in average tickets that greedy and non-greedy users hold after each allocation. Greedy users hold approximately $1.6 - 2.4\times$ more tickets than non-greedy users, decreasing as σ increases,

which may explain the significant greed incentive that we observe.

6 Conclusion

This paper presents a standardized implementation and evaluation of three single-resource allocation mechanisms with respect to strategy-proofness in dynamic demands. We introduce a new method for computing user welfare in auctions and an algorithm for allocating resources based on a ticket exchange system. We perform an analysis of these mechanisms under a real-world workload and show that none of them are effective for incentivizing users to remain honest.

While the presented algorithms are applicable to any single-resource environment, we only consider the general case, with specific implementations addressing each property differently depending on the context and workload. Future work may include exploring strategy-proofness over various user strategies (i.e. bidding strategy for MPSP), levels of "greediness" (i.e. always requesting higher than fair share), or algorithm parameters (i.e. *OD* or claim term for Sharp), as well as looking at different approaches for increasing user incentive to be honest.

References

[1] ATIKOGLU, B., XU, Y., FRACHTENBERG, E., JIANG, S., AND PALECZNY, M. Workload analysis of a large-scale key-value store. In *SIGMETRICS* (2012).

[2] BEN-YEHUDA, O. A., POSENER, E., BEN-YEHUDA, M., SCHUSTER, A., AND MU'ALEM, A. Ginseng: Market-driven memory allocation. *ACM SIGPLAN Notices* (2014).

[3] BHARDWAJ, R., KANDASAMY, K., BISWAL, A., GUO, W., HINDMAN, B., GONZALEZ, J., JORDAN, M., AND STOICA, I. Cilantro: Performance-aware resource allocation for general objectives via online feedback. In *SOSP* (2013).

[4] CUI, Y., HUANG, X., WU, D., AND ZHENG, H. Machine learning based resource allocation strategy for network slicing in vehicular networks. In *IEEE* (2023).

[5] FU, Y., CHASE, J., CHUN, B., SCHWAB, S., AND VAHDAT, A. Sharp: An architecture for secure resource peering. *ACM SIGOPS* (2003).

[6] FUNARO, L., BEN-YEHUDA, O. A., AND SCHUSTER, A. Ginseng: Market-driven llc allocation. *Proceedings of the 2016 USENIX Annual Technical Conference* (2016).

[7] GHODSI, A., ZAHARIA, M., HINDMAN, B., KONWINSKI, A., SHENKER, S., AND STOICA, I. Dominant resource fairness: Fair allocation of multiple resource types. In *NSDI* (2011).

[8] GRANDL, R., CHOWDHURY, M., AKELLA, A., AND ANANTHANARAYANAN, G. Altruistic scheduling in multi-resource clusters. In *OSDI* (2016).

[9] HONG, C.-Y., KANDULA, S., MAHAJAN, R., ZHANG, M., GILL, V., NANDURI, M., AND WATTENHOFER, R. Achieving high utilization with software-driven wan. In *SIGCOMM* (2013).

[10] LU, C., YE, K., XU, G., XU, C.-Z., , AND BAI, T. Imbalance in the cloud: An analysis on alibaba cluster trace. In *Big Data* (2017).

[11] MAHAJAN, K., BALASUBRAMANIAN, A., SINGHVI, A., VENKATARAMAN, S., AKELLA, A., PHANISHAYEE, A., AND CHAWLA, S. Themis: Fair and efficient gpu cluster scheduling. In *OSDI* (2019).

[12] MANAVI, M., ZHANG, Y., AND CHEN, G. Resource allocation in cloud computing using genetic algorithm and neural network. In *Preprint* (2023).

[13] MCCLURE, S., OUSTERHOUT, A., SHENKER, S., AND RATNASAMY, S. Efficient scheduling policies for microsecond-scale tasks. In *NSDI* (2022).

[14] NARAYANAN, D., KAZHAMIAKA, F., ABUZAID, F., KRAFT, P., AGRAWAL, A., KANDULA, S., BOYD, S., AND ZAHARIA, M. Solving large-scale granular resource allocation problems efficiently with pop. In *SOSP* (2021).

[15] NARAYANAN, D., SANTHANAM, K., KAZHAMIAKA, F., PHANISHAYEE, A., AND ZAHARIA, M. Heterogeneity-aware cluster scheduling policies for deep learning workloads. In *OSDI* (2020).

[16] OUSTERHOUT, K., WENDELL, P., ZAHARIA, M., AND STOICA, I. Sparrow: Distributed, low latency scheduling. In *SOSP* (2013).

[17] THOMSON, W. The manipulability of resource allocation mechanisms. *The Review of Economic Studies 51*, 3 (July 1984), 447–460.

[18] VARIAN, H. R., AND HARRIS, C. The vcg auction in theory and practice. *American Economic Review 104*, 5 (May 2014), 442–45.

[19] VEER, R., SIHMAN, S., AND RUPAVATH, B. A comprehensive study of resource allocation in cloud computing environments. *International Journal of Research Publication and Reviews* (2023).

[20] VUPPALAPATI, M., FIKIORIS, G., AND AGARWAL, R. Karma: Resource allocation for dynamic demands. *OSDI* (2024).

[21] VUPPALAPATI, M., MIRON, J., AGARWAL, R., TRUONG, D., MOTIVALA, A., AND CRUANES, T. Building an elastic query engine on disaggregated storage. *Networked Systems Design and Implementation* (2020).

[22] VUPPALAPATI, M., MIRON, J., AGARWAL, R., TRUONG, D., MOTIVALA, A., AND CRUANES, T. Building an elastic query engine on disaggregated storage. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)* (Santa Clara, CA, Feb. 2020), USENIX Association, pp. 449–462.

[23] V.VINOTHINA, SRIDARAN, D. R., AND GANAPATHI, D. A survey on resource allocation strategies in cloud computing. *International Journal of Advanced Computer Science and Applications* (2012).