FAIR INTERVAL SCHEDULING FOR MULTI-AGENT RESOURCE ALLOCATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Fair scheduling of shared computing resources aims to provide equitable access to competing groups while maintaining high throughput. We model this problem as offline interval scheduling, where each request belongs to a group, occupies a fixed time window, and competes with overlapping requests for exclusive access to a resource. To account for heterogeneous demand and conflicts within groups, we measure each group's allocation against its standalone optimum, defined as the maximum number of its requests that can be scheduled in isolation.

Our main result characterizes the optimal worst-case trade-off between fairness and throughput. For k groups and integer-endpoint intervals with lengths between 1 and Δ, we give an efficient randomized algorithm that guarantees every group at least a $1/\alpha$ fraction of its standalone optimum in expectation and achieves an approximation ratio of $\frac{\alpha\Delta}{(\alpha-k+1)\Delta+k-1}$ for every $\alpha \geq k$. We establish a matching lower bound, proving that this trade-off is optimal. The algorithm randomly selects between a maximum-throughput schedule and schedules that guarantee individual groups their standalone optima, with carefully chosen probabilities that meet the fairness requirements.

We also develop deterministic algorithms that guarantee each group a proportional share of its standalone optimum, rounded down to an integer. In particular, for two groups, we establish a tight throughput guarantee.

We also evaluate the algorithms on workload-derived instances to examine how the relationship between group membership and request structure affects fairness and throughput.

1 INTRODUCTION

Scheduling a shared computing resource involves balancing two objectives: serving as many requests as possible and ensuring fair access among users. Consider research teams reserving exclusive access to a GPU server. Accepting many short reservations may maximize the number of requests served while leaving teams requesting longer periods with little or no access. Balancing efficiency and fairness in situations like this has motivated machine learning systems such as Themis and Gavel (Mahajan et al., 2020; Narayanan et al., 2020).

We study this trade-off for competing groups whose requests require exclusive access to a shared resource during fixed time windows. Our goal is to achieve high throughput while guaranteeing each group a fair share of what it could schedule in isolation.

We model these decisions as offline interval scheduling with group labels. Each request occupies a fixed time window (an interval) and belongs to one group. Accepted requests must be pairwise non-overlapping. Throughput is the total number of accepted requests. Although earliest-finish-time greedy maximizes throughput, it offers no guarantee on how the accepted requests are distributed across groups.

To address this fairness issue, we need a meaningful measure of each group's entitlement. Simply counting its submitted requests can be misleading: a group with many overlapping requests may have less feasible demand than one with fewer, compatible requests. We therefore benchmark each group g against its *standalone optimum*, denoted OPT_g, which is the maximum number of its requests that can be scheduled if it has exclusive access to the resource. Our goal is to guarantee each group a common

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fraction of this benchmark. This approach accounts for differences in both demand volume and conflicts within groups; for example, duplicating a request does not increase a group's entitlement.

Fairness in interval scheduling has been studied in the context of allocating jobs to machines (Li et al., 2021; Borst et al., 2026) and selecting balanced sets of intervals in colored interval graphs (Bhore et al., 2021). We instead consider requests from groups competing for a single shared resource. Each group's entitlement is determined by its standalone optimum, while throughput is measured against the maximum number of requests that can be accepted without fairness constraints. Our central question is how much of this unrestricted throughput can be retained while guaranteeing each group a proportional share of its standalone optimum.

Guaranteeing each group a proportional share is challenging because requests are indivisible: a prescribed fraction of each group's standalone optimum may not be achievable in a single schedule. Randomization provides a way to address this issue by guaranteeing fairness in expectation while ensuring that every realized schedule remains feasible. Specifically, for a parameter α, we require each group g to receive at least $\textsc{Opt}_g/\alpha$ accepted requests in expectation on every instance.

As we show in Section 3, the strongest fairness guarantee achievable on every instance with k groups corresponds to $\alpha = k$. One simple way to achieve this guarantee is to select a group uniformly at random and return its standalone optimal schedule. However, this approach may leave compatible requests from other groups unserved, reducing overall throughput. This leads to our central question: *How much throughput can we retain while guaranteeing each group its proportional share of the standalone optimum?* We investigate this trade-off both at $\alpha = k$ and under weaker fairness requirements obtained by increasing α.

1.1 Contributions

We develop randomized and deterministic algorithms that balance group fairness and throughput in interval scheduling, establishing tight guarantees for key settings. Our results apply to intervals with integer endpoints and length at most Δ. We measure throughput relative to the unrestricted optimum, using expected throughput for randomized algorithms. Our main contributions are as follows.

- **Optimal randomized fairness–throughput trade-offs.** In § 4, we establish that, for every real $\alpha \geq k$, the optimal worst-case approximation ratio under α-fairness in expectation is
$$\frac{\alpha\Delta}{(\alpha - k + 1)\Delta + k - 1}.$$
We introduce the Group Protection Sampling Algorithm (GPSA) and show that it attains this bound. We also establish a matching lower bound that applies to every randomized algorithm with the same fairness guarantee. GPSA samples from at most $k + 1$ candidate schedules: an unrestricted optimum and schedules that preserve individual groups' standalone optima. It chooses the sampling probabilities to meet the fairness targets and returns the unrestricted optimum with certainty whenever that schedule already satisfies them.
- **Deterministic guarantees for individual schedules.** In §5, we introduce the deterministic Block Partitioning Algorithm (BPA), which guarantees every group at least $\lfloor \textsc{Opt}_g/\alpha \rfloor$ accepted requests for any number of groups and every integer $\alpha \geq k$. For two groups, we develop specialized algorithms and establish the tight approximation ratio $\frac{\alpha(\Delta-1)+1}{(\alpha-1)(\Delta-1)+1}$ for every integer $\alpha \geq 2$. We also establish feasibility and complexity results for unrounded fairness requirements.
- **Evaluation on workload-derived instances.** In § 6, we evaluate our algorithms on workload traces with different group assignments. We examine how fairness and throughput vary with the number of groups and the fairness requirements, and how these outcomes are affected by the relationship between group membership and request characteristics.

2 Related Work

Interval scheduling and applications. Interval scheduling is a classical optimization problem with a wide range of applications, including crew scheduling, telecommunications, and satellite observation. Numerous variants have been studied, including weighted requests, multiple machines, and availability constraints; see the survey by Kolen et al. (2007).

Fairness is also an important consideration in shared computing systems. Dominant Resource Fairness extends max-min sharing to settings with multiple resource types (Ghodsi et al., 2011). For GPU clusters, Themis introduces finish-time fairness (Mahajan et al., 2020), while Gavel accounts for differences among accelerators when designing policies that balance fairness and job completion time (Narayanan et al., 2020). Motivated by these systems, we study fair access to a single shared resource through fixed reservation requests, focusing on which requests from each group are accepted.

Fairness models and scheduling constraints. Bhore et al. (2021) study independent sets with prescribed, equal color quotas in interval graphs, which corresponds to selecting non-overlapping intervals subject to group quotas. They establish hardness results and develop parameterized algorithms for this problem. Unlike their prescribed, equal quotas, our fairness requirements are determined by each group's standalone optimum. Li et al. (2021) study maximin share and envy-freeness when jobs are allocated to agents' machines. Related work on fair allocation with conflicts includes Hummel & Hetland (2022), while Equbal et al. (2026) study envy-freeness up to one item (EF1) together with maximality, including settings with interval conflicts. In these models, fairness concerns the allocation received by each agent. In our setting, requests have fixed group labels and compete for admission to a single shared resource; overlapping requests cannot both be accepted, even when they belong to different groups.

A different notion of fairness concerns delays rather than admission. Zhao et al. (2025) study online preemptive scheduling with job-size predictions, aiming to minimize maximum stretch—the ratio of response time to processing time. In contrast, our fairness guarantee concerns admission: it measures the fraction of each group's feasible requests that are served.

Fairness, throughput, and randomization. The *price of fairness* captures the loss in efficiency caused by fairness requirements. Bertsimas et al. (2011) derive tight bounds for max-min and proportional fairness over convex utility sets. Their notion of proportional fairness maximizes the sum of logarithmic utilities, whereas our guarantees impose lower bounds relative to each group's standalone optimum. Fairness–efficiency trade-offs have also been studied in scheduling. Agnetis et al. (2019) consider two-agent single-machine scheduling with completion-time and tardiness objectives. More recently, Borst et al. (2026) study unweighted interval scheduling on identical machines under envy-freeness-up-to-one-item (EF1) fairness, obtaining an offline $3/2$-approximation with asymptotically matching lower bounds. In their model, jobs are allocated among machines; in ours, fixed requests from different groups compete for admission to a single shared resource. For additive valuations, Aziz et al. (2024) study randomized allocations that are envy-free in expectation while satisfying the weaker EF1 guarantee in every realized allocation. This distinction between expected and realized guarantees is also important in our setting. Our randomized schedules satisfy the group fairness targets in expectation while remaining feasible in every realization, whereas our deterministic schedules satisfy rounded group quotas in each realized schedule.

3 Preliminaries and Problem Formulation

3.1 Scheduling Model and Benchmarks

The input to the fair interval scheduling problem is a collection of n reservation requests $\mathcal{I} = \{I_1, \ldots, I_n\}$. Each request I occupies a fixed, half-open time interval $[\ell(I), r(I))$, where $\ell(I), r(I) \in \mathbb{Z}_{\geq 0}$ and $\ell(I) < r(I)$, and has length $\mathrm{len}(I) = r(I) - \ell(I)$. Distinct requests may have identical endpoints. Because endpoints are integral, intervals lie on a unit grid and have integer lengths. Let $\Delta = \max_{I \in \mathcal{I}} \mathrm{len}(I)$ denote the maximum interval length; hence every request has length in $\{1, \ldots, \Delta\}$. The requests are partitioned into $k = |G| \geq 2$ nonempty groups. Writing $g(i)$ for the group of I_i, let $\mathcal{I}_g = \{I_i \in \mathcal{I} : g(i) = g\}$. We work offline: all intervals and group labels are known before scheduling. Intervals I and J are compatible if $r(I) \leq \ell(J)$ or $r(J) \leq \ell(I)$. A schedule $S \subseteq \mathcal{I}$ is feasible if its requests are pairwise compatible. Requests have unit value, so throughput is $|S|$; write $S_g = S \cap \mathcal{I}_g$ for group g's allocation.

Let O and $O_g^\star$ be maximum-cardinality schedules on $\mathcal{I}$ and $\mathcal{I}_g$, respectively. Define the unrestricted optimum $\textsc{Opt} = |O|$ and group g's *standalone optimum* $\textsc{Opt}_g = |O_g^\star|$. These satisfy $1 \leq \textsc{Opt}_g \leq \textsc{Opt} \leq \sum_{h \in G} \textsc{Opt}_h$ for every $g \in G$. The benchmark $\textsc{Opt}_g$ measures the service attainable with exclusive access, accounts for conflicts within the group, and is unchanged by duplicating a request.

3.2 Fairness and Approximation Guarantees

For a real parameter $\alpha \geq 1$, we seek to guarantee each group a $1/\alpha$ fraction of its standalone optimum while maximizing throughput. Smaller α means stronger fairness.

Definition 3.1 (α-fairness in expectation)**.** A randomized algorithm ALG is *α-fair in expectation* if, on every input $\mathcal{I}$, its output $S = \text{ALG}(\mathcal{I})$ is feasible in every realization and satisfies

$$\mathbb{E}[|S_g|] \geq \text{OPT}_g/\alpha \qquad \text{for every } g \in G.$$

This is an *ex ante* fairness guarantee: fairness is required in expectation over the algorithm's randomization, while every realized schedule must only be feasible.

The next proposition identifies the threshold at which a universal fairness guarantee becomes possible.

Proposition 3.2 (Universal fairness bound)**.** *For any integer $k \geq 2$ and real $\alpha \geq 1$, there is a randomized algorithm that is α-fair on every instance with k groups if and only if $\alpha \geq k$.*

Proof. Fix an integer $B \geq k$ and give every group one request $[t, t+1)$ for each $t \in \{0, \ldots, B-1\}$. Then $\text{OPT}_g = B$ for every group, while each feasible schedule contains at most B requests. Fairness would imply $kB/\alpha \leq \sum_{g \in G} \mathbb{E}[|S_g|] = \mathbb{E}[|S|] \leq B$, hence $\alpha \geq k$. This holds for arbitrarily large B. Conversely, selecting a group uniformly at random and returning its standalone optimal schedule gives each group g expected service OPT_g/k, which is at least OPT_g/α whenever $\alpha \geq k$. □

Thus, $1/k$ is the strongest universal expected share. Our focus is the best throughput guarantee compatible with fairness for each $\alpha \geq k$.

Fairness for an individual schedule. We also consider *ex post* guarantees, where the fairness requirement must hold for the schedule actually returned.

Definition 3.3 (α-fair schedule)**.** A feasible schedule S is *α-fair* if $|S_g| \geq \text{OPT}_g/\alpha$ for every $g \in G$.

Equivalently, $|S_g| \geq \lceil \text{OPT}_g/\alpha \rceil$. We call the case $\alpha = k$ *exact k-fairness*; "exact" refers to the unrounded requirement and does not require equality.

Observation 3.4 (Infeasibility of unrounded fairness). For every integer $k \geq 2$ and finite real $\alpha \geq 1$, some instance admits no α-fair schedule.

Proof. Give each group one request $[0, 1)$. Since $\text{OPT}_g = 1$, fairness requires at least one accepted request per group, but a feasible schedule accepts at most one. At $\alpha = k$, infeasibility persists with arbitrarily large standalone optima: use the construction in Proposition 3.2 with B not divisible by k. Exact fairness would require $|S| \geq k\lceil B/k \rceil > B$. □

For deterministic guarantees, we therefore round the targets down.

Definition 3.5 (Floor-α-fair schedule)**.** A feasible schedule S is *floor-α-fair* if $|S_g| \geq q_g(\alpha)$ for every $g \in G$, where $q_g(\alpha) = \lfloor \text{OPT}_g/\alpha \rfloor$. The case $\alpha = k$ is *floor-k-fairness*.

Write q_g when α is fixed. Rounding reduces each target by less than one request, with $q_g = 0$ when $\text{OPT}_g < \alpha$. The definition allows real α, but our deterministic algorithms based on block partitioning assume integer $\alpha \geq k$.

Throughput guarantees. For $\rho \geq 1$, a randomized algorithm has approximation ratio at most ρ in expectation if it returns feasible schedules with $\mathbb{E}[|\text{ALG}(\mathcal{I})|] \geq \text{OPT}/\rho$ on every input in the stated class. For deterministic algorithms, the condition is $|\text{ALG}(\mathcal{I})| \geq \text{OPT}/\rho$. We state the required fairness guarantee alongside this bound. Comparison with the unrestricted optimum measures the throughput retained under fairness.

3.3 Basic Scheduling Properties

Greedy selection and completion. Earliest-finish-time greedy scans requests in nondecreasing order of right endpoint, accepting each request compatible with those already selected. Ties follow a fixed ordering of request indices. A single sort suffices to compute O and all $O_g^\star$ in total $O(n \log n)$ time. Given a feasible schedule P, apply greedy to all requests outside P compatible with every interval of P. Adding the result to P gives a maximum-cardinality schedule containing P.

Overlap and maximality. A feasible schedule is *maximal* if no further request can be added. The unit-grid assumption gives the following useful bounds.

Lemma 3.6 (Unit-grid overlap bound)**.** *An input interval I of length d overlaps at most d intervals of any feasible schedule Q, counting itself if $I \in Q$.*

Proof. Integer endpoints ensure that each overlap contains a unit slot $[t, t+1)$ within I. Choose one such slot per overlapping interval of Q. Feasibility makes these slots distinct, and I contains exactly d unit slots. □

Lemma 3.7 (Throughput of maximal schedules)**.** *Every maximal schedule S has $|S| \geq \textsc{Opt}/\Delta$.*

Proof. By maximality, every interval of O overlaps one of S, counting common intervals as overlapping themselves. By Lemma 3.6, each interval of S overlaps at most Δ intervals of O. Hence $\textsc{Opt} = |O| \leq \Delta|S|$. □

4 Optimal Randomized Fair Scheduling

We now turn to randomized scheduling and study the trade-off between expected fairness and throughput. By Proposition 3.2, the strongest fairness guarantee that can hold universally is a $1/k$ share of each group's standalone optimum. More generally, for every $\alpha \geq k$, we ask how much throughput can be retained while guaranteeing each group an expected share of at least $1/\alpha$.

Theorem 4.1 (Optimal randomized trade-off)**.** *Let $k \geq 2$ and $\Delta \geq 1$ be integers, and let $\alpha \geq k$ be real. On unit-grid instances with k groups and maximum interval length Δ, the optimal worst-case approximation ratio among randomized offline algorithms that are α-fair in expectation is*

$$\rho^\star(\alpha, k, \Delta) = \frac{\alpha\Delta}{(\alpha - k + 1)\Delta + k - 1}. \tag{1}$$

The Group Protection Sampling Algorithm (GPSA) attains this ratio in $O(n \log n)$ time.

At $\alpha = k$, the ratio is $k\Delta/(\Delta + k - 1)$; at $\Delta = 1$, fairness is compatible with optimal throughput. For fixed k and Δ, the ratio tends to 1 as α grows. GPSA does not require Δ as an input,

4.1 Group Protection Sampling Algorithm

GPSA considers the unrestricted optimal schedule together with one protection schedule for each group. The protection schedule for group g preserves all $\textsc{Opt}_g$ requests of a standalone optimal schedule for g, and then adds as many compatible requests from other groups as possible. GPSA then samples among these schedules, assigning more probability to groups that receive a smaller fraction of their standalone optimum in the unrestricted schedule.

Candidate schedules. Index the groups by $G = \{1, \ldots, k\}$. Compute O and all $O_g^\star$ by earliest-finish-time greedy with the fixed tie-breaking rule. For each g, let $\mathcal{R}_g$ contain the requests outside $O_g^\star$ compatible with every interval of $O_g^\star$. Let T_g be a maximum-cardinality schedule on $\mathcal{R}_g$, obtained by greedy, and define $A_g = O_g^\star \cup T_g$. Thus, A_g is a largest feasible schedule containing $O_g^\star$: it preserves the group's standalone optimum while accepting as many additional requests as possible.

Sampling probabilities. Set $c_g = |O \cap \mathcal{I}_g|$ and $r_g = c_g/\textsc{Opt}_g \in [0, 1]$. If $r_g \geq 1/\alpha$ for every group, GPSA returns O deterministically. Otherwise, it returns A_g with probability p_g and O with probability p_0, where

$$p_g = \frac{1 - r_g}{\alpha} \quad (g \in G), \tag{2}$$

$$p_0 = 1 - \sum_{g \in G} p_g = \frac{\alpha - k + \sum_{g \in G} r_g}{\alpha}. \tag{3}$$

In this sampling branch, each group's protection probability is proportional to its shortfall $1 - r_g$ from full standalone service. Increasing α shifts probability toward O; once O meets all targets, it is returned deterministically. Although the distribution has at most $k + 1$ candidates, only the sampled protection schedule is constructed, since the probabilities depend only on O and the standalone optima. Algorithm 1 illustrates GPSA.

Algorithm 1 Group Protection Sampling Algorithm (GPSA)

Require: Instance $\mathcal{I}$ with groups $G = \{1, \ldots, k\}$; real $\alpha \geq k$.
Ensure: A schedule S that is feasible in every realization and satisfies α-fairness in expectation.
1: Sort by intervals in I by their right endpoint; compute O and all $O_g^\star$ by greedy strategy.
2: For each $g \in G$, set $r_g \leftarrow |O \cap \mathcal{I}_g|/|O_g^\star|$.
3: **if** $r_g \geq 1/\alpha$ for every $g \in G$ **then**
4: **return** O.
5: Set $p_g \leftarrow (1 - r_g)/\alpha$ for each g and $p_0 \leftarrow 1 - \sum_{g \in G} p_g$.
6: Sample $J \in \{0, 1, \ldots, k\}$ with $\Pr[J = j] = p_j$.
7: **if** $J = 0$ **then**
8: **return** O.
9: Let $\mathcal{R}_J$ contain the requests outside $O_J^\star$ compatible with every interval of $O_J^\star$.
10: Compute T_J by applying greedy on $\mathcal{R}_J$.
11: **return** $O_J^\star \cup T_J$.

4.2 Fairness and Throughput Guarantees

We now establish the fairness and throughput guarantees of GPSA. The analysis relies on two key facts: the unrestricted optimum receives sufficient probability, and every protection schedule retains a controlled fraction of the optimal throughput.

Lemma 4.2 (Probability assigned to the optimum)**.** *The probabilities in equation 2–equation 3 form a distribution with* $p_0 \geq (\alpha - k + 1)/\alpha \geq 1/\alpha$.

Proof. For each group g, we have $0 \leq r_g = c_g/\textsc{Opt}_g \leq 1$, so $p_g = (1 - r_g)/\alpha \geq 0$. It remains to show that p_0 is sufficiently large. Since $\textsc{Opt}_g \leq \textsc{Opt}$ for every g, we have $r_g = c_g/\textsc{Opt}_g \geq c_g/\textsc{Opt}$. Summing over all groups and using $\sum_g c_g = \textsc{Opt}$ gives $\sum_g r_g \geq \sum_g c_g/\textsc{Opt} = 1$. Therefore, by equation 3, $p_0 = \frac{\alpha - k + \sum_g r_g}{\alpha} \geq \frac{\alpha - k + 1}{\alpha} \geq \frac{1}{\alpha}$, where the last inequality follows from $\alpha \geq k$.

Thus all probabilities are nonnegative, and they sum to one by the definition of p_0. □

Lemma 4.3 (Throughput of protection candidates)**.** *Every candidate* A_g *is feasible, has* $|A_g \cap \mathcal{I}_g| = \textsc{Opt}_g$*, and satisfies* $|A_g| \geq \textsc{Opt}/\Delta$.

Proof. Feasibility of $O_g^\star$ implies that A_g serves exactly $\textsc{Opt}_g$ requests from group g. As a largest feasible schedule containing $O_g^\star$, it is maximal. The throughput bound follows from Lemma 3.7. □

Theorem 4.4 (Guarantees of GPSA)**.** *For every real* $\alpha \geq k$*, GPSA is* α*-fair in expectation. Its output* S *is feasible in every realization and satisfies* $\mathbb{E}[|S|] \geq \textsc{Opt}/\rho^\star(\alpha, k, \Delta)$.

Proof. If O meets all fairness targets, GPSA returns it and both guarantees hold. Otherwise, every candidate is feasible. For any group g, O serves c_g requests and A_g serves $\textsc{Opt}_g$. Ignoring the nonnegative contributions from other protection candidates gives

$$\mathbb{E}[|S_g|] \geq p_0 c_g + p_g \textsc{Opt}_g = \frac{\textsc{Opt}_g}{\alpha} + \left(p_0 - \frac{1}{\alpha}\right) c_g \geq \frac{\textsc{Opt}_g}{\alpha},$$

where the last inequality follows from Lemma 4.2. For throughput, Lemma 4.3 yields

$$\mathbb{E}[|S|] = p_0 \textsc{Opt} + \sum_{g \in G} p_g |A_g| \geq \left(p_0 + \frac{1 - p_0}{\Delta}\right) \textsc{Opt} \geq \left(1 - \frac{k - 1}{\alpha}\left(1 - \frac{1}{\Delta}\right)\right) \textsc{Opt} = \frac{\textsc{Opt}}{\rho^\star(\alpha, k, \Delta)}.$$

The last inequality uses $1 - p_0 \leq (k - 1)/\alpha$ and $\Delta \geq 1$; the equality follows from equation 1. □

Running time. One sort and a scan maintaining a separate greedy state for each group compute O and all $O_g^\star$ in $O(n \log n)$ time. Counting group allocations, computing probabilities, and sampling take $O(n + k)$ time. For the sampled protection candidate, binary search in its sorted standalone schedule checks compatibility in $O(\log n)$ time per request, followed by one greedy scan. Since only the sampled candidate is completed, the total time is $O(n \log n)$.

4.3 MATCHING LOWER BOUND

We now show that the throughput guarantee achieved by GPSA is best possible under α-fairness in expectation. The lower-bound construction creates a direct conflict between fairness and throughput: requests from $k-1$ groups span entire blocks, while the remaining group has many short unit requests within each block. Serving the former is necessary to satisfy their fairness requirements, but every such choice displaces several short requests and therefore reduces total throughput.

Theorem 4.5 (Matching approximation lower bound)**.** *For integers $k \geq 2$ and $\Delta \geq 1$ and real $\alpha \geq k$, every randomized offline algorithm that guarantees α-fairness in expectation has worst-case approximation ratio at least $\rho^\star(\alpha, k, \Delta)$.*

Proof. Fix an integer $B \geq 1$ and partition $[0, B\Delta)$ into blocks $[t\Delta, (t+1)\Delta)$, $t = 0, \ldots, B-1$. In each block, every group $g < k$ has one request spanning the entire block, while group k has Δ unit requests partitioning it. Thus $\textsc{Opt}_g = B$ for $g < k$ and $\textsc{Opt}_k = \textsc{Opt} = B\Delta$.

For the output S of any algorithm that is α-fair in expectation, set $X = \sum_{g=1}^{k-1} |S_g|$ and $Y = |S_k|$. Fairness gives $\mathbb{E}[X] \geq (k-1)B/\alpha$. The X accepted length-Δ requests occupy distinct blocks, so $Y \leq \Delta(B - X)$ in every realization. Consequently,

$$\mathbb{E}[|S|] = \mathbb{E}[X + Y] \leq B\Delta - (\Delta - 1)\mathbb{E}[X] \leq B\Delta - \frac{(\Delta-1)(k-1)B}{\alpha} = \frac{\textsc{Opt}}{\rho^\star(\alpha, k, \Delta)}.$$

The second inequality uses $\Delta \geq 1$. Dividing $\textsc{Opt}$ by this upper bound proves the claim. Since B is arbitrary, the bound holds even with arbitrarily large standalone optima. □

5 FAIRNESS IN INDIVIDUAL SCHEDULES

Fairness in expectation allows a group's allocation to vary across realizations: a group may receive fewer intervals in one schedule and more in another. We now study deterministic algorithms whose returned schedule must satisfy every group's fairness quota.

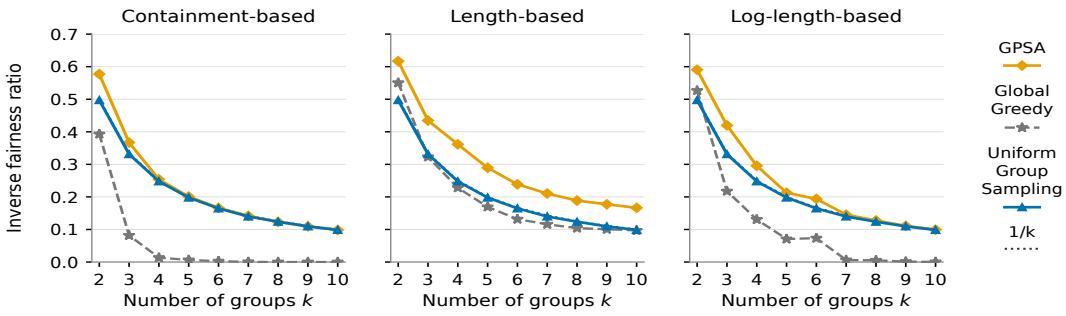
Specifically, we impose the floor quotas $|S_g| \geq \lfloor \textsc{Opt}_g/\alpha \rfloor$ from Definition 3.5. Rounding reduces each group's target by less than one interval. Our goal is to satisfy every group's quota while retaining as much total throughput as possible. For two groups, we characterize the optimal trade-off exactly.

Theorem 5.1 (Tight deterministic trade-off for two groups)**.** *For two groups and integers $\alpha \geq 2$ and $\Delta \geq 1$, a deterministic $O(n \log n)$-time algorithm returns a floor-α-fair schedule S satisfying $\frac{\textsc{Opt}}{|S|} \leq \frac{\alpha(\Delta-1)+1}{(\alpha-1)(\Delta-1)+1}$. This factor is tight: for every such α and Δ, there is an instance on which every floor-α-fair schedule has a ratio at least this large.*

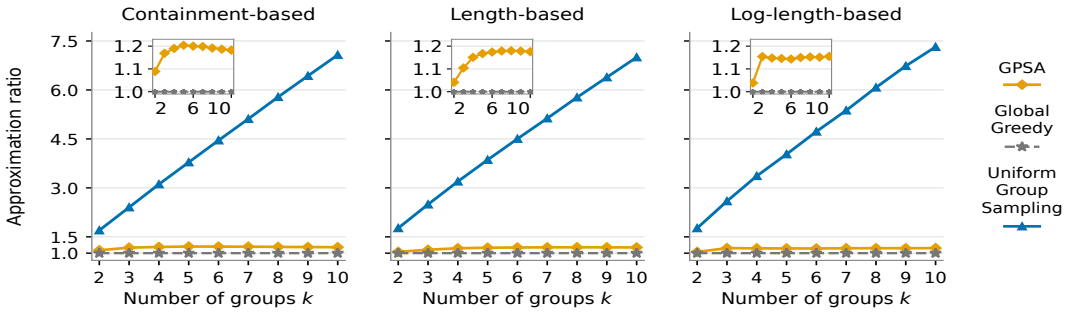
At $\alpha = 2$, the bound is $2 - 1/\Delta$ and is attained by *Augmented Two-Cut* (Algorithm 2), which combines portions of optimal schedules and then augments the result with additional compatible intervals (See § A.2). For integer $\alpha \geq 3$, *Minimum-Conflict Block Replacement* (Algorithm 3) repairs a deficient group's allocation by inserting a block from its individual optimum while preserving the other group's quota (See § A.3). The matching lower bound and proofs appear in Appendices A.1–A.3.

Block partitioning for general groups. For k groups and integer $\alpha \geq k$, the deterministic *Block Partitioning Algorithm* (BPA) first computes an unrestricted optimum and returns it if all quotas are met. Otherwise, it constructs a fair schedule by choosing one *block* of intervals from each group with a positive quota, then adding further compatible requests to improve throughput.

To form the blocks, BPA partitions each such group's individual optimum chronologically into α blocks whose sizes differ by at most one. Every block therefore contains enough intervals to meet its group's quota. A block's *span* extends from its first interval's start to its last interval's finish. BPA repeatedly selects the available block whose span ends earliest, then discards the remaining blocks of that group and all blocks whose spans overlap the selected span. See Algorithm 4 and Figure 3 for an illustration of BPA. The earliest-ending rule ensures that selecting a block leaves enough choices for the remaining groups. In particular, a selected span can overlap at most one available block from each other group: if it overlapped two, the earlier of those two blocks would end before the selected span, contradicting the selection rule. Thus, each selection discards at most one block from each



(a) Inverse fairness ratio $\widehat{C}$, with the $1/k$ reference.



(b) Approximation ratio $\widehat{\rho}$, with insets showing GPSA and Global Greedy near 1.

Figure 1: Fairness and throughput at $\alpha = k$ for GPSA, UGS, and Global Greedy.

unselected group. Since every group starts with $\alpha \geq k$ blocks and there are at most $k-1$ other groups, it retains an available block until its turn to be selected. The selected blocks have disjoint spans and meet all positive quotas, so their intervals form a feasible floor-α-fair schedule. BPA then augments this schedule using greedy on all original requests compatible with the selected intervals. The algorithm runs in $O(n \log n)$ time and achieves an $O(k)$ approximation at $\alpha = k$. Appendix A.4 gives the detailed explanation and analysis; Appendix A.5 provides complementary lower bounds.

For completeness, we also characterize exact quota-constrained scheduling: deciding exact k-fairness is NP-complete when k is part of the input, even for $\Delta = 2$, while for fixed k a dynamic program computes a maximum-throughput schedule subject to arbitrary prescribed group quotas. See Appendices A.6 and A.7 for details.

6 Experimental Evaluation

In this section, we experimentally evaluate how GPSA balances throughput with fairness.

Setup. We construct interval instances from four real-world workload traces in Standard Workload Format (Chapin et al., 1999): CTC-SP2-1996, LANL-CM5-1994, NASA-iPSC-1993, and SDSC-SP2-1998. These traces record jobs submitted to production high-performance computing systems at the Cornell Theory Center, Los Alamos National Laboratory, NASA Ames, and the San Diego Supercomputer Center, respectively. Each record includes a submission time s_j and runtime p_j; we map it to the fixed interval $[s_j, s_j + p_j)$ on one shared resource. The resulting instances contain between 18,066 and 122,058 requests.

To examine different relationships between group membership and request structure, we use three group assignments. The *length-based* assignment orders requests by length and divides them into approximately equal-sized groups. The *containment-based* assignment does the same after ordering requests by the number of other intervals they contain. The *log-length-based* assignment instead partitions the range of interval lengths into logarithmically spaced bins, allowing group sizes and

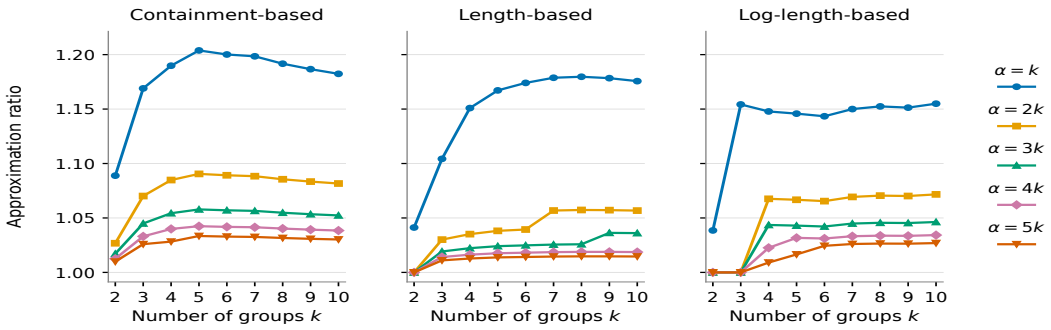


Figure 2: GPSA's approximation ratio $\widehat{\rho}$ for $\alpha \in \{k, 2k, 3k, 4k, 5k\}$.

standalone optima $\textsc{Opt}_g$ to vary substantially across groups. We consider $k \in \{2, \ldots, 10\}$. These are constructed group labels rather than the original users or accounts in the traces. Thus, the experiments preserve the timing and duration patterns of real workloads while evaluating the fixed-interval model studied in this paper. Appendix B.1 gives the full construction and implementation details.

Baselines and metrics. We compare GPSA with two baselines. *Global Greedy* applies earliest-finish-time greedy to all requests and therefore returns an unrestricted throughput-optimal schedule. *Uniform Group Sampling* (UGS) chooses a group uniformly at random and returns its standalone optimal schedule. Hence, UGS gives every group a $1/k$ fraction of its standalone optimum in expectation, but does not attempt to retain requests from other groups.

For GPSA and UGS, let $\bar{a}$ and $\bar{a}_g$ denote the mean total and group-g accepted counts over $100{,}000$ runs. For Global Greedy, these are its deterministic counts. We report $\widehat{\rho} = \textsc{Opt}/\bar{a}$, and $\widehat{C} = \min_{g \in G} \bar{a}_g / \textsc{Opt}_g$. Here, $\widehat{\rho}$ measures throughput loss relative to the unrestricted optimum (lower is better), while $\widehat{C}$ measures the normalized mean service of the worst-served group (higher is better). For the randomized algorithms, we average the accepted counts over runs before computing these metrics. Each plotted point is then obtained by computing the metric separately on each workload and averaging the four resulting values.

Fairness with high throughput. At the strongest universally feasible fairness level, $\alpha = k$, GPSA gives the worst-served group substantially more normalized service than Global Greedy in many of the tested settings (Figure 1(a)). The difference is most pronounced for containment-based and log-length-based groups as k grows, where Global Greedy's mean $\widehat{C}$ approaches zero. UGS follows the $1/k$ reference by construction, while GPSA achieves comparable or larger values, with the largest separation under the length-based assignment.

This improvement in group service comes with a relatively small throughput loss on these instances. As shown in Figure 1(b), GPSA's mean approximation ratio ranges from about 1.04 to just over 1.20. In contrast, UGS sacrifices substantially more throughput, with its mean ratio reaching roughly 7 at $k = 10$. The insets make the smaller difference between GPSA and the throughput-optimal Global Greedy visible. Per-workload results are reported in Appendix B.5.

Relaxing fairness. We next keep each instance and its group labels fixed and increase α. Moving from $\alpha = k$ to $\alpha = 2k$ already brings all plotted mean approximation ratios below 1.10 (Figure 2); further relaxation produces smaller additional gains. This is consistent with GPSA's sampling rule: as α increases, more probability is assigned to the unrestricted optimum.

Additional experiments. The appendix complements these randomized results in several ways. Appendix B.2 compares GPSA with the deterministic BPA at $\alpha = k$; because BPA enforces rounded quotas in every returned schedule while GPSA provides unrounded fairness in expectation, we report their results under their respective fairness measures rather than treating them as a direct head-to-head comparison. Appendix B.3 tracks both GPSA and BPA as α is relaxed. Appendix B.4 compares BPA with the specialized TwoCut and MCBR algorithms for two groups, and Appendix B.5 reports the results separately for each workload.

AI USE STATEMENT

We used ChatGPT (OpenAI, 2026a) and Claude (Anthropic, 2026) for grammar and language editing. We also consulted generative AI tools for feedback on mathematical arguments, experimental comparisons, and figure presentation. OpenAI Codex (OpenAI, 2026b) assisted with refactoring the experimental and plotting code. The authors reviewed the AI-assisted edits and suggestions. We checked text edits against the authors' draft and verified that the refactored code reproduced the original results with fixed random seeds. The authors take responsibility for the final manuscript, theoretical claims, code, and experimental results.

REPRODUCIBILITY STATEMENT

The scheduling model and assumptions are specified in Section 3. Proofs of the theoretical results are provided in the main text and Appendix A. Sections B and 6 describe the data sources, workload construction, group assignments, baselines, parameter settings, and evaluation protocol. The supplementary material includes code for constructing the instances, running the evaluated algorithms and baselines, computing unrestricted and per-group optima, and reproducing the figures.

REFERENCES

Alessandro Agnetis, Bo Chen, Gaia Nicosia, and Andrea Pacifici. Price of fairness in two-agent single-machine scheduling problems. *European Journal of Operational Research*, 276(1):79–87, 2019.

Anthropic. Claude, 2026. URL `https://claude.com/product/overview`.

Haris Aziz, Rupert Freeman, Nisarg Shah, and Rohit Vaish. Best of both worlds: Ex ante and ex post fairness in resource allocation. *Oper. Res.*, 72(4):1674–1688, 2024.

Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. The price of fairness. *Operations research*, 59(1):17–31, 2011.

Sujoy Bhore, Jan-Henrik Haunert, Fabian Klute, Guangping Li, and Martin Nöllenburg. Balanced independent and dominating sets on colored interval graphs. In *International Conference on Current Trends in Theory and Practice of Informatics*, pp. 89–103. Springer, 2021.

Sander Borst, Golnoosh Shahkarami, and Rohit Vaish. Fair division meets scheduling: Approximately envy-free interval scheduling. *arXiv preprint arXiv:2608.15159*, 2026.

Steve J Chapin, Walfredo Cirne, Dror G Feitelson, James Patton Jones, Scott T Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In *Workshop on job scheduling strategies for parallel processing*, pp. 67–90. Springer, 1999.

Sarfaraz Equbal, Rohit Gurjar, Ayumi Igarashi, Yatharth Kumar, Pasin Manurangsi, Swaprava Nath, Raghuvansh Saxena, Rohit Vaish, and Hirotaka Yoneda. Fair allocation under conflict constraints. *arXiv preprint arXiv:2605.09930*, 2026.

Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *8th USENIX symposium on networked systems design and implementation (NSDI 11)*, 2011.

Halvard Hummel and Magnus Lie Hetland. Fair allocation of conflicting items. *Autonomous Agents and Multi-Agent Systems*, 36(1):8, 2022.

Antoon WJ Kolen, Jan Karel Lenstra, Christos H Papadimitriou, and Frits CR Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.

Bo Li, Minming Li, and Ruilong Zhang. Fair scheduling for time-dependent resources. *Advances in Neural Information Processing Systems*, 34:21744–21756, 2021.

Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. Themis: Fair and efficient {GPU} cluster scheduling. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pp. 289–304, 2020.

Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. {Heterogeneity-Aware} cluster scheduling policies for deep learning workloads. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 481–498, 2020.

OpenAI. ChatGPT, 2026a. URL https://chatgpt.com/overview/.

OpenAI. Codex, 2026b. URL https://chatgpt.com/codex/.

Tianming Zhao, Xiaomin Chang, Chunhao Li, Wei Li, Albert Zomaya, et al. Competitive fair scheduling with predictions. In *International Conference on Learning Representations*, volume 2025, pp. 66548–66585, 2025.

A Fairness in Individual Schedules: Algorithms and Proofs

This appendix gives the algorithms and proofs behind the deterministic guarantees summarized in Section 5. We first prove the tight two-group trade-off, then give an algorithm and lower bounds for general k. We conclude with the complexity of unrounded fairness and an exact dynamic program for prescribed quotas.

We use the integer-endpoint model of Section 3, with interval lengths between 1 and Δ. Throughout the approximation results, $\alpha \geq k$ is an integer and $q_g = \lfloor \textsc{Opt}_g/\alpha \rfloor$ denotes group g's quota. Approximation ratios are measured against the unrestricted optimum Opt. For a schedule S, write $S_g = S \cap \mathcal{I}_g$. All algorithms use a fixed ordering of the input requests to break ties.

Greedy completion of a fixed schedule. Several algorithms first choose a feasible set F that meets the quotas. To complete it, retain every request outside F that is compatible with all intervals in F, and apply earliest-finish-time greedy to the remaining requests. The union with F is a maximum-cardinality feasible schedule containing F: every feasible extension draws its additional requests from this filtered set, and greedy finds a largest compatible subset of that set. Completion therefore preserves all lower quotas.

A.1 A Matching Lower Bound for Two Groups

The loss in Theorem 5.1 is imposed by the quotas themselves. The following construction applies even to an algorithm that can compute a largest fair schedule exactly.

Proposition A.1 (Two-group lower bound)**.** *For every pair of integers $\alpha \geq 2$ and $\Delta \geq 1$, there exists a two-group instance for which*

$$\frac{\textsc{Opt}}{\textsc{Opt}_{\text{fair}}} = \frac{\alpha(\Delta-1)+1}{(\alpha-1)(\Delta-1)+1},$$

where $\textsc{Opt}_{\text{fair}}$ is the maximum cardinality of a floor-α-fair schedule. At $\alpha = 2$, this ratio is $2 - 1/\Delta$.

Proof. First suppose that $\Delta \geq 2$. Divide $[0, \alpha\Delta)$ into α blocks $[j\Delta, (j+1)\Delta)$, $j = 0, \ldots, \alpha-1$. Group A contains one interval equal to each block, so $\textsc{Opt}_A = \alpha$ and $q_A = 1$.

For group B, place the interval $[j\Delta - 1, j\Delta + 1)$ across each internal boundary $j\Delta$, and fill every remaining unit slot with a unit interval. These intervals are pairwise compatible. There are $\alpha - 1$ crossing intervals, $\Delta - 1$ unit intervals in each end block, and $\Delta - 2$ in each interior block. Hence $\textsc{Opt}_B = N := \alpha(\Delta-1)+1$ and $q_B = \Delta - 1$.

Each interval of A overlaps exactly Δ intervals of B. Consecutive blocks share one conflicting interval, and nonconsecutive blocks share none. Selecting $r \geq 1$ intervals of A therefore excludes at least $r\Delta - (r-1)$ intervals of B, so the resulting schedule has cardinality at most $r + N - (r\Delta - (r-1)) = N - (\Delta - 1) - (r-1)(\Delta - 2)$. Since $\Delta \geq 2$, selecting all of B is optimal and $\textsc{Opt} = N$. Any fair schedule must select at least one interval of A, so its cardinality is at most $N - (\Delta - 1) = (\alpha - 1)(\Delta - 1) + 1$. This bound is attained by selecting any one interval of A and all compatible intervals of B. The latter contribute $N - \Delta = (\alpha-1)(\Delta-1) \geq q_B$ intervals, so the schedule is fair.

For $\Delta = 1$, give each group a copy of every unit interval $[j, j+1)$, $j = 0, \ldots, \alpha - 1$. Both quotas equal 1. A schedule of size α can meet both quotas, and no larger schedule is feasible. The ratio is therefore 1, as claimed. □

A.2 Augmented Two-Cut: The Case $\alpha = 2$

We begin with two groups and $\alpha = 2$. Proposition A.1 shows that floor-2-fairness can force a throughput loss of $2 - 1/\Delta$ relative to the unrestricted optimum. We match this bound with the *Augmented Two-Cut Algorithm*. The algorithm constructs a fair schedule by joining prefixes and suffixes of optimal schedules, retains a subset that meets both quotas, and then completes this subset using earliest-finish-time greedy.

Let the groups be A and B, with $q_g = \lfloor \textsc{Opt}_g/2 \rfloor$ for $g \in \{A, B\}$.

Compute an unrestricted optimal schedule O using earliest-finish-time greedy. If O satisfies both quotas, return O. Otherwise, exactly one group is deficient. The following lemma establishes this fact for any maximum-cardinality schedule; the proof for overlapping cuts will additionally use the greedy choice of O.

Lemma A.2 (At most one deficient group)**.** *For two groups and any integer $\alpha \geq 2$, at most one group violates its floor-α quota in a maximum-cardinality schedule.*

Proof. Let O be a maximum-cardinality schedule and set $q_g = \lfloor \textsc{Opt}_g/\alpha \rfloor$. Since a feasible schedule for either group is also feasible for the full instance, $|O| = \textsc{Opt} \geq \max\{\textsc{Opt}_A, \textsc{Opt}_B\} \geq 2\max\{q_A, q_B\}$. If both groups were deficient, then $|O| = |O_A| + |O_B| \leq q_A + q_B - 2 \leq 2\max\{q_A, q_B\} - 2$, a contradiction. □

Assume without loss of generality that A is deficient, and write $a = |O_A| < q_A$ and $b = |O_B| \geq q_B$. Compute an individual optimal schedule P_A for group A using earliest-finish-time greedy. Order its intervals chronologically as $A_1, \ldots, A_m$, where $m = \textsc{Opt}_A \geq 2q_A$. Similarly, write $O_B = (B_1, \ldots, B_b)$ in chronological order.

Two-cut construction. Define $t_L = \ell(B_{b-q_B+1})$ and $t_R = r(B_{q_B})$, and consider the two candidate schedules

$$\begin{aligned} S^L &= \{I \in P_A : r(I) \leq t_L\} \cup \{I \in O : \ell(I) \geq t_L\}, \\ S^R &= \{I \in O : r(I) \leq t_R\} \cup \{I \in P_A : \ell(I) \geq t_R\}. \end{aligned} \tag{4}$$

The first candidate retains the last q_B intervals of O_B, whereas the second retains its first q_B intervals. Among the candidates satisfying both quotas, choose one of maximum cardinality and denote it by S. We next show that such a candidate always exists.

Lemma A.3 (Two-cut feasibility and quota preservation)**.** *If A is deficient in O, both cut points are well-defined. Moreover, S^L and S^R are feasible and satisfy group B's quota.*

Proof. Since A is deficient, $q_A \geq 1$ and $a \leq q_A - 1$. Hence $b = \textsc{Opt} - a \geq \textsc{Opt}_A - a \geq 2q_A - (q_A - 1) = q_A + 1 \geq 2$. Since $\textsc{Opt}_B \geq b \geq 2$, we have $q_B \geq 1$. Together with $b \geq q_B$, this implies $q_B, b - q_B + 1 \in \{1, \ldots, b\}$, so both cut points exist.

Each part of S^L is feasible. Every interval in its first part ends by t_L, and every interval in its second part starts at or after t_L. Thus, their union is feasible. The same argument applies to S^R.

By construction, S^L contains $B_{b-q_B+1}, \ldots, B_b$, and S^R contains $B_1, \ldots, B_{q_B}$. Both candidates therefore satisfy group B's quota. □

To prove that at least one candidate also satisfies group A's quota, we distinguish separated and overlapping cuts.

Lemma A.4 (Separated cuts)**.** *If A is deficient in O and $b \geq 2q_B$, at least one of S^L, S^R satisfies group A's quota.*

Proof. The condition $b \geq 2q_B$ gives $b - q_B + 1 \geq q_B + 1$. Since the intervals of O_B are compatible, $t_R = r(B_{q_B}) \leq \ell(B_{b-q_B+1}) = t_L$. If $r(A_{q_A}) \leq t_L$, then $A_1, \ldots, A_{q_A}$ belong to S^L, so $|S^L_A| \geq q_A$. Otherwise, feasibility of P_A gives $t_R \leq t_L < r(A_{q_A}) \leq \ell(A_{q_A+1})$. Thus, $A_{q_A+1}, \ldots, A_{2q_A}$ belong to S^R, and $|S^R_A| \geq q_A$. □

For overlapping cuts, we use the standard stays-ahead property of earliest-finish-time greedy: its jth selected interval ends no later than the jth interval of any feasible schedule containing at least j intervals.

Lemma A.5 (Overlapping cuts)**.** *If A is deficient in the earliest-finish-time greedy schedule O and $q_B \leq b < 2q_B$, at least one of S^L, S^R satisfies group A's quota.*

Proof. Let $s = b - q_B + 1$. Since $s \leq q_B$, $t_L = \ell(B_s) \leq \ell(B_{q_B}) < r(B_{q_B}) = t_R$. Set $u = |\{I \in O_A : r(I) \leq t_L\}|$ and $v = |\{I \in O_A : \ell(I) \geq t_R\}|$. No interval of O strictly crosses either cut: such an interval would overlap B_s or B_{q_B}. Consequently, the window $[t_L, t_R)$ contains

Algorithm 2 Augmented Two-Cut Algorithm

Input: An interval instance $\mathcal{I}$ with two groups.
Output: A floor-2-fair schedule.
Compute O using earliest-finish-time greedy on $\mathcal{I}$.
for each group $g \in \{A, B\}$ **do**
    Compute P_g using earliest-finish-time greedy on $\mathcal{I}_g$.
    Set $q_g \leftarrow \lfloor |P_g|/2 \rfloor$.
if $|O \cap \mathcal{I}_g| \geq q_g$ for both groups **then**
    return O.
Relabel the groups so that A is deficient and B is the other group.
Order O_B chronologically as $B_1, \ldots, B_b$.
Set $t_L \leftarrow \ell(B_{b-q_B+1})$ and $t_R \leftarrow r(B_{q_B})$.
Construct S^L, S^R according to equation 4.
Choose a largest floor-2-fair candidate $S \in \{S^L, S^R\}$.
Let F_A contain the first q_A intervals of S_A chronologically.
Set $F \leftarrow F_A \cup S_B$.
Set $\mathcal{R} \leftarrow \{I \in \mathcal{I} \setminus F : I \text{ is compatible with every interval of } F\}$.
Compute T using earliest-finish-time greedy on $\mathcal{R}$.
return $F \cup T$.

$a - u - v$ intervals of O_A and $2q_B - b$ intervals of O_B; let $C = a - u - v + 2q_B - b$ be their total number. In particular, $C \geq 1$.

Suppose both candidates violate group A's quota. Set $x_L = |\{I \in P_A : r(I) \leq t_L\}|$ and $x_R = |\{I \in P_A : \ell(I) \geq t_R\}|$. Then $x_L + a - u \leq q_A - 1$ and $x_R + a - v \leq q_A - 1$. Since $m \geq 2q_A$, this gives $m - x_L - x_R \geq 2a - u - v + 2$. Also, $a + b = \textsc{Opt} \geq \textsc{Opt}_B \geq 2q_B$, so $a \geq 2q_B - b$ and therefore $m - x_L - x_R \geq C + 2$.

These $m - x_L - x_R$ intervals of P_A all end after t_L and start before t_R. Since they are pairwise compatible, at most one starts at or before t_L, and at most one ends at or after t_R. Discarding these at most two intervals leaves at least C intervals strictly inside (t_L, t_R). Choose exactly C of them and denote this set by Q.

Let N be the number of intervals of O ending by t_L. Its first N intervals, followed by Q, form a feasible schedule of $N + C$ intervals whose last interval ends strictly before t_R. However, the $(N + C)$th interval of O is B_{q_B} and ends at t_R. This contradicts the stays-ahead property of earliest-finish-time greedy. Thus, at least one candidate satisfies group A's quota. □

Lemma A.6 (Guarantee of the two-cut construction)**.** *For two groups, the two-cut construction returns a feasible floor-2-fair schedule S with $|S| \geq \textsc{Opt}/2$.*

Proof. If O satisfies both quotas, the algorithm returns O. Otherwise, Lemmas A.3 to A.5 ensure that a feasible candidate satisfies both quotas. Hence, $|S| \geq q_A + q_B$.

Since A is deficient, $a \leq q_A - 1$. Also, $b \leq \textsc{Opt}_B \leq 2q_B + 1$. Therefore,

$$\textsc{Opt} = a + b \leq (q_A - 1) + (2q_B + 1) = q_A + 2q_B \leq 2(q_A + q_B) \leq 2|S|.$$

□

Greedy augmentation. To obtain the tight guarantee, retain exactly q_A intervals from S_A and all intervals of S_B. Denote the retained intervals of A by F_A and set $F = F_A \cup S_B$. Any subset $F_A \subseteq S_A$ of size q_A suffices; Algorithm 2 uses the first q_A intervals in chronological order.

Let $\mathcal{R}$ contain every interval outside F that is compatible with all intervals of F. In particular, intervals removed from S_A remain eligible. Apply earliest-finish-time greedy to $\mathcal{R}$ to obtain T, and return $S' = F \cup T$. This is a maximum-cardinality feasible schedule containing F. Since S is one such schedule, $|S'| \geq |S|$.

Theorem A.7 (Tight approximation guarantee of augmented two-cut)**.** *For two groups, the augmented two-cut algorithm returns a feasible floor-2-fair schedule with approximation ratio at most $2 - 1/\Delta$. This guarantee matches Proposition A.1.*

Proof. If O satisfies both quotas, the algorithm returns an optimal schedule. Otherwise, assume that A is deficient. The fixed set $F = F_A \cup S_B$ is feasible and satisfies both quotas, and augmentation adds only compatible intervals. Thus, the output S' is feasible and floor-2-fair. Writing $M = |S'|$, we have $M \geq |S| \geq q_A + q_B$.

Let D contain all intervals of O that overlap an interval of F_A, including any intervals in $O \cap F_A$. By Lemma 3.6, each interval of F_A overlaps at most Δ intervals of O. Hence, $|D| \leq \Delta q_A$.

The schedule $F_A \cup (O \setminus D)$ is feasible and contains F: indeed, $S_B \subseteq O_B$, and no interval of S_B overlaps F_A. Since S' has maximum cardinality among feasible schedules containing F, we have $M \geq q_A + \textsc{Opt} - |D| \geq \textsc{Opt} - (\Delta - 1)q_A$. Also, $\textsc{Opt} = a + b \leq (q_A - 1) + (2q_B + 1) = q_A + 2q_B \leq 2M - q_A$, so $q_A \leq 2M - \textsc{Opt}$. Combining the two bounds, $\textsc{Opt} \leq M + (\Delta - 1)q_A \leq M + (\Delta - 1)(2M - \textsc{Opt})$. Rearranging yields $\Delta\textsc{Opt} \leq (2\Delta - 1)M$, and thus $\textsc{Opt}/|S'| \leq 2 - 1/\Delta$. Tightness follows from Proposition A.1. □

The algorithm runs in $O(n \log n)$ time. After sorting, the global and individual optimal schedules, both candidates, and the augmentation can be computed by linear scans. Compatibility with the chronologically ordered fixed set F can also be checked by a linear scan of the intervals in start-time order.

A.3 Minimum-Conflict Block Replacement: The Case $\alpha \geq 3$

We next consider integer $\alpha \geq 3$. The *Minimum-Conflict Block Replacement* algorithm (MCBR) also starts from an unrestricted optimal schedule. If one group is deficient, it considers consecutive blocks of that group's individual optimum, each large enough to meet its quota. MCBR inserts a block that conflicts with the fewest intervals of the unrestricted optimum and removes those conflicts. We show that this replacement preserves the other group's quota and attains the lower bound in Proposition A.1.

For $g \in \{A, B\}$, let $q_g = \lfloor \textsc{Opt}_g/\alpha \rfloor$. Compute an unrestricted optimal schedule O using earliest-finish-time greedy. If O satisfies both quotas, return O. Otherwise, exactly one group is deficient by Lemma A.2.

Candidate blocks and replacement. Assume that A is deficient. Let $O_A^\star = (I_1, \ldots, I_m)$ be an individual optimal schedule for A, in chronological order, with $m = \textsc{Opt}_A$. We have $q_A \geq 1$ and $\alpha q_A \leq m$. Partition its first αq_A intervals into α consecutive blocks $B_i = \{I_{(i-1)q_A+1}, \ldots, I_{iq_A}\}$, $i = 1, \ldots, \alpha$. The remaining $m - \alpha q_A$ intervals are not used in the candidate blocks. Let $L(B_i) = \ell(I_{(i-1)q_A+1})$ and $R(B_i) = r(I_{iq_A})$. The block spans $[L(B_i), R(B_i))$ are pairwise disjoint, with $R(B_i) \leq L(B_{i+1})$ for consecutive blocks.

For each block, let $C_i = \{J \in O : J \cap I \neq \varnothing \text{ for some } I \in B_i\}$. Thus, C_i contains exactly the intervals of O that conflict with the block; an interval lying entirely in a gap between block intervals is not counted. Choose $i^\star \in \arg\min_{1 \leq i \leq \alpha} |C_i|$ and return $S = B_{i^\star} \cup (O \setminus C_{i^\star})$. Ties may be broken arbitrarily. MCBR returns this replacement schedule directly.

Algorithm 3 Minimum-Conflict Block Replacement (MCBR)

Input: An interval instance with two groups and an integer $\alpha \geq 3$.
Output: A floor-α-fair schedule.
Compute an unrestricted optimal schedule O and individual optimal schedules $O_A^\star, O_B^\star$ using earliest-finish-time greedy.
Set $q_g \leftarrow \lfloor |O_g^\star|/\alpha \rfloor$ for $g \in \{A, B\}$.
if $|O \cap \mathcal{I}_g| \geq q_g$ for both groups **then**
    return O.
Let A be the deficient group and B the other group.
Partition the first αq_A intervals of $O_A^\star$ into consecutive blocks $B_1, \ldots, B_\alpha$ of size q_A.
for $i = 1, \ldots, \alpha$ **do**
    Set $C_i \leftarrow \{J \in O : J \cap I \neq \varnothing \text{ for some } I \in B_i\}$.
Choose $i^\star \in \arg\min_{1 \leq i \leq \alpha} |C_i|$.
return $B_{i^\star} \cup (O \setminus C_{i^\star})$.

Lemma A.8 (Conflict bound)**.** *The candidate blocks satisfy*

$$\sum_{i=1}^{\alpha} |C_i| \leq \textsc{Opt} + \alpha - 1.$$

Consequently,

$$|C_{i^\star}| \leq \left\lceil \frac{\textsc{Opt}}{\alpha} \right\rceil \leq \left\lfloor \frac{\textsc{Opt}}{\alpha} \right\rfloor + 1.$$

Proof. For $i = 1, \ldots, \alpha - 1$, let $b_i = R(B_i)$. Consider the sets $C_1, \ldots, C_\alpha$ in order. At most $\textsc{Opt}$ intervals of O can be counted for the first time.

Fix $i \geq 2$ and an interval $J \in C_i$ that also belongs to C_j for some $j < i$. Overlap with B_j gives $\ell(J) < R(B_j) \leq R(B_{i-1}) = b_{i-1}$, while overlap with B_i gives $r(J) > L(B_i) \geq R(B_{i-1}) = b_{i-1}$. Thus, J strictly crosses b_{i-1}. Since O is feasible, at most one of its intervals can strictly cross this point. Thus, for each $i \geq 2$, at most one interval in C_i was counted earlier, proving $\sum_{i=1}^{\alpha} |C_i| \leq \textsc{Opt} + \alpha - 1$. By the choice of $i^\star$ and integrality, $|C_{i^\star}| \leq \lfloor (\textsc{Opt} + \alpha - 1)/\alpha \rfloor = \lceil \textsc{Opt}/\alpha \rceil \leq \lfloor \textsc{Opt}/\alpha \rfloor + 1$. □

Lemma A.9 (Floor-α-fairness)**.** *For every integer $\alpha \geq 3$, MCBR returns a feasible floor-α-fair schedule.*

Proof. If O satisfies both quotas, the claim is immediate. Otherwise, assume that A is deficient. Removing $C_{i^\star}$ eliminates every conflict between O and $B_{i^\star}$, so $S = B_{i^\star} \cup (O \setminus C_{i^\star})$ is feasible. The inserted block contains q_A intervals of A and therefore satisfies its quota.

To check group B's quota, set $u = \lfloor \textsc{Opt}/\alpha \rfloor$. Since $\textsc{Opt}_g \leq \textsc{Opt}$, we have $q_A, q_B \leq u$. Deficiency of A gives $|O_A| \leq q_A - 1 \leq u - 1$, and Lemma A.8 gives $|C_{i^\star}| \leq u + 1$. Therefore

$$|S_B| \geq \textsc{Opt} - |O_A| - |C_{i^\star}| \geq \textsc{Opt} - 2u \geq u \geq q_B.$$

The last line uses $\textsc{Opt} \geq \alpha u \geq 3u$. Thus, both quotas are satisfied. □

Theorem A.10 (Tight approximation guarantee of MCBR)**.** *For integers $\alpha \geq 3$ and $\Delta \geq 1$, MCBR returns a feasible floor-α-fair schedule for two groups with approximation ratio at most*

$$\frac{\alpha(\Delta - 1) + 1}{(\alpha - 1)(\Delta - 1) + 1}.$$

This guarantee matches the lower bound in Proposition A.1.

Proof. Feasibility and fairness follow from Lemma A.9. If O satisfies both quotas, MCBR returns an optimal schedule. Otherwise, $q_A \geq 1$. Every interval in $O \cap B_{i^\star}$ belongs to $C_{i^\star}$, so $B_{i^\star}$ and $O \setminus C_{i^\star}$ are disjoint. Hence $|S| = \textsc{Opt} - |C_{i^\star}| + q_A$. The conflict lemma and the integer-endpoint overlap bound give, respectively, $|C_{i^\star}| \leq (\textsc{Opt} + \alpha - 1)/\alpha$ and $|C_{i^\star}| \leq \Delta q_A$. The second inequality follows because each of the q_A inserted intervals overlaps at most Δ intervals of O, by Lemma 3.6.

If $\Delta = 1$, the second inequality implies $|S| \geq \textsc{Opt}$, and therefore $|S| = \textsc{Opt}$. Now suppose that $\Delta > 1$. We have $\textsc{Opt} - |S| = |C_{i^\star}| - q_A \leq (\Delta - 1) q_A$. Combining the conflict bounds yields

$$\begin{aligned}
|S| &= \textsc{Opt} - |C_{i^\star}| + q_A \\
&\geq \textsc{Opt} - \frac{\textsc{Opt} + \alpha - 1}{\alpha} + q_A \\
&= \frac{\alpha - 1}{\alpha} \textsc{Opt} + \frac{\alpha q_A - \alpha + 1}{\alpha} \\
&\geq \frac{\alpha - 1}{\alpha} \textsc{Opt} + \frac{q_A}{\alpha} \\
&\geq \frac{\alpha - 1}{\alpha} \textsc{Opt} + \frac{\textsc{Opt} - |S|}{\alpha(\Delta - 1)}.
\end{aligned}$$

The penultimate inequality follows from $(\alpha - 1)(q_A - 1) \geq 0$. Rearranging gives

$$|S| \geq \frac{(\alpha - 1)(\Delta - 1) + 1}{\alpha(\Delta - 1) + 1} \textsc{Opt}.$$

The claimed bound follows, and Proposition A.1 establishes tightness. □

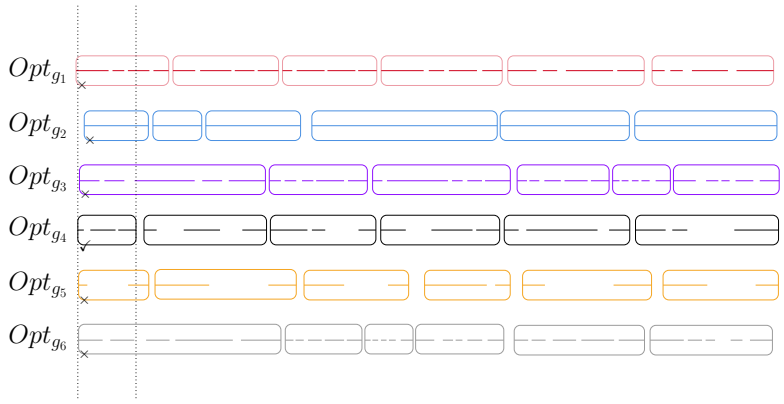


Figure 3: Example of the Block Partitioning Algorithm with $k = 6$.

Greedy augmentation can also be applied to the intervals outside MCBR's output that are compatible with it. This preserves fairness and may increase cardinality. Its worst-case approximation ratio remains the tight bound in Theorem A.10, because Proposition A.1 applies to every floor-α-fair schedule.

MCBR runs in $O(n \log n)$ time. After sorting and computing the optimal schedules, a sweep through the two compatible interval sequences O and $O^\star_A$ computes all candidate conflict sets in linear time. Repeated conflicts within a block are counted only once. The total number of overlap pairs between the two sequences is linear in their combined size, since each sweep step advances past at least one interval.

Proof of Theorem 5.1. At $\alpha = 2$, the asserted ratio equals $2 - 1/\Delta$ and is attained by Theorem A.7. For integer $\alpha \geq 3$, Theorem A.10 gives the same formula. Both algorithms run in $O(n \log n)$ time. Proposition A.1 supplies the matching lower bound for every integer $\alpha \geq 2$ and $\Delta \geq 1$. □

A.4 Block Partitioning for General k

The two-group algorithms exploit the fact that repairing one group leaves only one other quota to preserve. For general k, we instead build a feasible schedule that meets all positive quotas before maximizing its extension. Fix an integer fairness parameter $\alpha \geq k$. The *Block Partitioning Algorithm* (BPA) selects a block from each group with a positive quota and then completes the resulting schedule using earliest-finish-time greedy.

Compute an unrestricted optimal schedule O and an individual optimal schedule $O^\star_g$ for each group using earliest-finish-time greedy. Set $\textsc{Opt}_g = |O^\star_g|$ and $q_g = \lfloor \textsc{Opt}_g/\alpha \rfloor$. If O satisfies all quotas, return O. Otherwise, let $P = \{g \in G : q_g > 0\}$ and proceed as follows.

Block construction. For each $g \in P$, write $\textsc{Opt}_g = \alpha q_g + s_g$ with $0 \leq s_g < \alpha$. Partition $O^\star_g$ chronologically into α consecutive blocks. The first s_g blocks contain $q_g + 1$ intervals each, and the remaining blocks contain q_g intervals each. Every block is nonempty, and larger blocks precede smaller ones. For a block B, let its span be $[L(B), R(B))$, where $L(B) = \min_{I \in B} \ell(I)$ and $R(B) = \max_{I \in B} r(I)$.

Block selection and augmentation. Initialize $F = \varnothing$. Select a remaining block with minimum right endpoint $R(B)$ and add its intervals to F. Delete all blocks of its group and every remaining block whose span overlaps the selected span. Repeat until one block has been selected from each group in P. The proof below shows that a block is always available.

Let $\mathcal{R}$ be the requests in $\mathcal{I} \setminus F$ that are compatible with every interval in F. Apply earliest-finish-time greedy to $\mathcal{R}$ to obtain T, and return $S = F \cup T$. This is a maximum-cardinality feasible schedule containing F. Augmentation may use intervals from deleted blocks and from groups with zero quotas.

Algorithm 4 Block Partitioning Algorithm (BPA)

Input: An interval instance with k groups and an integer $\alpha \geq k$.
Output: A floor-α-fair schedule.
Compute an unrestricted optimal schedule O and all individual optimal schedules $O_g^\star$ using earliest-finish-time greedy.
Set $q_g \leftarrow \lfloor |O_g^\star|/\alpha \rfloor$ for every $g \in G$.
if $|O \cap \mathcal{I}_g| \geq q_g$ for every $g \in G$ **then**
    return O.
Set $H \leftarrow \{g \in G : q_g > 0\}$ and $F \leftarrow \varnothing$.
Partition each $O_g^\star$, $g \in H$, into α balanced consecutive blocks, with larger blocks first.
while $H \neq \varnothing$ **do**
    Select a remaining block $B^\star$ with minimum $R(B^\star)$; let $g^\star$ be its group.
    Set $F \leftarrow F \cup B^\star$ and $H \leftarrow H \setminus \{g^\star\}$.
    Delete all blocks of $g^\star$ and all remaining blocks whose spans overlap the span of $B^\star$.
Set $\mathcal{R} \leftarrow \{I \in \mathcal{I} \setminus F : I \text{ is compatible with every interval in } F\}$.
Compute T using earliest-finish-time greedy on $\mathcal{R}$.
return $F \cup T$.

Theorem A.11 (Guarantee of block partitioning)**.** *For all integers $k \geq 2$ and $\alpha \geq k$, BPA returns a feasible floor-α-fair schedule S with approximation ratio at most*

$$\min\left\{\Delta,\ \max\left\{\alpha,\ k-1+\frac{k(k-1)}{2\alpha}\right\}\right\}.$$

Proof. If O satisfies all quotas, BPA returns an optimal schedule. Otherwise, let $m = |P|$. At least one positive quota is violated, so $1 \leq m \leq k$.

Feasibility and fairness. A selected block $B^\star$ overlaps at most one remaining block span from each other group. Indeed, suppose that it overlaps the spans of two blocks C, D of the same group, with C preceding D. Then $R(C) \leq L(D) < R(B^\star)$, contradicting the choice of $B^\star$ as a block with minimum right endpoint.

Every group in P initially has α blocks. Before round t, each unprocessed group has lost at most $t-1$ blocks. Since $t \leq m \leq k \leq \alpha$, at least $\alpha - t + 1 \geq 1$ remain. Thus, every group in P selects a block. Selected spans are pairwise disjoint, and the block selected for group g contains at least q_g intervals. Groups outside P have zero quotas. Consequently, F is feasible and meets all quotas, and augmentation preserves both properties.

Cardinality bound. For $g \in P$, let t_g be its selection round and v_g the number of intervals in its selected block. Then $|F| = \sum_{g \in P} v_g$. If $v_g = q_g$, all s_g larger blocks of group g must have been deleted before round t_g: otherwise, an available larger block would end earlier than the selected block. Each preceding round deletes at most one block of group g, so $s_g \leq t_g - 1$ and $\textsc{Opt}_g = \alpha q_g + s_g \leq \alpha v_g + t_g - 1$. If $v_g = q_g + 1$, then $\textsc{Opt}_g = \alpha(v_g - 1) + s_g \leq \alpha v_g - 1 \leq \alpha v_g + t_g - 1$. Thus, in either case,

$$\textsc{Opt}_g \leq \alpha v_g + t_g - 1. \tag{5}$$

Let z be the group selected in the last round. Before that round, at least $\alpha - m + 1$ of its blocks remain. Their spans are pairwise disjoint and compatible with all previously selected spans. After selecting one, at least $\alpha - m$ other blocks can therefore be added to F, each contributing at least q_z intervals. Since S is a largest feasible schedule containing F,

$$|S| \geq |F| + (\alpha - m)q_z \geq m + (\alpha - m) = \alpha. \tag{6}$$

The second inequality uses $|F| \geq m$ and $q_z \geq 1$.

Write $O_g = O \cap \mathcal{I}_g$. Some group $h \in P$ is deficient in O, so $|O_h| \leq q_h - 1$ and $\textsc{Opt}_h - |O_h| \geq \alpha q_h - (q_h - 1) = (\alpha - 1)q_h + 1 \geq \alpha$. For every other group, $|O_g| \leq \textsc{Opt}_g$, and therefore

$\textsc{Opt} = \sum_{g\in G} |O_g| \le \sum_{g\in G} \textsc{Opt}_g - \alpha$. Each group outside P has $\textsc{Opt}_g \le \alpha - 1$. Applying equation 5 and summing the selection rounds gives

$$\begin{aligned}\textsc{Opt} &\le \sum_{g\in P} (\alpha v_g + t_g - 1) + (k-m)(\alpha-1) - \alpha \\ &= \alpha|F| + \frac{m(m-1)}{2} + (k-m)(\alpha-1) - \alpha.\end{aligned}$$

By equation 6,

$$\begin{aligned}\textsc{Opt} &\le \alpha|S| - \alpha(\alpha-m)q_z + \frac{m(m-1)}{2} + (k-m)(\alpha-1) - \alpha \\ &= \alpha|S| + \frac{m(m+1)}{2} - k - \alpha(\alpha-k+1) - \alpha(\alpha-m)(q_z-1) \\ &\le \alpha|S| + \frac{k(k-1)}{2} - \alpha(\alpha-k+1).\end{aligned}$$

The last inequality uses $m \le k \le \alpha$ and $q_z \ge 1$.

Set $D = k(k-1)/2 - \alpha(\alpha-k+1)$. If $D \le 0$, then $\textsc{Opt}/|S| \le \alpha$. If $D > 0$, use $|S| \ge \alpha$ to obtain

$$\frac{\textsc{Opt}}{|S|} \le \alpha + \frac{D}{|S|} \le \alpha + \frac{D}{\alpha} = k - 1 + \frac{k(k-1)}{2\alpha}.$$

Thus,

$$\frac{\textsc{Opt}}{|S|} \le \max\left\{\alpha,\ k-1+\frac{k(k-1)}{2\alpha}\right\}.$$

The Δ bound. The output S is maximal, since otherwise an additional interval would give a larger feasible schedule containing F. By Lemma 3.7, $\textsc{Opt} \le \Delta|S|$. Combining the bounds proves the theorem. □

Corollary A.12. *At $\alpha = k$, BPA returns a floor-k-fair schedule with approximation ratio at most $\min\{\Delta, 2\}$ for $k = 2$ and $\min\{\Delta, (3k-3)/2\}$ for $k \ge 3$.*

Proof. Substituting $\alpha = k$ into Theorem A.11 gives $\max\{k, (3k-3)/2\}$. This equals 2 for $k = 2$ and $(3k-3)/2$ for $k \ge 3$. □

Running time. For each group with a positive quota, $\alpha \le \textsc{Opt}_g$. The total number of blocks is therefore at most $\sum_{g\in P} \textsc{Opt}_g \le n$. Sort the blocks by their right endpoints and scan them in this order, skipping blocks of groups already selected and blocks whose spans overlap the last selected span. This implements block selection in $O(n \log n)$ time. The initial greedy computations and the final augmentation also take $O(n \log n)$ time, so BPA runs in $O(n \log n)$ time overall.

The bound in Theorem A.11 need not decrease with α. BPA recomputes its block partition when α changes, so the analysis does not establish monotonicity of the returned cardinality under weaker quotas.

A.5 Lower Bounds for General k

We next give lower bounds on the throughput loss forced by floor quotas for general k. For an integer $\alpha \ge k$, let $\textsc{Opt}_{\text{fair}}$ denote the maximum cardinality of a floor-α-fair schedule. The following constructions require $k-1$ groups to receive long intervals, each of which excludes intervals from the remaining group. The resulting bounds apply to every algorithm that returns a floor-α-fair schedule.

Theorem A.13 (Lower bound under floor-α-fairness)**.** *Let $k \ge 2$, $\alpha \ge k$, and $\Delta \ge 3$ be integers, and set $t = \alpha - k + 1$. There exists an instance such that*

$$\frac{\textsc{Opt}}{\textsc{Opt}_{\text{fair}}} \ge \max\{L_1, L_2, L_3, L_4\},$$

where

$$L_1 = \frac{\alpha\Delta}{k-1+t\Delta}, \qquad L_2 = \frac{\alpha\Delta - \lceil \alpha/2 \rceil}{k-1+t(\Delta-1)+\lfloor t/2 \rfloor},$$
$$L_3 = \frac{\alpha\Delta - \alpha - 1}{k-2+t(\Delta-1)}, \qquad L_4 = \frac{\alpha(\Delta-1)+1}{k-1+t(\Delta-1)}.$$

For $\Delta = 2$, the bound holds with $\max\{L_1, L_2, L_4\}$. For $\Delta = 1$, the lower bound is 1.

Proof. Partition $[0, \alpha\Delta)$ into blocks $B_j = [j\Delta, (j+1)\Delta)$, $j = 0, \ldots, \alpha - 1$.

Each of the $k-1$ sparse groups contains one interval equal to each block. Every sparse group therefore has individual optimum α and floor-α quota 1.

In the first three constructions, the remaining group H consists of pairwise compatible intervals. Each block contains exactly $M \geq 1$ intervals of H, and another $c < \alpha$ intervals cross block boundaries. Each crossing interval intersects exactly two adjacent blocks. Thus $\textsc{Opt}_H = \alpha M + c$ and $q_H = \lfloor \textsc{Opt}_H / \alpha \rfloor = M$.

Replacing a selected sparse interval by an interval of H contained in the same block preserves feasibility and cardinality. Replacing all selected sparse intervals therefore shows that $\textsc{Opt} = \textsc{Opt}_H$. For a fair schedule, instead retain one interval from each sparse group and replace only its additional intervals. This preserves all quotas and cardinality. Hence, some optimal fair schedule selects exactly one interval per sparse group, occupying $k-1$ distinct blocks and leaving $t = \alpha - k + 1$ blocks available to H.

For any choice of these t blocks, we can select all tM contained intervals and every crossing interval whose two blocks are both available. Let c_t be the maximum number of such crossing intervals over all choices of t blocks. Assigning the other $k-1$ blocks to distinct sparse groups gives a fair schedule because $tM \geq M = q_H$. Consequently, $\textsc{Opt} = \alpha M + c$ and $\textsc{Opt}_{\text{fair}} = k - 1 + tM + c_t$.

First construction ($\Delta \geq 1$). Partition each block into Δ unit intervals of H, with no crossing intervals. Then $M = \Delta$ and $c = c_t = 0$, giving $\textsc{Opt}/\textsc{Opt}_{\text{fair}} = \alpha\Delta/(k-1+t\Delta)$.

Second construction ($\Delta \geq 2$). For each odd $j \in \{1, \ldots, \alpha - 1\}$, place an interval $[j\Delta - 1, j\Delta + 1)$ of H across the boundary $j\Delta$. These intervals join $\lfloor \alpha/2 \rfloor$ disjoint pairs of adjacent blocks. Each block has at least $\Delta - 1$ free unit slots; place unit intervals of H in exactly $\Delta - 1$ of them. All intervals of H are pairwise compatible, with $M = \Delta - 1$ and $c = \lfloor \alpha/2 \rfloor$.

Among t available blocks, at most $\lfloor t/2 \rfloor$ pairs can be complete. This number is attained by choosing $\lfloor t/2 \rfloor$ complete pairs and, if t is odd, one additional block. Therefore, $c_t = \lfloor t/2 \rfloor$, and $\textsc{Opt}/\textsc{Opt}_{\text{fair}} = (\alpha\Delta - \lceil \alpha/2 \rceil)/(k-1+t(\Delta-1)+\lfloor t/2 \rfloor)$.

Third construction ($\Delta \geq 3$). Place an interval $[j\Delta - 1, j\Delta + 1)$ of H across every internal boundary, for $j = 1, \ldots, \alpha - 1$. Each block has at least $\Delta - 2$ free unit slots; place unit intervals of H in exactly $\Delta - 2$ of them. All intervals of H are pairwise compatible, with $M = \Delta - 2$ and $c = \alpha - 1$.

A set of t blocks contains at most $t-1$ adjacent pairs, attained by choosing t consecutive blocks. Thus, $c_t = t - 1$, and $\textsc{Opt}/\textsc{Opt}_{\text{fair}} = (\alpha\Delta - \alpha - 1)/(k - 2 + t(\Delta - 1))$.

Fourth construction ($\Delta \geq 2$). Place an interval $[j\Delta - 1, j\Delta + 1)$ of H across each internal boundary, for $j = 1, \ldots, \alpha - 1$, and fill every remaining unit slot with a unit interval of H. All intervals of H are pairwise compatible. Each end block contains $\Delta - 1$ unit intervals, and each interior block contains $\Delta - 2$. Therefore $\textsc{Opt}_H = N := \alpha(\Delta - 1) + 1$ and $q_H = \lfloor N/\alpha \rfloor = \Delta - 1$.

Every sparse interval conflicts with exactly Δ intervals of H. Adjacent blocks share one conflicting interval of H, whereas nonadjacent blocks share none. List the selected sparse intervals chronologically. The first excludes Δ intervals of H, and each subsequent interval excludes at least $\Delta - 1$ additional intervals. Relative to selecting all of H, the net cardinality loss is therefore at least $\Delta - 1$ for the first sparse interval and $\Delta - 2$ for each subsequent one.

Since $\Delta \geq 2$, selecting all intervals of H is optimal, so $\textsc{Opt} = N$. Every fair schedule selects at least $k-1$ sparse intervals and consequently has cardinality at most $N - (\Delta - 1) - (k-2)(\Delta - 2) = k - 1 + t(\Delta - 1)$.

To attain this bound, assign $k-1$ consecutive blocks to distinct sparse groups. These blocks exclude exactly $(k-1)\Delta-(k-2)$ intervals of H. Selecting all compatible intervals of H leaves $N-((k-1)\Delta-(k-2))=t(\Delta-1)\geq q_H$ intervals from H. This schedule satisfies all quotas and attains the upper bound, so $\textsc{Opt}/\textsc{Opt}_{\text{fair}}=(\alpha(\Delta-1)+1)/(k-1+t(\Delta-1))$.

Each applicable construction has integer endpoints, minimum interval length 1, and maximum interval length Δ. Choosing a construction with the largest ratio proves the bound. When $\Delta=1$, the first construction gives ratio 1. □

Setting $\alpha=k$ gives the following special case.

Corollary A.14 (Lower bound under floor-k-fairness)**.** *For integers $k\geq 2$ and $\Delta\geq 3$, there exists an instance such that*

$$\frac{\textsc{Opt}}{\textsc{Opt}_{\text{fair}}}\geq \max\left\{\frac{k\Delta}{k+\Delta-1},\frac{k\Delta-\lceil k/2\rceil}{k+\Delta-2},\frac{k\Delta-k-1}{k+\Delta-3}\right\},$$

where $\textsc{Opt}_{\text{fair}}$ is the maximum cardinality of a floor-k-fair schedule. For $\Delta=2$, the bound holds with the first two terms. For $\Delta=1$, the lower bound is 1.

Proof. Set $\alpha=k$ and $t=1$ in Theorem A.13. The first three terms give the stated bounds. The fourth has the same denominator as the second and a numerator no larger, since $\lfloor k/2\rfloor\geq 1$. □

Unlike the two-group characterization, these bounds need not match the guarantee of BPA. They quantify the loss forced by the quotas and do not establish the optimal trade-off for general k.

A.6 Feasibility of Unrounded Fairness

The preceding algorithms use floor quotas. We now return to unrounded fairness. Although exact k-fairness cannot be guaranteed on every instance by Observation 3.4, an exact k-fair schedule may exist on a particular instance. We call the corresponding decision problem Exact-k-Fair Interval Scheduling (Exact-k-FIS): given an instance with k groups, determine whether it admits a feasible schedule satisfying $|S_g|\geq\lceil\textsc{Opt}_g/k\rceil$ for every group g.

Theorem A.15 (NP-completeness of exact k-fairness)**.** Exact-k-FIS *is NP-complete when k is part of the input, even if $\Delta=2$ and each group contains at most three intervals.*

Proof. Given a schedule, we can check feasibility and count its intervals from each group in polynomial time. Earliest-finish-time greedy computes each $\textsc{Opt}_g$ in polynomial time, so all quota inequalities can also be checked in polynomial time. Thus, the problem belongs to NP.

For NP-hardness, we reduce from the Job Interval Scheduling Problem with at most three candidate intervals per job (JISP$_3$). This problem asks whether one candidate per job can be selected so that all selected intervals are compatible. It is NP-complete for half-open intervals of length 2 with integer endpoints; see the discussion of this restriction in Bhore et al. (2021).

Consider a JISP$_3$ instance with $m\geq 1$ jobs. Translate all intervals by a common integer, if necessary, to make their endpoints nonnegative. Create one group for each job, containing its candidate intervals. Let T be the largest right endpoint. Add two new groups, containing only $[T,T+1)$ and $[T+1,T+2)$, respectively. These intervals are compatible with each other and with every original interval.

The construction takes polynomial time and produces $k=m+2\geq 3$ groups, each containing at most three intervals. The minimum and maximum interval lengths are 1 and 2, respectively. For every group, $1\leq\textsc{Opt}_g\leq 3\leq k$, so $\lceil\textsc{Opt}_g/k\rceil=1$. An exact k-fair schedule must therefore include at least one interval from each group.

If the original instance has a feasible selection, adding the two new intervals gives an exact k-fair schedule. Conversely, from any exact k-fair schedule in the constructed instance, retain one selected interval from each original group and discard the rest. The resulting schedule is feasible and selects exactly one candidate per original job. This proves NP-hardness. □

A.7 Exact Optimization with Prescribed Quotas

For fixed k, throughput can be optimized exactly subject to prescribed group quotas. We adapt the dynamic program of Bhore et al. (2021) to allow different quotas and maximize cardinality. Given an integer quota vector $\mathbf{q} = (q_g)_{g \in G}$ with $0 \le q_g \le \textsc{Opt}_g$, the objective is to find a largest feasible schedule S satisfying $|S_g| \ge q_g$ for every group g. This includes unrounded α-fairness with $q_g = \lceil \textsc{Opt}_g/\alpha \rceil$ and floor-α-fairness with $q_g = \lfloor \textsc{Opt}_g/\alpha \rfloor$.

Sort the intervals by nondecreasing right endpoint as $I_1, \dots, I_n$, and define $p(i) = \max(\{j < i : r(I_j) \le \ell(I_i)\} \cup \{0\})$. For $i \in \{0, \dots, n\}$ and an integer requirement vector $\mathbf{a} = (a_g)_{g \in G}$ with $0 \le a_g \le q_g$, let $\mathrm{DP}[i, \mathbf{a}]$ be the maximum cardinality of a feasible schedule from $\{I_1, \dots, I_i\}$ satisfying every requirement a_g. An infeasible state has value $-\infty$. The base cases are

$$\mathrm{DP}[0, \mathbf{a}] = \begin{cases} 0, & \mathbf{a} = \mathbf{0}, \\ -\infty, & \mathbf{a} \neq \mathbf{0}. \end{cases}$$

If I_i belongs to group h, define the residual requirement vector $\mathbf{b}$ by

$$b_g = \begin{cases} \max\{0, a_h - 1\}, & g = h, \\ a_g, & g \neq h. \end{cases}$$

The recurrence is

$$\mathrm{DP}[i, \mathbf{a}] = \max\left\{\mathrm{DP}[i-1, \mathbf{a}], 1 + \mathrm{DP}[p(i), \mathbf{b}]\right\}, \tag{7}$$

where $1 + (-\infty) = -\infty$. The two terms correspond to omitting and selecting I_i, respectively.

Theorem A.16 (Exact quota-constrained optimization)**.** *For any integer quota vector* $\mathbf{q}$ *with* $0 \le q_g \le \textsc{Opt}_g$*, the dynamic program returns a maximum-cardinality feasible schedule satisfying all quotas, or determines that no such schedule exists.*

Proof. We prove the stated value of every state by induction on i. For $i = 0$, the empty schedule is the only available schedule and satisfies precisely the zero requirement vector.

Suppose the claim holds for all smaller indices. A feasible schedule for state $(i, \mathbf{a})$ either omits or contains I_i. If it omits I_i, the best cardinality is $\mathrm{DP}[i-1, \mathbf{a}]$ by the induction hypothesis.

If it contains I_i, every other selected interval must end by $\ell(I_i)$. Indeed, all intervals in the current prefix end no later than $r(I_i)$, so a compatible interval in this prefix cannot lie after I_i. The remaining selected intervals therefore belong to $\{I_1, \dots, I_{p(i)}\}$ and satisfy $\mathbf{b}$. Conversely, any feasible schedule for $(p(i), \mathbf{b})$ can be combined with I_i to satisfy $\mathbf{a}$. The best cardinality in this case is $1 + \mathrm{DP}[p(i), \mathbf{b}]$.

Taking the larger value proves Equation (7), including infeasible states. Thus, $\mathrm{DP}[n, \mathbf{q}] = -\infty$ exactly when the quotas are infeasible. Otherwise, backtracking through stored choices reconstructs a maximum-cardinality schedule satisfying them. □

There are $(n+1)\prod_{g \in G}(q_g + 1)$ states. Each transition reads two table entries and changes at most one requirement. With precomputed array strides, the residual state can be accessed in constant time, so each state takes constant time. Sorting and computing all predecessor indices take $O(n \log n)$ time. Thus the running time is $O(n \log n + n \prod_{g \in G}(q_g + 1))$ and the space usage is $O(n \prod_{g \in G}(q_g + 1))$. For fixed k, the running time is $O(n^{k+1})$.

B Experimental Details and Additional Results

This appendix supplements Section 6 with implementation details, deterministic comparisons, and additional parameter studies.

B.1 Experimental Setup

Workload construction. We use four cleaned real-world workload traces in Standard Workload Format (SWF) (Chapin et al., 1999). Each record with submission time $s_j \ge 0$ and runtime $p_j > 0$ becomes the interval $I_j = [s_j, s_j + p_j)$. We preserve the original time values and treat records with

identical endpoints as distinct requests. Our model uses one shared resource and does not incorporate processor requirements or queueing delays.

Table 1 summarizes the resulting instances. Each workload has minimum runtime $p_{\min} = 1$, so $\Delta = p_{\max}/p_{\min} = p_{\max}$.

Table 1: Workload-derived interval instances after filtering.

Workload	Number of intervals	Length ratio Δ
CTC-SP2-1996	77,205	71,998
LANL-CM5-1994	122,058	239,892
NASA-iPSC-1993	18,066	62,643
SDSC-SP2-1998	54,034	510,209

Group assignments. The *length-based* and *containment-based* assignments sort requests by length and containment count, respectively, and divide the resulting orderings into approximately equal-sized groups. Length ties are broken by job identifier, while containment-count ties follow source-record order. Containment uses inclusive endpoint comparisons, so distinct requests with identical endpoints count as containing one another.

The *log-length-based* assignment instead divides each workload's runtime range into k logarithmically spaced bins, allowing group sizes and standalone optima to vary substantially across groups. For $p_{\min} < p_{\max}$, the boundaries are $b_j = p_{\min}(p_{\max}/p_{\min})^{j/k}$ for $j = 0, \ldots, k$. Bin j contains lengths in $[b_{j-1}, b_j)$, except that the final bin includes $p_{\max}$. Bin sizes may differ, and some bins may be empty. If all runtimes are equal, all requests are assigned to the first bin. All three assignments construct groups from interval characteristics rather than from the original users or accounts.

Algorithms and parameter ranges. We compare GPSA and BPA with Global Greedy at $\alpha = k$ for $k \in \{2, \ldots, 10\}$, and evaluate GPSA and BPA at $\alpha \in \{k, 2k, 3k, 4k, 5k\}$. The intervals and group labels remain fixed throughout each α sweep.

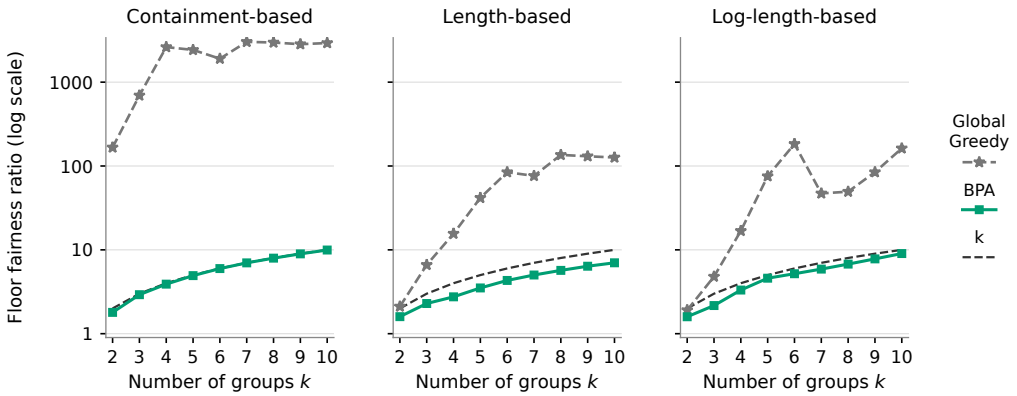
At $k = \alpha = 2$, we compare BPA with Augmented Two-Cut (Algorithm 2), denoted *TwoCut*. Its implementation includes quota retention and greedy completion, so both algorithms enforce the same floor-2 quotas. For $k = 2$ and $\alpha \in \{3, 4, 5, 6, 7\}$, we compare MCBR, BPA, and Global Greedy. MCBR returns its block-replacement schedule without additional greedy augmentation.

Randomized evaluation and unrounded fairness. GPSA and UGS each use $R = 100{,}000$ runs per setting. For GPSA, the same random draws are reused across values of α. We use $\widehat{\rho}$ and $\widehat{C}$ as defined in Section 6. Total and per-group accepted counts are averaged over runs before computing the metrics: $\widehat{\rho}$ uses the mean accepted count, while $\widehat{C}$ takes the minimum normalized mean service across groups. For deterministic algorithms, we use the returned counts.

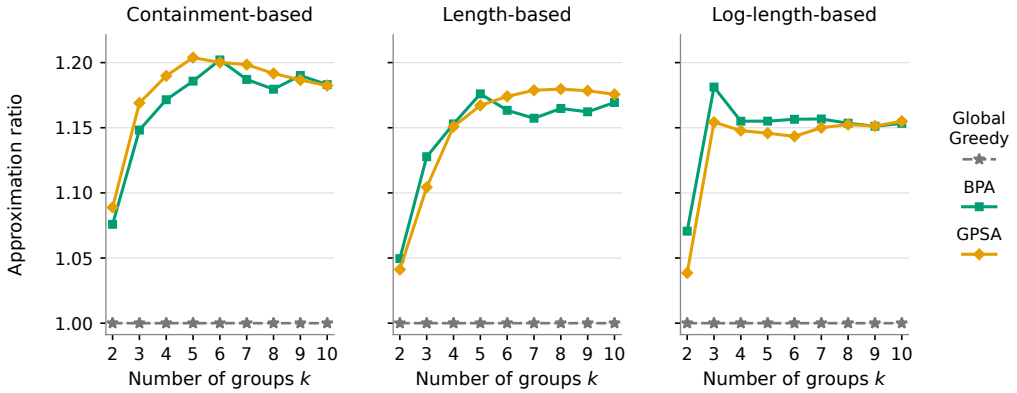
We also report the unrounded fairness ratio $\widehat{F} = 1/\widehat{C}$, with $\widehat{F} = \infty$ when $\widehat{C} = 0$. Lower values are better.

Floor fairness. For a deterministic schedule S, define $F_{\text{floor}}(S) = \max\{1, \max_{g \in G} \textsc{Opt}_g/(|S \cap \mathcal{I}_g| + 1)\}$. This is the infimum of the real parameters $a \geq 1$ for which S is floor-a-fair, since group g's quota is met exactly when $a > \textsc{Opt}_g/(|S \cap \mathcal{I}_g| + 1)$. The infimum need not itself be attained. Figures refer to this quantity as the *floor fairness ratio*; lower values are better. It measures the rounded requirement and should be interpreted separately from $\widehat{F}$.

Aggregation. We compute each metric separately for each workload and then average over the four workloads. Thus, the mean of $\widehat{F}$ is not generally the reciprocal of the mean of $\widehat{C}$, and the reciprocal of the mean of $\widehat{\rho}$ is not the mean normalized throughput.



(a) Floor-fairness scores of BPA and Global Greedy on a logarithmic scale.



(b) Approximation ratios of BPA, GPSA, and Global Greedy.

Figure 4: Comparison of BPA, GPSA, and Global Greedy at $\alpha = k$.

B.2 Comparison with Deterministic Floor Fairness

Figure 4(a) shows that BPA has lower mean floor-fairness scores than Global Greedy across all tested assignments and group counts at $\alpha = k$. The difference is largest for containment-based groups: Global Greedy's mean score reaches the thousands at larger k, whereas BPA's remains below k.

The throughput comparison in Figure 4(b) depends on the group assignment. BPA generally has lower mean approximation ratios than GPSA for containment-based and length-based groups. Under the log-length-based assignment, their ordering varies with k, and their ratios are close at larger group counts.

BPA satisfies rounded quotas in every returned schedule, whereas GPSA satisfies unrounded quotas in expectation. These comparisons therefore do not show that either algorithm dominates under a common fairness requirement.

B.3 Effect of Relaxing Fairness

We vary $\alpha \in \{k, 2k, 3k, 4k, 5k\}$ while keeping the intervals and group labels fixed. Thus, comparisons across parameter curves hold k, the workload, and the group assignment fixed.

GPSA. The throughput curves appear in Figure 2. The corresponding fairness ratios in Figure 5 increase as the fairness requirement is relaxed. The change is most visible for containment-based and log-length-based groups at larger k. Length-based groups have lower mean fairness ratios at the same parameter values. Together, the two figures show the expected trade-off: increasing α improves throughput while reducing the worst group's normalized mean service.

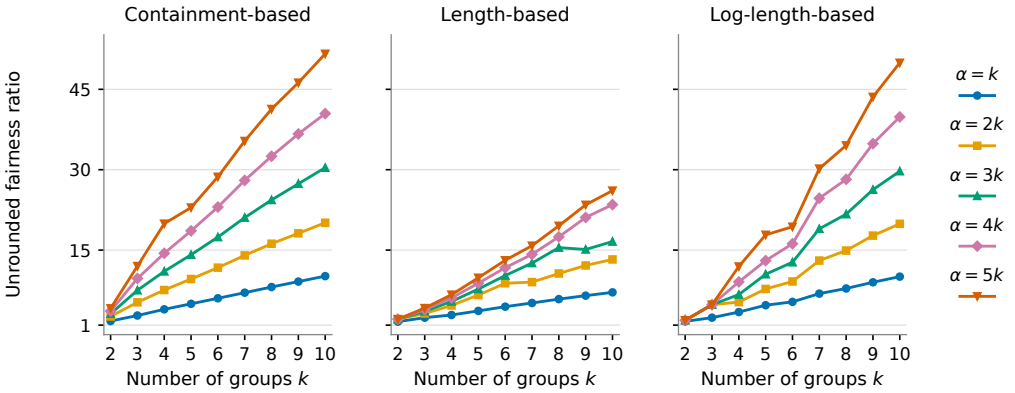
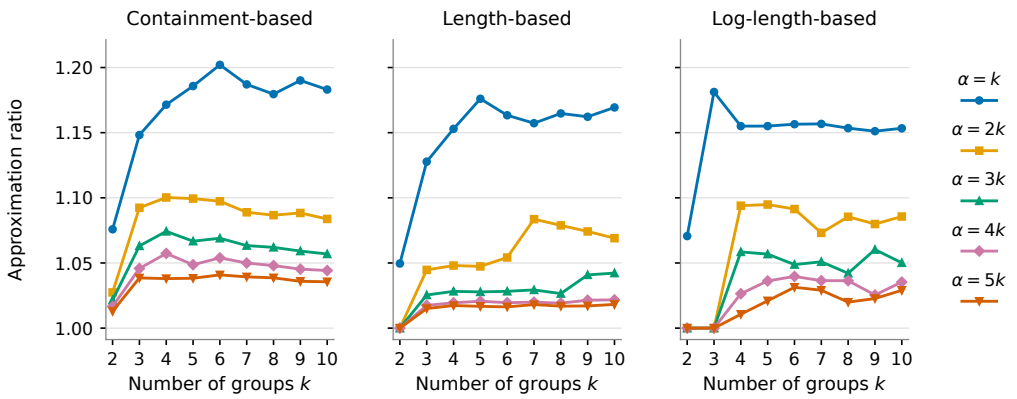
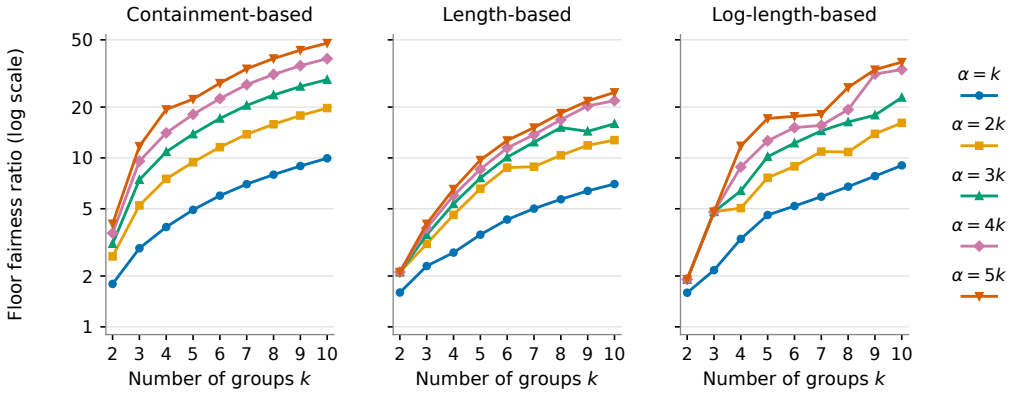


Figure 5: GPSA's unrounded fairness ratio $\widehat{F}$ as α varies, averaged over four workloads. The corresponding throughput curves appear in Figure 2.

BPA. Under the same parameter sweep, relaxing BPA's quotas generally reduces its mean approximation ratio (Figure 6(a)), with the largest reductions often occurring between $\alpha = k$ and $\alpha = 2k$. The corresponding increase in mean floor-fairness scores is especially pronounced for containment-based groups (Figure 6(b)). These are empirical trends: BPA recomputes its block partition as α changes, and the analysis does not guarantee monotone throughput on every instance.



(a) Approximation ratio $\widehat{\rho}$.



(b) Floor-fairness score on a logarithmic scale.

Figure 6: Approximation ratios and floor-fairness scores of BPA for different values of α/k.

B.4 Deterministic Comparisons for Two Groups

The strongest rounded requirement. At $k = \alpha = 2$, BPA and TwoCut satisfy the same floor-2 quotas. Table 2 shows that both have lower mean floor-fairness scores than Global Greedy while keeping their mean approximation ratios below 1.10 for all three assignments.

Neither algorithm is better on both metrics. For containment-based groups, TwoCut has the lower mean floor-fairness score (1.681 versus BPA's 1.795), while BPA has the lower mean approximation ratio (1.076 versus 1.091). Global Greedy's mean floor-fairness score is 166.408 in this setting. For length-based and log-length-based groups, TwoCut has lower mean approximation ratios, whereas BPA has lower mean floor-fairness scores. Thus, even under identical quotas, the algorithms produce different balances between throughput and group service.

Table 2: Approximation ratios and floor-fairness scores at $k = \alpha = 2$, averaged over four workloads.

Algorithm	Containment-based	Length-based	Log-length-based
Approximation ratio			
Global Greedy	1.000	1.000	1.000
BPA	1.076	1.050	1.071
TwoCut	1.091	1.039	1.055
Floor-fairness score			
Global Greedy	166.408	2.110	1.908
BPA	1.795	1.597	1.592
TwoCut	1.681	1.639	1.624

MCBR under relaxed quotas. Figure 7 compares MCBR, BPA, and Global Greedy for $k = 2$ and $\alpha \in \{3, 4, 5, 6, 7\}$. For containment-based groups, MCBR has a lower mean approximation ratio than BPA throughout the tested range. Both methods substantially improve the floor-fairness score relative to Global Greedy, while their throughput approaches the optimum as α increases.

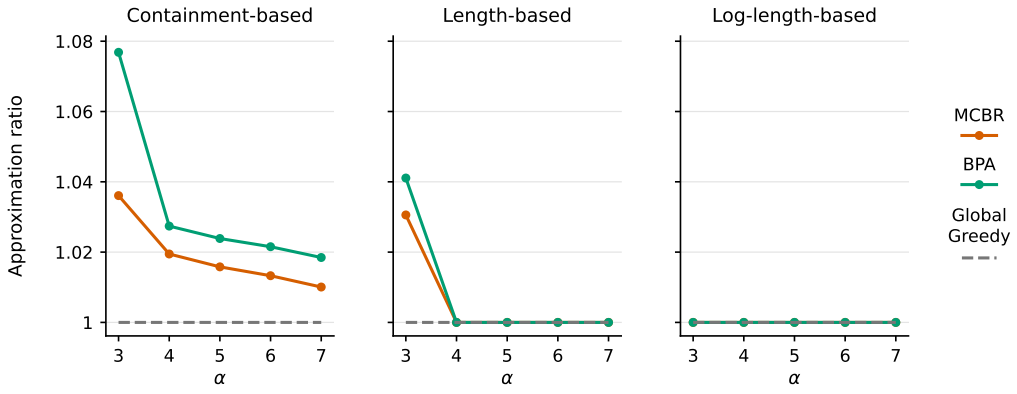
For length-based groups, all three methods have the same reported values for both metrics from $\alpha = 4$ onward. For log-length-based groups, they coincide throughout the tested range. The benefit of the fair algorithms is therefore most visible in assignments for which Global Greedy serves some groups poorly.

B.5 Results on Individual Workloads

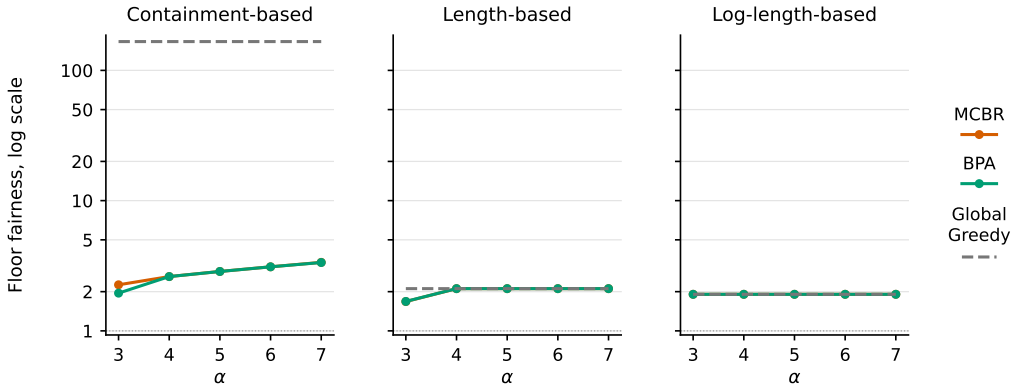
Figures 8 to 10 reports results separately for each workload.

CTC-SP2-1996 exhibits the largest throughput losses among the tested workloads, with approximation ratios approaching 1.4 for containment-based and length-based groups. On NASA-iPSC-1993 with length-based groups, BPA, GPSA, and Global Greedy coincide in both unrounded service and throughput, with approximation ratio one.

For containment-based groups, BPA substantially reduces the floor-fairness scores on CTC-SP2-1996, LANL-CM5-1994, and SDSC-SP2-1998 relative to Global Greedy. On SDSC-SP2-1998 with log-length-based groups, BPA's minimum normalized service approaches zero at $k = 9, 10$, illustrating the distinction between rounded quotas and unrounded service.



(a) Approximation ratios $\widehat{\rho}$.



(b) Floor-fairness scores on a logarithmic scale.

Figure 7: Approximation ratios and floor-fairness scores of MCBR, BPA, and Global Greedy at $k = 2$.

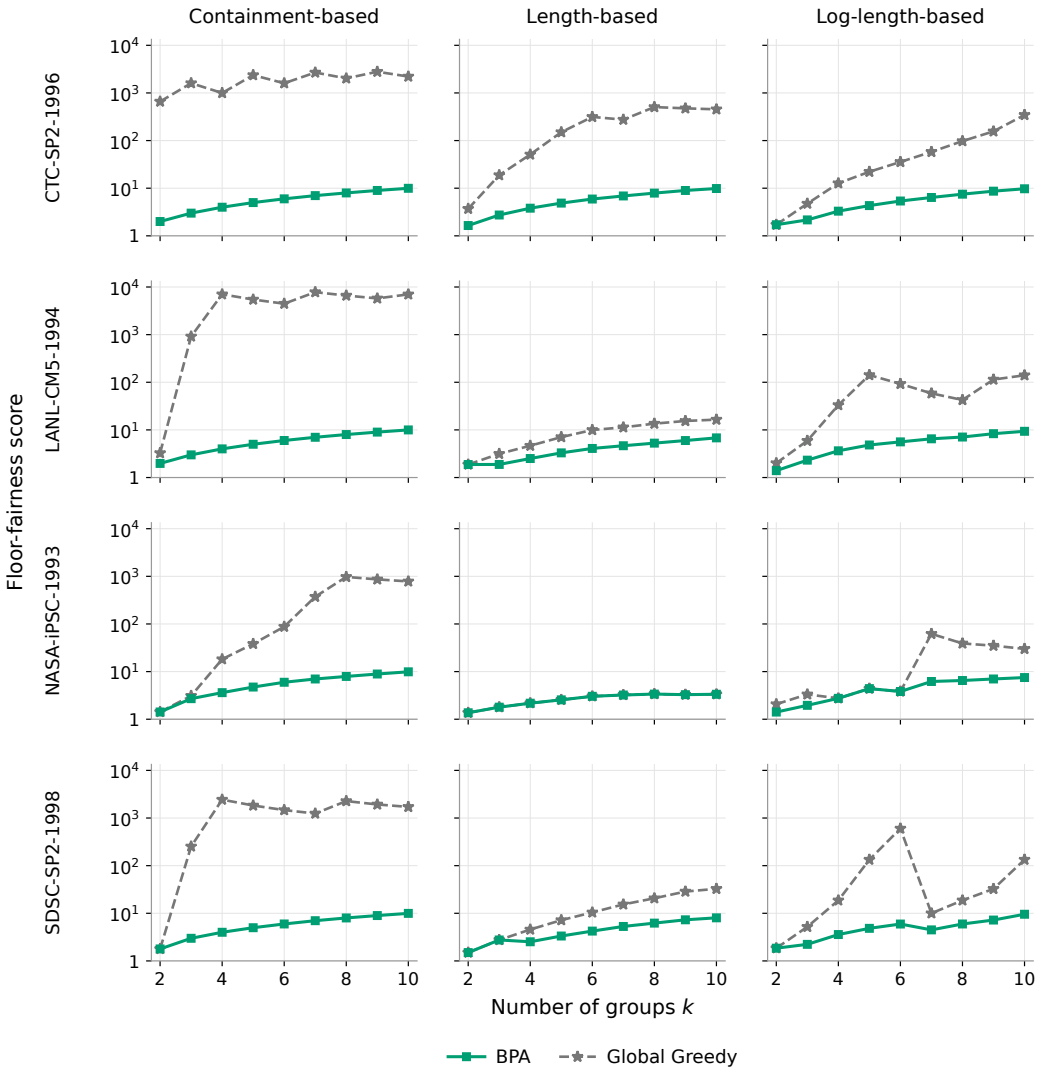


Figure 8: Per-workload floor-fairness scores at $\alpha = k$.

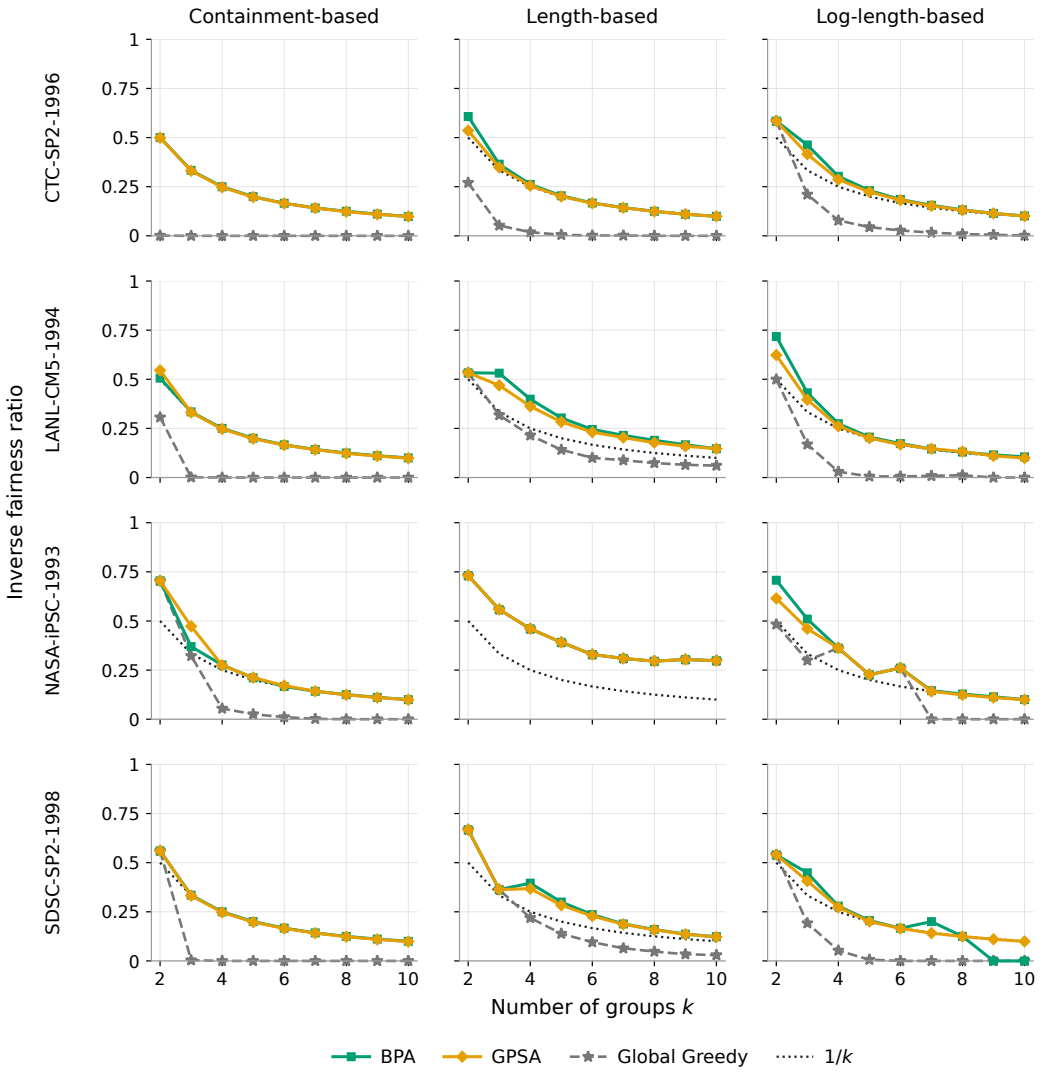


Figure 9: Per-workload inverse fairness ratio at $\alpha = k$.

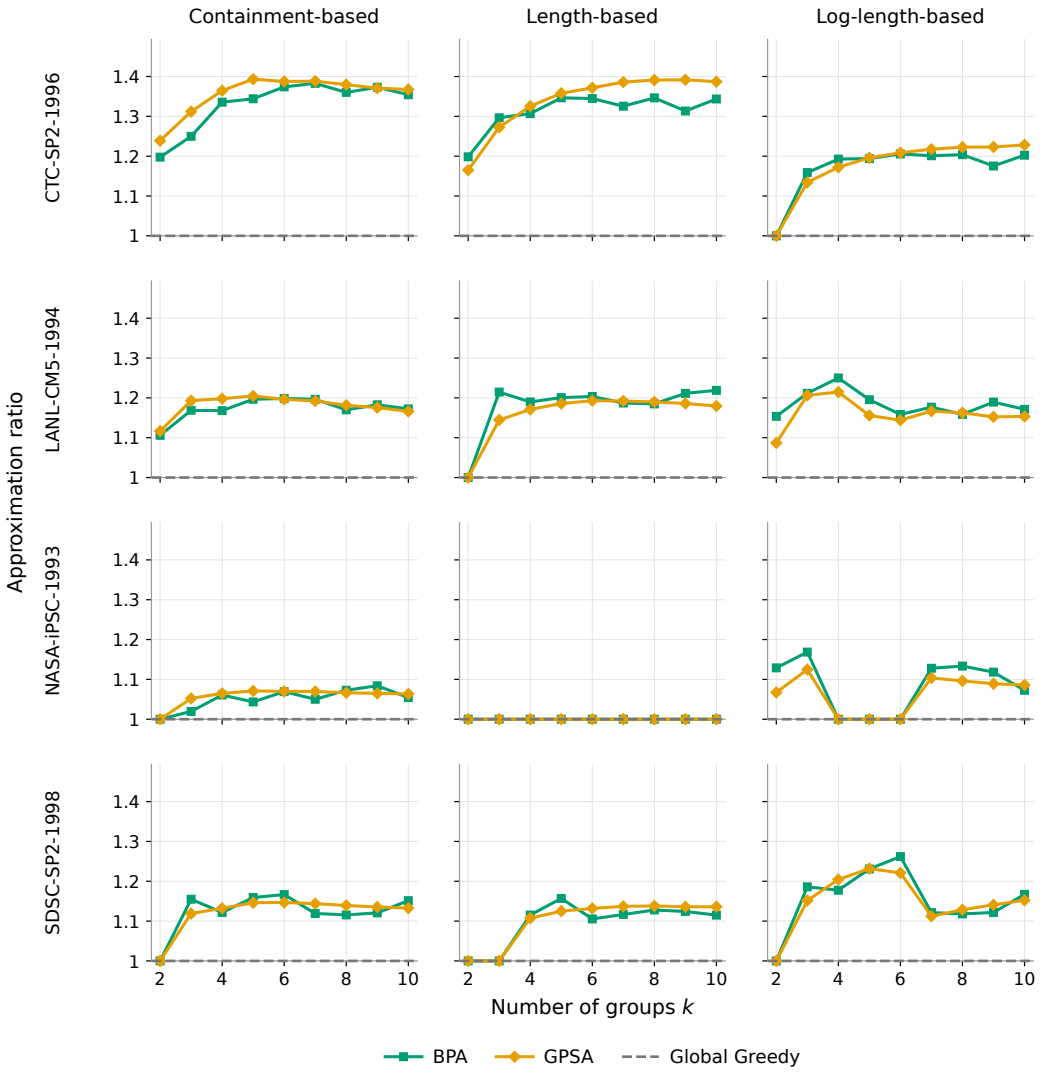


Figure 10: Per-workload approximation ratios at $\alpha = k$.