Efficient Algorithms and Hardness Results for the Weighted k-Server Problem

Anupam Gupta
Computer Science, Carnegie Mellon University, Pittsburgh, USA

Amit Kumar
Computer Science and Engineering Department, Indian Institute of Technology, Delhi.

Debmalya Panigrahi
Computer Science, Duke University, Durham, NC, USA

Abstract

In this paper, we study the weighted k-server problem on the uniform metric in both the offline and online settings. We start with the offline setting. In contrast to the (unweighted) k-server problem which has a polynomial-time solution using min-cost flows, there are strong computational lower bounds for the weighted k-server problem, even on the uniform metric. Specifically, we show that assuming the unique games conjecture, there are no polynomial-time algorithms with a sub-polynomial approximation factor, even if we use c-resource augmentation for $c < 2$. Furthermore, if we consider the natural LP relaxation of the problem, then obtaining a bounded integrality gap requires us to use at least ℓ resource augmentation, where ℓ is the number of distinct server weights. We complement these results by obtaining a constant-approximation algorithm via LP rounding, with a resource augmentation of $(2+\varepsilon)\ell$ for any constant $\varepsilon > 0$.

In the online setting, an $\exp(k)$ lower bound is known for the competitive ratio of any randomized algorithm for the weighted k-server problem on the uniform metric. In contrast, we show that 2ℓ-resource augmentation can bring the competitive ratio down by an exponential factor to only $O(\ell^2 \log \ell)$. Our online algorithm uses the two-stage approach of first obtaining a fractional solution using the online primal-dual framework, and then rounding it online.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online Algorithms, Weighted k-server, Integrality Gap, Hardness

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.12

Category APPROX

Funding *Anupam Gupta*: NSF awards CCF-1955785, CCF-2006953, and CCF-2224718
Debmalya Panigrahi: NSF awards CCF-1750140 (CAREER) and CCF-1955703

1 Introduction

The k-Server problem is a foundational problem in online algorithms and has been extensively studied over the last 30 years [10]. In this problem, there are a set of k servers that must serve requests arriving online at the vertices of an n-point metric space. The goal is to minimize the total movement cost of the servers. The k-Server problem was defined by Manasse et al. [22], who also showed a lower bound of k on the competitive ratio of any deterministic algorithm for this problem. Koutsoupias and Papadimitriou [20] gave a $(2k-1)$-compeititive algorithm for k-Server. There has been much progress in the recent past on obtaining randomized algorithms with polylogarithmic (in k and n) competitive ratio [2, 13, 21, 14]. The Weighted k-Server version of this problem, introduced by Fiat and Ricklin [17], allows the servers to have non-uniform positive weights; the cost of moving a server is now scaled by its weight. In this paper, we consider the Weighted k-Server problem on a uniform metric, namely when all n points of the metric space are at unit distance from each other, which means that the cost of moving a server between any two distinct points is simply the weight of the server. Note that the corresponding unweighted problem for the uniform metric is the extensively studied PAGING problem [10]. Indeed, one of the original motivations for studying the WEIGHTED k-SERVER problem came from a version of paging with non-uniform replacement costs for different cache slots [17]. In the rest of this paper, we will implicitly assume that the underlying metric space is a uniform metric.

The original paper of Fiat and Ricklin [17] introducing the WEIGHTED k-SERVER problem (on uniform metrics) gave a deterministic algorithm with a competitive ratio of about $2^{2^{2k}}$. They also showed a lower bound of $(k+1)!/2$ for deterministic algorithms. Chiplunkar and Viswanathan [15] improved this lower bound to $(k+1)!-1$, and gave a randomized algorithm that is 1.6^{2^k}-competitive against *adaptive* online adversaries; this also implies a deterministic competitive ratio of $2^{2^{k+1}}$ using the simulation technique of Ben-David et al. [8]. Bansal, Elias, and Koumutsos [6] showed that this competitive ratio is essentially tight for deterministic algorithms by showing a lower bound of $2^{2^{k-4}}$. They also gave a deterministic *work function algorithm* with a competitive ratio of $2^{2^{k+O(\log k)}}$. If the number of distinct server weights is ℓ and there are k_j servers of weight W_j, then the competitive ratio of their algorithm is $\exp(O(\ell k^3 \prod_{j=1}^{\ell}(k_j+1)))$, which is an exponential improvement over the general bound when ℓ is a constant. Unlike the k-SERVER and PAGING problems, it is unknown if randomization qualitatively improves the competitive ratio for WEIGHTED k-SERVER, although the best known lower bound for randomized algorithms against oblivious adversaries is only singly exponential in k [1] as against the doubly exponential lower bound for deterministic algorithms.

The above competitive ratios depend only on k, and are independent of the size n of metric space. Moreover, the hard instances are for metric spaces with the number of points n that are exponentially larger than the number of servers k. This is not a coincidence, since better algorithms exist for smaller values of n. Indeed, the WEIGHTED k-SERVER problem can be modeled as a metrical task system, where each state ω is a configuration (specifying the location of each of the k servers), and the distance between any two states ω, ω' is the cost to move between the configurations. Since there are $N = n^k$ states, one can obtain an n^k-competitive deterministic algorithm [11], and an $O(\text{poly}(k \log n))$-competitive randomized algorithm against *oblivious* adversaries [7, 3, 12, 16]; all these algorithms use $\text{poly}(n^k)$ time to explicitly maintain and manipulate the entire metric space, and hence are not efficient.

In this paper we ask: *is it possible to get efficient (randomized)* online *algorithms that have competitive ratios of the form* $\text{poly}(k \log n)$*, or even better? Is it possible to get such approximation ratios even in the* offline *setting?* We show that obtaining improved competitive or approximation ratios in polynomial time is possible, provided we allow for *resource augmentation* in the number of servers.

Resource augmentation in online algorithms has been widely studied in paging and scheduling settings (see e.g. [19, 23]). It is often a much needed assumption that allows for obtaining bounded or improved competitive ratios for such problems. Bansal et al. [5] studied the k-SERVER problem on trees under resource augmentation.

1.1 Our Results

Our first result establishes computational hardness of approximating the WEIGHTED k-SERVER problem in the offline setting. Unlike PAGING or k-SERVER, which are exactly solvable offline in polynomial time, we show that under the Unique Games conjecture, the WEIGHTED k-SERVER problem cannot be approximated to any subpolynomial factor even when we allow c-resource augmentation for any constant $c < 2$.

▶ **Theorem 1** (Hardness). *For any constant $\varepsilon > 0$, it is UG-hard to obtain an $N^{1/2-\varepsilon}$-approximation algorithm for* Weighted *k*-Server *with two weight classes, even when we are allowed c-resource augmentation for any constant $c < 2$. Here N represents the size of the input (including the request sequence length).*

Next, we show that the natural time indexed LP relaxation for Weighted k-Server (see LP) has a large integrality gap, unless we allow for a resource augmentation of almost ℓ, the number of distinct server weights.

▶ **Theorem 2** (Integrality Gap). *For any constant $\varepsilon > 0$, the integrality gap of the relaxation LP for* Weighted *k*-Server *is unbounded, even with $(\ell - \varepsilon)$-resource augmentation.*

It is worth noting that an optimal fractional solution to LP can be easily rounded to give an ℓ-approximation ratio with ℓ-resource augmentation. Indeed, we know that for each request, there exists a weight class which services this request to an extent of at least $1/\ell$. We can then scale this fractional solution by a factor ℓ and reduce this problem to ℓ instances of standard Paging problem. The integrality gap result shows that any rounding algorithm with bounded competitive ratio must incur almost ℓ-resource augmentation. We complement this integrality gap result with our main technical result, which gives an offline $O(1/\varepsilon)$-approximation with $(2 + \varepsilon)\ell$-resource augmentation, for any $\varepsilon \in (0, 1)$.

▶ **Theorem 3** (Offline Algorithm). *Let $\mathcal{I}$ be an instance of* Weighted *k*-Server *with k_j servers of weight W_j for all $j \in [\ell]$. For any $\varepsilon \in (0, 1)$, there is a polynomial time algorithm for $\mathcal{I}$ that uses at most $2(1 + \varepsilon)\ell \cdot k_j$ servers of weights W_j for each $j \in [\ell]$ and has server movement cost at most $O(1/\varepsilon)$ times the optimal cost of $\mathcal{I}$.*

Finally, we obtain an online algorithm for Weighted k-Server with 2ℓ-resource augmentation. The competitive ratio of the online algorithm is $O(\ell^2 \log \ell)$. (In constrast to the offline setting, it is no longer clear how to achieve an ℓ-competitive algorithm even if we augment the number of servers by a factor of ℓ.)

▶ **Theorem 4** (Online Algorithm). *Let $\mathcal{I}$ be an instance of* Weighted *k*-Server *with k_j servers of weight W_j for all $j \in [\ell]$. There is a randomized (polynomial time) online algorithm for $\mathcal{I}$ that uses at most $2\ell k_j$ servers of weights W_j for each $j \in [\ell]$ and has expected server movement cost at most $O(\ell^2 \log \ell)$ times the optimal cost of $\mathcal{I}$.*

Since $\ell \leq k$, the competitive ratio of the online algorithm is $O(k^2 \log k)$. This implies that an $O(\ell^2)$-resource augmentation achieves at least an exponential improvement in the competitive ratio of the Weighted k-Server problem. Moreover, by rounding the weights to powers of 2, we can assume that $\ell \leq O(\log W)$, where W is the aspect ratio of the server weights. Hence, the competitive ratio of the online algorithm is $O(\log^2 W \log \log W)$. Finally, note that for $\ell = O(1)$, the above theorem gives a $O(1)$-competitive online algorithm with $O(1)$-resource augmentation. This can be seen as a generalization of the classic result for the Paging problem that achieves a randomized competitive ratio of $O(\log \frac{k}{k-h+1})$ where the algorithm's cache has k slots while the adversary's has only $h < k$ slots [24].

1.2 Our Techniques

In this section, we give an overview of the main techniques in the paper. The UG hardness of Weighted k-Server is based on a reduction from the Vertex Cover problem. Given an instance of the vertex cover problem, the corresponding Weighted k-Server consists of one point in the uniform metric space for each vertex of the graph. The main observation is that if we know the minimum vertex cover size, we can keep one heavy weight server at each point corresponding to this vertex cover, which never change their positions. One can then generate an input sequence where the optimal solution pays a small cost, whereas an algorithm which does not cover an edge using heavy servers pays a much higher cost. The UG-hardness result for Vertex Cover translates to a corresponding resource augmentation lower bound for Weighted k-Server. Extending this approach to more than two weight classes (with stronger lower bounds on resource augmentation) turns out to be more challenging because the length of the input sequence becomes exponential in n. Instead, we show that the natural LP relaxation has a large integrality gap. The large gap instance consists of cycling through a sequence of subsets of the metric spaces with carefully varying frequency. The fractional solution is able to maintain a low cost by uniformly spreading servers over such cycles, but the integral solution is forced to service some of the cycles by small number of servers only.

Our main technical result shows how to round a solution to the LP relaxation. The relaxation has both covering and packing type constraints, and the rounding carefully addresses one set of constraints without violating the other. We first scale the LP by a factor of about 2ℓ, thus increasing both the resource augmentation and the cost. As a result, each request σ_t is covered to an extent of 2ℓ, and we can split this coverage across those weight classes which cover σ_t to an extent of at least 1. Now for a fixed weight class, we consider the requests which are covered by it to an extent of at least 1. We show how to integrally round this solution so that this coverage property is satisfied and yet, we do not violate any packing constraint. After this, we show that the packing constraints can be ignored. This allows to scale down the LP solution by a factor ℓ (which saves the cost by this factor) and uses total unimodularity of the constraint matrix to round it.

We extend our approximation algorithm to the online setting. The first step is to maintain an online fractional solution to the LP relaxation. Standard (weighted) paging algorithms for this problem rely on the fact that even the optimal offline algorithm needs to place a server at a requested location. But this turns out to be trickier here as we do not know the weight of the server which serves this location in the optimal solution. So we serve a request by ensuring that fractional mass from each weight classes is transferred at the same rate. The overall analysis proceeds by a careful accounting in the potential function. The online fractional solution satisfies the stronger guarantee that each request is served by servers of a particular weight class only. This allows us to reduce the rounding problem to independent instances of the Paging problem.

We now give an overview of the rest of the submission. In §2, we give details of the integrality gap construction; we defer the UG hardness proof to §A. The offline rounding of the LP relaxation is given in §3, and then we extend this algorithm to the online case in §4.

1.3 Preliminaries

In the Weighted k-Server problem on the uniform metric, we are given a set of n points $V = \{1, \ldots, n\}$, such that $d(v, v') = 1$ for each $v \neq v'$. There are k servers, labeled $1, \ldots, k$, with server i having weight $w_i \geq 0$. The input specifies a request sequence $(\sigma_1, \ldots, \sigma_T)$ of length T, with each request σ_t arriving at *time* t being a point in V. A solution $f : [k] \times \{0, \ldots, T\} \to V$ specifies the position of each server at each time $t \in [T]$ (where the initial positions $f(i, 0)$ are specified as part of the problem statement) such that for each time t there exists some server i_t such that $f(i_t, t) = \sigma_t$. The cost of the solution f is the total weighted distance travelled by the servers, i.e.,

$$1/2 \sum_{i=1}^{k} w_i \sum_{t=1}^{T} \mathbb{1}[f(i,t) \neq f(i,t-1)].$$

The goal is to find a solution with the minimum cost. We say that an instance has ℓ *weight classes* if the set $\{w_1, \ldots, w_k\}$ has cardinality ℓ. For an instance with ℓ different weight classes, we denote the distinct weights by $W_1, \ldots, W_\ell$, and let k_j denote the number of servers of weight W_j, with $\sum_j k_j = k$. For such an instance and a parameter $c \geq 1$, we say that the algorithm uses *c-resource augmentation* if it uses $\lfloor ck_j \rfloor$ servers of weight W_j for each $j = 1, \ldots, \ell$.

We now describe the natural LP relaxation for WEIGHTED k-SERVER. It has a variable $x(v,j,t)$ for each request time t, weight class $j \in [\ell]$ and vertex $v \in V$; it denotes the fractional mass of servers of weight W_j that are present at point v at time t. Let σ_t denote the vertex requested at time t. It is easy to verify that this is a valid relaxation.

$$\min 1/2 \sum_{j \in [\ell]} W_j \sum_t \sum_{v \in V} |x_{v,j,t} - x_{v,j,t-1}| \tag{LP}$$

$$\sum_{v \in V} x_{v,j,t} \leq k_j \qquad \forall t, j \in [\ell] \tag{1}$$

$$\sum_{j \in [\ell]} x_{\sigma_t,j,t} \geq 1 \qquad \forall t \tag{2}$$

$$x_{v,j,t} \geq 0 \qquad \forall t, v \in V, j \in [\ell]$$

2 An Integrality Gap for the Natural Linear Program

In this section, we show that the relaxation LP for WEIGHTED k-SERVER has a large integrality gap, unless we allow for a resource augmentation of almost ℓ, the number of distinct server weights.

Recall that the ℓ weights are denoted $W_1 \gg \cdots \gg W_\ell$, and there are k_j servers of weight W_j. Our theorem is the following:

▶ **Theorem 2** (Integrality Gap). *For any constant $\varepsilon > 0$, the integrality gap of the relaxation LP for WEIGHTED k-SERVER is unbounded, even with $(\ell - \varepsilon)$-resource augmentation.*

An Instance for Two Classes. To gain some intuition, we first consider the special case of $\ell = 2$, and prove the result without giving any resource augmentation. There are $n/2$ servers of weight W and $n/4$ servers of weight 1, thereby giving a total of $k = 3n/4$ servers. The input is given in "phases". Each phase is specified by a distinct subset S of V, where $|S| = n/2$. During the phase corresponding to a subset S, we cycle through all subsets S' of S with $|S'| = |S|/2 = n/4$. Given such a subset S' of S, we send requests which cycle through the points in S' for L times, where L is large enough.

One fractional solution for such a sequence is defined as follows: we assign $1/2$ unit of weight-W server at each of the n locations. During the phase for a subset S, we assign $1/2$ unit of server of unit weight at each of the locations in S. The cost of the fractional solution is at most $Z := \binom{n}{n/2} \cdot n/4$ (not accounting for the initial movement of the servers). However, an integral solution either moves at least one heavy server, or else pays at least L during one of the phases, thereby must pay at least $\min(W, L)$. Assuming $W, L \gg Z$ gives an arbitrarily large integrality gap. (We can account for the initial movement of the fractional servers by repeating the process some M times: the integral solution would pay at least $\min(W, L)$ in each such iteration and the fractional solution would pay at most Z, so that the initial movement cost would get amortized away.)

The Instance for ℓ Classes. We extend this construction to larger values of ℓ by defining phases in a recursive manner on a nested sequence of subsets of V, with each phase containing several repetitions of the same sequence. Instead of decreasing by a factor 2, the number of servers of each weight class now goes down by a factor of $C \geq \ell$. This allows the integrality gap result to hold even when the integral solution is allowed a resource augmentation of nearly ℓ.

For some $r \leq \ell - 1$, we call a tuple $(S_0, \ldots, S_r)$ *valid* if (i) $S_0 = V$ and each $S_j \subseteq S_{j-1}$, and (ii) $|S_j| = |S_{j-1}|/C = {}^{n}/_{C^j}$. The request sequence is generated by calling Algorithm 1 with the trivial valid sequence $(S_0 = V)$. Given a valid tuple $(S_0, \ldots, S_r)$, the procedure cycles through all subsets $S \subseteq S_r$ of size $|S_r|/C$ and recursively calls $\mathsf{Generate}(S_0, \ldots, S_r, S)$; this process is repeated L_r times. Finally, in the base case when $r = \ell - 1$, it cycles through all the locations in S_ℓ for $L_{\ell-1}$ times. For a suitably large choice of M, we define for each $r \in [\ell]$:

$$L_r := M^r \qquad \text{and} \qquad W_r := M^{\ell - r}. \tag{3}$$

Finally, the number of servers of weight W_r is given by $k_r := \frac{n}{\ell C^{r-1}}$.

Algorithm 1 Procedure $\mathsf{Generate}(S_0, S_1, \ldots, S_r)$.

```
1.1 Input: A valid tuple (S_0, S_1, ..., S_r)
1.2 repeat
1.3     if r is equal to ℓ − 1 then
1.4         Send a request at each location in S_{ℓ−1}.
1.5     else
1.6         for each subset S of S_r with |S| = |S_r|/C do
1.7             // Move 1/ℓ mass of servers of weight W_{r+2} to S
1.8             Call Generate(S_0, ..., S_r, S).
1.9 until L_r iterations have occurred
```

2.1 Analyzing the Integrality Gap

We bound the cost of the optimal fractional solution for the above input sequence.

▶ **Lemma 5.** *There is a fractional solution of total cost $O(f(n)M^{\ell-2})$ for the input sequence constructed by Algorithm 1, where $f(n)$ is a function solely of n.*

Proof. Our fractional solution maintains the invariant: when the procedure $\mathsf{Generate}(S_0, \ldots, S_r)$ is called, we have ${}^{1}/_{\ell}$ fractional mass of servers of weight $W_1, \ldots, W_{r+1}$ respectively at each location in S_r. For the base case $r = 0$, we place ${}^{1}/_{\ell}$ server mass at each location in $S_0 = V$; recall that $k_1 = {}^{n}/_{\ell}$. For the inductive step, suppose this invariant is satisfied for a certain value of r where $0 \leq r < \ell - 1$; we need to show that it is satisfied for $r+1$ as well. Indeed, the induction hypothesis implies that we have ${}^{1}/_{\ell}$ amount of server mass of weight $W_1, \ldots, W_{r+1}$ at each location in S_r, and hence at each location in S_{r+1}. Moreover, as line 1.7 indicates, we move ${}^{1}/_{\ell}$ fractional mass of servers of weight W_{r+2} to each location in S_{r+1} to satisfy the invariant condition. This costs $W_{r+2}\, k_{r+2}/\ell$; moreover, this is feasible because the total number of servers of weight W_{r+2} needed is $\frac{|S_{r+1}|}{\ell} = \frac{n}{\ell C^{r+1}} = k_{r+2}$. Finally, when $r = \ell - 1$, the invariant shows that 1 unit of server mass is present at each of the locations in S_ℓ, and hence the requests generated in line 1.4 can be served without any additional movement of servers.

We now account for the movement cost. The total server movement cost during $\textsf{Generate}(S_0, \ldots, S_r)$ (not counting the movement costs in the recursive calls) is at most $O(L_r\, k_{r+1}\, W_{r+2}) = O(k_{r+1}\, M^{\ell-2})$. Since $k_{r+1} \le n$ and the number of calls to Generate is a function only of n, the overall movement cost can be expressed as $O(f(n) \cdot M^{\ell-2})$. (Again, by repeating the entire process multiple times we can amortize away the initial movement cost; we avoid this step for the sake of clarity.) ◀

The next lemma shows that any integral solution must have much higher cost.

▶ **Lemma 6.** *Let $\varepsilon > 0$ be a small enough constant. Assume that the integral solution is allowed $(\ell - \varepsilon)k_r$ servers of weight W_r for each $r \in [\ell]$. Any integral solution for the input sequence generated by Algorithm 1 (with $C = \ell/\varepsilon$) has movement cost at least $M^{\ell-1}$.*

We defer the proof to Appendix B; combining Lemma 5 and Lemma 6 proves Theorem 2.

3 An Offline Algorithm via LP Rounding

We now show an algorithm for the offline setting, that rounds any fractional solution to the LP relaxation (LP), and achieves the following guarantee:

▶ **Theorem 3** (Offline Algorithm). *Let $\mathcal{I}$ be an instance of WEIGHTED k-SERVER with k_j servers of weight W_j for all $j \in [\ell]$. For any $\varepsilon \in (0,1)$, there is a polynomial time algorithm for $\mathcal{I}$ that uses at most $2(1+\varepsilon)\ell \cdot k_j$ servers of weights W_j for each $j \in [\ell]$ and has server movement cost at most $O(1/\varepsilon)$ times the optimal cost of $\mathcal{I}$.*

Instead of working with the relaxation (LP), we work with an equivalent relaxation which turns out to be easier to interpret. For each vertex $v \in V$, index $j \in [\ell]$ and time interval I, we have a variable $y_{v,j,I}$, which denotes the fractional mass of server of weight W_j residing at v during the entire time interval I. The variable $x_{v,j,t}$ used in (LP) can be expressed as follows:

$$x_{v,j,t} = \sum_{I: t \in I} y_{v,j,I}. \tag{4}$$

Let $\mathbf{I}$ denote the set of all intervals during the request timeline. The new linear program relaxation for WEIGHTED k-SERVER is the following:

$$\min \, 1/2 \sum_{j \in [\ell]} W_j \sum_{I \in \mathbf{I}} \sum_{v \in V} y_{v,j,I} \tag{LP2}$$

$$\text{s.t.} \sum_{j \in [\ell]} \sum_{I: t \in I} y_{\sigma_t,j,I} \ge 1 \qquad \forall t \tag{5}$$

$$\sum_{v \in V} \sum_{I: t \in I} y_{v,j,I} \le k_j \qquad \forall t, j \in [\ell] \tag{6}$$

$$y_{v,j,I} \ge 0 \qquad \forall t, j \in [\ell], v \in V.$$

Note that the covering constraint (5) enforces having at least one unit of (fractional) server mass at the location σ_t requested for each time t. The packing constraint (6) enforces that the total (fractional) server mass of weight W_j used at any time t is at most the number of

Algorithm 2 Procedure $\mathsf{ScaleRound}(x, y, v, W_j)$.

Input: A fractional solution $(y_{v,j,I}, x_{v,j,t})$ to LP2, a location v and a weight W_j
Initialize variables $\bar{y}_{v,j,I}$ to 0 for all intervals I.
(Scale): Define $\widetilde{y}_{v,j,I} = (2 + \varepsilon/2)\,\ell \cdot y_{v,j,I}$ and therefore,
$\widetilde{x}_{v,j,t} = \sum_{I:t\in I} \widetilde{y}_{v,j,I} = (2 + \varepsilon/2)\,\ell \cdot x_{v,j,t}$ for each $I \in \mathbf{I}$.
(Round): **for** $h = 1, 2, \ldots, \ell$ **do**
  Initialize $\mathsf{LastEvent} = \mathsf{DOWN}, \mathsf{LastTime} = 0$.
  repeat
    if $\mathsf{LastEvent} = \mathsf{UP}$ **then**
      Let t be the first DOWN after $\mathsf{LastEvent}$
      Update $\mathsf{LastEvent} = \mathsf{DOWN}, \mathsf{LastTime} = t$.
    else
      $(\mathsf{LastEvent} = \mathsf{DOWN})$ Let t be the first UP after $\mathsf{LastEvent}$
      Add $I = [\mathsf{LastTime}, t)$ to $\mathbf{I}_{v,j}(h)$ and increase $\bar{y}_{v,j,I}$ by 1.
      Update $\mathsf{LastEvent} = \mathsf{DOWN}, \mathsf{LastTime} = t$.
  until *we have reached the end of the timeline* $[0, T]$

servers of this weight, namely k_j. Given a solution $y_{v,j,I}$ to LP2, the variables $x_{v,j,t}$ defined using (4) define a feasible solution to LP of the same cost.

Fix any constant $\varepsilon \in (0, 1)$. We now prove Theorem 3 by rounding an optimal fractional solution $y_{v,j,I}$ to LP2. The rounding algorithm has two stages. The first stage scales and discretizes the LP variables to integers such that

1. the packing constraints are satisfied up to a factor of $(2 + \varepsilon)\ell$,
2. the covering constraints are satisfied with a scaled covering requirement of ℓ instead of 1, i.e., $\sum_j \sum_{I:t\in I} y_{\sigma_t,j,I} \geq \ell$, for all times t, and
3. the cost of the fractional solution increases by a factor of $O(\ell/\varepsilon)$.

In the second stage, we remove the packing constraints from the LP; this results in the resulting interval covering LP being integral. Next, we scale the solution from the first stage down by ℓ, getting a feasible fractional solution to the standard LP relaxation for the interval covering problem. Finally, we use the integrality of the interval covering LP relaxation to obtain an integral solution for LP2. We present these two stages in the next two sections.

3.1 Stage I: Scaling and Discretization

The first stage of the rounding algorithm operates independently on each location $v \in V$ and for each server weight W_j; the formal algorithm $\mathsf{ScaleRound}(x, y, v, W_j)$ is given in Algorithm 2. We work with both the $y_{v,j,I}$ variables and the equivalent $x_{v,j,t}$ variables defined in (4); this representational flexibility makes it convenient to explain the algorithm. To begin, we scale the LP variables $y_{v,j,I}$ by a factor $(2 + \varepsilon/2)\ell$ to obtain $\widetilde{y}_{v,j,I}$ (we also define the auxiliary variables $\widetilde{x}_{v,j,t}$ by scaling $x_{v,j,t}$ similarly).

Discretization. Next we discretize the scaled variables $\widetilde{y}_{v,j,I}$ and $\widetilde{x}_{v,j,t}$ to nonnegative integers $\bar{y}_{v,j,I}$ and $\bar{x}_{v,j,t}$ respectively. To start, let us describe the discretization of $\widetilde{x}_{v,j,t}$ to obtain $\bar{x}_{v,j,t}$. Intuitively, we would like to define $\bar{x}_{v,j,t}$ as $\lfloor \widetilde{x}_{v,j,t} \rfloor$, i.e., the largest step function with unit step sizes entirely contained in $\widetilde{x}_{v,j,t}$, but this can amplify small fluctuations around integer values, and hence may increases the cost. To avoid this, we introduce *hysteresis* in our discretization, by setting different thresholds for increasing and decreasing the value of

$\widetilde{x}_{v,j,t}$. We view $\widetilde{x}_{v,j,t}$ as a time-varying profile and define horizontal *slabs* in it corresponding to the restriction of the range of $\widetilde{x}_{v,j,t}$ to $[h, h+1)$ for some integer h. For each such slab, we identify intervals I of width at most 1 and at least $1/2$ and set the increase the corresponding $\bar{y}_{v,j,I}$ value by 1. In more detail, for each such level h, we identify a subset $\mathbf{I}_{v,j}(h)$ of intervals for which the corresponding $\bar{y}_{v,j,I}$ variable is to be increased by 1. We identify an alternating sequence of *up* and *down* events in the timeline $[0, T]$ as follows:

- UP event: At time t, there is an UP event at level h if $\widetilde{x}_{v,j,t^-} < h$ and $\widetilde{x}_{v,j,t} \geq h$, and the previous event at level h was a DOWN event.
- DOWN event: At time t, there is a DOWN event at level h if the previous event at level h was an UP, and $\widetilde{x}_{v,j,t^-} > h - \varepsilon/2$ and $\widetilde{x}_{v,j,t} \leq h - \varepsilon/2$, or $t = T$, the end of the timeline. (The reader should think of $\varepsilon/2$ as the "hysteresis gap" between the up and down events at any level.)

To make the definition complete, we set $\widetilde{x}_{v,j,t}$ to 0 at $t = 0^-$ and at $t = T^+$, and start with a DOWN at time 0. Finally, we add intervals stretching from each UP to the next DOWN to the set $\mathbf{I}_{v,j}(h)$ of intervals. By construction, these intervals are mutually disjoint. Finally, whenever an interval I is added to such a set $\mathbf{I}_{v,j}(h)$, we increment the corresponding variable $\bar{y}_{v,j,I}$. Thus we have:

$$\bar{y}_{v,j,I} = |\{h : I \in \mathbf{I}_{v,j}(h)\}|, \text{ and correspondingly, } \bar{x}_{v,j,t} = \sum_{I: t \in I} \bar{y}_{v,j,I}.$$

The next lemma shows that $\bar{x}_{v,j,t}$ can be thought of as a discretized form of $\widetilde{x}_{v,j,t}$:

▶ **Lemma 7.** *The following holds for variables $\bar{x}_{v,j,t}$:*

$$\widetilde{x}_{v,j,t} - 1 < \bar{x}_{v,j,t} < \widetilde{x}_{v,j,t} + \varepsilon/2. \tag{7}$$

Proof. Suppose $\widetilde{x}_{v,j,t} \in [r, r+1)$. Consider the **for** loop in line 2.4 in Algorithm 2 for a value $h \leq r$. We claim that at time t, the value of the variable LastEvent must be UP. Suppose not. Let t' be the value of LastTime at time t (i.e., t' is the last time before and including t when an UP or a DOWN occurred). Since a DOWN event happened at time t', $\widetilde{x}_{v,j,t'} < h$. Since $\widetilde{x}_{v,j,t} \geq h$, an UP event must occur during $(t', t]$, a contradiction. Therefore must have added an interval containing time t to $\mathbf{I}_{v,j}(h)$. Thus, $\bar{x}_{v,j,t}$ gets increased during each such iteration, i.e., $\bar{x}_{v,j,t} \geq r > \widetilde{x}_{v,j,t} - 1$. This proves the first inequality in (7).

We now prove the second inequality. Let h be an integer satisfying $h \geq \widetilde{x}_{v,j,t} + \varepsilon/2$. Consider the iteration of the **for loop** in Algorithm 2 for this particular value of h. We claim that the value of the variable LastEvent at time t must be DOWN. Suppose not, and let t' denote the value of the variable LastTime. Then an UP happened at time t' and so $\widetilde{x}_{v,j,t'} \geq h$. Since $\widetilde{x}_{v,j,t} \leq h - \varepsilon/2$, a DOWN event must have happened during $(t', t]$, a contradiction. Hence, we do not add any interval containing time t to the set $\mathbf{I}_{v,j}(h)$. Therefore, $\bar{x}_{v,j,t} < \widetilde{x}_{v,j,t} + \varepsilon/2$, which proves the second inequality in (7). ◀

The next lemma establishes the key properties of the variables $\bar{y}_{v,j,I}$ and $\bar{x}_{v,j,t}$.

▶ **Lemma 8.** *The following properties hold the for the variables $\bar{y}_{v,j,I}$:*

(i) (Cost) The LP cost increases by at most $O(\ell/\varepsilon)$ when the original variables $y_{v,j,I}$ are replaced by the new variables $\bar{y}_{v,j,I}$:

$$\sum_{v,j,I} W_j \cdot \bar{y}_{v,j,I} \leq O(\ell/\varepsilon) \cdot \sum_{v,j,I} W_j \cdot y_{v,j,I}.$$

(ii) (Covering) The variables $\bar{y}_{v,j,I}$ satisfy the scaled covering constraints of (LP2)

$$\sum_{j,I:t\in I} \bar{y}_{v,j,I} \geq \ell \quad \forall t.$$

(iii) (Packing) The variables $\bar{y}_{v,j,I}$ approximately satisfy the packing constraints of (LP2)*:*

$$\sum_{v,I:t\in I} \bar{y}_{v,j,I} \leq (2+\varepsilon)\ell k_j \quad \forall j \in [\ell], t.$$

Proof. We first prove the cost bound: the cost of the solution $\bar{y}_{v,j,I}$ is the weight of all intervals added to the sets $\mathbf{I}_{v,j}(h)$ for all v, j, h. I.e.,

$$\sum_{v,j,I} W_j \cdot \bar{y}_{v,j,I} = \sum_{v,j} W_j \cdot \sum_{h\in[\ell]} |\mathbf{I}_{v,j}(h)|. \tag{8}$$

Fix a vertex v and indices j, h. For a non-negative number x, and non-negative integer h, define the *h-level truncation* of x to be $\mathsf{trunc}_h(x) := \min(1, (x-h)^+)$, where $(a)^+ := \max(a, 0)$ for any real a. Observe that $x = \sum_{h\geq 0} \mathsf{trunc}_h(x)$. In fact, for any two non-negative integers x and y:

$$|x-y| = \sum_{h'\geq 0} |\mathsf{trunc}_{h'}(x) - \mathsf{trunc}_{h'}(y)|. \tag{9}$$

Now let $I_1 = [s_1, t_1), \ldots, I_u = [s_u, t_u)$ be the intervals added to $\mathbf{I}_{v,j}(h)$ (in left to right order). Define $t_0 = 0$. We know that for any $i \in [u]$, an UP happens at s_u and a DOWN happens at t_u. Therefore, $\mathsf{trunc}_h(\widetilde{x}_{v,j,s_u}) - \mathsf{trunc}_h(\widetilde{x}_{v,j,t_{u-1}}) \geq \varepsilon/2$. Hence,

$$\begin{aligned}
\varepsilon W_j/2 \cdot |\mathbf{I}_{v,j}(h)| &\leq W_J \cdot \sum_{i=1}^{u} |\mathsf{trunc}_h(\widetilde{x}_{v,j,s_u}) - \mathsf{trunc}_h(\widetilde{x}_{v,j,t_{u-1}})| \\
&\leq W_j \cdot \sum_{t'=1}^{T} |\mathsf{trunc}_h(\widetilde{x}_{v,j,t-1}) - \mathsf{trunc}_h(\widetilde{x}_{v,j,t})|,
\end{aligned}$$

where the last inequality follows from triangle inequality. Summing over all h and using (9), we get

$$\varepsilon W_j/2 \cdot \bar{y}_{v,j,I} \leq W_j \cdot \sum_{t'=1}^{T} |\widetilde{x}_{v,j,t-1}) - \widetilde{x}_{v,j,t}|.$$

Summing over all vertices v and indices $j \in [\ell]$, we see that the cost of the solution $\bar{y}_{v,j,I}$ is at most $2/\varepsilon$ times that of $\widetilde{y}_{v,j,I}$. Finally, the fact that $\widetilde{y}_{v,j,I}$ are obtained by scaling $y_{v,j,I}$ by a factor $(2+\varepsilon/2)\ell$, we get the desired bound on the cost of $\bar{y}_{v,j,I}$ solution.

Next, we prove the covering property. Since $x_{v,j,t}$ is a feasible solution to LP2, we have for any time t:

$$\sum_j x_{\sigma_t,j,t} \geq 1, \text{ and therefore, } \sum_j \widetilde{x}_{\sigma_t,j,t} \geq (2+\varepsilon/2)\ell.$$

Using Lemma 7, we have $\widetilde{x}_{\sigma_t,j,t} < \bar{x}_{\sigma_t,j,t} + 1$, so

$$\sum_{j\in\ell} (\bar{x}_{\sigma_t,j,t} + 1) > (2+\varepsilon/2)\ell, \text{ and therefore, } \sum_j \bar{x}_{\sigma_t,j,t} > \ell.$$

Finally, we prove the packing property. Since $x_{v,j,t}$ is a feasible solution to the LP, we have for any $j \in [\ell]$ and time t,

$$\sum_v x_{v,j,t} \leq k_j, \text{ and therefore, } \sum_v \widetilde{x}_{v,j,t} \leq (2 + \varepsilon/2)\ell k_j.$$

Again Lemma 7 gives $\widetilde{x}_{v,j,t} > \bar{x}_{v,j,t} - \varepsilon/2$, which implies

$$\sum_j (\bar{x}_{v,j,t} - \varepsilon/2)^+ < (2 + \varepsilon/2)\ell k_j. \tag{10}$$

Since $\bar{x}_{v,j,t}$ is a nonnegative integer,

$$\bar{x}_{v,j,t} > 0 \implies \bar{x}_{v,j,t} \geq 1 \overset{\text{Lemma 7}}{\implies} \widetilde{x}_{v,j,t} > \bar{x}_{v,j,t} - \varepsilon/2 \geq 1 - \varepsilon/2.$$

Since $\sum_v \widetilde{x}_{v,j,t} \leq k_j$, it follows that the number of locations v for which $\bar{x}_{v,j,t} > 0$ is at most $\frac{k_j}{1-\varepsilon/2} < 2k_j$, if $\varepsilon < 1$. Using this fact in Equation (10), we get

$$\begin{aligned}\sum_v \bar{x}_{v,j,t} &= \sum_{v:\bar{x}_{v,j,t}>0} \bar{x}_{v,j,t} = \sum_{v:\bar{x}_{v,j,t}>0} (\bar{x}_{v,j,t} - \varepsilon/2) + \sum_{v:\bar{x}_{v,j,t}>0} \varepsilon/2 \\ &\leq \sum_v (\bar{x}_{v,j,t} - \varepsilon/2)^+ + 2k_j \cdot \varepsilon/2 \leq (2 + \varepsilon/2)\ell k_j + \varepsilon k_j.\end{aligned}$$

Since $\ell \geq 2$ (otherwise, we have the unweighted problem), we get

$$\sum_v \bar{x}_{v,j,t} \leq (2 + \varepsilon)\ell k_j.$$ ◀

3.2 Stage II: Weighted Interval Cover

In the second stage of the rounding algorithm, we first scale the (integer-valued) variables $\bar{y}_{v,j,I}$ down by a factor of ℓ to obtain new variables $y^*_{v,j,I}$:

$$y^*_{v,j,I} := \bar{y}_{v,j,I}/\ell \text{ and therefore, } x^*_{v,j,t} = \sum_{I:t\in I} y^*_{v,j,I} = \bar{x}_{v,j,t}/\ell. \tag{11}$$

Our goal is to round the fractional variables $y^*_{v,j,I}$ to $\{0, 1\}$ values. In fact, our rounding will ensure that if the rounded value equals 1 then $y^*_{v,j,I} > 0$. Since $\bar{y}_{v,j,I}$ is integral, the packing property in Lemma 8 implies that for any time t, vertex v, and index $j \in [\ell]$, there are at most $(2 + \varepsilon)\ell k_j$ intervals $I \ni t$ for which $\bar{y}_{v,j,I} > 0$. The rounding property of our algorithm will ensure that the final integral solution, which lies in the support of $y^*_{v,j,I}$, will also satisfy that there are at most $(2 + \varepsilon)\ell k_j$ intervals containing any time t. Since we are allowed a resource augmentation of $(2 + \varepsilon)\ell$ factor in the number of servers of weight W_j, we can serve the requests with the set of available servers. Henceforth, we can ignore the packing constraint (6) for our rounded solution. As a result, the relaxation LP2 decouples into n independent relaxations, one for each location $v \in V$.

In this decoupled instance, we get the following LP relaxation for each location v, where for each class $j \in [\ell]$, we define $\mathbf{I}_{v,j} := \{I \mid y^*_{v,j,I} > 0\}$ as the set of intervals I with a nonzero value of $y^*_{v,j,I}$ and $\mathcal{R}(v)$ as the set of times t when v is requested:

$$\begin{aligned} &\min \, 1/2 \sum_{j\in[\ell]} W_j \cdot \sum_{I\in\mathbf{I}_{v,j}} y_{v,j,I} && \text{(LP}_v\text{)} \\ &\text{s.t.} \sum_j \sum_{I\in\mathbf{I}_{v,j}:t\in I} y_{v,j,I} \geq 1 && \forall t \in \mathcal{R}_v \\ &\qquad y_{v,j,I} \geq 0. \end{aligned}$$

By the covering property of Lemma 8, the variables $y^*_{v,j,I}$ defined in (11) are feasible solutions for (LP$_v$) for all locations v. Furthermore, by the lemma's cost property (and the scaling down by ℓ), the total cost $\sum_v \sum_j W_j \cdot \sum_I y^*_{v,j,I}$ is at most $O(1/\varepsilon)$ times the optimal cost of (LP2).

Finally, the constraint matrix for (LP$_v$) satisfies the consecutive-ones property: if the constraints are ordered chronologically, then a variable $y_{v,j,I}$ appears in the constraints corresponding to times $t \in I$ where $\sigma_t = v$, which is a contiguous subsequence of all times t where $\sigma_v = j$. Constraint matrices with this property are totally unimodular (see, e.g., [18]). Therefore, each of the solutions $\{y^*_{v,j,I} : j \in [\ell], I \in \mathbf{I}_{v,j}\}$ for LP$_v$ can be rounded to a feasible integral solution without any increase in cost, which proves Theorem 3.

4 Online Algorithm

In this section, we describe an efficient online algorithm for Weighted k-Server and prove the following result:

▶ **Theorem 4** (Online Algorithm). *Let $\mathcal{I}$ be an instance of Weighted k-Server with k_j servers of weight W_j for all $j \in [\ell]$. There is a randomized (polynomial time) online algorithm for $\mathcal{I}$ that uses at most $2\ell k_j$ servers of weights W_j for each $j \in [\ell]$ and has expected server movement cost at most $O(\ell^2 \log \ell)$ times the optimal cost of $\mathcal{I}$.*

We begin by re-writing the LP relaxation (LP2) in terms of the "anti-page" variables, as in [4]. Recall that (LP2) has variables $y_{v,j,I}$ representing the (fractional) weight W_j server mass present at location v during the interval I; instead we first rewrite it in terms of the "page" variables $x_{v,j,t}$, which denote the total amount of weight W_j server mass at location v at time t, as given in (4). The objective of this LP in terms of $x_{v,j,t}$ is:

$$\sum_{v,j,I} W_j \cdot y_{v,j,I} = \sum_{v,j,I} W_j \cdot (x_{v,j,t} - x_{v,j,t^-})^+.$$

We can constrain any algorithm to values $x_{v,j,t} \in [0,1]$ for all v, j, t (since having multiple servers at a location is not beneficial). This allows us to work with non-negative *anti-page* variables $z_{v,j,t} := 1 - x_{v,j,t}$. The objective, now rewritten in terms of these new variables $z_{v,j,t}$, becomes:

$$\sum_{v,j,I} W_j \cdot (x_{v,j,t} - x_{v,j,t^-})^+ = \sum_{v,j,I} W_j \cdot (z_{v,j,t^-} - z_{v,j,t})^+. \tag{12}$$

We shall also maintain the following invariant for each server weight W_j and time t:

$$\sum_v x_{v,j,t} = k_j \qquad \iff \qquad \sum_v z_{v,j,t} = n - k_j \quad \forall j, t. \tag{13}$$

We write the covering constraint (5) (or equivalently (2)) in terms of $z_{v,j,t}$ as:

$$\sum_j z_{\sigma_t,j,t} \leq \ell - 1 \tag{14}$$

The algorithm follows the standard relax-and-round paradigm in the online setting. The first step is to compute a feasible fractional solution to an LP consisting of objective (12) and constraints (13) and (14), in an online setting. We show in §4.1 that we can find a fractional solution that uses $O(\ell k_j)$ servers of weight W_j for each class j, and has a competitive ratio of $O(\ell^2)$. The second step is to give an online rounding algorithm to convert this fractional solution to an integral solution: our rounding algorithm given in §4.2 uses the standard online rounding algorithm for the paging problem and increases the cost of the solution by a constant factor.

4.1 Online Fractional Algorithm

In this section, we give an online algorithm for maintaining a fractional solution to the LP involving $z_{v,j,t}$ variables. We obtain the following result:

▶ **Theorem 9.** *There is a deterministic (polynomial time) online fractional algorithm that maintains the condition that for every request time t, there exists an index $j \in [\ell]$ such that there is unit server mass of weight W_j at location σ_t at time t. The algorithm uses $2\ell k_j$ servers of weight W_j for each $j \in [\ell]$, and whose cost is at most $O(\ell^2 \log \ell)$ times that of an optimal fractional solution.*

Note that the condition in the theorem is stronger than (14), the feasibility condition for (LP2), because we are using server from a single weight class to service this request.

Consider a time t, and the request arriving at location σ_t. We first set $z_{v,j,t} = z_{v,j,t^-}$ for all $v \in V, j \in \ell$. Now the algorithm moves fractional server mass to σ_t until a relaxed version of the covering constraint (14) for time t gets satisfied. The relaxed constraint is given by

$$\exists j \in [\ell] \text{ such that } z_{\sigma_t,j,t} \leq 1 - \frac{1}{2\ell}. \tag{15}$$

Indeed, if the constraint is violated, then for each vertex $v \neq \sigma_t$ and each $j \in [\ell]$, if v has non-zero server mass of weight W_j (i.e., $z_{v,j,t} < 1$), then the algorithm moves server mass of weight W_j from v to σ_t using the following differential equation. (The derivative is with respect to a variable s which starts from 0 and increases at uniform rate.)

$$\dot{z}_{v,j,t} = \frac{1}{W_j|S_j|} \cdot (z_{v,j,t} + \delta) \qquad \forall j \in [\ell], \forall v \in S_j. \tag{16}$$

Here, $S_j \subseteq V$ denotes the instantaneous set of locations (i.e., at the current value of the variable s) that have $z_{v,j,t} < 1$, not including the location σ_t, and $\delta > 0$ is a parameter that we shall fix later. Correspondingly, we reduce $z_{\sigma_t,j,t}$ by the total amount of server mass of weight W_j entering σ_t:

$$\dot{z}_{\sigma_t,j,t} = -\frac{1}{W_j|S_j|} \cdot \sum_{v \in S_j} (z_{v,j,t} + \delta) \qquad \forall j \in [\ell]. \tag{17}$$

Note that server mass is moved away other locations and into location σ_t only if $z_{\sigma_t,j,t} > 1 - \frac{1}{2\ell}$ for all j. Since $z_{\sigma_t,j,t} \leq 1$ for all j, it follows that $z_{v,j,t} \in [1 - \frac{1}{2\ell}, 1]$ for all j, t. Hence,

$$z_{v,j,t} \geq 1 - \frac{1}{2\ell} \text{ for all } j, t \quad \implies \quad |S_j| \geq 2\ell k_j - 1 \geq \frac{3\ell k_j}{2} \geq 3 \text{ for all } j, t, \tag{18}$$

since $\ell \geq 2, k_j \geq 1$.

To analyze the algorithm, we use a potential function Φ. The potential function depends on the offline (integral) optimal solution—let us call it $\mathcal{O}$, and let $\text{opt}_{v,j,t}$ be the indicator variable for the location v containing a server of weight W_j at time t. The potential at time t is defined as follows:

$$\Phi(t) := \sum_{v,j:\text{opt}_{v,j,t}=0} W_j \cdot \ln\left(\frac{1+\delta}{z_{v,j,t}+\delta}\right).$$

Let $\mathsf{cost}(t)$ denote the algorithm's server movement cost at time t and $\mathsf{cost}^{\mathcal{O}}(t)$ denote the corresponding quantity for the optimum solution $\mathcal{O}$. Our goal is to show that:

$$\frac{\mathsf{cost}(t)}{4\ell} + \Phi(t+1) - \Phi(t) \leq \ln(1 + 1/\delta) \cdot \mathsf{cost}^{\mathcal{O}}(t). \tag{19}$$

The following properties of $\Phi(t)$ can verified easily:

- **Nonnegativity:** Φ is always nonnegative, since $z_{v,j,t} \leq 1$.
- **Lipschitzness property:** When the optimal solution moves a server of weight W_j from one location to another, the increase in Φ is at most $W_j \cdot \ln(1 + 1/\delta)$.

The Lipschitzness property implies that (19) holds when $\mathcal{O}$ serves the request at σ_t. It remains the analyze the cost and change in potential when the algorithm changes its solution. Consider the process when we transfer server mass to σ_t.

We first bound the online algorithm's cost. Since all the weight classes incur the same server movement cost while transferring to σ_t, the movement cost is ℓ times the movement cost incurred while transferring servers of a fixed class, say $j^\star$. The latter is at most

$$W_{j^\star} \sum_{v \in S_{j^\star}} \dot{z}_{v,j^\star,t} \stackrel{(16)}{=} \frac{1}{|S_{j^\star}|} \sum_{v \in S_{j^\star}} (z_{v,j^\star,t} + \delta) \quad = \frac{|S_{j^\star}| + 1 - k_{j^\star} + \delta |S_{j^\star}|}{|S_{j^\star}|} \leq 1 + \delta. \tag{20}$$

Thus, the upper bound on the $\frac{\mathsf{cost}(t)}{4\ell}$ term in the LHS of (19) is at most $\frac{1+\delta}{4} \leq 1/3$ provided $\delta \leq 1/3$.

Next, we lower bound the rate of decrease of potential Φ. We begin by bounding the rate of decrease in potential due to because of server mass leaving all locations except σ_t:

$$\begin{aligned}\Delta^- &= -\sum_{j \in [\ell], v \neq \sigma_t : \mathrm{opt}_{v,j,t} = 0} \frac{W_j}{z_{v,j,t} + \delta} \cdot \dot{z}_{v,j,t} \quad \stackrel{(16)}{=} -\sum_{j, v \in S_j : \mathrm{opt}_{v,j,t} = 0} \frac{1}{z_{v,j,t} + \delta} \cdot \frac{z_{v,j,t} + \delta}{|S_j|} \\ &= -\sum_j \frac{|\{v \in S_j : \mathrm{opt}_{v,j,t} = 0\}|}{|S_j|} \quad \stackrel{(18)}{\leq} -\sum_j \frac{|S_j| - k_j}{|S_j|} \leq -\ell \left(1 - \frac{2}{3\ell}\right) = -\ell + 2/3.\end{aligned} \tag{21}$$

Next, we bound the rate of increase in potential due to server classes $j \neq j^*$ because of server mass entering σ_t:

$$\begin{aligned}\Delta^+ &= \sum_{j \neq j^*} \frac{W_j}{z_{\sigma_t,j,t} + \delta} \cdot \dot{z}_{\sigma_t,j,t} \quad \stackrel{(16)}{=} \sum_{j \neq j^*, v \in S_j} \frac{W_j}{z_{\sigma_t,j,t} + \delta} \cdot \frac{z_{v,j,t} + \delta}{|S_j| W_j} \\ &= \sum_{j \neq j^*} \frac{\sum_{v \in S_j} (z_{v,j,t} + \delta)}{|S_j| (z_{\sigma_t,j,t} + \delta)} \quad = \sum_{j \neq j^*} \frac{(|S_j| - k_j + (1 - z_{\sigma_t,j,t})) + \delta \cdot |S_j|}{|S_j| (z_{\sigma_t,j,t} + \delta)} \\ &\stackrel{(18)}{\leq} \sum_{j \neq j^*} \frac{(|S_j| - k_j + 1/2\ell) + \delta \cdot |S_j|}{|S_j| (1 - 1/2\ell + \delta)} \quad \stackrel{(18)}{\leq} \sum_{j \neq j^*} \frac{1 - 2/3\ell + 1/6\ell + \delta}{1 - 1/2\ell + \delta} \quad \leq \ell - 1,\end{aligned}$$

provided $\delta = 1/2\ell$. Combining with (21), we see that the overall change in potential is $\Delta^- + \Delta^+ \leq -1/3$. Consequently, we get that the change in potential pays for the increase in the algorithm's cost (divided by 4ℓ)—which shows (19)—when the fractional solution changes.

This implies that we have an algorithm for maintaining $z_{v,j,t}$ that satisfies (15). In terms of the competitive ratio, the algorithm loses 4ℓ in (19) and $\ln(1 + 1/\delta) = O(\log \ell)$ in the Lipschitzness of the potential function. Note that (15) implies that for all t, there exists j such that $x_{\sigma_t,j,t} \geq \frac{1}{2\ell}$. We scale the fractional variables to obtain $\widetilde{x}_{v,j,t} := \min(2\ell x_{v,j,t}, 1)$; then, for all t, there exists j such that $\widetilde{x}_{\sigma_t,j,t} = 1$. Note that this satisfies the condition in Theorem 9. Equivalently, the corresponding "anti-page" variables $\widetilde{z}_{v,j,t} := 1 - \widetilde{x}_{v,j,t}$ satisfy the following condition for all t:

$$\exists j \text{ such that } \widetilde{z}_{\sigma_t,j,t} = 0. \tag{22}$$

The last scaling step creates a resource augmentation of 2ℓ, and increases the competitive ratio to $O(\ell^2 \log \ell)$. This completes the proof of Theorem 9.

4.2 Rounding the Fractional Solution Online

We round the fractional solution for each weight class j separately. Let T_j represent the request times t when (22) is satisfied by weight class j. Note that the solution $\widetilde{z}_{v,j,t}$ for weight class j represents a feasible fractional solution for an instance of the paging problem with $2\ell k_j$ cache slots, where there is a page request for each time $t \in T_j$ at location σ_t.

We now invoke the following known online rounding algorithm for the paging problem separately in each weight class j to complete the proof of Theorem 4.

▶ **Lemma 10.** *[9] There is a randomized (polynomial time) online algorithm that converts any feasible fractional solution for an instance of the PAGING problem to an integral solution using the same number of cache slots, and incurs constant times the cost of the fractional solution.*

5 Discussion

In this work, we have given the first efficient offline and online algorithms with non-trivial guarantees for WEIGHTED k-SERVER. Several interesting problems remains open:

1. For the case of two distinct weight classes, we show in Appendix A that it is UG-Hard to obtain an $\Omega(N^c)$-approximation algorithm for some constant $c > 0$, even with $(2 - \varepsilon)$-resource augmentation. Can we extend such a hardness result to more weight classes? For example, can we show that for three distinct weight classes, it is UG-Hard to obtain a C-approximation algorithm for any *constant* C, even with $(3 - \varepsilon)$-resource augmentation? The principal reason why our hardness proof for $\ell = 2$ does not extend here is because one needs to recursively cycle through all subsets (of a certain size) of V to create an integrality gap instance for the natural LP relaxation. If the size of these subsets is large, then the length of the input becomes very large. If the size of these subsets is small, then it is not clear how to extend this to a hardness proof.
2. In Section 3, we give an offline constant approximation algorithm which requires slightly more than 2ℓ-resource augmentation. Can we get a constant approximation algorithm (or even an optimal algorithm) with exactly ℓ-resource augmentation? We conjecture that the integrality gap of LP is constant (or even 1) if the integral solution is allowed ℓ-resource augmentation.
3. In the online case, we give a $O(\ell^2 \log \ell)$-competitive algorithm with 2ℓ-resource augmentation in Section 4. Can we get a constant-competitive algorithm with $O(\ell)$-resource augmentation, i.e., a result in the same vein as our offline algorithm?

References

1 Nikhil Ayyadevara and Ashish Chiplunkar. The randomized competitive ratio of weighted k-server is at least exponential. In *29th Annual European Symposium on Algorithms*, volume 204 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 9, 11. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021.

2 Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40:1–40:49, 2015. doi:10.1145/2783434.

3 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Metrical task systems and the k-server problem on HSTs. In *Automata, languages and programming. Part I*, volume 6198 of *Lecture Notes in Comput. Sci.*, pages 287–298. Springer, Berlin, 2010. URL: https://doi.org/10.1007/978-3-642-14165-2_25, doi:10.1007/978-3-642-14165-2_25.

4 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19, 2012.

5 Nikhil Bansal, Marek Eliás, Lukasz Jez, and Grigorios Koumoutsos. The (h, k)-server problem on bounded depth trees. *ACM Trans. Algorithms*, 15(2):28:1–28:26, 2019. doi:10.1145/3301314.

6 Nikhil Bansal, Marek Eliáš, and Grigorios Koumoutsos. Weighted k-server bounds via combinatorial dichotomies. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*, pages 493–504. IEEE Computer Soc., Los Alamitos, CA, 2017. doi:10.1109/FOCS.2017.52.

7 Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *STOC '97 (El Paso, TX)*, pages 711–719. ACM, New York, 1999.

8 S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994. doi:10.1007/BF01294260.

9 Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 450–458. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814617.

10 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, USA, 1998.

11 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. Assoc. Comput. Mach.*, 39(4):745–763, 1992. doi:10.1145/146585.146588.

12 Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM J. Comput.*, 50(3):909–923, 2021. doi:10.1137/19M1237879.

13 Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 3–16. ACM, 2018. doi:10.1145/3188745.3188798.

14 Niv Buchbinder, Anupam Gupta, Marco Molinaro, and Joseph (Seffi) Naor. k-servers with a smile: Online algorithms via projections. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 98–116. SIAM, 2019. doi:10.1137/1.9781611975482.7.

15 Ashish Chiplunkar and Sundar Vishwanathan. Randomized memoryless algorithms for the weighted and the generalized k-server problems. *ACM Trans. Algorithms*, 16(1):Art. 14, 28, 2020. doi:10.1145/3365002.

16 Christian Coester and James R. Lee. Pure entropic regularization for metrical task systems. *Theory Comput.*, 18:Paper No. 23, 24, 2022. doi:10.4086/toc.2022.v018a023.

17 Amos Fiat and Moty Ricklin. Competitive algorithms for the weighted server problem. *Theoret. Comput. Sci.*, 130(1):85–99, 1994. doi:10.1016/0304-3975(94)90154-6.

18 D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835 – 855, 1965.

19 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 214–221. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492478.

20 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995. doi:10.1145/210118.210128.

21 James R. Lee. Fusible hsts and the randomized k-server conjecture. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 438–449. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00049.

22 Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990. doi:10.1016/0196-6774(90)90003-W.

23 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

24 Neal E. Young. On-line caching as cache size varies. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California, USA*, pages 241–250. ACM/SIAM, 1991. URL: http://dl.acm.org/citation.cfm?id=127787.127832.

Appendix

A The Unique Games Hardness

In this section, we consider the special case of Weighted k-Server when there are only two weight classes. Assume wlog that the two distinct weights are 1 and W, where $W \gg 1$. Our first main result shows that getting a good approximation algorithm with $(2-\varepsilon)$-resource augmentation for any constant $\varepsilon > 0$ is as hard as getting a better-than-two approximation for the vertex cover problem.

▶ **Theorem 1** (Hardness). *For any constant $\varepsilon > 0$, it is UG-hard to obtain an $N^{1/2-\varepsilon}$-approximation algorithm for Weighted k-Server with two weight classes, even when we are allowed c-resource augmentation for any constant $c < 2$. Here N represents the size of the input (including the request sequence length).*

Proof. We give a reduction from the Vertex Cover problem. Let $d = d(\varepsilon)$ be a constant to be fixed later, and let $c < 2$ be a constant as in the statement of the theorem. Let $\mathcal{I} = (G = (V, E), t)$ be an instance of the Vertex Cover problem on n vertices. We know that it is UG-hard to distinguish between the following two cases: (i) G has a vertex cover of size at most t, or (ii) every vertex cover of G must have size strictly larger than ct.

We map $\mathcal{I}$ to an instance $\mathcal{I}'$ of Weighted k-Server as follows: the set of points in $\mathcal{I}'$ is given by $V \cup \{v_0\}$, where v_0 is a special vertex. There are t servers of weight $W = n^d$ and one server of unit weight. Let the edges in E be $e_1, \ldots, e_m$. A subsequence of the request sequence consists of m *phases*, where we have a phase for each edge e_i. During phase i corresponding to edge $e_i = (u_i, v_i)$, the request sequence toggles between u_i and v_i for W times. Finally, the subsequence is repeated W times. In other words, it is the following sequence

$$\Big(\underbrace{u_1, v_1, u_1, v_1, \ldots, u_1, v_1}_{W \text{ times}}, \ldots, \underbrace{u_m, v_m, u_m, v_m, \ldots, u_m, v_m}_{W \text{ times}}\Big)^W.$$

We also have to specify the initial location of the servers. Assume that all servers are at location v_0 in the beginning. This completes the description of the instance $\mathcal{I}'$. Observe that N, the number of requests in instance $\mathcal{I}'$ is $O(m \cdot n^{2d})$.

▷ Claim 11. Suppose G has a vertex cover of size at most t. Then the cost of the optimal solution for $\mathcal{I}'$ is at most $2mW$.

Proof. Let $V' \subseteq V$ be a vertex cover of size t. Consider the following solution: we move the t heavy servers from v_0 to V' at the beginning. From now on, the heavy servers will not move at all. During a phase corresponding to an edge $e_i = (u_i, v_i)$, we know that at least one of these vertices will be occupied by a heavy server. If the other end-point, say v_i, is not occupied by a heavy server, we move the server of weight 1 to v_i. Now we have two servers occupying u_i

and v_i respectively until the end of this phase. The total movement cost is incurred either at the beginning (which is tW overall), or at the beginning of each phase (when the cost is 1). Since there are mW phases, the overall cost is at most $tW + mW \leq 2mW$. ◀

▷ Claim 12. Suppose every vertex cover in G has size strictly larger than ct. Then cost of the optimal solution for $\mathcal{I}'$, even with c-resource augmentation, is at least W^2.

Proof. Consider any solution for $\mathcal{I}'$. Recall that the input consists of W subsequences, call these $S_1, \ldots, S_W$, where each subsequence S_j consists of m phases, one for each edge of G. We claim that during each such subsequence S_j, the solution must pay movement cost of at least W. Indeed, consider a subsequence S_j. If the solution moves a heavy server during S_j, then the claim follows directly. Else observe that the size of any vertex cover is strictly larger than the number of heavy servers ct, so there is some edge $e_i = (u_i, v_i)$ not covered by the heavy servers during S_j. Now the phase for e_i in S_j would require the unit weight server to toggle between u_i and v_i for W times. In either case, the cost of each subsequence is at least W, and the overall cost of the solution is at least W^2. ◀

The above two results along with the UG-hardness result for VERTEX COVER impliy that it is UG-hard to obtain a $\frac{W^2}{2mW}$-approximation for WEIGHTED k-SERVER with two weight classes. This ratio is equal to $\frac{W}{2m} \geq n^{d-2} \geq N^{1/2-\varepsilon}$, assuming d is $\Omega(1/\varepsilon)$, which proves Theorem 1. ◀

B Missing proofs from §2

▶ **Lemma 6.** *Let $\varepsilon > 0$ be a small enough constant. Assume that the integral solution is allowed $(\ell - \varepsilon)k_r$ servers of weight W_r for each $r \in [\ell]$. Any integral solution for the input sequence generated by Algorithm 1 (with $C = \ell/\varepsilon$) has movement cost at least $M^{\ell-1}$.*

Proof of Lemma 6. We prove the following more general statement by reverse induction on r: any integral solution for the sequence generated by $\mathsf{Generate}(S_0, \ldots, S_r)$ for a valid tuple $(S_0, \ldots, S_r)$ which does not use any server of weight class $W_1, \ldots, W_r$ (at any location in S_r) has cost at least $M^{\ell-1}$. It suffices to prove this statement, because the case when $r = 0$ implies the lemma.

Consider the base case when $r = \ell - 1$. Consider the sequence generated by such a procedure $\mathsf{Generate}(S_0, \ldots, S_r)$ such that no server of weight $W_1, \ldots, W_{\ell-1}$ is used for serving the requests at $S_{\ell-1}$. Thus all requests generated by this procedure must be served by servers of weight W_ℓ. Now, $|S_{\ell-1}| = \frac{n}{C^{\ell-1}}$, whereas the number of weight W_ℓ servers available to the algorithm is $(\ell - \varepsilon)k_\ell < \frac{n}{C^{\ell-1}}$. Therefore, during each iteration of the **repeat-until** loop in lines 1.2–1.8 in Algorithm 1, at least one server of weight W_ℓ must move. So the overall movement cost during this input sub-sequence is at least $W_\ell \cdot L_{\ell-1} = M^{\ell-1}$. This proves the base case.

The inductive case is proved in an analogous manner. Suppose the statement is true for $r+1$, and now consider the sub-sequence generated by $Gen(S_0, \ldots, S_r)$ for some valid tuple $(S_0, \ldots, S_r)$. Assume that no server of weight $W_1, \ldots, W_r$ is present at any node in S_r during this time. We claim that the algorithm must incur movement cost of at least W_{r+1} during each iteration of the **repeat-until** loop. Indeed, fix such an iteration. Two cases arise: (a) The algorithm moves a server of weight W_{r+1} then the claim follows trivially, or (b) No server of weight W_{r+1} is moved during this period. Now observe that $|S_r| = \frac{n}{C^r}$, and the number of weight W_{r+1} servers available to the algorithm is $(\ell - \varepsilon)k_{r+1} = |S_r| - \varepsilon k_{r+1} = |S_r| \left(1 - \frac{1}{C}\right)$. Thus, there is a subset S_{r+1} of S of size $\frac{|S_r|}{C} = \frac{n}{C^{r+1}}$ where no server of weight W_{r+1} appears during this input sub-sequence. Consider the recursive call Generate$(S_0, \ldots, S_r, S_{r+1})$ in line 1.8. The induction hypothesis implies that the movement cost during this recursive call is at least $M^{\ell-1} \geq W_{r+1}$.

Thus, we have shown that the movement cost during each iteration of the **repeat-until** loop during Generate$(S_0, \ldots, S_r)$ is at least W_{r+1}. Since there are L_r such iterations, the overall movement cost is at least $W_{r+1} \cdot L_r = M^{\ell-1}$. This completes the proof of the induction hypothesis, and implies the lemma. ◀