A ROBUSTIFIED GREEDY ALGORITHM FOR ONLINE TRANSPORTATION WITH IMPROVED COMPETITIVE GUARANTEES

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the *online transportation problem*, in which n requests arriving sequentially in a metric space must be irrevocably assigned to k capacitated facilities. Beyond classical logistics applications, this problem models resource-allocation tasks arising in machine learning, including online facility assignments, recommender systems, and mixture-of-experts routing.

We introduce *Robustified Greedy* (RG), a deterministic generalization of the Robust Matching algorithm that achieves a competitive ratio of $6.66k - 2.89$, improving upon the state-of-the-art bounds of $8k - 7$ (Arndt et al., SOSA 2026) and $8k - 5$ (Harada and Itoh, ICALP 2025). RG also retains the metric-sensitive guarantee established for Robust Matching (RM) (Nayyar and Raghvendra, FOCS 2017), achieving a competitive ratio of $O(k^{1-1/d} \log^2 n)$ in d-dimensional Euclidean spaces for fixed $d > 1$. No comparable metric-sensitive guarantee is known for the transportation algorithms of Arndt et al. or Harada and Itoh.

Beyond these competitive guarantees, RG provides a simple explanation for its decisions. It favors the natural nearest-neighbor assignment and, for suitable parameters, departs from this choice only when it identifies a reassignment that reduces the cost of its maintained auxiliary matching, thereby correcting accumulated assignment costs. We also prove that nearest-neighbor assignments account for a guaranteed fraction of RG's total cost, approaching one-half for appropriate parameters, even under adversarial arrivals. Experiments on real-world datasets corroborate the theory: RG achieves lower cost-to-OPT ratios than the competing algorithms while retaining a substantial nearest-neighbor component in its cost.

1 INTRODUCTION

In the *online metric transportation problem*, we are given a set S of k server locations in a metric space. Each $s \in S$ has a positive integral capacity b_s, with $\sum_{s \in S} b_s = n$. A sequence of n requests arrives online, each of which must be irrevocably assigned to a server with available capacity, consuming one unit of capacity. The objective is to minimize the total assignment distance. When every server has unit capacity, this is the *online metric matching problem* (Kalyanasundaram & Pruhs, 1993). These problems model sequential allocation in logistics and resource assignment (Harada & Itoh, 2025; Arndt et al., 2026), as well as machine-learning settings in which requests and resources are embeddings, costs reflect embedding-space dissimilarity, and resources have limited capacities. Examples include capacity-constrained recommendation (Mladenov et al., 2020; Li et al., 2024), balanced token-to-expert routing in sparse mixture-of-experts models (Lewis et al., 2021; Zhou et al., 2022), and 1-Wasserstein-based imitation learning in robotics (Dadashi et al., 2021).

Assignments must be made before future requests are known but are judged against an optimal solution that knows the entire sequence. The standard measure is the competitive ratio (Sleator & Tarjan, 1985): an algorithm is c-competitive if, for every request sequence and arrival order, its cost is at most c times the optimum. For transportation, the goal is a guarantee depending on the number of server locations k, independently of the total capacity n.

The natural Greedy algorithm assigns each request to a nearest available server. This minimizes the current cost and makes the decision easy to explain; in emergency response, for example, proximity

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.

to an available responder may be an immediate priority. Greedy also performs well empirically: Tong et al. (2016) found that it outperformed all other evaluated algorithms on almost all tests, including algorithms with substantially better worst-case guarantees. Under independent uniform sampling of servers and requests in $[0, 1]^d$, Greedy is $O(1)$-competitive for every fixed $d \neq 2$ and $\Theta(\sqrt{\log n})$-competitive for $d = 2$ (Balkanski et al., 2023; Yang & Yu, 2026).

Nevertheless, Greedy has a worst-case competitive ratio of $2^n - 1$. To see the lower bound without relying on tie-breaking, fix an arbitrarily small $\varepsilon > 0$ and consider unit-capacity servers on the line at

$$-(1 + \varepsilon),\ 1,\ 3,\ 7,\ \ldots,\ 2^{n-1} - 1,$$

with requests arriving at

$$0,\ 1,\ 3,\ 7,\ \ldots,\ 2^{n-1} - 1.$$

For request 0, server 1 is uniquely nearest. Subsequently, whenever a request arrives at an occupied server, the next server to its right is closer than the remaining server at $-1 - \varepsilon$. Greedy therefore assigns request 0 to server 1, request 1 to server 3, and so on, leaving the final request to be assigned to server $-1 - \varepsilon$. Its costs are $1,\ 2,\ 4,\ \ldots,\ 2^{n-2},\ 2^{n-1} + \varepsilon$, adding up to $2^n - 1 + \varepsilon$. The optimum instead assigns the first request to the server at $-(1 + \varepsilon)$ and every other request to its own location, at total cost $1 + \varepsilon$. Hence the competitive ratio is $\left(\frac{2^n - 1 + \varepsilon}{1 + \varepsilon}\right)$ which approaches $2^n - 1$ as $\varepsilon \to 0$. Each greedy decision is locally optimal but consumes capacity that the next request could have used at zero cost, causing a cascade of successively longer assignments. Thus, a rule that performs well on many inputs can be exploited by an adversarial sequence.

A retrospective alternative to Greedy maintains an optimal offline matching of the requests seen so far and uses it to guide the next irrevocable assignment. For online metric matching, the Permutation algorithm (Kalyanasundaram & Pruhs, 1993) follows this approach and achieves the optimal deterministic competitive ratio of $2n - 1$. The RM algorithm (Raghvendra, 2016) instead combines immediate assignment costs with retrospective information. Using a parameter $t > 1$, it balances the cost of serving the current request against the quality of a maintained auxiliary matching. Its analysis also adapts to the underlying metric, yielding competitive ratios of $O(\sqrt{n} \log^2 n)$ in two-dimensional Euclidean space and $O(n^{1-1/d} \log^2 n)$ in metrics of doubling dimension $d > 1$ (Nayyar & Raghvendra, 2017).

These guarantees for online matching do not directly provide the desired guarantees for online transportation, where the number of server locations k can be much smaller than the total capacity n. Representing each unit of capacity as a separate server allows matching algorithms to be applied, but their general-metric guarantees may still depend on n. Kalyanasundaram & Pruhs (1998) conjectured that online transportation admits a deterministic $(2k - 1)$-competitive algorithm, matching the known lower bound. Obtaining a guarantee linear in k, independently of the total capacity, remained open for more than two decades. Recent work has made substantial progress toward this conjecture. Harada & Itoh (2025) gave a deterministic $8k - 5$-competitive algorithm for online transportation and Arndt et al. (2026) simplified this algorithm and improved the analysis to $8k - 7$.

The difficulty of extending guarantees on matching algorithms to the transportation setting is particularly clear for the Permutation algorithm. Despite being worst-case optimal for online metric matching, its competitive ratio can be as large as $2n - 1$ for the online transportation problem on the line with only four distinct server locations (Nicoletti et al., 2026). For RM, Arndt et al. (2026) observe an $O(k \log^2 n)$ bound and identify an $O(k)$ guarantee as an open analysis question. RM also retains metric-sensitive guarantees, whereas comparable guarantees are not known for the transportation algorithms of Harada and Itoh or Arndt et al.

Although these algorithms provide strong competitive guarantees, none of the online matching or transportation algorithms are known to guarantee that a fixed positive fraction of their total online cost comes from nearest-neighbor greedy assignments. This motivates the following question:

> *Can we design an online metric transportation algorithm that favors greedy assignments while providing strong worst-case guarantees?*

Our Contribution. We introduce *Robustified Greedy* (RG), a parameterized generalization of RM. As in RM, the algorithm maintains an irrevocable online matching and an auxiliary offline matching. RG uses a second parameter $\alpha \geq 1$. Setting $\alpha = t$ recovers RM, whereas $\alpha < t$ favors assignments to nearest available servers.

Theorem 1. *The competitive ratio of RG is upper bounded by* $6.6604k - 2.89$.

This guarantee is independent of the total capacity n and improves upon the previous best-known bound of $8k - 7$ by Arndt et al. (2026). We provide a somewhat simpler analysis that yields a competitive ratio of $(20/3)k - 3$ in the main paper while the tighter analysis appears in Appendix A.4.

Using our analysis framework, we could also prove a competitive ratio of $11.66k$ for the RM algorithm ($\alpha = t$ regime for RG), thereby resolving a question raised in Arndt et al. (2026) (see Appendix B. However, we give a much more nuanced analysis for RG, utilizing the greedy choices that the algorithm explicitly makes. Favoring such greedy assignments creates persistent structure in the auxiliary matching - every positive-cost greedy edge remains unchanged until its server becomes saturated. We track this structure through *server saturation epochs* to obtain the improved competitive bound of $6.6604k$ for $t = 2 + \sqrt{5}, \alpha = \sqrt{t}$.

We also recover the metric-sensitive guarantees of RM for the RG algorithm.

Theorem 2 (Metric-sensitive guarantee)**.** *For every fixed $t > 1$ and $1 \le \alpha \le t$, RG has competitive ratio $O\big(k^{1-1/d} \log^2 n\big)$ in d-dimensional Euclidean spaces, for every fixed $d > 1$.*

The parameter α biases the RG algorithm towards making more nearest neighbor choices. Indeed, we show that with suitable choice of parameters, a significant fraction of the online cost of RG comes from nearest neighbor assignments, while it still retains a guarantee of $O(k)$ for the competitive ratio. However, greedy assignments increase the cost of the maintained auxiliary offline matching and the accumulated cost can potentially start affecting the online cost. The algorithm makes a non-greedy assignment *only when it finds evidence that this accumulated cost can be reduced*.

Theorem 3. *Suppose $t \ge 3 + 2\sqrt{2}$ and $1 \le \alpha \le \sqrt{t}$ are fixed constants. Then (a) RG algorithm is $O(k)$-competitive, and, (b) The total fraction of online cost of RG from greedy assignments is at least $\left(\frac{1}{2} - \frac{\alpha+1}{2(t-\alpha)}\right)$.*

Note that for fixed $\alpha = o(t)$, the guaranteed fraction in Theorem 3(b) approaches 1/2 as t increases. Complete proofs of Theorems 2 and 3 appear in Appendices C and A.4 respectively.

Experiments. Experiments on real-world datasets show that RG retains Greedy's strong performance on natural arrival orders while being more robust to correlated arrivals, improving over Greedy by up to 15.02%. RG also consistently outperforms the algorithm due to Arndt et al. (2026) while preserving a high fraction of nearest-neighbor assignments.

Preliminaries. Let (X, d) be a metric space, where $d(u, v)$ denotes the distance between any two points $u, v \in X$. For a set of edges E, define its *cost* by $w(E) := \sum_{(u,v)\in E} d(u, v)$.

Recall that there are k servers in S, each with a positive integral capacity, and that the total capacity of the servers is n. Let R denote a set of n requests. A *matching* M assigns requests in R to servers in S such that each request is incident to at most one edge of M, and the number of edges in M incident to any server does not exceed its capacity. A request is *free* with respect to M if it is unmatched, while a server is *free* if it has unused capacity. A server whose capacity is fully used is said to be *saturated*. The *size* of M, denoted by $|M|$, is the number of requests matched by M. Let OPT denote the cost of a minimum-cost matching of all n requests to the servers, subject to their capacities.

An *alternating path* with respect to a matching M is a path whose edges alternate between edges in M and edges not in M. An *augmenting path* is a simple alternating path that starts at a free request and ends at a free server. Given an augmenting path P, augmenting M along P removes the edges $M \cap P$ from M and adds the edges $P \setminus M$. We denote the resulting matching by $M \leftarrow M \oplus P$. Augmenting along an augmenting path increases the size of the matching by exactly one.

Fix parameters $t > 1$ and $1 \le \alpha \le t$. An augmenting path consisting of a single edge between a free request and a free server is called a *direct augmenting path*. All other augmenting paths contain at least three edges and are called *non-direct augmenting paths*.

The (t, α)-net-cost of an augmenting path P with respect to M^* is

$$\phi^{M^*}_{t,\alpha}(P) = \begin{cases} \alpha d(r, s), & P = \{(r, s)\} \text{ is direct,} \\ t\, w(P \setminus M^*) - w(P \cap M^*), & P \text{ is non-direct.} \end{cases} \tag{1}$$

For simplicity, we omit the superscript M^* when the matching is clear from the context and write $\phi_{t,\alpha}(P)$. When $\alpha = t$, this definition coincides with the t-net-cost used in the Robust Matching algorithm from Nayyar & Raghvendra (2017).

2 Robustified Greedy Algorithm

Fix parameters $t > 1$ and $1 \le \alpha \le t$. Our algorithm maintains an online matching M and an offline matching M^*, both initially empty. After each arrival, both matchings serve all requests received so far and assign the same number of requests to each server. To process a new request r, the algorithm performs the following steps:

1. Compute an augmenting path P from r to a free server s that minimizes the (t, α)-net-cost with respect to M^*.
2. Augment the offline matching along P, setting $M^* \leftarrow M^* \oplus P$.
3. Assign r to s in the online matching, setting $M \leftarrow M \cup \{(r, s)\}$.

These updates ensure that M and M^* continue to assign the same number of requests to each server. Augmenting M^* increases the number assigned to the terminal server s by one and leaves this number unchanged at every other server. The online assignment makes the same change in M. Thus, the two matchings have identical remaining capacities and the same free servers after every arrival.

For a request r, the (t, α)-net-cost of a direct augmenting path to a free server s is $\alpha d(r, s)$. This cost is minimized by a nearest free server. Therefore, whenever the algorithm selects a direct augmenting path, it makes a greedy assignment.

When $\alpha = t$, the (t, α)-net-cost coincides with the t-net-cost defined in Raghvendra (2016), so our algorithm recovers the Robust Matching algorithm. Choosing $\alpha < t$ discounts direct augmenting paths while leaving the cost of non-direct paths unchanged, thereby favoring nearest-server assignments.

The following lemma relates the online matching cost to the total (t, α)-net-cost of the selected paths and the contribution of greedy choices. It is central both to explaining the algorithm's decisions and to analyzing its competitive ratio.

Consider any consecutive sequence of arrivals. Let M^*_{start} and M^*_{end} be the offline matchings immediately before and after this sequence. Let G be the total weight of the selected greedy edges, and let Φ be the sum of the (t, α)-net-costs of all selected paths.

Lemma 4 (Cost accounting)**.** *The online cost incurred during the sequence is at most*

$$\frac{2\Phi + 2(t-\alpha)G + (t+1)\big(w(M^*_{\text{start}}) - w(M^*_{\text{end}})\big)}{t-1}. \tag{2}$$

Proof. For each selected non-direct path P, let

$$I = w(P \setminus M^*), \qquad D = w(P \cap M^*),$$

where M^* is the matching immediately before augmentation. Since $w(P) = I + D$, $\phi_{t,\alpha}(P) = tI - D$, and augmentation changes $w(M^*)$ by $I - D$,

$$2\phi_{t,\alpha}(P) = 2(tI - D) = (t-1)w(P) + (t+1)(I - D)$$

Let L be the total weight of the selected non-direct paths. The direct paths contribute αG to Φ and increase $w(M^*)$ by G; hence the total change due to the non-direct paths is $w(M^*_{\text{end}}) - w(M^*_{\text{start}}) - G$. Summing the preceding identity gives

$$2\Phi = 2\alpha G + (t-1)L + (t+1)\big(w(M^*_{\text{end}}) - w(M^*_{\text{start}}) - G\big).$$

Therefore,

$$(t-1)(G+L) = 2\Phi + 2(t-\alpha)G + (t+1)\big(w(M^*_{\text{start}}) - w(M^*_{\text{end}})\big).$$

Each online edge joins the endpoints of its selected path and hence, by the triangle inequality, costs at most the path weight. The online cost is therefore at most $G + L$, and the result follows. □

2.1 Greedy Assignment Fraction

The following structural guarantees are independent of the competitive-ratio analysis; their proofs are deferred to Appendix A.

Lemma 5. *Suppose $1 \le \alpha < (t-1)/2$. On each arrival, the algorithm either selects a greedy edge or selects a non-direct path P satisfying*

$$w(M^*) - w(M^* \oplus P) \ge \frac{t-1-2\alpha}{t+1}\, w(P).$$

Lemma 6 (Greedy-cost bound)**.** *Let G be the total cost of selected greedy edges, and let M and M^* be the final online and offline matchings. For $1 \le \alpha < (t-1)/2$,*

$$G \ge \left(\frac{1}{2} - \frac{\alpha+1}{2(t-\alpha)}\right) w(M).$$

Proof. On each arrival, the direct edge to a nearest free server is a candidate augmenting path. The selected path therefore has net-cost at most α times the nearest-free-server distance. This distance is at most the cost of the online assignment, since the assignment also uses a free server. Summing over all arrivals gives $\Phi \le \alpha w(M)$. Applying Lemma 4 to the entire execution, whose initial offline matching is empty, yields

$$(t-1)w(M) \le 2\Phi + 2(t-\alpha)G - (t+1)w(M^*).$$

Substituting $\Phi \le \alpha w(M)$ and rearranging gives

$$2(t-\alpha)G \ge (t-1-2\alpha)w(M) + (t+1)w(M^*).$$

The lemma follows using $w(M^*) \ge 0$. □

Lemma 5 quantifies the decrease in the offline matching cost behind every non-greedy choice, while Lemma 6 shows that the fraction of online cost contributed by greedy edges approaches $1/2$ for fixed α, or more generally $\alpha = o(t)$. This positive fraction relies on discounting direct paths: when $\alpha = t$ (RM), the same argument gives no positive lower bound on G.

3 Primal-Dual Implementation

In this section, we present a primal-dual implementation of the Robustified-Greedy algorithm that processes each request in $O(nk \log n)$ time. More importantly, the evolution of the dual weights maintained by the algorithm plays a critical role in our analysis of its competitive ratio (see Section 4).

Feasibility. Recall that each server $s \in S$ has an integer capacity b_s. The algorithm maintains a dual weight $y(v)$ for every request and server v.

A matching M^*, together with the dual weights $y(\cdot)$, is (t, α)-*feasible* if every request-server pair (r, s) satisfies

$$y(r) + y(s) \le \begin{cases} \alpha d(r,s), & \text{if both } r \text{ and } s \text{ are free,} \\ t\, d(r,s), & \text{otherwise,} \end{cases} \tag{3}$$

and every matching edge satisfies

$$y(r) + y(s) \ge d(r,s) \qquad \text{for every } (r,s) \in M^*. \tag{4}$$

Slack. For each non-matching edge (r, s), define its slack as

$$\sigma(r,s) = \begin{cases} \alpha d(r,s) - y(r) - y(s), & \text{if both } r \text{ and } s \text{ are free,} \\ t\, d(r,s) - y(r) - y(s), & \text{otherwise.} \end{cases} \tag{5}$$

For each matching edge (r, s), define its slack as

$$\sigma(s,r) = y(r) + y(s) - d(r,s). \tag{6}$$

Feasibility guarantees that all these slacks are nonnegative.

Residual graph. Given a (t, α)-feasible matching M^* and its dual weights, we construct the residual graph G_{M^*} on the requests and servers, together with an auxiliary sink vertex z. Each non-matching edge (r, s) is directed from r to s and assigned weight $\sigma(r, s)$. Each matching edge (r, s) is directed from s to r and assigned weight $\sigma(s, r)$. Finally, we add a directed edge of weight zero from every free server to z.

3.1 Processing a request via primal-dual algorithm

Initially, before any request has arrived the matching is empty and the dual weights for every server is set to zero. As requests arrive, our algorithm will update the dual weights to find the desired augmenting path. We describe this procedure for any request r.

Before each arrival, the algorithm maintains a set of dual weights $y(\cdot)$ on all servers and requests and a (t, α)-feasible matching M^* When a new request r arrives, we initialize $y(r) = 0$, construct the residual graph G_{M^*}, and perform the following steps.

1. **Find an augmenting path.** Execute Dijkstra's shortest-path algorithm from r in G_{M^*}. Let $\ell(v)$ denote the shortest-path distance from r to v, and let $\delta = \ell(z)$. Choose a shortest r-to-z path and truncate it at its first free server s to obtain P.
2. **Update the dual weights.** For each request or server v with $\ell(v) < \delta$, update
$$\begin{aligned} y(v) &\leftarrow y(v) + \delta - \ell(v), \quad \text{if } v \in R, \\ y(v) &\leftarrow y(v) - \delta + \ell(v), \quad \text{if } v \in S. \end{aligned} \tag{7}$$
All other dual weights remain unchanged.
3. **Augment the matchings.** Assign r to s in the online matching and augment the offline matching along P:
$$M \leftarrow M \cup \{(r, s)\}, \qquad M^* \leftarrow M^* \oplus P.$$

Every server is initialized with dual weight zero. Each dual update decreases the weight of a server s by $\delta - \ell(s)$ when $\ell(s) < \delta$ and leaves it unchanged otherwise. Thus, all server dual weights remain nonpositive throughout the execution.

Efficiency. The runtime for the RG algorithm per request arrival is dominated by the time to run Dijkstra's algorithm on a graph with $O(nk)$ edges, which is $O(nk \log n)$ (since $k \le n$).

Lemma 7. *The Robustified Greedy algorithm processes each request in $O(nk \log n)$ time.*

Correctness. Initializing the new request with dual weight zero preserves feasibility because server dual weights are nonpositive. Next, we show that the dual update and augmentation preserve (t, α)-feasibility and that the selected path minimizes the (t, α)-net-cost. The following lemma establishes feasibility after the remaining steps and shows that the selected path has zero slack immediately before augmentation.

Lemma 8. *The dual update and augmentation preserve (t, α)-feasibility. Immediately before augmentation, every residual edge of P has zero slack.*

Corollary 9. *The selected augmenting path P has minimum (t, α)-net-cost.*

3.2 Properties used in the analysis

We first show that direct retain their dual contribution while their server remains free.

Lemma 10. *Suppose $\alpha > 1$. Until a server s becomes saturated, every positive-weight edge (r, s) inserted into M^* by a direct augmenting path remains in M^* and satisfies*
$$y(r) + y(s) = \alpha d(r, s) \tag{8}$$

We next bound the weight of the offline matching by the dual objective.

Lemma 11. *After processing any arrival,*
$$w(M^*) \le \sum_r y(r) + \sum_s b(s) y(s) \le t \operatorname{OPT}, \tag{9}$$
where the first sum ranges over the requests that have arrived.

4 COMPETITIVE RATIO

For the analysis, index requests by their arrival order. Let M_i and M_i^* be the online and offline matchings after arrival i, let P_i be the selected path, and let d_i be the distance from r_i to a nearest free server before it is served. Write $R_i = \{r_1, \dots, r_i\}$ and $\mathrm{ALG} = w(M_n)$. In this section we establish that the competitive ratio of RG is upper bounded by $20k/3 - 3$ for $t = 4, \alpha = 2$.

Saturation Epochs. A saturation event occurs when a server uses its last unit of available capacity. Each arrival increases the load of exactly one server by one. Hence each server becomes saturated once, and at most one server becomes saturated on an arrival. Let $0 = \tau_0 < \tau_1 < \cdots < \tau_k = n$ be the saturation times. Epoch h consists of arrivals $\tau_{h-1} + 1, \dots, \tau_h$. Fix an epoch and write $a = \tau_{h-1}$ and $b = \tau_h$. Immediately before every arrival in this epoch, every server that was free at time a is still free.

Let

$$G = \sum_{\substack{a<i\le b \\ P_i \text{ direct}}} d_i, \qquad \Phi = \sum_{i=a+1}^{b} \phi_{t,\alpha}^{M_{i-1}^*}(P_i).$$

Lemma 12 (Direct-cost bound)**.** *For every epoch,* $w(M_b^*) + (\alpha - 1)G \le t\,\mathrm{OPT}$

Proof. For $\alpha = 1$, the claim follows from Lemma 11. Assume $\alpha > 1$.

For every positive-length direct edge $e = (r, s)$ inserted during the epoch, Lemma 10 implies that $e \in M_b^*$ at the end of the epoch and $y(r) + y(s) = \alpha d(e)$. These edges have total weight G. Every remaining edge $e' = (r', s') \in M_b^*$ satisfies $y(r') + y(s') \ge d(e')$ by the matching-edge lower bound (4). Therefore,

$$\begin{aligned} w(M_b^*) + (\alpha - 1)G &\le \sum_{(r,s)\in M_b^*} \big(y(r) + y(s)\big) \\ &= \sum_{r\in R_b} y(r) + \sum_{s} \deg_{M_b^*}(s) y(s) = \sum_{r\in R_b} y(r) + \sum_{s} b(s) y(s). \end{aligned}$$

The last equality holds because an unsaturated server has $y(s) = 0$, whereas a saturated server has $\deg_{M_b^*}(s) = b(s)$. Lemma 11 bounds the final expression by $t\,\mathrm{OPT}$. □

Lemma 13 (Nearest-free-distance bound)**.** *For every epoch,* $\sum_{i=a+1}^{b} d_i \le \mathrm{OPT} + w(M_a^*)$

Proof. Fix a final optimal matching and let B be its restriction to R_b. In $M_a^* \oplus B$, direct the edges of $B \setminus M_a^*$ from requests to servers and the edges of $M_a^* \setminus B$ from servers to requests. Each old request has equal indegree and outdegree, whereas each new request has one outgoing edge and no incoming edge. At a server s, indegree minus outdegree equals $\deg_B(s) - \deg_{M_a^*}(s)$.

Decompose this directed graph into edge-disjoint paths from vertices with excess outgoing edges to vertices with excess incoming edges, together with cycles. This can be obtained by pairing incoming and outgoing edges at each vertex and removing cycles. Each new request r_i starts one such path, which must end at a server s_i with $\deg_B(s_i) > \deg_{M_a^*}(s_i)$. Since $\deg_B(s_i) \le b(s_i)$, that server is free at time a.

No server that is free at time a becomes saturated before the last arrival of the epoch. Thus s_i is available immediately before arrival i. The triangle inequality gives

$$d_i \le d(r_i, s_i) \le w(\text{path from } r_i \text{ to } s_i).$$

Summing over the new requests and using edge-disjointness gives $\sum_{i=a+1}^{b} d_i \le w(M_a^*) + w(B) \le w(M_a^*) + \mathrm{OPT}$. □

A direct edge to a nearest free server has net-cost αd_i, so minimum-net-cost selection gives $\phi_{t,\alpha}(P_i) \le \alpha d_i$. A selected direct path consequently has weight d_i. Summing and applying the nearest-free-distance bound gives

$$\Phi \le \alpha(\mathrm{OPT} + w(M_a^*)). \tag{10}$$

Set $t = 4$ and $\alpha = 2$. Lemma (12) and (10) give

$$w(M_b^*) + G \le 4\,\mathrm{OPT}, \qquad \Phi \le 2(\mathrm{OPT} + w(M_a^*)).$$

Consequently,

$$\begin{aligned} \Phi + 2G &\le 2(\mathrm{OPT} + w(M_a^*)) + 2(4\,\mathrm{OPT} - w(M_b^*)) \\ &= 10\,\mathrm{OPT} + 2(w(M_a^*) - w(M_b^*)). \end{aligned} \tag{11}$$

Applying Lemma 4 with $t = 4$ and $\alpha = 2$, the online cost incurred during the epoch satisfies

$$w(M_b) - w(M_a) \le \frac{2\Phi + 4G + 5\big(w(M_a^*) - w(M_b^*)\big)}{3}.$$

Using (11), we obtain

$$w(M_b) - w(M_a) \le \frac{20}{3}\mathrm{OPT} + 3\big(w(M_a^*) - w(M_b^*)\big).$$

Summing this inequality over the k epochs and using $w(M_0^*) = 0$ and $w(M_n^*) \ge \mathrm{OPT}$ gives

$$\mathrm{ALG} \le \frac{20}{3}k\,\mathrm{OPT} + 3\big(w(M_0^*) - w(M_n^*)\big) \le \left(\frac{20}{3}k - 3\right)\mathrm{OPT}.$$

5 Experiments

Setup. We compare RG with Greedy, RM, the algorithm of Arndt et al. (2026), and the offline optimum (OPT). Every method receives the same requests, sites, capacities, and arrival order. We report mean cumulative assignment cost, per-request runtime, and the fraction of decisions and cost attributable to nearest-available-server assignments for RG and, on NYC, Arndt et al. (ties count as nearest). Results are averaged over ten instances of $10{,}000$ requests. Our C++ implementation runs on one core of an Apple M2 processor with 8 GB RAM.

Datasets. NYC-Taxi uses the samples of Seth et al. (2026) from the NYC Taxi data (H, 2021): pickups are requests, drop-offs form the server-side point cloud, and costs are Euclidean distances in the original two-dimensional coordinates. Each sample contains $10{,}000$ trips from the first week of a fixed month. **MovieLens-25M** (Harper & Konstan, 2015) uses the 1,128-dimensional Tag Genome vectors (Vig et al., 2012) available for 13,816 movies. We standardize each tag coordinate and normalize each movie vector. A user's location is the normalized weighted sum of genome movies that the user rated at least four stars, with weight equal to the rating minus three. Restricting to users with at least 20 such ratings leaves 13,758 request vectors; the server-side points are the 10,000 genome movies with the most ratings. We fit PCA on these movies, map movies and users to 100 dimensions, and normalize them. For both datasets, k-means clusters the 10,000 server-side points into k sites, with each site's capacity equal to its cluster size; clustering is rerun whenever k changes.

Arrival orders and parameters. For NYC, we use both the natural timestamp order and a *directional* order obtained by sorting the same requests by their y-coordinate. The latter is a simple, non-adaptive stress test for spatially correlated arrivals: it changes only the order, while retaining every real request and server location. We set $k = 500$, $t = 6$, and $\alpha \in \{1, 1.5, 2\}$; we also vary $k \in \{50, 100, 200, 500\}$ at $(t, \alpha) = (6, 2)$. For MovieLens, we use a directional arrival order with $k = 500$, $t = 1.12$, and $\alpha \in \{1.02, 1.05, 1.08, 1.10\}$. For computational efficiency, all online algorithms operate on the same fixed 12-dimensional Gaussian projection, while their resulting assignments and OPT are evaluated in the 100-dimensional reference space.

Assignment cost. On NYC's timestamp traces, RG stays within 0.5% of Greedy for every tested α, while both cost about 31% less than Arndt et al. (Figure 1(a) and Appendix D). RM costs 1.0% more than Greedy; but still costs about 31% less than Arndt et al. This is the expected behavior on natural traces, where Greedy is already known to be empirically strong (Tong et al., 2016): RG preserves that favorable behavior while retaining its worst-case guarantee.The directional trace exposes the benefit of robustification: as α increases from 1 to 2, RG's reduction in final cost relative to Greedy grows from 10.75% to 15.02% (Figure 1(b)). The gain also strengthens with the number of sites, rising from 8.54% at $k = 50$ to 15.02% at $k = 500$ (Figure 1(c)). RM extends this trend to its limit; at $k = 500$, $t = 6$, it reduces cost by 24.3% relative to Greedy.

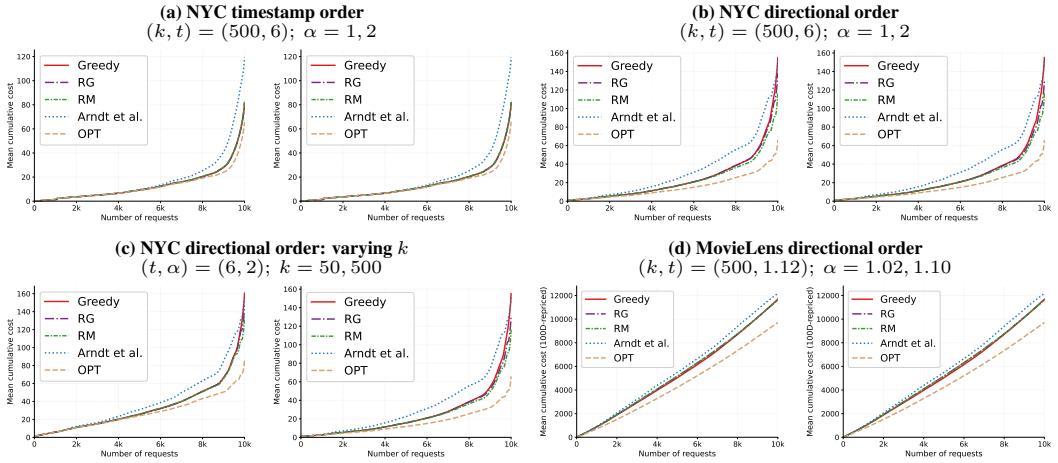

Figure 1: Mean cumulative assignment cost over ten instances, shown at the endpoints of each parameter sweep. RG tracks Greedy on NYC's timestamp order, but is distinctly better under directional ordering; this advantage grows with k. On MovieLens, RG consistently improves on Greedy and both outperform Arndt et al. Complete sweeps appear in Appendix D.

On MovieLens, RG costs less than Greedy in all ten repetitions for every tested $\alpha, t = 1.12$, with mean reductions between 0.41% and 0.51% across the sweep (Figure 1(d) and Appendix D). Arndt et al. has higher cost than both methods in every run. RG therefore retains a consistent edge even in this high-dimensional setting where Greedy is already a strong baseline Tong et al. (2016). For MovieLens, we show the results for $t = 1.12$ in the main text since it reveals a regime in which RG makes a substantial fraction of non-greedy assignments. At $t = 6$, as used for NYC, RG matches Greedy's cost on MovieLens, while both outperform Arndt et al.; these results appear in Appendix D. Additional results for other tested value of t, α and k on both the datasets are reported in Appendix D.

Runtime and nearest-neighbor behavior.

Runtime per request (ms)

Algorithm	NYC	MovieLens
RG	0.04291	0.2647
RM	0.06312	0.2674
Greedy	0.00114	0.0018
Arndt et al.	0.00152	0.0033
OPT	0.5403	1.1660

Nearest-neighbor contribution

Dataset	Algorithm	α	Decisions (%)	Cost (%)
NYC	RG ($t = 6$)	1.00	98.05	87.78
		1.50	97.62	85.48
		2.00	97.37	84.05
	RM ($t = 6$)	–	93.44	62.46
	Arndt et al.	–	51.34	9.06
MovieLens	RG ($t = 1.12$)	1.02	63.76	52.53
		1.10	51.79	40.79
	RM ($t = 1.12$)	–	47.40	36.66
	Arndt et al.	–	14.10	7.17

Runtime trade-off. The runtime and nearest-neighbor entries are averaged over ten instances of 10,000 requests; the runtime comparison uses $(t, \alpha) = (6, 2)$ for NYC and $(1.12, 1.10)$ for MovieLens. RG and RM performs more computation than the two online baselines, while both remaining substantially faster than exact offline optimization. This is the computational trade-off for simultaneously obtaining a robust competitive guarantee and explicitly favoring nearest-neighbor assignments. The latter remain dominant: they constitute 97.37–98.05% of RG's NYC decisions and 51.79–63.76% of its MovieLens decisions. For comparison, Arndt et al.'s nearest-server assignments on NYC constitute only 51.34% of its decisions and 9.06% of its cost.

Key takeaway. RG preserves Greedy's strong performance on natural traces, but is markedly more stable when arrivals have even mild, fixed spatial directionality: on unchanged NYC locations it improves over Greedy by up to 15.02% in the regime $t = 6, \alpha < (t - 1)/2$, with a widening advantage as k grows. Across both datasets it also consistently outperforms Arndt et al., while a majority of its decisions remain nearest-neighbor assignments.

AI USE STATEMENT

In this work, we used generative AI tools for formulating mathematical claims, provide critical ingredients for proving mathematical claims and assisting in the writing of proofs, implementing methods and provide feedback on experiments. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, propose or refine hypotheses, clean and reformat dataset, interpret results, and 'assist with translation, support qualitative and thematic data analysis' are not applicable to this work. We used generative AI tools for brainstorming proof ideations. Specifically, the authors designed the proposed RG algorithms and formalized an analysis framework by themselves. LLMs were then used for parametric search in order to provide the tightest possible bound. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have uploaded code for the implementation of all algorithms used in our experiments as supplementary material.

REFERENCES

Stephen Arndt, Benjamin Moseley, Kirk Pruhs, and Marc Uetz. Competitive online transportation simplified. In *Proceedings of the SIAM Symposium on Simplicity in Algorithms (SOSA 2026)*, 2026.

Eric Balkanski, Yuri Faenza, and Noemie Périvier. The power of greedy for online minimum cost matching on the line. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC)*, pp. 185–205, 2023. doi: 10.1145/3580507.3597794.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. In *International Conference on Learning Representations*, 2021.

M Yasser H. Nyc taxi trip duration. `https://www.kaggle.com/datasets/yasserh/nyc-taxi-trip-duration`, 2021. Kaggle.

Tsubasa Harada and Toshiya Itoh. A nearly optimal deterministic algorithm for the online transportation problem. In *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *LIPIcs*, pp. 94:1–94:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015. doi: 10.1145/2827872.

Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3): 478–488, 1993.

Bala Kalyanasundaram and Kirk Pruhs. On-line network optimization problems. In Amos Fiat and Gerhard J. Woeginger (eds.), *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pp. 268–280. Springer, 1998. doi: 10.1007/BFb0029573.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274, 2021.

Yuantong Li, Guang Cheng, and Xiaowu Dai. Two-sided competing matching recommendation markets with quota and complementary preferences constraints. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 27589–27614, 2024.

Martin Mladenov, Elliot Creager, Omer Ben-Porat, Kevin Swersky, Richard Zemel, and Craig Boutilier. Optimizing long-term social welfare in recommender systems: A constrained matching approach. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6987–6998, 2020.

Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 505–515, 2017. doi: 10.1109/FOCS.2017.53.

Alberto Nicoletti, Enoch Peserico, and Michele Scquizzato. Permutation for online metric matching with m distinct servers. *Operations Research Letters*, 68:107484, 2026. doi: 10.1016/j.orl.2026.107484.

Sharath Raghvendra. A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans (eds.), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*, volume 60 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 18:1–18:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-018-7. doi: 10.4230/LIPIcs.APPROX-RANDOM.2016.18. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX-RANDOM.2016.18.

Ritesh Seth, Mrinal Garg, Sujoy Bhore, Sharath Raghvendra, and Syamantak Das. Efficient algorithms for incremental metric bipartite matching. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wnIanx0r0w.

Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. doi: 10.1145/2786.2793.

Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. Online minimum matching in real-time spatial data: Experiments and analysis. *Proceedings of the VLDB Endowment*, 9(12):1053–1064, 2016.

Jesse Vig, Shilad Sen, and John Riedl. The Tag Genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems*, 2(3):13:1–13:44, 2012. doi: 10.1145/2362394.2362395.

Mingwei Yang and Sophie H. Yu. Optimal analysis of greedy for stochastic online euclidean matching. *arXiv preprint arXiv:2609.23947*, 2026.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y. Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7103–7114, 2022.

A Missing Proofs from Sections 2–4

We restate and prove all lemmas from Sections 2–4. We conclude with the tighter parameter analysis used to prove Theorem 1.

A.1 Proofs for Robustified Greedy

Consider any consecutive sequence of arrivals. Let M^*_{start} and M^*_{end} be the offline matchings immediately before and after this sequence. Let G be the total weight of the selected greedy edges, and let Φ be the sum of the (t, α)-net-costs of all selected paths.

Lemma 4 (Cost accounting)**.** *The online cost incurred during the sequence is at most*

$$\frac{2\Phi + 2(t-\alpha)G + (t+1)\big(w(M^*_{\text{start}}) - w(M^*_{\text{end}})\big)}{t-1}. \tag{2}$$

Proof. For each selected non-direct path P, let

$$I = w(P \setminus M^*), \qquad D = w(P \cap M^*),$$

where M^* is the matching immediately before augmentation. Since $w(P) = I + D$, $\phi_{t,\alpha}(P) = tI - D$, and augmentation changes $w(M^*)$ by $I - D$,

$$2\phi_{t,\alpha}(P) = 2(tI - D) = (t-1)w(P) + (t+1)(I - D)$$

Let L be the total weight of the selected non-direct paths. The direct paths contribute αG to Φ and increase $w(M^*)$ by G; hence the total change due to the non-direct paths is $w(M^*_{\text{end}}) - w(M^*_{\text{start}}) - G$. Summing the preceding identity gives

$$2\Phi = 2\alpha G + (t-1)L + (t+1)\big(w(M^*_{\text{end}}) - w(M^*_{\text{start}}) - G\big).$$

Therefore,

$$(t-1)(G+L) = 2\Phi + 2(t-\alpha)G + (t+1)\big(w(M^*_{\text{start}}) - w(M^*_{\text{end}})\big).$$

Each online edge joins the endpoints of its selected path and hence, by the triangle inequality, costs at most the path weight. The online cost is therefore at most $G + L$, and the result follows. □

Lemma 5. *Suppose $1 \le \alpha < (t-1)/2$. On each arrival, the algorithm either selects a greedy edge or selects a non-direct path P satisfying*

$$w(M^*) - w(M^* \oplus P) \ge \frac{t-1-2\alpha}{t+1}\, w(P).$$

Proof. Suppose that P is non-direct, and write

$$I = w(P \setminus M^*), \qquad D = w(P \cap M^*).$$

If d is the distance from the new request to a nearest free server, then the corresponding direct edge has net-cost αd. Since P minimizes net-cost and its terminal server is free, the triangle inequality gives

$$tI - D \le \alpha d \le \alpha w(P) = \alpha(I + D).$$

Using the identity from Lemma 4,

$$(t+1)(D-I) = (t-1)(I+D) - 2(tI - D) \ge (t-1-2\alpha)(I+D).$$

Therefore,

$$D - I \ge \frac{t-1-2\alpha}{t+1}(I+D).$$

Augmentation decreases the offline matching cost by $D - I$, and the coefficient is positive under the stated assumption. □

Lemma 6 (Greedy-cost bound)**.** *Let G be the total cost of selected greedy edges, and let M and M^* be the final online and offline matchings. For $1 \le \alpha < (t-1)/2$,*

$$G \ge \left(\frac{1}{2} - \frac{\alpha+1}{2(t-\alpha)}\right) w(M).$$

Proof. On each arrival, the direct edge to a nearest free server is a candidate augmenting path. The selected path therefore has net-cost at most α times the nearest-free-server distance. This distance is at most the cost of the online assignment, since the assignment also uses a free server. Summing over all arrivals gives

$$\Phi \le \alpha w(M).$$

Applying Lemma 4 to the entire execution, whose initial offline matching is empty, yields

$$(t-1)w(M) \le 2\Phi + 2(t-\alpha)G - (t+1)w(M^*).$$

Substituting $\Phi \le \alpha w(M)$ and rearranging gives

$$2(t-\alpha)G \ge (t-1-2\alpha)w(M) + (t+1)w(M^*).$$

Since $\alpha < t$, division by $2(t-\alpha)$ proves the first inequality. The second follows from $w(M^*) \ge 0$ and

$$\frac{t-1-2\alpha}{2(t-\alpha)} = \frac{1}{2} - \frac{\alpha+1}{2(t-\alpha)}.$$

□

A.2 PROOFS FOR THE PRIMAL-DUAL IMPLEMENTATION

Lemma 7. *The Robustified Greedy algorithm processes each request in $O(nk \log n)$ time.*

Proof. The residual graph has $O(nk)$ edges. Executing Dijkstra's algorithm to find an augmenting path takes $O(nk \log n)$ time. Updating the dual weights and augmenting the matching each take $O(n)$ time. Therefore, each request is processed in $O(nk \log n)$ time. □

Lemma 8. *The dual update and augmentation preserve (t, α)-feasibility. Immediately before augmentation, every residual edge of P has zero slack.*

Proof. For the analysis, define

$$\bar{\ell}(v) = \min\{\ell(v), \delta\},$$

where $\bar{\ell}(v) = \delta$ if v is unreachable. The update adds $\delta - \bar{\ell}(v)$ to each request weight and subtracts $\delta - \bar{\ell}(v)$ from each server weight. Let y' denote the updated weights, and let σ denote the slacks before the update.

Consider a non-matching edge (r', s'), directed from r' to s'. The shortest-path inequality gives

$$\ell(s') \le \ell(r') + \sigma(r', s').$$

Since $\sigma(r', s') \ge 0$, it follows that

$$\bar{\ell}(s') - \bar{\ell}(r') \le \sigma(r', s').$$

The change in its dual sum is

$$\begin{aligned} y'(r') + y'(s') - y(r') - y(s') &= (\delta - \bar{\ell}(r')) - (\delta - \bar{\ell}(s')) \\ &= \bar{\ell}(s') - \bar{\ell}(r') \\ &\le \sigma(r', s'). \end{aligned}$$

Thus, the dual sum increases by at most its slack, preserving the upper-bound constraint.

Now consider a matching edge (r', s'), directed from s' to r'. The shortest-path inequality gives

$$\ell(r') \le \ell(s') + \sigma(s', r'),$$

and hence

$$\bar{\ell}(r') - \bar{\ell}(s') \le \sigma(s', r').$$

Therefore,

$$\begin{aligned} y'(r') + y'(s') - d(r', s') &= \sigma(s', r') + \bar{\ell}(s') - \bar{\ell}(r') \\ &\ge 0. \end{aligned}$$

This preserves the matching-edge lower bound.

We must also verify the upper bound on each matching edge. The only incoming residual edge of the matched request r' is its edge from s'. Consequently, when these vertices are reachable,

$$\ell(r') = \ell(s') + \sigma(s', r') \ge \ell(s').$$

Thus, $\bar{\ell}(r') \ge \bar{\ell}(s')$. If the vertices are unreachable, both truncated distances equal δ. In either case,

$$\begin{aligned} y'(r') + y'(s') &= y(r') + y(s') + \bar{\ell}(s') - \bar{\ell}(r') \\ &\le y(r') + y(s') \\ &\le t\, d(r', s'). \end{aligned}$$

Hence the matching-edge upper bound is preserved.

For every free server s', the zero-weight edge $s' \to z$ gives

$$\delta = \ell(z) \le \ell(s').$$

Therefore, free servers receive no update and retain dual weight zero. All server weights remain nonpositive because they only decrease.

Along the selected shortest path, every residual edge $u \to v$ satisfies

$$\ell(v) = \ell(u) + \sigma(u, v).$$

Every vertex on the path has distance at most δ, so $\bar{\ell}(v) = \ell(v)$ there. The slack after the update is therefore

$$\sigma(u, v) + \bar{\ell}(u) - \bar{\ell}(v) = 0.$$

Thus, every residual edge of P has zero slack immediately before augmentation.

It remains to verify feasibility after augmentation. If P is direct, its inserted edge has dual sum $\alpha d(r, s)$. If P is non-direct, it has no internal free server, so none of its inserted edges joins two free vertices before augmentation. Each such edge (r', s') therefore has dual sum $t\, d(r', s')$. Since $1 \le \alpha \le t$, every inserted edge satisfies both matching-edge bounds.

Every deleted edge (r', s') has dual sum $d(r', s')$, because its residual slack is zero. Its request remains matched after augmentation, so its new upper bound is $t\, d(r', s')$, which is satisfied.

Finally, augmentation matches the new request and may saturate the terminal server. No request or server becomes free. These changes can only relax an upper bound from $\alpha d(r', s')$ to $t\, d(r', s')$. All remaining constraints are unchanged. Therefore, feasibility is preserved. □

Corollary 9. *The selected augmenting path P has minimum (t, α)-net-cost.*

Proof. Consider the dual weights and residual slacks immediately after the update and before augmentation. Let Q be any augmenting path from the new request r to a free server s'. Summing the slack definitions along Q cancels the dual contributions of all internal vertices, giving

$$\phi_{t,\alpha}(Q) \ge y(r) + y(s') + \sum_{e \in Q} \sigma(e).$$

The inequality allows for a discounted first edge when a non-direct path first visits a free server: the slack definition uses coefficient α on that edge, whereas the net-cost of a non-direct path uses coefficient t. Since $y(s') = 0$ and all residual slacks are nonnegative,

$$\phi_{t,\alpha}(Q) \ge y(r).$$

The selected path P has no internal free server. Consequently, the first inequality is an equality for P. By Lemma 8, all its residual edges have zero slack. Hence

$$\phi_{t,\alpha}(P) = y(r),$$

which proves that P has minimum (t, α)-net-cost. □

Lemma 10. *Suppose $\alpha > 1$. Until a server s becomes saturated, every positive-weight edge (r, s) inserted into M^* by a direct augmenting path remains in M^* and satisfies*

$$y(r) + y(s) = \alpha d(r, s) \tag{8}$$

Proof. On insertion, the direct edge has zero forward slack, so its dual sum is $\alpha d(r, s)$. Its backward slack is therefore

$$\gamma = (\alpha - 1) d(r, s) > 0.$$

Consider a subsequent search while s is free. Its zero-length edge to z implies $\ell(s) \ge \delta$. The only incoming residual edge of r is its matching edge from s. If reachable, it therefore satisfies

$$\ell(r) = \ell(s) + \gamma > \delta.$$

Neither endpoint receives a dual update. Moreover, a shortest path of length δ cannot traverse this backward edge and delete the matching edge. Unreachable endpoints also receive no update and cannot occur on the selected path.

Applying this argument to successive arrivals proves the claim. □

Lemma 11. *After processing any arrival,*

$$w(M^*) \le \sum_r y(r) + \sum_s b(s) y(s) \le t\, \mathrm{OPT}, \tag{9}$$

where the first sum ranges over the requests that have arrived.

Proof. Let $\deg_{M^*}(s)$ be the number of requests assigned to server s. Summing the matching-edge lower bounds gives

$$w(M^*) \le \sum_r y(r) + \sum_s \deg_{M^*}(s) y(s).$$

If s is saturated, then $\deg_{M^*}(s) = b(s)$. Otherwise, $y(s) = 0$. Thus,

$$w(M^*) \le \sum_r y(r) + \sum_s b(s) y(s).$$

Fix a final optimal matching, and let B be its restriction to the requests that have arrived. Since $\deg_B(s) \le b(s)$ and $y(s) \le 0$,

$$\begin{aligned}\sum_r y(r) + \sum_s b(s) y(s) &\le \sum_r y(r) + \sum_s \deg_B(s) y(s) \\ &= \sum_{(r,s)\in B} \big(y(r) + y(s)\big) \\ &\le t\, w(B) \le t\,\mathrm{OPT}.\end{aligned}$$

The penultimate inequality follows from the pairwise upper bounds. □

A.3 Proofs for the competitive-ratio analysis

Let

$$0 = \tau_0 < \tau_1 < \cdots < \tau_k = n$$

be the saturation times. Epoch h consists of arrivals $\tau_{h-1} + 1, \ldots, \tau_h$.

Fix an epoch and write $a = \tau_{h-1}$ and $b = \tau_h$. Immediately before every arrival in this epoch, every server that was free at time a is still free.

Let

$$G = \sum_{\substack{a<i\le b \\ P_i \text{ direct}}} d_i, \qquad \Phi = \sum_{i=a+1}^{b} \phi_{t,\alpha}^{M_{i-1}^*}(P_i).$$

Lemma 12 (Direct-cost bound)**.** *For every epoch,* $w(M_b^*) + (\alpha - 1)G \le t\,\mathrm{OPT}$

Proof. For $\alpha = 1$, the claim follows from Lemma 11. Assume $\alpha > 1$.

For every positive-length direct edge $e = (r, s)$ inserted during the epoch, Lemma 10 implies that $e \in M_b^*$ at the end of the epoch and

$$y(r) + y(s) = \alpha d(e).$$

These edges have total weight G. Every remaining edge $e' = (r', s') \in M_b^*$ satisfies

$$y(r') + y(s') \ge d(e')$$

by the matching-edge lower bound (4). Therefore,

$$\begin{aligned}w(M_b^*) + (\alpha - 1)G &\le \sum_{(r,s)\in M_b^*} \big(y(r) + y(s)\big) \\ &= \sum_{r\in R_b} y(r) + \sum_s \deg_{M_b^*}(s) y(s) \\ &= \sum_{r\in R_b} y(r) + \sum_s b(s) y(s).\end{aligned}$$

The last equality holds because an unsaturated server has $y(s) = 0$, whereas a saturated server has $\deg_{M_b^*}(s) = b(s)$. Lemma 11 bounds the final expression by $t\,\mathrm{OPT}$. □

Lemma 13 (Nearest-free-distance bound)**.** *For every epoch,* $\sum_{i=a+1}^{b} d_i \le \mathrm{OPT} + w(M_a^*)$

Proof. Fix a final optimal matching and let B be its restriction to R_b. In $M_a^* \oplus B$, direct the edges of $B \setminus M_a^*$ from requests to servers and the edges of $M_a^* \setminus B$ from servers to requests. Each old request has equal indegree and outdegree, whereas each new request has one outgoing edge and no incoming edge. At a server s, indegree minus outdegree equals $\deg_B(s) - \deg_{M_a^*}(s)$.

Decompose this directed graph into edge-disjoint paths from vertices with excess outgoing edges to vertices with excess incoming edges, together with cycles. This can be obtained by pairing incoming and outgoing edges at each vertex and removing cycles. Each new request r_i starts one such path, which must end at a server s_i with $\deg_B(s_i) > \deg_{M_a^*}(s_i)$. Since $\deg_B(s_i) \le b(s_i)$, that server is free at time a.

No server that is free at time a becomes saturated before the last arrival of the epoch. Thus s_i is available immediately before arrival i. The triangle inequality gives

$$d_i \le d(r_i, s_i) \le w(\text{path from } r_i \text{ to } s_i).$$

Summing over the new requests and using edge-disjointness gives

$$\sum_{i=a+1}^{b} d_i \le w(M_a^*) + w(B) \le w(M_a^*) + \mathrm{OPT}.$$

□

A.4 Tighter Analysis Using Parameter Optimization

The epoch bounds hold for every $t > 1$ and $1 \le \alpha \le t$. Choosing $\alpha = \sqrt{t}$ makes the offline matching terms telescope with equal coefficients.

Corollary 14. *For $t > 1$ and $\alpha = \sqrt{t}$,*

$$\mathrm{ALG} \le \left[\frac{2\sqrt{t}(t+1)}{t-1} k - \frac{t+1+2\sqrt{t}}{t-1} \right] \mathrm{OPT}. \tag{12}$$

Proof. Since $\alpha > 1$, the direct-cost bound implies

$$G \le \frac{t\,\mathrm{OPT} - w(M_b^*)}{\alpha - 1}.$$

For $t = \alpha^2$,

$$\frac{t-\alpha}{\alpha-1} = \alpha.$$

Combining this with (10),

$$\begin{aligned} \Phi + (t-\alpha)G &\le \alpha(\mathrm{OPT} + w(M_a^*)) + \alpha(t\,\mathrm{OPT} - w(M_b^*)) \\ &= \alpha(t+1)\mathrm{OPT} + \alpha(w(M_a^*) - w(M_b^*)). \end{aligned}$$

Substitution into Lemma 4 gives

$$(t-1)\big(w(M_b) - w(M_a)\big) \le 2\alpha(t+1)\mathrm{OPT} + (2\alpha + t + 1)(w(M_a^*) - w(M_b^*)).$$

Sum over the epochs and use $w(M_0^*) = 0$ and $w(M_n^*) \ge \mathrm{OPT}$. □

The coefficient of k in (12) is

$$K(t) = \frac{2\sqrt{t}(t+1)}{t-1}.$$

Its derivative is

$$K'(t) = \frac{t^2 - 4t - 1}{\sqrt{t}(t-1)^2}.$$

Hence the minimum over $t > 1$ occurs at

$$t = 2 + \sqrt{5}, \qquad \alpha = \sqrt{2 + \sqrt{5}}.$$

For these parameters, the competitive ratio in (12) is approximately

$$6.660381353571k - 2.890053638264.$$

This proves Theorem 1. Furthermore, note that if we set $t > 3 + 2\sqrt{2}$, then $\alpha = \sqrt{t}$ also implies $\alpha < (t-1)/2$. Hence, we can apply Lemma 6 and Corollary 14 to obtain the proof for Theorem 3.

B Competitive Analysis for the RM Algorithm

In this section, prove the competitive ratio of RM algorithm.

Theorem 15 (Competitive ratio of RM)**.** *For $t = 1 + \sqrt{2}$, the RM algorithm is*

$$(6 + 4\sqrt{2})k < 11.66k$$

competitive for online metric transportation.

Proof. RM is the special case of RG with $\alpha = t$. Retain the saturation epochs from Section 4, and write

$$W_h = w(M^*_{\tau_h}), \qquad W_0 = 0.$$

For epoch h, let

$$\Phi_h = \sum_{i=\tau_{h-1}+1}^{\tau_h} \phi_{t,t}^{M^*_{i-1}}(P_i).$$

A direct edge from r_i to a nearest free server is a candidate path of net-cost td_i. Hence, by minimum-net-cost selection and Lemma 13,

$$\Phi_h \le t \sum_{i=\tau_{h-1}+1}^{\tau_h} d_i \le t(\mathrm{OPT} + W_{h-1}).$$

Since $\alpha = t$, the greedy-cost term in Lemma 4 vanishes. Thus, the online cost C_h incurred during epoch h satisfies

$$\begin{aligned} C_h &\le \frac{2\Phi_h + (t+1)(W_{h-1} - W_h)}{t-1} \\ &\le \frac{2t\,\mathrm{OPT} + (3t+1)W_{h-1} - (t+1)W_h}{t-1}. \end{aligned}$$

Summing over the k epochs gives

$$\mathrm{ALG} \le \frac{2tk\,\mathrm{OPT} + 2t\sum_{h=1}^{k-1} W_h - (t+1)W_k}{t-1}.$$

By Lemma 11, $W_h \le t\,\mathrm{OPT}$ for every h, while $W_k \ge \mathrm{OPT}$ because $M^*_{\tau_k}$ is a feasible matching of all requests. Therefore,

$$\mathrm{ALG} \le \left[\frac{2t(t+1)}{t-1}k - \frac{2t^2 + t + 1}{t-1}\right]\mathrm{OPT}.$$

The leading coefficient

$$\frac{2t(t+1)}{t-1}$$

is minimized over $t > 1$ at $t = 1 + \sqrt{2}$. Substituting this value yields

$$\mathrm{ALG} \le \big((6 + 4\sqrt{2})k - (5 + 4\sqrt{2})\big)\mathrm{OPT} \le (6 + 4\sqrt{2})k\,\mathrm{OPT}.$$

□

C Proof of the input-sensitive guarantee

The preceding competitive analysis depends only on the number of server locations. We now give a complementary guarantee that adapts to their geometry. For $S' \subseteq S$, let $\mathrm{DIAM}(S') = \max_{u,v \in S'} d(u, v)$, and let $\mathrm{TSP}(S')$ be the length of a minimum closed tour through S'. Define

$$\mu_X(S) = \max_{\substack{S' \subseteq S \\ \mathrm{DIAM}(S') > 0}} \frac{\mathrm{TSP}(S')}{\mathrm{DIAM}(S')}. \tag{13}$$

When $\mathrm{DIAM}(S) = 0$, we set $\mu_X(S) = 1$.

Theorem 16 (Restatement of Theorem 2)**.**

$$\mathrm{ALG} \le O_{t,\alpha}\big(\mu_X(S)\log^2 n\big)\,\mathrm{OPT}.$$

For server locations embedded metric Euclidean spaces with dimensions $d > 1$, $\mu_X(S) = O_d(k^{1-1/d})$, yielding the desired competitive ratio of $Theorem\ 2$

We use the notation of the main paper. Thus, M_i and M_i^* are the online and offline matchings after request r_i is processed, P_i is the selected augmenting path, and

$$\phi_i = \phi_{t,\alpha}^{M_{i-1}^*}(P_i).$$

In particular, $\mathrm{ALG} = w(M_n)$. Fix a minimum-cost feasible matching M_{OPT} of all requests, so that $w(M_{\mathrm{OPT}}) = \mathrm{OPT}$, and let $\mathrm{opt}(r)$ be the server to which r is assigned in M_{OPT}. For a request set $R' \subseteq R$, let $\sigma(R')$ denote its requests in arrival order.

C.1 Algorithmic facts used by the geometric argument

We first isolate the three consequences of the current primal–dual implementation that replace the corresponding RM properties in Nayyar & Raghvendra (2017).

Lemma 17 (Basic net-cost bounds)**.** *For every arrival i,*

$$0 \le \phi_i \le t\mathrm{OPT}, \qquad w(M_i^*) \le t\mathrm{OPT}.$$

Moreover, immediately before augmentation in phase i,

$$\phi_i = y(r_i).$$

Proof. The equality $\phi_i = y(r_i)$ follows from Lemma 4 and Corollary 5 of the main paper: after the dual update, every residual edge of the selected path has zero slack. Request weights are initialized to zero and only increase, so $\phi_i = y(r_i) \ge 0$. The bound

$$w(M_i^*) \le t\mathrm{OPT}$$

is Lemma 7 of the main paper.

It remains to prove $\phi_i \le t\mathrm{OPT}$. Orient the edges of

$$M_{\mathrm{OPT}} \triangle M_{i-1}^*$$

from requests to servers when they belong to M_{OPT}, and from servers to requests when they belong to M_{i-1}^*. Pair incoming and outgoing edges at requests and servers, and consider the resulting directed trail beginning at the new request r_i. Repeatedly erase any closed directed subtrail. The result is a simple alternating path Q ending at a server whose M_{i-1}^*-load is smaller than its M_{OPT}-load, and hence at a free server. Truncate Q at the first free server it encounters.

If Q is direct, then

$$\phi_{t,\alpha}^{M_{i-1}^*}(Q) = \alpha w(Q) \le t\, w(M_{\mathrm{OPT}} \cap Q) \le t\mathrm{OPT}.$$

If Q is non-direct, then

$$\phi_{t,\alpha}^{M_{i-1}^*}(Q) = t\, w(Q \setminus M_{i-1}^*) - w(Q \cap M_{i-1}^*) \le t\, w(M_{\mathrm{OPT}} \cap Q) \le t\mathrm{OPT}.$$

By Corollary 5 of the main paper, P_i is a minimum-(t,α)-net-cost augmenting path. Therefore

$$\phi_i \le \phi_{t,\alpha}^{M_{i-1}^*}(Q) \le t\mathrm{OPT}.$$

□

Lemma 18 (Online cost from net cost)**.** *Let*

$$\Phi = \sum_{i=1}^{n} \phi_i.$$

Then

$$w(M_n) \le \frac{2t}{\alpha(t-1)}\,\Phi.$$

Proof. Let G be the total weight of the selected direct paths. The total contribution of these paths to Φ is αG. Every selected non-direct path has nonnegative net cost by Lemma 17. Consequently,

$$\alpha G \le \Phi.$$

Apply Lemma 2 of the main paper to the complete execution. Since the initial offline matching is empty, it gives

$$w(M_n) \le \frac{2\Phi + 2(t-\alpha)G - (t+1)w(M_n^*)}{t-1}.$$

Dropping the nonpositive terminal term and using $G \le \Phi/\alpha$, we obtain

$$w(M_n) \le \frac{2\Phi + 2(t-\alpha)\Phi/\alpha}{t-1} = \frac{2t}{\alpha(t-1)}\,\Phi.$$

□

Lemma 19 (Free-server separation)**.** *Suppose request r is about to be processed and s is free with respect to the current offline matching. Then*

$$d(r,s) \ge \frac{\phi_{t,\alpha}(r)}{\alpha},$$

where $\phi_{t,\alpha}(r)$ denotes the net cost selected while processing r.

Proof. Immediately before augmentation, r and s are both free. By Lemma 17, $\phi_{t,\alpha}(r) = y(r)$. The free–free dual constraint and the free-server normalization therefore give

$$\phi_{t,\alpha}(r) = y(r) \le \alpha d(r,s) - y(s) = \alpha d(r,s).$$

□

The denominator in Lemma 19 is α, rather than t as in the RM proof, because Robustified Greedy discounts direct free–free edges.

C.2 Dyadic net-cost groups

Assume henceforth that $\mathrm{OPT} > 0$. If $\mathrm{OPT} = 0$, Lemmas 17 and 18 immediately give $w(M_n) = 0$.

Partition the requests into R' and R'' as follows. Put $r_i \in R'$ if at least one of

$$\phi_i \le \frac{\mathrm{OPT}}{n}, \qquad \phi_i \le 16\alpha\, d(r_i, \mathrm{opt}(r_i)) \tag{14}$$

holds, and put all remaining requests in R''. Then

$$\sum_{r_i \in R'} \phi_i \le (16\alpha + 1)\mathrm{OPT}. \tag{15}$$

Indeed, charge the requests satisfying the first condition to the n copies of OPT/n. Charge every remaining request in R' to its distinct edge in M_{OPT} using the second condition.

Partition R'' into groups $R_1, \dots, R_m$, where $r_i \in R_j$ if

$$\frac{2^{j-1}\mathrm{OPT}}{n} \le \phi_i < \frac{2^j \mathrm{OPT}}{n}. \tag{16}$$

By Lemma 17, we may take

$$m = 1 + \lceil \log_2(tn) \rceil. \tag{17}$$

We will prove that, for every j,

$$\sum_{r_i \in R_j} \phi_i \le O_{t,\alpha}(\mu_X(S)\log(2n))\,\mathrm{OPT}. \tag{18}$$

C.3 CLUSTERS AND OPTIMAL PATHS

Fix a group R_j, write

$$\sigma(R_j) = (r'_1, \dots, r'_N),$$

and define

$$r^j_{\mathrm{IB}} = \frac{2^{j-1}\mathrm{OPT}}{\alpha n}. \tag{19}$$

Process $\sigma(R_j)$ in arrival order. The first request starts a cluster and is its center. Subsequently, assign a request to an existing cluster if its distance from the nearest center is less than $r^j_{\mathrm{IB}}/2$; otherwise, start a new cluster with that request as its center. Denote the resulting clusters by $C_1, \dots, C_{\widetilde{k}}$, and denote the center of C_k by $c^{(k)}$. The construction gives

$$d(r, c^{(k)}) < \frac{r^j_{\mathrm{IB}}}{2} \qquad \text{for every } r \in C_k, \tag{20}$$

$$d(c^{(k)}, c^{(k')}) \geq \frac{r^j_{\mathrm{IB}}}{2} \qquad \text{for } k \neq k'. \tag{21}$$

Let $M^{(k)}$ be the offline matching immediately before $c^{(k)}$ arrives. Because $c^{(k)}$ is the first request of C_k in arrival order, every request of C_k is free with respect to $M^{(k)}$.

Orient

$$M_{\mathrm{OPT}} \triangle M^{(k)}$$

as in the proof of Lemma 17, and fix one edge-disjoint decomposition into directed alternating trails and cycles. Repeatedly erase closed directed subtrails from each trail. Every arrived request is balanced, every unarrived request has one unit of excess outdegree, and a server s has indegree excess

$$b(s) - \deg_{M^{(k)}}(s) \geq 0,$$

because M_{OPT} saturates every server. Thus every $r_g \in C_k$ is a source of the resulting decomposition, and its path ends at a server with positive indegree excess, equivalently at a server s with

$$\deg_{M^{(k)}}(s) < b(s).$$

The terminal server is therefore free with respect to $M^{(k)}$.

Let Q_g be this path, truncated when it first reaches a free server. We call Q_g the *optimal path* of r_g. Define

$$\delta_g = \max_{v \in V(Q_g)} d(c^{(k)}, v). \tag{22}$$

The paths Q_g, for $r_g \in C_k$, are edge-disjoint. This is the capacitated analogue of the vertex-disjoint path family used by Nayyar & Raghvendra (2017).

Lemma 20 (Radius of an optimal path)**.** *For every $r_g \in C_k$,*

$$r^j_{\mathrm{IB}} \leq \delta_g \leq \left(t + 1 + \frac{t}{2\alpha}\right)\mathrm{OPT}.$$

Proof. The terminal server of Q_g is free with respect to $M^{(k)}$. By Lemma 19, every such server s satisfies

$$d(c^{(k)}, s) \geq \frac{\phi_{t,\alpha}(c^{(k)})}{\alpha} \geq \frac{2^{j-1}\mathrm{OPT}}{\alpha n} = r^j_{\mathrm{IB}}.$$

This proves the lower bound.

The path Q_g uses only edges of M_{OPT} and $M^{(k)}$. Therefore, by Lemma 17,

$$w(Q_g) \leq w(M_{\mathrm{OPT}}) + w(M^{(k)}) \leq (t+1)\mathrm{OPT}.$$

Furthermore, by (20),

$$d(c^{(k)}, r_g) < \frac{r^j_{\mathrm{IB}}}{2} \leq \frac{\phi_{t,\alpha}(r_g)}{2\alpha} \leq \frac{t}{2\alpha}\mathrm{OPT}.$$

The triangle inequality now gives the upper bound. □

Lemma 21 (Optimal-path density)**.** *For every $r_g \in C_k$,*

$$w(Q_g) \geq \frac{\delta_g}{2}, \tag{23}$$

$$w(M_{\mathrm{OPT}} \cap Q_g) \geq \frac{w(Q_g)}{t+1}. \tag{24}$$

Proof. By the triangle inequality,

$$\delta_g \leq d(c^{(k)}, r_g) + w(Q_g).$$

Equations (20) and the lower bound in Lemma 20 imply

$$d(c^{(k)}, r_g) < \frac{\delta_g}{2},$$

which proves (23).

For (24), the claim is immediate when Q_g is direct, because its only edge belongs to M_{OPT}. Suppose therefore that Q_g is non-direct. Consider the feasible dual vector associated with $M^{(k)}$, extended by assigning dual weight zero to requests that have not yet arrived. This extension remains feasible because all server weights are nonpositive.

Since $\alpha \leq t$, the free–free constraint is stronger than the ordinary t-constraint. Consequently, every forward edge (r, s) of Q_g satisfies

$$y(r) + y(s) \leq t\, d(r, s),$$

regardless of whether its endpoints are free. Every backward matching edge satisfies

$$y(r) + y(s) \geq d(r, s).$$

Summing these inequalities along Q_g cancels the internal dual weights. Its initial request has dual weight zero, while its terminal server is free and therefore also has dual weight zero. Hence

$$t\, w(Q_g \setminus M^{(k)}) - w(Q_g \cap M^{(k)}) \geq 0.$$

The forward edges belong to M_{OPT}, and the backward edges belong to $M^{(k)}$. Write

$$A = w(M_{\mathrm{OPT}} \cap Q_g), \qquad B = w(M^{(k)} \cap Q_g).$$

The preceding inequality is $tA \geq B$. Therefore

$$w(Q_g) = A + B \leq (t+1)A,$$

which proves (24). □

A direct optimal path has optimal-edge fraction one. For a non-direct optimal path, the ordinary t-bound remains valid on every forward edge because $\alpha \leq t$.

C.4 Outer groups and weighted balls

Partition the requests of R_j into outer groups R^l, for $l \geq j$, by placing r_g in R^l when

$$\frac{2^{l-1}\mathrm{OPT}}{\alpha n} \leq \delta_g < \frac{2^{l}\mathrm{OPT}}{\alpha n}. \tag{25}$$

Lemma 20 implies that there are at most

$$L = 1 + \left\lceil \log_2\left(\alpha n \left(t + 1 + \frac{t}{2\alpha}\right)\right)\right\rceil \tag{26}$$

nonempty outer-group indices.

Fix l. Let

$$C_k^l = C_k \cap R^l, \qquad \beta_k^l = |C_k^l|,$$

omitting empty sets. Associate with C_k^l two concentric weighted balls centered at $c^{(k)}$:

$$\mathrm{IB}_k^l = B(c^{(k)}, r_{\mathrm{IB}}^j), \qquad \text{weight } \beta_k^l,$$
$$\mathrm{OB}_k^l = B(c^{(k)}, r_{\mathrm{OB}}^l), \qquad r_{\mathrm{OB}}^l = \frac{2^l \mathrm{OPT}}{\alpha n}, \qquad \text{weight } \beta_k^l.$$

Because $l \geq j$,

$$r_{\mathrm{IB}}^j < r_{\mathrm{OB}}^l.$$

Moreover, every optimal path Q_g with $r_g \in C_k^l$ is contained in OB_k^l.

Lemma 22 (Inner-ball bounds)**.** *For the fixed outer group R^l,*

$$\sum_{r_i \in R^l} \phi_i < 2\alpha \sum_k \beta_k^l r_{\mathrm{IB}}^j, \tag{27}$$

$$|\widetilde{S}| \frac{r_{\mathrm{IB}}^j}{4} \leq \mathrm{TSP}(\widetilde{S}) \tag{28}$$

for every set $\widetilde{R}$ of at least two cluster centers, where

$$\widetilde{S} = \mathrm{opt}(\widetilde{R}).$$

Proof. For every $r_i \in R_j$, equations (16) and (19) give

$$\frac{\phi_i}{2\alpha} < r_{\mathrm{IB}}^j.$$

Summing over the partition of R^l into the sets C_k^l proves (27).

Now consider two distinct centers $c^{(k)}$ and $c^{(k')}$. Since both belong to R'', equations (14) and (16) imply

$$d(c^{(k)}, \mathrm{opt}(c^{(k)})) < \frac{\phi_{t,\alpha}(c^{(k)})}{16\alpha} < \frac{r_{\mathrm{IB}}^j}{8},$$

and the same inequality holds for $c^{(k')}$. Combining this with (21) gives

$$d(\mathrm{opt}(c^{(k)}), \mathrm{opt}(c^{(k')})) > \frac{r_{\mathrm{IB}}^j}{4}.$$

In particular, different centers in $\widetilde{R}$ have different optimal server locations. Every edge of a tour through $\widetilde{S}$ therefore has length at least $r_{\mathrm{IB}}^j/4$, proving (28). □

For a ball $B \subseteq X$, write $M_{\mathrm{OPT}} \cap B$ for the set of optimal edges both of whose endpoints lie in B.

Lemma 23 (Optimal weight in an outer ball)**.** *For every nonempty C_k^l,*

$$w(M_{\mathrm{OPT}} \cap \mathrm{OB}_k^l) \geq \frac{\beta_k^l r_{\mathrm{OB}}^l}{4(t+1)}.$$

Proof. For every $r_g \in C_k^l$, equation (25) and Lemma 21 give

$$w(M_{\mathrm{OPT}} \cap Q_g) \geq \frac{w(Q_g)}{t+1} \geq \frac{\delta_g}{2(t+1)} \geq \frac{r_{\mathrm{OB}}^l}{4(t+1)}.$$

The paths associated with the requests of C_k^l are edge-disjoint and lie inside OB_k^l. Summing the displayed inequality over these β_k^l paths proves the claim. □

C.5 WEIGHTED VITALI SELECTION

We use the same weighted variant of the Vitali selection as Nayyar & Raghvendra (2017). Given a collection of equal-radius weighted balls, repeatedly select a remaining ball of maximum weight and remove it together with every remaining ball that intersects it. The selected balls are pairwise disjoint, and their associated intersecting sets partition the original collection. Every ball in an intersecting set lies inside the threefold expansion of its selected ball, and its weight is no larger than the weight of the selected ball.

Apply this procedure to the outer balls $\{\mathrm{OB}_k^l\}_k$. Write the selected balls as

$$\mathrm{OB}_{s_1}^l, \ldots, \mathrm{OB}_{s_q}^l,$$

and let K_{s_p} be the indices of the balls assigned to $\mathrm{OB}_{s_p}^l$. Set

$$\kappa_{s_p} = \frac{\beta_{s_p}^l r_{\mathrm{OB}}^l}{4(t+1)}, \qquad \theta_{s_p} = |K_{s_p}| \beta_{s_p}^l r_{\mathrm{IB}}^j.$$

Lemma 23 and the disjointness of the selected outer balls give

$$\sum_{p=1}^{q} \kappa_{s_p} \le \mathrm{OPT}. \tag{29}$$

The maximal-weight selection and the partition into intersecting sets give

$$\sum_{k} \beta_k^l r_{\mathrm{IB}}^j \le \sum_{p=1}^{q} \theta_{s_p}. \tag{30}$$

Lemma 24 (One outer group)**.** *For every j and l,*

$$\sum_{r_i \in R^l} \phi_i \le 192\alpha(t+1)\mu_X(S)\mathrm{OPT}.$$

Proof. We first bound $\theta_{s_p}/\kappa_{s_p}$. If $|K_{s_p}| = 1$, then $l \ge j$ gives

$$\frac{\theta_{s_p}}{\kappa_{s_p}} = 4(t+1)\frac{r_{\mathrm{IB}}^j}{r_{\mathrm{OB}}^l} \le 2(t+1).$$

Suppose $|K_{s_p}| > 1$. Let $\widetilde{R}_{s_p}$ be the centers of the balls indexed by K_{s_p}, and let

$$\widetilde{S}_{s_p} = \mathrm{opt}(\widetilde{R}_{s_p}).$$

By Lemma 22,

$$|K_{s_p}| r_{\mathrm{IB}}^j \le 4\,\mathrm{TSP}(\widetilde{S}_{s_p}).$$

Furthermore,

$$\mathrm{opt}(c^{(k)}) \in \mathrm{IB}_k^l \subseteq \mathrm{OB}_k^l$$

for every k: the proof of Lemma 22 gives

$$d(c^{(k)}, \mathrm{opt}(c^{(k)})) < \frac{r_{\mathrm{IB}}^j}{8}.$$

Every ball indexed by K_{s_p} lies inside $3\mathrm{OB}_{s_p}^l$. Hence

$$\mathrm{DIAM}(\widetilde{S}_{s_p}) \le 6 r_{\mathrm{OB}}^l.$$

It follows that

$$\begin{aligned}
\frac{\theta_{s_p}}{\kappa_{s_p}} &= 4(t+1)\frac{|K_{s_p}| r_{\mathrm{IB}}^j}{r_{\mathrm{OB}}^l} \\
&\le 16(t+1)\frac{\mathrm{TSP}(\widetilde{S}_{s_p})}{r_{\mathrm{OB}}^l} \\
&\le 96(t+1)\frac{\mathrm{TSP}(\widetilde{S}_{s_p})}{\mathrm{DIAM}(\widetilde{S}_{s_p})} \\
&\le 96(t+1)\mu_X(S).
\end{aligned}$$

The same final bound covers $|K_{s_p}| = 1$. Indeed, when $\mathrm{DIAM}(S) > 0$, every closed tour through a positive-diameter set has length at least twice its diameter, and hence $\mu_X(S) \geq 2$. Combining (29) and (30), we obtain

$$\sum_k \beta_k^l r_{\mathrm{IB}}^j \leq 96(t+1)\mu_X(S)\mathrm{OPT}.$$

Equation (27) now gives

$$\sum_{r_i \in R^l} \phi_i < 2\alpha \sum_k \beta_k^l r_{\mathrm{IB}}^j \leq 192\alpha(t+1)\mu_X(S)\mathrm{OPT}.$$

□

C.6 Completing the proof

Summing Lemma 24 over the at most L outer groups gives, for every net-cost group R_j,

$$\sum_{r_i \in R_j} \phi_i \leq 192\alpha(t+1)L\mu_X(S)\mathrm{OPT}. \tag{31}$$

Summing (31) over the m net-cost groups and adding (15) yields

$$\Phi \leq \Big(16\alpha + 1 + 192\alpha(t+1)mL\mu_X(S)\Big)\mathrm{OPT}. \tag{32}$$

Here

$$m = 1 + \lceil \log_2(tn) \rceil,$$
$$L = 1 + \left\lceil \log_2\left(\alpha n \left(t + 1 + \frac{t}{2\alpha}\right)\right)\right\rceil.$$

For fixed t and α, both m and L are $O(\log(2n))$.

Applying Lemma 18 to (32), we obtain the explicit estimate

$$\mathrm{ALG} = w(M_n) \leq \frac{2t}{\alpha(t-1)}\Big(16\alpha + 1 + 192\alpha(t+1)mL\mu_X(S)\Big)\mathrm{OPT}. \tag{33}$$

If $\mathrm{DIAM}(S) = 0$, all server locations coincide and every feasible assignment has the same cost, so the theorem is immediate. Otherwise $\mu_X(S) \geq 2$, and the additive term $16\alpha + 1$ is absorbed by the $\mu_X(S)mL$ term. Since $mL = O_{t,\alpha}(\log^2 n)$, we conclude that

$$\mathrm{ALG} \leq O_{t,\alpha}\big(\mu_X(S)\log^2 n\big)\,\mathrm{OPT}.$$

This proves Theorem 16. □

D Additional Experimental Results

Figure 2, 3, 4 and 5 reports the complete parameter sweeps on underlying NYC and MovieLens data. All settings, instances, and evaluation procedures are identical to those in Section 5.

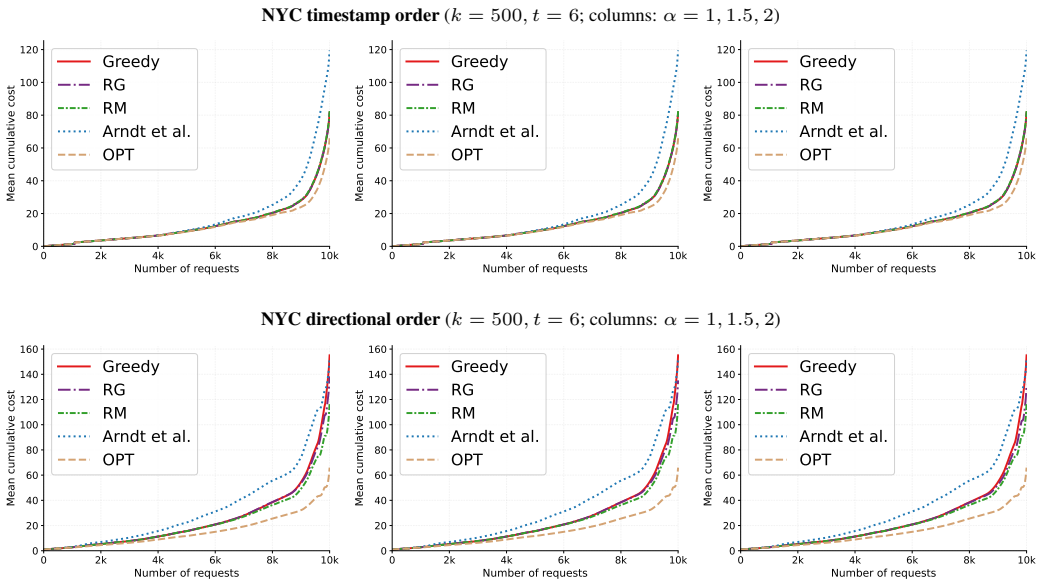

Figure 2: Complete mean cumulative assignment-cost sweeps over ten instances. The two rows vary α on NYC's timestamp and directional orders.

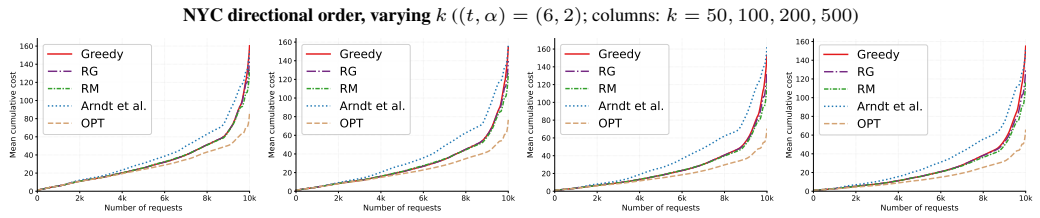

Figure 3: Complete mean cumulative assignment-cost sweeps over ten instances. Varies k on NYC's directional order.

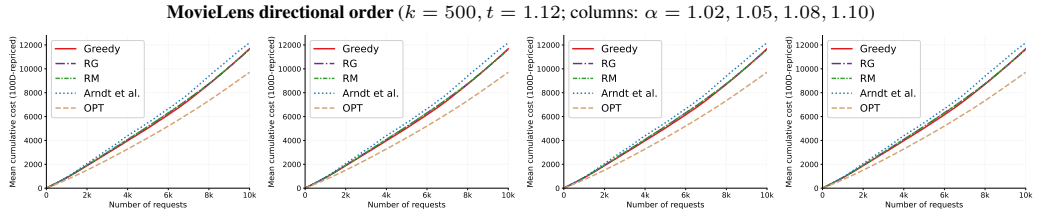

Figure 4: Complete mean cumulative assignment-cost sweeps over ten instances on MovieLens for $k = 1.12$ and α.

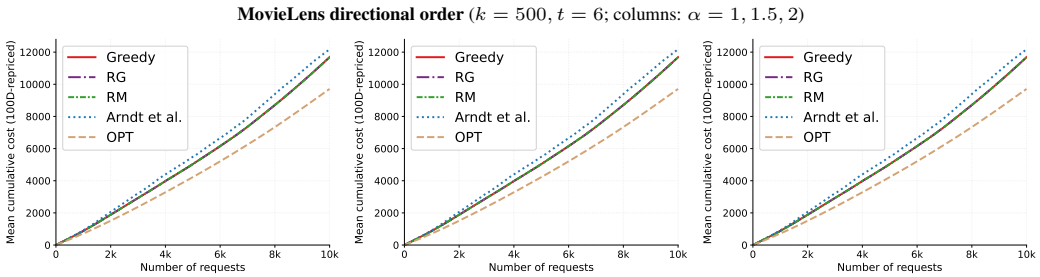

Figure 5: Complete mean cumulative assignment-cost sweeps over ten instances on MovieLens for $k = 6$ and α.