Learning-Augmented Online Algorithms for Nonclairvoyant Joint Replenishment Problem with Deadlines

Michael Dinitz* Jeremy T. Fineman† Seeun William Umboh‡

November 21, 2025

Abstract

This paper considers using predictions in the context of the online Joint Replenishment Problem with Deadlines (JRP-D). Prior work includes asymptotically optimal competitive ratios of $O(1)$ for the clairvoyant setting and $O(\sqrt{n})$ of the nonclairvoyant setting, where n is the number of items. The goal of this paper is to significantly reduce the competitive ratio for the nonclairvoyant case by leveraging predictions: when a request arrives, the true deadline of the request is not revealed, but the algorithm is given a predicted deadline.

The main result is an algorithm whose competitive ratio is $O(\min(\eta^{1/3}\log^{2/3}(n), \sqrt{\eta}, \sqrt{n}))$, where n is the number of item types and $\eta \leq n^2$ quantifies how flawed the predictions are in terms of the number of "instantaneous item inversions." Thus, the algorithm is robust, i.e., it is never worse than the nonclairvoyant solution, and it is consistent, i.e., if the predictions exhibit no inversions, then the algorithm behaves similarly to the clairvoyant algorithm. Moreover, if the error is not too large, specifically $\eta < o(n^{3/2}/\log^2(n))$, then the algorithm obtains an asymptotically better competitive ratio than the nonclairvoyant algorithm. We also show that all deterministic algorithms falling in a certain reasonable class of algorithms have a competitive ratio of $\Omega(\eta^{1/3})$, so this algorithm is nearly the best possible with respect to this error metric.

*Johns Hopkins University. Email: mdinitz@cs.jhu.edu. Supported in part by NSF award 2228995.

†Georgetown University. Email: jf474@georgetown.edu. Supported in part by NSF awards 2106759 and 1918989.

‡The University of Melbourne, and ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA). Email: william.umboh@unimelb.edu.au. Supported by the Australian Government through the Australian Research Council DP240101353.

1 Introduction

In many practical online optimization problems, we are not required to serve a request immediately upon arrival but are allowed to either postpone servicing the request either up to a deadline or by paying a delay cost that depends on the waiting time between arrival time and service time. Moreover, at any point in time, the algorithm can aggregate a subset of available requests and serve them together as a batch, leveraging economies of scale. These problems are called *online problems with deadlines or delay*. An important example of such a problem is the Joint Replenishment Problem, which we describe next.

The Joint Replenishment Problem (JRP) is a cornerstone problem in Operations Research and plays an important role in supply chain management. At a high level, the problem consists of a sequence of requests that arrive over time where each request is associated with one of n item types (also called "items"). At any time, we may choose to serve a set of requests Q and incur a fixed cost, called the ***joint ordering cost***, plus a marginal cost, called the ***item ordering cost***, for each item type associated with a request in Q. (Note that the marginal cost is per item type, not request, so servicing additional requests of the same item type is effectively free.) In the general version of the problem, each request also has a (possibly different) delay cost that depends on how long it waited between its arrival and service times. This paper considers the well-studied and important special case called JRP with deadlines (JRP-D): here instead of a delay cost, each request has a deadline and must be served no later than its deadline.

JRP has been studied in both the offline and online settings. In the offline setting, the problem is known to be APX-hard [NS09, BBC$^+$15] and admits small constant-factor approximations (c.f. [BBC$^+$14]). This paper focuses on the online setting: at each time t, we only know of requests whose arrival times are not later than t, and the decision is which (if any) items to serve at time t.

Nonclairvoyance vs Clairvoyance. For online problems with deadlines or delay, a standard distinction is made between the clairvoyant and nonclairvoyant settings. In the ***clairvoyant*** setting, when a request arrives, we also receive its deadline or its delay cost function for all future times. Online JRP was first studied in the clairvoyant setting by Buchbinder et al. [BKL$^+$13] who gave a deterministic primal-dual algorithm that is 3-competitive for the general delay version and 2-competitive for JRP-D. Later, Bienkowski et al. [BBC$^+$14] gave a deterministic greedy 2-competitive algorithm for JRP-D and proved a matching lower bound.

While most previous work on online problems with deadlines or delay has focused on the clairvoyant setting, there has been recent interest in studying the ***nonclairvoyant*** setting. In this setting, at each time t, for every request that has arrived but not yet been served, the algorithm only knows of the marginal increase in the delay cost incurred by the request if it is not served at the current time t. For online problems with deadlines, nonclairvoyance translates into knowing only if the deadline of a request is at the current time t, in which case it must be served immediately, or later than t. It turns out that the nonclairvoyant setting is significantly harder. For JRP, Le et al. [LUX23] gave a deterministic $O(\sqrt{n})$-competitive algorithm. Moreover, for JRP-D and hence also JRP, this is best-possible even for randomized algorithms [AGGP21, LUX23]. Beyond JRP, there has been work on nonclairvoyant algorithms for generalizations of JRP [ACKT20, Tou23, ELP$^+$24], as well as online matching with delay [DU23] and online service with delay [Tou25].

Using Predictions to Overcome Nonclairvoyance. There is an enormous gap between the clairvoyant setting (where we have a 2-competitive algorithm) and the nonclairvoyant setting (where the best possible competitive ratio is $\Omega(\sqrt{n})$). The goal of this paper is to circumvent the difficulty of the nonclairvoyant setting through the use of learning-augmented algorithms, also known as "algorithms with predictions".

In the algorithms with predictions framework, in addition to the regular input the algorithm is also given "predictions" of some sort (presumably the output of some machine learning algorithm or system). The goal is to use the predictions to get beyond traditional worst-case bounds: if the predictions are "good" then we would like our algorithm to perform extremely well, while if the prediction are "bad" then we would still like to have traditional worst-case guarantees. In other words, we want the best of both worlds: good performance in the average case thanks to machine learning, but also robustness and good performance in the worst-case from traditional algorithms.

While similar ideas have appeared in many places in the past, the formal study of algorithms with predictions was pioneered by Lykouris and Vassilvitskii [LV21]. While this framework can apply to many different algorithmic settings (and has been applied to settings like combinatorial algorithms [DIL$^+$21, DMVW23], differential privacy [ADKV22], data structures [DIL$^+$24, LLW22, VKMK21] and mechanism design [ABG$^+$22], to name just a few), the focus of [LV21] was on online caching, and the vast majority of papers in this area continue this line of work by considering online problems. We note that not only are predictions useful (as in caching) to allow the online algorithm to perform more like the offline, but they have also been used to allow nonclairvoyant algorithms to perform more like clairvoyant algorithms (see, e.g., [IKMQP21, LM25b]). So we build on this line of work by studying nonclairvoyant JRP with predictions.

Predictions for JRP. The first question for any problem in the algorithms with predictions framework is obvious: what should the prediction be? We make a natural choice: when a request q arrives at time a_q, because we are in the nonclairvoyant setting we are not given its true deadline d_q, but are instead given a ***predicted deadline*** $\tilde{d}_q$. We note that while sometimes using non-obvious predictions is a useful tool that can lead to improved performance (see, e.g., [LLMV20]), it is standard and common (particularly in online algorithms) for the prediction to be essentially the "missing data", i.e., whatever information is missing that would allow for improved performance. So our choice of deadline as the prediction is the obvious and standard choice.

We note that another obvious choice would be an *ordering* of requests with respect to their predicted deadlines. Permutation predictions have been used in the past for similar problems, most notably by [LM25b] in the context of nonclairvoyant scheduling. It turns out that, thanks to the error metric that we adopt (see following discussion), the way we use predicted deadlines is essentially equivalent to having a permutation prediction.

Error Metric

Once we have decided on what the prediction is, the next question in the framework is to fix an ***error metric***. One extreme is the "consistency vs robustness" framework, where we design an algorithm with two properties. First, it does well if the predictions are perfect ($\tilde{d}_q = d_q$ for all requests q), and in particular does approximately as well as would be possible in the clairvoyant case; this is called a ***consistency*** guarantee. Second, if the predictions are not accurate, then the worst-case guarantee is not much worse than the traditional nonclairvoyant guarantee; this is called a ***robustness*** guarantee. While much of the initial work on algorithms with predictions studied

the tradeoff between consistency and robustness, more recent research aims to give refined bounds that quantify the performance of the algorithm as a function of *how inaccurate* the predictions are (see the excellent list of papers in this area at [LM25a]). In particular, the goal is to obtain a competitive ratio that is a function of both the problem instance (notably n, the number of items here) and the error in the predictions. If the predictions are perfectly accurate we get back the consistency guarantee, but if they are "close" to being accurate then we should still get nontrivial guarantees that are much better than the completely worst-case robustness guarantee.

Numeric. So we need a way to quantify how close the predictions $\{\tilde{d}_q\}$ are from the true deadline $\{d_q\}$, and we want to design an algorithm whose performance degrades gracefully as a function of this distance. In the case of JRP-D, the choice of this error metric turns out to be not quite as obvious for us as the choice of prediction was. A natural first try is to consider these predictions as a vector (indexed by request q) and use a traditional ℓ_p vector norm, e.g., the maximum deadline error $\|d-\tilde{d}\|_\infty$, the total deadline error $\|d-\tilde{d}\|_1$, or some intermediate norm like $\|d-\tilde{d}\|_2$. Unfortunately, for nonclairvoyant JRP these norms are somewhat unsatisfactory. In particular, they are not scale-invariant: if we simply change the time scale on which we operate, e.g., by multiplying all times by some constant factor c, then any of these norms will also increase by a factor of c. But we have not really made the predictions any more or less accurate; we have simply changed the timescale. Moreover, it is not hard to see that the important information for the clairvoyant case is not the *value* of the deadlines, but rather their *ordering*. For example, suppose the predicted deadlines are all incorrect but their order is correct. Then the standard 2-competitive clairvoyant algorithm would still be 2-competitive—in fact, all known algorithms for JRP-D and its generalizations are invariant under order-preserving perturbations of deadlines—but any error metric based on norms could be large. So we would intuitively like our error metric to be a function of the induced ordering, rather than the values themselves.

Inversions. Clearly any collection of predicted or true deadlines gives an ordering of the requests. A natural notion of error then is the number of ***inversions*** between these orderings; that is, the number of pairs of requests q, q' where q is before q' in the true ordering but q' is before q in the predicted ordering. Unfortunately, this notion of error has a similar issue to the scale-invariant issue earlier: the number of requests can be arbitrarily larger than the number n of item types. So, for example, given a sequence of requests with some number of inversions between the true deadlines and the predicted deadlines, if we simply *repeat* this sequence again we end up with double the error, but without any meaningful change in the problem instance. Moreover, since there exists a worst-case nonclairvoyant solution that is $O(\sqrt{n})$ competitive, our error metric should ideally be upper bounded by a function of n, not the arbitrarily large number of requests.

But the bad example of repeating a sequence gives a natural notion of error that fixes these problems: ***item inversions***. We say that two *items* are inverted if there exists two requests for these items that are inverted. More formally, items i and j are inverted if there exist requests q to item i and q' to item j such that $d_q < d_{q'}$ but $\tilde{d}_q > \tilde{d}_{q'}$. This is one of the main notions of inversions that we will use, and in fact our lower bound (Theorem 1.4) is in terms of item inversions.

However, item inversions have another issue: it is trivial to design instances where there are many item inversions, but at any point in time there are very few actual request inversions. In such a scenario we would like to think that our predictions are actually quite good. So we arrive at our final error metric: ***instantaneous item inversions***. We define this formally in Section 2, but

it is essentially the maximum over all time points t of the number of item inversions for requests whose interval includes t. Let η denote the total number of instantaneous item inversions. To state performance bounds more concisely (they have a multiplicative dependence on item inversions), we define η to be 1 when there are zero inversions. Then we now have our final question: *can we design an algorithm with good performance as a function of η?*

1.1 Our Results

Our main technical result is an algorithm for JRP-D with predictions with good performance as a function of η.

Theorem 1.1. *There is an algorithm for nonclairvoyant JRP-D with predictions with competitive ratio at most $O(\eta^{1/3} \log^{2/3} n)$.*

This theorem implies that we start getting nontrivial bounds (better than the $O(\sqrt{n})$-competitive nonclairvoyant algorithm) when $\eta \leq O(n^{3/2}/\log^2 n)$.

We also show (Section 4) that as long as an algorithm for JRP-D satisfies a few key properties, it can be combined with other such algorithms to achieve the *best* of the competitive ratios offered by any of the algorithms. So we give two more algorithms and bounds to improve extreme regimes where Theorem 1.1 is weak. First, we are able to prove a $O(\sqrt{\eta})$-competitive ratio for a slight variant of our new algorithm from Theorem 1.1, which is better when η is very small but worse when η is extremely large. Second, while the $O(\sqrt{n})$-competitive nonclairvoyant algorithm of [LUX23] does *not* satisfy the properties we need for combinability, we show that it can be modified to have these properties without affecting its competitive ratio. This gives an improved bound when η is large. Putting these together, we obtain the following theorem:

Theorem 1.2. *There is an algorithm for nonclairvoyant JRP-D with predictions with competitive ratio at most $O(\min(\eta^{1/3} \log^{2/3} n, \sqrt{\eta}, \sqrt{n})$.*

Our final algorithm is thus both consistent and robust. Moreover its competitive ratio grows relatively slowly as a function of the error. We remark that all of our algorithms are deterministic.

1.1.1 Techniques and Approach

To understand the strength of this result and the techniques we use to get it, consider an extreme scenario of "useless" predictions. The worst-case number of instantaneous item inversions is $\eta = \Theta(n^2)$. Moreover, if the predictions are completely uninformative then we are essentially back in the traditional nonclairvoyant setting without predictions, where an $O(\sqrt{n})$ competitive ratio is possible. Therefore the best competitive ratio as a function of η that we can hope to achieve is $\Theta(\sqrt{n}) = \Theta(\eta^{1/4})$. So a natural goal for us is to aim for a competitive ratio of $O(\eta^{1/4})$.

With this as our goal, a natural approach (as in all algorithms with predictions settings) is to see what happens if we simply run the state of the art greedy 2-competitive clairvoyant algorithm [BBC+14] directly on the predictions. We call this algorithm Classic-Greedy and describe it formally in Section A, but informally it works as follows. When the deadline of a pending request is reached at some time t, we sort the requests by (predicted) deadline and keep adding items to our set in this order and stops when adding the next item causes the total of item ordering costs of our set to exceed the joint ordering cost; in other words, we choose the largest prefix of items whose total item ordering cost does not exceed the joint ordering cost. We then serve all pending requests for these items.

Classic-Greedy fails. In Section A we modify Classic-Greedy slightly to make it easier to analyze albeit sacrificing constants—instead of choosing the smallest prefix of items whose total item ordering cost does not exceed the joint ordering cost—and call it Folklore-Greedy. We show that the competitive ratio of Folklore-Greedy is at most $O(\eta)$, so we get an improvement over the nonclairvoyant algorithm when $\eta \leq O(\sqrt{n})$. This is a nontrivial result and so we include it for completeness since the greedy algorithm is of fundamental interest; in particular, variants of Classic-Greedy and Folklore-Greedy are subroutines in generalizations of JRP. Unfortunately, there is a fundamental roadblock: neither Classic-Greedy nor Folklore-Greedy beats $\Omega(\sqrt{\eta})$, and hence fall far from our goal.

Lemma 1.3. *Classic-Greedy and Folklore-Greedy has competitive ratio $\Omega(\sqrt{\eta})$.*

Proof. Create $2k$ items $b_1, \ldots, b_k$ (the *black items*) and $r_1, \ldots, r_k$ (the *red items*). Set the joint ordering cost at 1 and all item costs at $1/k$, meaning the greedy algorithm would always try to service k additional items on any deadline (if possible). For each red item r_i, create a single request with release time 0, deadline $2i$, and predicted deadline $2i$. For each black item b_i, create k requests: the j'th request has arrival time $2j-1$, deadline $3k$, and predicted deadline $2j+1$. (Unlike the red items, the black requests do not depend on the item number i.) Note that every red item is inverted with every black item, so $\eta = \Theta(k^2)$. Since all requests for red items are available at time 0 and all requests for black items are available at time $3k$, clearly we can just use two services (one at time 0 and one at time $3k$), each of which services k items with service cost $1 + k \cdot (1/k) = 2$. So OPT ≤ 4.

We now consider Classic-Greedy inductively. At each even time step $2i$, Classic-Greedy encounters a deadline for the lone request on r_i. At this point there are outstanding requests for red items $r_{i+1}, r_{i+2}, \ldots, r_k$ as well as requests for all of the black items (notably those released at time $2i-1$). But each of the black items has a request with predicted deadline $2i+1$, whereas the next red item's request has a (predicted and true) deadline of $2i+2$. Thus Classic-Greedy prioritizes the black items and services r_i and all of the black items at time $2i$. Thus Classic-Greedy has cost $\Omega(k)$, and so has competitive ratio $\Omega(k) = \Omega(\sqrt{\eta})$. The same argument works for Folklore-Greedy as well since both algorithms behave exactly the same on this instance. □

Improved algorithm. To beat a competitive ratio of $\sqrt{\eta}$, we thus need a new algorithm. We design the algorithm Local-Greedy to get around the above example. While the exact algorithm is somewhat complex (we give it formally in Section 3), intuitively it takes inspiration from the classical Marking algorithm for caching [FKL+91] and exploits the fact that OPT is lower bounded by the joint ordering cost times the number of requests with disjoint intervals. The algorithm explicitly keeps track of phases, where a new phase starts when a deadline hits for a request with interval that is disjoint from the interval that started the phase. And then in each phase we essentially run the greedy algorithm, but only on requests that were active at the beginning of the phase. So this algorithm is local in the sense that when a phase begins, we fix the set of requests that we might consider serving anytime in the phase, and any new requests that get released are deferred until the next phase.

It is not hard to see that Local-Greedy is actually optimal on our previous bad instance: the first phase starts at time 0 and includes only the red items, so at the first deadline the algorithm will service all of the red items and none of the black items. And since all red items have been serviced the next deadline that hits is the common deadline of all black requests, so it will service all the black items at that time. Unfortunately, it turns out that we are still in trouble: we can

construct another bad example on which Local-Greedy also has competitive ratio $\Omega(\sqrt{\eta})$. This is a little more complex, so we present this instance in Section 3.1.1.

We can in fact show a matching upper bound: the competitive ratio of Local-Greedy on *every* instance is at most $O(\sqrt{\eta})$ (see Section 3.1). This is a useful bound when η is extremely small, but it is still far from our goal. To improve on this, we note an interesting feature of the bad example: unlike the first example (showing a lower bound on the standard greedy algorithm), in this instance different items have different item ordering costs. If we are willing to lose an $O(\log n)$ factor, we can bucket the items into weight classes so that all items in a class have roughly the same ordering cost (up to a constant factor). It is not hard to see that this basically resolves the above bad example we had for Local-Greedy. Our main contribution is to show that this actually implies a better bound for Local-Greedy in general: we show that when all items have the same ordering costs, Local-Greedy is actually $O(\eta^{1/3})$-competitive. This fact coupled with the bucketing then implies Theorem 1.1.

1.1.2 Lower bound

Theorem 1.1 unfortunately does not achieve our original goal, of getting competitive ratio of $O(\eta^{1/4})$. But we show that this is for a good reason: while $\Omega(\eta^{1/4})$ is a natural lower bound based on completely uninformative predictions, we prove a stronger lower bound (nearly matching our upper bound) of $\Omega(\eta^{1/3})$ against a class of algorithms which we call "semi-memoryless" in the "limited information model". Formal definitions of these terms can be found in Section 5, but informally, in the limited information model we make the assumption that if we service a request strictly before its deadline then we do not actually find out when its deadline would have been, and we say that an algorithm is semi-memoryless if its memory is empty whenever there are no outstanding requests. Semi-memorylessness is a very natural property that all previous algorithms for JRP have had (and which Local-Greedy has), so this is not a major assumption. Similarly, for most applications it is completely reasonable to assume that servicing a request means that we will never find out its actual deadline since it will never trigger a required service.

So under these two assumptions, we get our nearly-matching lower bound. We note that in the following theorem we can even take η to be the number of item inversions rather than instantaneous item inversions, giving a stronger lower bound.

Theorem 1.4. *For every constant $c \geq 1$ and for a sufficiently large number of item types, every deterministic semi-memoryless algorithm in the limited information model has competitive ratio at least* $\max\left(\frac{\eta^{1/3}}{c^{1/3}(c+2)}, c\right)$.

Theorem 1.4 provides a construction that sets deadlines one of two possible ways depending on how the algorithm chooses to service requests. In one case, there are $\eta = \Theta(n^{3/2})$ inversions, but this case is chosen only if it would cause the algorithm to have competitive ratio at least $\Omega(\sqrt{n}) = \Omega(\eta^{1/3})$. In the other case, there are no inversions ($\eta = 1$), but the algorithm has competitive ratio at least c. Because the theorem holds for all constants $c \geq 1$, that effectively means that either the algorithm fails to be constant competitive when $\eta = 1$, or it is $\Omega(\eta^{1/3})$ competitive when $\eta = \Theta(n^{3/2})$

We note that Theorem 1.4 uses an instance where all item ordering costs are the same, for which our Local-Greedy is also $O(\eta^{1/3})$ competitive. For uniform weights, Local-Greedy is thus effectively optimal with respect to the error metric of (instantaneous) item inversions.

1.2 Paper Outline

We begin in Section 2 with preliminaries and definitions. In Section 3 we define the Local-Greedy algorithm and give our analysis of it, proving Theorem 1.1 as well as a weaker $O(\eta^{1/2})$-competitive bound along the way. In Section 4 we show how to combine different algorithms to get "best of all worlds" guarantee. Then in Section 5 we prove our main lower bound, Theorem 1.4. In Section A we study the modified greedy algorithm Folklore-Greedy (even though it gives weaker bounds) since it is a natural and simple algorithm which is well-studied in the clairvoyant setting.

2 Preliminaries and Notation

In the Joint Replenishment Problem with Deadlines (JRP-D), there is a set of n items, each with an *item ordering cost* w_i, and a *joint ordering cost* $w_0 \geq w_i$ for every item i. Then, m requests arrive over time; each request q has an associated item v_q, arrival time a_q, and deadline d_q. For a set of requests Q', we use $v(Q')$ to denote the set of items associated with Q'. Serving a set Q' of requests at time t at cost $c(Q') := w_0 + \sum_{i \in v(Q')} w_i$. Note that item i contributes w_i to the cost if there is at least one request in Q' on item i. We use λ to denote a service, Q_λ to be the set of requests it serves, and t_λ to denote the time of the service. We say that λ *transmits* item i if $i \in v(Q_\lambda)$. A feasible solution is a set of services Λ such that every request q is served at some time in the interval $[a_q, d_q]$. Denote by Λ_i the subset of Λ that transmits item i. The cost of Λ is

$$\sum_{\lambda \in \Lambda} w_0 + \sum_{i \in v(Q_\lambda)} w_i = |\Lambda| w_0 + \sum_i w_i |\Lambda_i|.$$

We call $|\Lambda| w_0$ the *joint cost* of Λ and $\sum_i w_i |\Lambda_i|$ the *item cost* of Λ.

As is common in the JRP literature, we assume that the request deadlines are *distinct*. This is without loss of generality as one can convert an instance to have this property by fixing an arbitrary permutation of the items and applying infinitesimal perturbations to the deadlines according to that permutation.[1] (Our lower-bound examples use repeated deadlines for clarity, but it easy to perturb them to have distinct deadlines.)

Clairvoyant vs Nonclairvoyant. In the *clairvoyant* setting, the online algorithm is given the deadline d_q of request q at its arrival time.

In the *nonclairvoyant* setting, the online algorithm is not given d_q when q arrives; instead, it is only given d_q at time d_q. Notably for our lower bound, we focus on the ***limited information*** setting, where d_q is only revealed if the request is still pending at that time (i.e., it is not served at any time $t < d_q$).

Deadline Predictions

This paper considers the prediction variant of JRP-D. As in the nonclairvoyant setting, the actual deadlines d_q are not known when a request q arrives. Instead, a prediction $\tilde{d}_q$ is provided. As in the nonclairvoyant setting, the limited information model means that the true deadline d_q is revealed at time d_q only if the request is still pending at that time.

[1] Note that different permutations may result in a different number of item inversions, but our analysis would apply to whichever permutation is being considered.

As discussed in Section 1, the primary way in which we quantify error is with respect to inversions, most notably instantaneous item inversions.

Definition 2.1 (Request Inversion)**.** *Requests q and q' are said to be* **inverted** *if they are on different items, i.e., $v_q \neq v_{q'}$, and their deadlines and predicted deadlines have different orderings, i.e., $d_q < d_{q'}$ and $\tilde{d}_q > \tilde{d}_{q'}$. Each pair of requests that are inverted is called a* **request inversion**. *We use R to denote the set of inverted request pairs, and $\#\textsc{ReqInv}$ to denote the total number of request inversions $|R|$ if there are any, or 1 if there are none.*

Definition 2.2 (Item Inversion)**.** *Items i and j, $i \neq j$, are said to be* **inverted** *if there exist requests q on item i and q' on item j that are inverted. Each pair of items that are inverted is called an* **item inversion**. *We use η to denote the total number of item inversions if there are any, or 1 if there are none.*

Definition 2.3 (Instantaneous Inversions)**.** *The* **instantaneous request inversions at time** t *is set of inversions from requests that overlap time t, i.e., $R_t = \{(q, q') \in R : t \in [a_q, d_q] \cap [a_{q'}, d_{q'}]\}$. The* **number of instantaneous request inversions** *refers to the maximum at any time, i.e., $\max_t |R_t|$.*

Similarly, the **instantaneous item inversions at time** t *refers to any item inversions that overlap at time t, i.e., $I_t = \{(v_q, v_{q'}) : v_q \neq v_{q'} \text{ and } (q, q') \in R_t\}$. The* **number of instantaneous item inversions** *refers to the maximum at any time, i.e., $\max_t |I_t|$.*

We will use η to denote the number of instantaneous item inversions in an input, with the exception that if there are no such inversions then we will set $\eta = 1$. We slightly abuse notation by letting η be the number of item inversions when we prove our lower bound (Theorem 1.4). Note that this is a stronger lower bound than if we used instantaneous item inversions.

3 Main Upper Bound: the Local-Greedy Algorithm

Algorithm description. Our algorithm, which we call Local-Greedy, is given in Algorithm 1. The algorithm keeps track of phases, defined as follows. The first phase starts at the first deadline; subsequently, the current phase ends and a new phase starts when a deadline hits for a request whose interval is disjoint from the request interval that started the phase. When a new phase starts, the service that is also triggered at the start of the new phase belongs to the new phase. Next, we describe how the algorithm chooses the requests to serve in a service. Suppose a service is triggered within a phase that starts at time s when the deadline of an unserved request q is reached. The request q is called the *triggering request* of the service. The only requests that are *eligible* for service are those whose intervals intersect with s. The algorithm goes through the eligible requests in ascending order of deadline, adding each request's item to its transmission set I, and stops once I served has total item cost $w(I) \geq w_0$, or there is no eligible request left. It then transmits I, serving all pending requests on items in I. The remaining eligible requests are called *leftover requests*.

Overview of analysis. Let Λ be the set of services produced by Local-Greedy. Let P be the set of requests whose deadlines correspond to the start of a phase. Observe that the optimal solution makes at least $|P|$ services since P is a set of disjoint requests and so $\text{OPT} \geq w_0|P|$. Next, consider some phase and let Λ' be the services within the phase. Since each service of Local-Greedy costs at most $3w_0$, the above lower bound on OPT lets us pay for the cost of one service the phase. Observe

Algorithm 1 Local-Greedy Algorithm

initialize $s = -\infty$ ▷ s tracks the start time of the current phase
when the deadline of an unserved request q is reached:
  if $a_q > s$ **then**
    start a new phase and set $s = d_q$
  end if
  let E_λ be the set of pending requests q' with $a_{q'} \leq s$
  initialize $I = \{v_q\}$;
  for each $q' \in E_\lambda$ in ascending order of deadline **do**
    add $v_{q'}$ to I
    if $w(I) \geq w_0$ **then**
      break
    end if
  end for
  transmit I and serve all pending requests on items I
end when

that each of the first $|\Lambda'| - 1$ services of Λ' have item cost at least w_0. This is because a service can only have item cost less than w_0 if there are no requests leftover, and thus, is the last service of the phase. In the clairvoyant setting, the item costs of these services can be charged to the item cost of OPT because there are sufficiently many disjoint requests on each item. In the nonclairvoyant setting with predicted deadlines, we run Local-Greedy with predicted deadlines. Here, we need to divide these services into safe and unsafe services. For the safe services, their item cost can be analyzed as before, while the item cost of unsafe services are charged to $w_0|P|$ by showing that there are at most $O(\sqrt{\eta})$ unsafe services per phase.

3.1 Useful Properties and Non-Uniform Bound

We first prove some useful properties of the optimal solution and Local-Greedy that hold independent of the accuracy of the predictions. We will then be able to use these to get our first bound: an $O(\sqrt{\eta})$ competitive ratio.

Lower bounds on OPT**.** Consider an optimal solution Λ^* with cost OPT. Let Λ^*_i be the subset of Λ^* that transmit item i. Let $\text{OPT}_0 := w_0|\Lambda^*|$ denote the joint cost of the optimal solution and $\text{OPT}_i := w_i|\Lambda^*_i|$ be its item i cost.

Proposition 3.1. *Let R be a set of disjoint requests and for each item i, let R_i be set of disjoint requests on item i. Then,* $\text{OPT}_0 \geq w_0|R|$ *and* $\text{OPT}_i \geq w_i|R_i|$ *for each item i.*

Proof. Since the requests in R are disjoint, the optimal solution must have at least $|R|$ services and so $|\Lambda^*| \geq |R|$. Let Λ^*_i be the set of services in OPT that transmit item i. Since there are $|R_i|$ disjoint requests on item i, the optimal solution must transmit item i at least $|R_i|$ times and so $|\Lambda^*_i| \geq |R_i|$. □

Bounding the cost of charged services. The crux of the analysis of Local-Greedy is in bounding the cost of charged services, defined as follows.

Definition 3.2 (Charged services)**.** *A service λ is a* charged *if it is not the last service of its phase, and* uncharged *otherwise.*

Let Λ° be the set of charged services. The following proposition implies that to bound the cost of charged services, it suffices to bound their total item cost.

Proposition 3.3. *Let λ be a charged service. Then, $w(\lambda) \geq w_0$.*

Proof. Suppose, towards a contradiction, that $w(\lambda) < w_0$. Let λ' be the next service of the phase and q' be its triggering request. Let s be the start of the phase. Since λ and λ' are in the same phase, q' intersects s and so is eligible when λ was triggered. Since $w(\lambda) < w_0$, Local-Greedy would have added it to λ, a contradiction. □

Let Λ°_i be the subset of charged services that transmit item i. To bound the total item cost of Λ°, we will bound, for each item i, the number of charged services transmitting item i in terms of $|\Lambda^*_i|$, the number of times the optimal solution transmits item i. To that end, we consider the requests that caused Local-Greedy to transmit item i.

Definition 3.4 (i-triggering requests)**.** *For each service $\lambda \in \Lambda^\circ_i$, for each item i, let q^i_λ be the earliest-deadline pending request that caused λ to include item i in its transmission. Let $Q_i := \{q^i_\lambda\}_{\lambda \in \Lambda^\circ_i}$. The requests in Q_i are called* i-triggering requests.

Proposition 3.5. *During a phase, every item is transmitted at most once.*

Proof. Suppose, towards a contradiction, that there a two services λ and λ' in a phase that transmits item i. Let s be the start of the phase, t, t' be their respective service times, and suppose $t < t'$. Let q' be the i-triggering request of λ'. Since q' was served by λ', we have $a_{q'} < s$ and $d_{q'} \geq t' > t$. Thus, when λ transmitted item i, it would have served q' as well. However, this contradicts the assumption that q' was served by λ'. □

Bounds in terms of unsafe services. We are now ready to develop the tools needed to relate the competitive ratio of Local-Greedy with the prediction error. The main tool is the following notion of safe and unsafe services, which capture the intuition that wrong predictions hurt Local-Greedy by causing it to serve a request too early.

Definition 3.6 (Phase boundary and safe requests/services)**.** *The* boundary *of a phase is the time of its last service. A request is* unsafe *if its deadline is later than the boundary of the phase it is served in and* safe *otherwise. A charged service is* safe *if every request q in the service is safe, and* unsafe *otherwise.*

We will later show that the item cost of safe services can be charged to the item cost of the optimal solution (Lemma 3.7), and prove a bound on the competitive ratio of Local-Greedy in terms of unsafe services (Lemma 3.8). Together, they show that a set of predictions is bad for Local-Greedy, in the sense that they cause Local-Greedy to have large competitive ratio, only if they cause Local-Greedy to have many unsafe services in a single phase.

Lemma 3.7. *Let Λ'_i be a set of safe services that transmit item i. Then, $w_i|\Lambda'_i| \leq 2\,\mathrm{OPT}_i$.*

Proof. Let Q'_i be the set of i-triggering requests that caused the algorithm to add item i to the services in Λ'_i. We will show that the depth of Q' is at most 2, which in turn implies that there are at least $|\Lambda'_i|/2$ disjoint requests on item i.

Suppose, towards a contradiction, that requests $q_1, q_2, q_3 \in Q'_i$ overlap. Suppose that these requests were served by safe services $\lambda_1, \lambda_2, \lambda_3$ at times $t_1 < t_2 < t_3$, respectively. By Proposition 3.5, the requests were served in different phases. Let s_1, s_2, s_3 be the start times of the respective phases, and b_1, b_2, b_3 be the boundaries of the respective phases. Note that $b_1 < s_2 \leq t_2 \leq b_2 < s_3 \leq t_3$. Since λ_1 served q_1 and λ_1 is a safe service, the deadline of q_1 is earlier than b_1. On the other hand, q_3 has deadline no earlier than t_3, since it was served at that time. Thus, it must have arrived after t_2, as otherwise it would already have been served at t_2. Therefore, q_1 and q_3 do not overlap, a contradiction.

Thus, there are at least $|\Lambda'_i|/2$ disjoint requests on item i. Combining this with Proposition 3.1 gives us the lemma. □

Lemma 3.8. *Suppose that there are at most D unsafe services per phase. Then, the cost of* Local-Greedy *is at most $(2D+3)\,\mathrm{OPT}_0 + 4\sum_i \mathrm{OPT}_i \leq (2D+4)\,\mathrm{OPT}$.*

Proof. Recall that P is the set of disjoint requests whose deadlines mark the start of a phase. We show that the cost of uncharged services $\Lambda \setminus \Lambda^\circ$ is at most $3\,\mathrm{OPT}_0$ and the cost of charged services is at most $2D\,\mathrm{OPT}_0 + 4\sum_i \mathrm{OPT}_i$.

By definition of Classic-Greedy, the cost of every service λ satisfies $w_0 \leq c(\lambda) \leq 3w_0$. So, the cost of each uncharged service is at most $3w_0$. Since each phase has at most one uncharged service and $|P|$ is exactly the number of phases, Proposition 3.1 implies

$$c(\Lambda \setminus \Lambda^\circ) \leq 3\,\mathrm{OPT}_0\,.$$

Consider a charged service $\lambda \in \Lambda^\circ$. Since it is charged, we have that $w(\lambda) \geq w_0$ and so $c(\lambda) = w_0 + w(\lambda) \leq 2w(\lambda)$. Let Λ' be the subset of Λ° that is safe, and Λ'_i be the subset of Λ°_i that is safe. The total cost of charged services is at most

$$\begin{aligned}
\sum_{\lambda\in\Lambda^\circ} 2 \sum_{i\in v(Q_\lambda)} w_i &= 2\sum_i w_i|\Lambda^\circ_i| \\
&\leq 2\sum_i w_i|\Lambda'_i| + 2\sum_i w_i|\Lambda^\circ_i \setminus \Lambda'_i| \\
&\leq 4\sum_i \mathrm{OPT}_i + 2w_0|\Lambda^\circ \setminus \Lambda'| && \text{(Lemma 3.7 and } w_0 \geq w_i \text{ for every item } i\text{)} \\
&\leq 4\sum_i \mathrm{OPT}_i + 2Dw_0|P| && \text{(at most } D \text{ unsafe services per phase)} \\
&\leq 4\sum_i \mathrm{OPT}_i + 2D\,\mathrm{OPT}_0\,. && \text{(}P \text{ is disjoint and Proposition 3.1)}
\end{aligned}$$

Together with the above bound on the cost of uncharged services, we have that the total cost of Local-Greedy is at most

$$(2D+3)\,\mathrm{OPT}_0 + 4\sum_i \mathrm{OPT}_i \leq 2D\,\mathrm{OPT}_0 + 4\,\mathrm{OPT} \leq (2D+4)\,\mathrm{OPT}\,.$$

This completes the proof of the lemma. □

Bounding unsafe services in terms of prediction error. One nice property of Local-Greedy is that it has good consistency: if all predictions are accurate, then its competitive ratio is quite good. We remark that we have not attempted to optimize the constants in both the design and analysis of Local-Greedy.

Theorem 3.9. *In the clairvoyant setting—all predictions are accurate—the competitive ratio of Local-Greedy is at most* 4.

Proof. We claim that in the clairvoyant setting, every charged service is safe. Consider a phase starting at time s and has boundary b. Let q' be the triggering request of the last service λ' of the phase, i.e. $d_{q'} = b$. Since λ' is part of the phase, $a_{q'} \leq s \leq b$. Let λ be a charged service of the phase. It has service time t where $s \leq t \leq b$. Thus, q' was eligible for λ. Since q' was not served by λ, it must have deadline no earlier than those served by λ. Therefore, λ is safe. This concludes the proof of the claim. The claim implies that we can apply Lemma 3.8 with $D = 0$ which yields the desired bound on the competitive ratio of Local-Greedy. □

We can now use the previous lemmas and properties to prove our main upper bound on Local-Greedy (although not our main upper bound, which we get later in Section 3.1 by bucketing items by weight and running Local-Greedy in each bucket).

Theorem 3.10. *In the predictions setting, the competitive ratio of Local-Greedy is at most* $O(\sqrt{\eta})$.

Proof. It suffices to show that if a phase has D unsafe services, then there are at least $\Omega(D^2)$ instantaneous item inversions. Consider a phase with D unsafe services that starts at time s and has boundary b. Let $\lambda_1, \ldots, \lambda_D$ be the D unsafe services and $t_1 < t_2 < \ldots < t_D$ be their respective service times. For λ_j, let q_j be the request served by λ_j whose deadline is after the phase boundary, and let q'_j be the leftover request that triggered the next service immediately after λ_j. For convenience, we write $d_j = d_{q_j}$ and $d'_j := d'_{q_j}$.

We claim that there are $\Omega(D^2)$ inversions among the requests $q_1, q'_1, \ldots, q_D, q'_D$. Observe that by definition of d'_j, we get

$$t_1 < d'_1 \leq t_2 < d'_2 \leq \ldots < t_{D-1} < d'_{D-1} \leq t_D < d'_D.$$

By definition of d_j, we also have $d_j > b > d'_k$ for every $j, k \in [1, D]$. Next, we re-index as follows: let $q'_{(j)}$ be the leftover triggering request with the j-th earliest predicted deadline, $\lambda_{(j)}$ be the charged service it is leftover from, and $q_{(j)}$ be the request served by $\lambda_{(j)}$ whose deadline is after the phase boundary, i.e. , i.e. $d_{q_j} > b$. Since Local-Greedy chose to add $q_{(j)}$ to $\lambda_{(j)}$ instead of $q'_{(j)}$, we get that $\tilde{d}_{(j)} < \tilde{d}'_{(j)}$. Thus, for $j \leq k \leq D$, $\tilde{d}_{(j)} \leq \tilde{d}'_{(k)}$ and so $q_{(j)}$ is inverted with $q'_{(k)}$. Therefore, we conclude that the total number of inversions among $q_1, q'_1, \ldots, q_D, q'_D$ is at least $\sum_{j=1}^{D}(D - j + 1) = \Omega(D^2)$.

Since $q_1, q'_1, \ldots, q_D, q'_D$ are eligible requests of their respective services, their intervals contain the start time of the phase. Thus, these inversions are instantaneous. Moreover, each of them is on a distinct item, by Proposition 3.5. Therefore, there are $\Omega(D^2)$ instantaneous item inversions and so $D = O(\sqrt{\eta})$. Applying Lemma 3.8 completes the proof of the lemma. □

3.1.1 Tightness of Analysis

We now show that our analysis above is tight (as discussed in Section 1).

Theorem 3.11. *There is an instance on which* Local-Greedy *has competitive ratio* $\Omega(\sqrt{\eta})$.

Proof. Let $w_0 = 1$. Our instance will have $2n$ items: cheap items $C = \{c_1, \ldots, c_n\}$ together with expensive items $E = \{e_1, \ldots, e_n\}$. Each c_i has item ordering cost $1/n$, and each e_i has item ordering cost 1.

Our requests come in n phases, where phase i starts at time $t_i = 2n(i-1)$ and has length $2n$. These phases will correspond with the phases of Local-Greedy. In phase i, a request for each item is released at time t_i. The request for an expensive item e_j has true deadline $3n^2$ (i.e., after all the phases have finished) and predicted deadline $t_i + 2(j-1) + 1$. The request for a cheap item c_j has true and predicted deadline $t_i + 2(j-1)$. Note that c_j is inverted with $e_{j'}$ for all $j' < j$, and hence there are $\Theta(n^2)$ item inversions.

Since all of the expensive item requests have true deadline at $3n^2$, we can use one service at that time to handle all of those requests (with cost $n+1$), and then in each phase use one service to handle all of the cheap item requests (so each such service costs $1 + n(1/n) = 2$). Thus $\mathrm{OPT} \leq 2n + n + 1 = 3n + 1$.

On the other hand, in each phase when Local-Greedy hits a deadline (for a cheap item c_j) it will do a service for $\{c_j, e_j\}$ with cost $2 + \frac{1}{n}$. Hence the total cost of Local-Greedy on this instance is $n(2 + \frac{1}{n})$ per phase, and so $\Theta(n^2)$ overall. Thus the competitive ratio is $\Omega(n) = \Omega(\sqrt{\eta})$ as claimed. □

3.2 Refined Analysis for Uniform Weights

To bypass the lower bound of Theorem 3.11, we show that when run on uniform weights, Local-Greedy actually performs better. In particular, we prove the following theorem.

Theorem 3.12. *Local-Greedy is* $O(\sqrt[3]{\eta})$*-competitive when every item has ordering cost* $0 \leq w \leq 1$.

Overview. The main idea for the improved analysis is in observing that Lemma 3.8 charges much more to OPT_0, joint cost of the optimal solution, than to the item ordering cost, $\sum_i \mathrm{OPT}_i$. To fix this, we use a more refined notion of unsafe services (Definition 3.14). Roughly speaking, instead of saying the entire service is unsafe and charging its entire cost to OPT_0 if it has even a single unsafe request, we only do so if there are many unsafe requests. Together with a bound on Local-Greedy (Proposition 3.13) in terms of the uniform item ordering cost w, we get the improved analysis.

Proposition 3.13. *Local-Greedy is* $O(1/w)$*-competitive.*

Proof. We begin by observing that the cost of each service of Local-Greedy is at most 2 and so the total cost of Local-Greedy is at most $2|\Lambda|$. Let $\hat{\Lambda}_i$ be the set of services triggered by a request on item i. Since the triggering requests on the same item are disjoint, we get $\mathrm{OPT}_i \geq w|\hat{\Lambda}_i|$ for each item i. Since $|\Lambda| = \sum_i |\hat{\Lambda}_i|$, we now have that

$$|\Lambda| = \sum_i |\hat{\Lambda}_i| \leq \frac{1}{w} \sum_i \mathrm{OPT}_i,$$

as desired. □

Definition 3.14 (τ-unsafe services)**.** *A charged service is* τ-unsafe *if at least* τ *fraction of its requests are unsafe and* τ-safe *otherwise.*

Lemma 3.15. *For every $0 < \tau < 1$, the cost of Local-Greedy's solution is at most*

$$\sqrt{\frac{w}{\tau}\eta} \cdot \mathrm{OPT}_0 + \frac{1}{1-\tau} \sum_i \mathrm{OPT}_i$$

and so Local-Greedy is $O(\max\{\sqrt{\frac{w}{\tau}\eta}, \frac{1}{1-\tau}\})$-competitive.

Proof. We will show that the total cost of τ-unsafe services is at most $\sqrt{\frac{w}{\tau}\eta} \cdot \mathrm{OPT}_0$ and the total cost of τ-safe services is at most $\frac{1}{1-\tau}\sum_i \mathrm{OPT}_i$.

Let p be the number of phases. Since the triggering requests of the first services of the phases are disjoint, we get $\mathrm{OPT}_0 \geq p$. We now bound the number of τ-unsafe services per phase.

Claim 3.16. *Every phase has at most $O(\sqrt{\frac{w}{\tau}\eta})$ τ-unsafe services.*

Proof. We prove the contrapositive: If some phase has D τ-unsafe services, then $\eta \geq \Omega(D^2\tau/w)$.

Consider a phase with D τ-unsafe services that starts at time s and has boundary b. Let $\lambda_1, \ldots, \lambda_D$ be the D unsafe services and $t_1 < t_2 < \ldots < t_D$ be their respective service times. For λ_j, let $Q_j \subseteq Q(\lambda_j)$ be consist of requests q with deadline $d_q > b$ and let q'_j be the leftover request that triggered the next service after λ_j. Thus, we have

$$t_1 < d'_1 \leq t_2 < d'_2 \leq \ldots < t_{D-1} < d'_{D-1} \leq t_D < d'_D < b.$$

We first observe that by definition, $d_q > b > d'_k$ for every $q \in Q_j$ and $j, k \in [1, D]$. Next, we re-index as follows: let $q'_{(j)}$ be the leftover triggering request with the j-th earliest predicted deadline, $\lambda_{(j)}$ be the charged service it is leftover from, and $Q_{(j)}$ be the requests served by $\lambda_{(j)}$ whose deadlines are after the phase boundary. Since the greedy algorithm chose to add $Q_{(j)}$ to $\lambda_{(j)}$ instead of $q'_{(j)}$, we get that $\tilde{d}_q < \tilde{d}'_{(j)}$ for every $q \in Q_{(j)}$. Thus, for $j \leq k \leq D$ and $q \in Q_{(j)}$, we have $\tilde{d}_q \leq \tilde{d}'_{(k)}$ and so every request in $Q_{(j)}$ is inverted with $q'_{(k)}$. Since $|Q_{(j)}| \geq \tau/w$, we conclude that the total number of item inversions is at least $\sum_{j=1}^{D}(k-j+1)\tau/w = \Omega(D^2\tau/w)$. □

This implies that each phase has at most $O(\sqrt{\frac{w}{\tau}\eta})$ τ-unsafe services. Thus, the total cost of τ-unsafe services is at most $p \cdot 2\sqrt{\frac{w}{\tau}\eta} \leq 2\sqrt{\frac{w}{\tau}\eta} \cdot \mathrm{OPT}_0$. This completes the first half of the proof.

It remains to bound the total cost of τ-safe services. The cost of a τ-safe service is at most 2 and so is at most $2/(1-\tau)$ times the item cost of its safe requests. More precisely, let Q'_i be the set of i-triggering requests that are safe. Then, the total item cost of safe requests is $\sum_i w|Q'_i|$ and we have that the total cost of t-safe services is at most $O\left(\frac{1}{1-\tau}\sum_i w|Q'_i|\right)$.

Claim 3.17. *For every item i, the item ordering cost of Local-Greedy due to item i is $w|Q'_i| \leq 2\,\mathrm{OPT}_i$.*

Proof. Let Q'_i be the set of i-triggering requests. We will show that the depth of Q'_i is at most 2, which in turn implies that there are at least $|Q'_i|/2$ disjoint requests on item i.

Suppose, towards a contradiction, that requests $q_1, q_2, q_3 \in Q'_i$ overlap. Suppose that these requests were served by services $\lambda_1, \lambda_2, \lambda_3$ at times $t_1 < t_2 < t_3$, respectively. By Proposition 3.5, the requests were served in different phases. Let s_1, s_2, s_3 be the start times of the respective phases, and b_1, b_2, b_3 be the boundaries of the respective phases. Note that $b_1 < s_2 \leq t_2 \leq b_2 < s_3 \leq t_3$. Since λ_1 served q_1 and q_1 is a safe request, the deadline of q_1 is earlier than b_1. On the other hand,

q_3 has deadline no earlier than t_3, since it was served at that time. Thus, it must have arrived after t_2, as otherwise it would already have been served at t_2. Therefore, q_1 and q_3 do not overlap, a contradiction.

Thus, there are at least $|Q'_i|/2$ disjoint requests on item i. Combining this with Proposition 3.1 gives us the claim. □

Thus, the total cost of τ-safe services is at most $O\left(\frac{1}{1-\tau}\sum_i w|Q'_i|\right) \leq O\left(\frac{1}{1-\tau}\sum_i \mathrm{OPT}_i\right)$. This completes the second half of the proof. □

Proof of Theorem 3.12: Applying Lemma 3.15 with $\tau = 1 - 1/\sqrt{w \cdot \eta} \geq 1/2$, we get that Local-Greedy is $O\left(\sqrt{w \cdot \eta}\right)$-competitive. By Proposition 3.13, we get that the competitive ratio of Local-Greedy is at most

$$O(\min\{\sqrt{w \cdot \eta}, 1/w\}) = O(\sqrt[3]{\eta}),$$

as desired. □

3.3 Final Bound

We can now bucket by weights and apply Theorem 3.12 to each bucket. We call this algorithm "Local-Greedy with bucketing". This will give our main theorem, which we restate here for completeness.

Theorem 1.1. *There is an algorithm for nonclairvoyant JRP-D with predictions with competitive ratio at most $O(\eta^{1/3}\log^{2/3} n)$.*

Proof. Our algorithm is a simple variant of Local-Greedy. For $j \in [\log n]$, let $B_j = \{i : w_0/2^i < w_i \leq w_0/2^{i-1}\}$. Let $B_{\log n+1} = \{i : w_i \leq w_0/n\}$. For each bucket B_i with $i \leq \log n$, we create a modified instance where all ordering costs have been rounded up to $w_0/2^{i-1}$. We then run Local-Greedy independently on the modified instance for each bucket, except for bucket $B_{\log n+1}$ we run the trivial algorithm that always services all open requests whenever a deadline is reached.

To analyze this algorithm, we need to set up some notation. For each $i \in [\log n + 1]$, let ALG_i denote the cost of the algorithm on the jobs in bucket B_i. Let $\mathrm{ALG} = \sum_{i=1}^{\log n+1} \mathrm{ALG}_i$ be the total cost of our algorithm. Let η_i be the number of instantaneous item inversions in the instance restricted to the items in B_i, and note that $\eta_i \leq \eta$ for all i. Let OPT_i denote the optimal cost of the original instance restricted to items in B_i, and let OPT'_i denote the optimal cost of the modified instance restricted to items in B_i. Clearly $\mathrm{OPT}_i \leq \mathrm{OPT}$ for all i, and also $\mathrm{OPT}'_i \leq 2\,\mathrm{OPT}_i$ for all $i \leq \log n$.

For bucket $B_{\log n+1}$, we know by the definition of the ordering costs that every time we perform a service that total cost is at most $w_0 + n\frac{w_0}{n} \leq 2w_0$. Let α denote the total number of times the algorithm performs a service. We know that the triggering requests for any two services must correspond to disjoint intervals, since otherwise by the definition of the algorithm they would both be served at the same time, and hence $\mathrm{OPT} \geq w_0\alpha$. Thus

$$\mathrm{ALG}_{\log n+1} \leq 2w_0\alpha \leq 2\,\mathrm{OPT}\,. \tag{1}$$

Putting these together we have that

$$\begin{aligned}
\mathrm{ALG} &= \sum_{i=1}^{\log n} \mathrm{ALG}_i + \mathrm{ALG}_{\log n+1} \\
&\leq \sum_{i=1}^{\log n} \mathrm{ALG}_i + 2\,\mathrm{OPT} && \text{(Eq. (1))} \\
&\leq \sum_{i=1}^{\log n} O(\eta_i^{1/3}) \cdot \mathrm{OPT}'_i + 2\,\mathrm{OPT} && \text{(Theorem 3.12)} \\
&\leq \sum_{i=1}^{\log n} O(\eta_i^{1/3}) \cdot \mathrm{OPT} + 2\,\mathrm{OPT} \\
&\leq \left(\sum_{i=1}^{\log n} O(\eta_i) \right)^{1/3} \log^{2/3} n \cdot \mathrm{OPT} + 2\,\mathrm{OPT} && \text{(Hölder's inequality)} \\
&\leq O(\eta^{1/3} \log^{2/3} n) \cdot \mathrm{OPT}
\end{aligned}$$

as claimed. □

4 Combining Algorithms

We now show that it is possible to combine different algorithms to get the "best of all worlds" guarantee, thereby obtaining Theorem 1.2. This is a relatively standard goal in online algorithms, both with predictions and a more general case of combining different algorithms where we do not know ahead of time which will be better (see, e.g., [MNS12]). The standard technique for this is to simply use a switching argument: if we want to combine two algorithms then we start running one of them, but keep track of how much the other algorithm *would* have cost if we had run it on the same sequence of requests. If we notice that the other algorithm is significantly cheaper, then we switch. Unfortunately we cannot apply this idea to our setting, since when we do a service with some set of requests then we will never find out the true deadlines for any requests other than the triggering requests, and so cannot keep track of the counterfactual cost. So instead of switching between algorithms, we instead simply take the *union* of the algorithms.

Definition 4.1 (Combinable Algorithm)**.** *We say that an algorithm A for JRP-D is* combinable *if it satisfies the following three properties.*

1. *A only does a service when an active request hits its deadline.*
2. *Whenever A does a service, its cost is $O(w_0)$.*
3. *Consider some set of requests Q. Let $Q' \subseteq Q$. Then A behaves the same (makes the exact same services of the exact same requests) when run only on Q' as it does if it is run on Q except the requests in $Q \setminus Q'$ are not serviced by A but are instead removed at no cost at some arbitrary time between their release dates and deadlines.*

We will now show that it is possible to combine algorithms that are combinable to get the stronger of the two.

Theorem 4.2. *Let A_1 and A_2 be combinable algorithms for JRP-D with predictions with competitive ratios of $\gamma_1(\eta, n)$ and $\gamma_2(\eta, n)$, respectively, where each γ_i is a nondecreasing function of η and n. Then there is an algorithm with competitive ratio $O(\min(\gamma_1(\eta, n), \gamma_2(\eta, n)))$.*

Proof. Consider the following algorithm A, which essentially just runs both A_1 and A_2 in parallel.

- Initialize both A_1 and A_2 as usual.
- When a deadline for an active request hits, let S_1 be the service (set of items) that A_1 would do and let S_2 be the service that A_2 would do. Service $S_1 \cup S_2$.

Note that this algorithm is well-defined thanks to property 1 of Definition 4.1.

To analyze this algorithm, consider some instance (set of requests) Q. For any $R \subseteq Q$, let $A_i(R \mid Q)$ be the cost of running A_i on input Q but where the requests in $Q \setminus R$ are removed at no cost at some point between their release dates and deadlines, and let $\alpha_i(R \mid Q)$ denote the *number* of services that A_i does in the same setting. From property 2 of Definition 4.1 we know that $A_i(R \mid Q) = O(w_0 \alpha_i(R \mid Q))$.

Now let $Q_1 \subseteq Q$ be the requests serviced by A_1 when we run A on Q, and let $Q_2 \subseteq Q$ be the requests serviced by A_2 (note that during a service both algorithms could choose to service overlapping requests, and in fact will always both choose to service the request triggering the service, so these are not disjoint). We also know that $\alpha_1(Q_1 \mid Q) = \alpha_2(Q_2 \mid Q)$, since A will do a service exactly when an active request hits its deadline (property 1 of Definition 4.1) which implies that this triggering request is in both Q_1 and Q_2 and will contribute 1 to both $\alpha_1(Q_1 \mid Q)$ and $\alpha_2(Q_1 \mid Q)$. So let $\alpha = \alpha_1(Q_1 \mid Q) = \alpha_2(Q_2 \mid Q)$. Moreover, by property 3 of Definition 4.1 we know that $A_i(Q_i \mid Q) = A_i(Q_i \mid Q_i)$ and $\alpha_i(Q_i \mid Q) = \alpha_i(Q_i \mid Q_i)$.

Thus we have that

$$
\begin{aligned}
A(Q) &\le A_1(Q_1 \mid Q) + A_2(Q_2 \mid Q) && \text{(definition of } \mathsf{A}) \\
&\le O(w_0 \alpha_1(Q_1 \mid Q)) + O(w_0 \alpha_2(Q_2 \mid Q)) && \text{(property 2 of Definition 4.1)} \\
&= O(w_0 \alpha) \\
&= O(\min(A_1(Q_1 \mid Q), A_2(Q_2 \mid Q))) \\
&= O(\min(A_1(Q_1 \mid Q_1), A_2(Q_2 \mid Q_2))) && \text{(property 3 of Definition 4.1)} \\
&\le O(\min(\gamma_1(\eta(Q_1), |Q_1|) \cdot \mathrm{OPT}(Q_1), \gamma_2(\eta(Q_2), |Q_2|) \cdot \mathrm{OPT}(Q_2))) && \text{(def of } \gamma_1, \gamma_2) \\
&\le O(\mathrm{OPT}(Q) \cdot \min(\gamma_1(\eta(Q_1), |Q_1|), \gamma_2(\eta(Q_2), |Q_2|))) && \text{(monotonicity of OPT)} \\
&\le O(\mathrm{OPT}(Q) \cdot \min(\gamma_1(\eta, n), \gamma_2(\eta, n))) && \text{(monotonicity of } \eta \text{ and } \gamma_i)
\end{aligned}
$$

as claimed. □

Now that we know that we can combine algorithms satisfying Definition 4.1, we need to show that the algorithms we consider are in fact combinable.

Lemma 4.3. *Both variants of Local-Greedy, i.e., with or without bucketing, are combinable.*

Proof. Local-Greedy obviously satisfies properties 1 and 2 of Definition 4.1, by construction. We just need to show property 3. As it progresses, Local-Greedy only makes decisions depending on (1) the start time s of the current phase, and (2) the set of pending requests that were released before time s. A phase boundary always starts at a deadline for a pending request that was released after

s, which must by definition be in Q'. Thus, the phase boundaries behave as though any removed requests never existed. Similarly, when Local-Greedy chooses request to include in a service, it has a total priority order that depends only on the set of still pending requests (i.e., in Q') requests that arrived before s. □

4.1 A Combinable nonclairvoyant Algorithm

In order to make our main result robust, as required by Theorem 1.2, we also need a combinable algorithm that is $O(\sqrt{n})$-competitive, where n is the number if item types. Unfortunately, the existing nonclairvoyant algorithm [LUX23] without predictions does not directly satisfy property 2 of Definition 4.1. Here we provided a simplified algorithm for nonclairvoyant JRP-D that is combinable, building on the same tools. For completeness, we also provide a simple argument that it achieves the desired competitive ratio. (The algorithm from [LUX23] solves a more general problem of multi-level aggregation, but the ideas are similar.)

A main component of the algorithm, as in [LUX23], is dividing the items into heavy and light items. We call an item i ***heavy*** if it has item ordering cost $w_i \geq w_0/\sqrt{n}$ and ***light*** otherwise. The remainder of the algorithm, however, is a bit different in the way light items are handled to make it fit our definition of combinable.

Algorithm 2 Nonclairvoyant JRP-D

```
partition the ≤ n light items arbitrarily into groups G_1, ..., G_√n such that |G_j| ≤ √n.
when the deadline of an unserved request q is reached:
    let i = v_q be the corresponding item
    if i is heavy, i.e., w_i ≥ w_0/√n then
        serve all requests for item i
    else
        serve all requests for items in the group G_j ∋ i
    end if
end when
```

Algorithm 2 starts by grouping the light items arbitrarily into groups $G_1, G_2, \ldots, G_{\sqrt{n}}$, each with $O(\sqrt{n})$ light items. Each heavy item and each group are then handled separately. That is, when a request for heavy item i reaches its deadline, then only that item is served. If instead a request for light item i reaches its deadline, all light items in the same group as i are served.

Theorem 4.4. *Algorithm 2 is $O(\sqrt{n})$ competitive for nonclairvoyant JRP-D.*

Proof. We start by considering a heavy item i. Let R_i be the set of triggering requests on that item, i.e., the requests that caused i to perform a service. By construction, the requests in R_i are disjoint as any request that overlaps the deadline d_q of a triggering request $q \in R_i$ are also served at time d_q. Thus, by Proposition 3.1, $\mathrm{OPT}_i \geq w_i|R_i|$. In contrast, Algorithm 2 has a cost of $(w_0 + w_i)|R_i|$ to serve these requests. Since i is heavy, $(w_0+w_i)|R_i| \leq (2\sqrt{n}w_i)|R_i| = O(\sqrt{n}\,\mathrm{OPT}_i)$. Summing across heavy jobs, we thus get our cost for heavy jobs is at most $\sum_{\text{heavy i}} \sqrt{n}\,\mathrm{OPT}_i) \leq O(\sqrt{n}\sum_i \mathrm{OPT}_i)$

We next consider each particular group G_j of light items. Again, by construction, the triggering requests R for this particular group are all disjoint. Thus, by Proposition 3.1, $\mathrm{OPT}_0 \geq w_0|R|$. The corresponding services that Algorithm 2 performs each include at most $|G_j| \leq \sqrt{n}$ items and thus have a cost of at most $w_0 + \sum_{i \in G_j} w_i \leq w_0 + \sqrt{n}w_i \leq 2w_0$. So the total cost of these services of

group G_j are at most $2w_0|R| = O(\mathrm{OPT}_0)$. Summing across all $\sqrt{n}$ groups, the total cost of the services performed for light groups is $O(\sqrt{n}\,\mathrm{OPT}_0)$.

Adding the heavy and light costs gives a total cost of $O(\sqrt{n}(\mathrm{OPT}_0 + \sum_i \mathrm{OPT}_i)) = O(\sqrt{n}\,\mathrm{OPT})$. □

Lemma 4.5. *Algorithm 2 for nonclairvoyant JRP-D is combinable.*

Proof. Again, each of the properties of Definition 4.1 follow directly from the algorithm. Property 1 is obvious, and Property 2 follows from the fact that $w_i \le w_0$ for heavy items and $\sum_{i \in G_j} w_i \le w_0$ for groups of light items. Finally, as with Classic-Greedy, this algorithm's servicing decisions are driven only by the deadline the set of pending request at each particular time, not any history, so removing requests before they are served has no impact. □

4.2 Tying it all together

We are now ready to complete the proof of Theorem 1.2.

Theorem 1.2. *There is an algorithm for nonclairvoyant JRP-D with predictions with competitive ratio at most* $O(\min(\eta^{1/3}\log^{2/3} n, \sqrt{\eta}, \sqrt{n})$.

Proof. We simply combine three algorithms: unbucketed Local-Greedy, bucketed local, and the nonclairvoyant algorithm. These algorithms have competitive ratios $O(\sqrt{\eta})$ from Theorem 3.10, $O(\eta^{1/3}\log^{2/3}(n))$ from Theorem 1.1, and $O(\sqrt{n})$ from Theorem 4.4, respectively. Lemmas 4.3 and 4.5 state that these algorithms are combinable. We can thus apply Theorem 4.2 to conclude that we achieve the min of all three bounds. □

5 Lower Bound

Our goal in this section is to prove Theorem 1.4. We first give important defintions of the setting.

Definition 5.1. *In the* **limited information model***, if a request is serviced before its deadline then the algorithm does not learn its deadline.*

Definition 5.2. *An algorithm is* **semi-memoryless** *if its memory is empty if there are no outstanding requests, and the algorithm does not use the timestamp when making decisions.*

In other words, if the system is empty (no outstanding requests), then the algorithm is not allowed to remember anything that has happened previously (and in particular cannot remember which items have already been inverted). Thus if we give the algorithm a sequence of requests which it handles, and then once there are no outstanding requests we give it the exact same sequence of requests, the fact that its memory was empty and it is not allowed to use the time when making decisions means that it will make the exact same decisions on the second copy of the sequence as it did on the first.

Now that the setting has been defined, we can now prove Theorem 1.4, restated here for convenience.

Theorem 1.4. *For every constant* $c \ge 1$ *and for a sufficiently large number of item types, every deterministic semi-memoryless algorithm in the limited information model has competitive ratio at least* $\max\left(\frac{\eta^{1/3}}{c^{1/3}(c+2)}, c\right)$.

Proof. Consider the following construction, which extends a version of the nonclairvoyant lower bound [AGGP21, LUX23] to include predictions. We describe our adversarial construction as adaptive; in reality, because the algorithm is assumed to be deterministic, the adversary could simulate what the algorithm would do before making choices, so it need not be adaptive.

The goal is to construct an instance on n item types where either (1) the algorithm is at least c-competitive when there are no item inversions, or (2) the algorithm is at least $\Omega(\eta^{1/3})$-competitive when $\eta = \Theta(n^{1/3})$.

We start by setting the joint ordering cost $w_0 = 1$ and the item ordering costs all to $w_i = 1/\sqrt{n}$.

The construction is broken down into a sequence of s *phases*, where s is to be set later. The rth phase starts at time $2rn$ and ends at time time $2rn + n$. We now describe each phase, where $t_0 = 2rn$ is the start time. At time t_0: for each item type $i \in [n]$, release a request for item i with release time t_0 and predicted deadline $\tilde{d}_i = t_0 + i$. Then advance through time steps $t_0+1, \ldots, t_0+n$ as follows. At time $t_0 + i$: if the request for item i has not yet been served by the algorithm then set the actual deadline $d_i = t_0 + i = \tilde{d}_i$, triggering a service of that request right now. Note that thus far we have not set the deadlines of requests that the algorithm chooses to service before their deadlines. To do this, we split the construction into two cases:

1. If the algorithm services an average of at most $\sqrt{n}/c$ requests per service (i.e., the total number of services is at least $c\sqrt{n}$), then set the deadline for the request for item i to $d_i = t_0 + i = \tilde{d}_i$.

2. If the algorithm services an average of at least $\sqrt{n}/c$ requests per services (so the total number of services is at most $c\sqrt{n}$), then the yet-unset deadlines are instead $d_i = 3sn + t_0 + i \gg \tilde{d}_i$, where s is the number of phases.

Observe that by construction, each request is serviced no later than its predicted deadline. Thus, each phase starts with no ouststanding requests. Because the algorithm is assumed to be deterministic and semi-memoryless, the algorithm makes exactly the same decisions in each phase.

To analyze this instance, we break into two cases corresponding to the cases from the construction. The case depends on the number x of services the algorithm performs during the each phase.

Case 1. Suppose that $x \geq c\sqrt{n}$. The algorithm thus pays at least $xw_0 + \sum_i w_i \geq c\sqrt{n} \cdot 1 + \sum_i (1/\sqrt{n}) = (c+1)\sqrt{n}$ per phase, for a total of at least $s(c+1)\sqrt{n} \geq sc(\sqrt{n}+1)$ for sufficiently large n (i.e., $\sqrt{n} > c$). On the other hand, in each phase one could simple service all requests in a single service at the beginning, so the optimal cost is at most $w_0 + \sum_i w_i = 1 + \sqrt{n}$ per phase or $s(\sqrt{n}+1)$ total. Thus, the competitive ratio is at least $\frac{sc(\sqrt{n}+1)}{s(\sqrt{n}+1)} = c$. Since this case sets all item deadlines to their predicted deadlines, there are no item inversions (i.e., $\eta = 1$), and the competitive ratio of the algorithm is indeed at least $\max\left(\frac{\eta^{1/3}}{c^{1/3}(c+2)}, c\right)$.

Case 2. Now suppose that $x < c\sqrt{n}$. It follows that most requests are serviced before their predicted deadlines and hence also true deadlines. As in Section 3, we use the term *triggering requests* to refer to those requests served at their deadlines (i.e., those with $\tilde{d}_i = d_i = t_0 + i$), and x here equals the number of triggering requests. Note that all triggering requests have deadlines that fall during the phase, and all non-triggering requests have deadlines after the final phase.

Since the algorithm services every item in each phase and uses at least one service to do so, the algorithm pays $xw_0 + \sum_i w_i \geq 1 + \sqrt{n}$ per phase or at least $s(1 + \sqrt{n})$ total. A better solution

would instead serve the x triggering requests at the start of each phase, and those nontriggering requests (with deadlines $d_i = 3sn + t_0 + i$) at time $3sn$. The cost of the optimal algorithm is thus at most $s(w_0 + xw_i) + (w_0 + (n-x)w_i) \leq s(1+c) + (1+\sqrt{n})$. Putting these together, we get a competitive ratio for the algorithm of at least

$$\frac{s(1+\sqrt{n})}{s(c+1)+(1+\sqrt{n})} \geq \frac{s\sqrt{n}}{s(c+1)+s} \text{if } s \geq \sqrt{n}+1 \ .$$

Setting $s = n$ thus suffices, giving us a competitive ratio ρ of at least $\rho \geq \frac{s\sqrt{n}}{s(c+2)} = \frac{\sqrt{n}}{c+2}$.

It remains to relate this competitive ratio to the number of inversions. By the semi-memoryless property, in every phase the algorithm makes the exact same decisions, and in particular services requests for the exact same items in the exact same order. So each item either has a corresponding triggering request in every phase, or none of its requests are triggering. Thus we can partition the *items* into x *triggering items* and $n-x$ *nontriggering items*. Notice that (1) triggering items are never inverted with triggering items because their predicted deadlines are all correct, and (2) nontriggering items are never inverted with nontriggering items because the $d_i - \tilde{d}_i = 3sn$ for every nontriggering request, so order is preserved. The number of item inversions η is thus at most $\eta = x(n-x) \leq xn \leq cn^{3/2}$, where the last step follows from the assumed bound on x. Taking the cube root, we have $\eta^{1/3} \leq c^{1/3}\sqrt{n}$ and hence $\frac{\eta 1/3}{c^{1/3}(c+2)} \leq \frac{\sqrt{n}}{c+2} \leq \rho$ as desired, which dominates the maximum when n is large enough relative to c. □

6 Conclusions and Future Work

In this work, we used deadline predictions to bridge the gap between clairvoyant and nonclairvoyant JRP-D. We showed that running the state of the art greedy algorithms Classic-Greedy and Folklore-Greedy on the predicted deadlines yields competitive ratios at most $O(\eta)$ and at least $\Omega(\sqrt{\eta})$. To get closer to the lower bound of $\Omega(\eta^{1/4})$, we designed a new greedy algorithm Local-Greedy that achieves $O(\eta^{1/3})$. Finally, we showed that semi-memoryless algorithms (which includes the above greedy algorithms) have competitive ratios that are $\Omega(\eta^{1/3})$.

There are several interesting directions for future work. The most immediate ones are to: close the gap between $O(\eta^{1/3})$ and $\Omega(\eta^{1/4})$; extend to generalizations of JRP-D such as Cardinality JRP and Multi-Level Aggregation (see [CKU22, ELP+24] and references therein); and extend to the general delay version. It would also be interesting to consider nonclairvoyant and learning-augmented algorithms for the setting with holding costs that penalize early service of a request [MNR25, AL26, SSU26].

References

[ABG+22] Priyank Agrawal, Eric Balkanski, Vasilis Gkatzelis, Tingting Ou, and Xizhi Tan. Learning-augmented mechanism design: Leveraging predictions for facility location. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 497–528, 2022.

[ACKT20] Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay - clairvoyance is not required. In Fabrizio Grandoni, Grzegorz Herman, and

Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[ADKV22] Kareem Amin, Travis Dick, Mikhail Khodak, and Sergei Vassilvitskii. Private algorithms with private predictions. *CoRR*, abs/2210.11222, 2022.

[AGGP21] Yossi Azar, Arun Ganesh, Rong Ge, and Debmalya Panigrahi. Online service with delay. *ACM Trans. Algorithms*, 17(3):23:1–23:31, 2021.

[AL26] Yossi Azar and Shahar Lewkowicz. Online joint replenishment problem with arbitrary holding and backlog costs. In *Proceedings of the Thirty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, Canada, January 11-14, 2026*, page to appear, 2026.

[BBC+14] Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jez, Dorian Nogneng, and Jirí Sgall. Better approximation bounds for the joint replenishment problem. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 42–54. SIAM, 2014.

[BBC+15] Marcin Bienkowski, Jarosław Byrka, Marek Chrobak, Neil Dobbs, Tomasz Nowicki, Maxim Sviridenko, Grzegorz Świrszcz, and Neal E Young. Approximation algorithms for the joint replenishment problem with deadlines. *Journal of Scheduling*, 18(6):545–560, 2015.

[BKL+13] Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: Primal-dual competitive algorithms. *Oper. Res.*, 61(4):1014–1029, 2013.

[CKU22] Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[DIL+21] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 10393–10406, 2021.

[DIL+24] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, Aidin Niaparast, and Sergei Vassilvitskii. Binary search with distributional predictions. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 90456–90472. Curran Associates, Inc., 2024.

[DMVW23] Sami Davies, Benjamin Moseley, Sergei Vassilvitskii, and Yuyan Wang. Predictive flows for faster ford-fulkerson. In *International Conference on Machine Learning*, pages 7231–7248. PMLR, 2023.

[DU23] Lindsey Deryckere and Seeun William Umboh. Online matching with set and concave delays. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[ELP+24] Tomer Ezra, Stefano Leonardi, Michal Pawlowski, Matteo Russo, and Seeun William Umboh. Universal optimization for non-clairvoyant subadditive joint replenishment. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK*, volume 317 of *LIPIcs*, pages 12:1–12:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[FKL+91] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[IKMQP21] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '21, page 285–294, New York, NY, USA, 2021. Association for Computing Machinery.

[LLMV20] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1859–1877. SIAM, 2020.

[LLW22] Honghao Lin, Tian Luo, and David Woodruff. Learning augmented binary search trees. In *International Conference on Machine Learning*, pages 13431–13440. PMLR, 2022.

[LM25a] Alexander Lindermayr and Nicole Megow. Algorithms with predictions, 2025. `https://algorithms-with-predictions.github.io/`.

[LM25b] Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. *ACM Trans. Parallel Comput.*, 12(2):4:1–4:26, 2025.

[LUX23] Ngoc Mai Le, Seeun William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1594–1610. SIAM, 2023.

[LV21] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

[MNR25] Benjamin Moseley, Aidin Niaparast, and R. Ravi. Putting off the catching up: Online joint replenishment problem with holding and backlog costs. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 3865–3883. SIAM, 2025.

[MNS12] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Trans. Algorithms*, 8(1), January 2012.

[NS09] Tim Nonner and Alexander Souza. A 5/3-approximation algorithm for joint replenishment with deadlines. In Ding-Zhu Du, Xiaodong Hu, and Panos M. Pardalos, editors, *Combinatorial Optimization and Applications, Third International Conference, COCOA 2009, Huangshan, China, June 10-12, 2009. Proceedings*, volume 5573 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2009.

[SSU26] David Shmoys, Varun Suriyanarayana, and Seeun William Umboh. Improved online algorithms for inventory management problems with holding and delay costs: Riding the wave makes things simpler, stronger, & more general. In *Proceedings of the Thirty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, Canada, January 11-14, 2026*, page to appear, 2026.

[Tou23] Noam Touitou. Frameworks for nonclairvoyant network design with deadlines or delay. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 105:1–105:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Tou25] Noam Touitou. Nearly-optimal algorithm for non-clairvoyant service with delay. In Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen, editors, *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, March 4-7, 2025, Jena, Germany*, volume 327 of *LIPIcs*, pages 74:1–74:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[VKMK21] Kapil Vaidya, Eric Knorr, Michael Mitzenmacher, and Tim Kraska. Partitioned learned bloom filters. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

A Greedy Algorithm

In this section, we study greedy algorithms as they are particularly natural and one of them is known to be 2-competitive in the clairvoyant setting [BBC+14].

We begin by defining the greedy algorithms.

Classic-Greedy [BBC+14] does the following: when the deadline of a pending request q is reached,

1. initialize $I = \{v_q\}$;
2. for each pending request q' in ascending order of (predicted) deadline: if $w(I \cup \{v_{q'}\}) \geq w_0$, break; else add $v_{q'}$ to I;
3. serve all pending requests on items I

For our purposes, it is more convenient to analyze a slightly modified greedy algorithm Folklore-Greedy. Local-Greedy is also based on Folklore-Greedy. Folklore-Greedy does the following: when the deadline of a pending request q is reached,

1. initialize $I = \{v_q\}$;
2. for each pending request q' in ascending order of (predicted) deadline: add $v_{q'}$ to I; if $w(I) \geq w_0$, break
3. serve all pending requests on items I

A.1 Generic properties of Folklore-Greedy

In this section, we prove properties of Folklore-Greedy that independent of the quality of the predictions.

Let Λ be the set of services produced by Folklore-Greedy.

Definition A.1 (Triggering and leftover requests/items)**.** *For a service $\lambda \in \Lambda$, the request q whose deadline triggered λ is called its* triggering request *and the pending requests that were not served due to the budget constraint are called* leftover requests*. The item v_q of the triggering request is called the* triggering item*.*

Definition A.2 (Primary, complete, charged services)**.** *A service λ is called:*

- primary *if its triggering request q_λ was not leftover from a previous service, and* secondary *otherwise;*
- complete *if there are no leftover requests, and* incomplete *otherwise;*
- charged *if the next service λ' was triggered by one of its leftover request and say that λ'* charges *λ; the service λ is* uncharged *otherwise. The triggering request of λ' is said to to be the* leftover triggering request *of λ*

Let Λ_1 be the set of primary services. Let Λ° be the set of charged services and Λ_i° be the subset that transmit item i. A set of consecutive services is called a *chain* if the first service is a primary service and the last service is uncharged. Note that every service is in a chain. Thus, the number of uncharged services is at most the number of primary services.

Lemma A.3. *The cost of Folklore-Greedy is at most*

$$3w_0|\Lambda_1| + 2\sum_i w_i|\Lambda_i^\circ|.$$

Proof. We show that the cost of uncharged services is at most $w_0|\Lambda_1|$ and the cost of charged services is at most $3\sum_i w_i|\Lambda_i^\circ|$.

By definition of Folklore-Greedy, the cost of every service λ satisfies $w_0 \le c(\lambda) \le 3w_0$. So, the cost of each uncharged service is at most $3w_0$. Since the number of uncharged services is equal to the number of primary services, we get that

$$c(\Lambda \setminus \Lambda^\circ) \le 3w_0|\Lambda_1|.$$

Consider a charged service $\lambda \in \Lambda^\circ$. Since it is charged, we have that $w(\lambda) \ge w_0$ and so $c(\lambda) = w_0 + w(\lambda) \le 2w(\lambda)$. Thus, the total cost of charged services is at most

$$\sum_{\lambda\in\Lambda^\circ} 3 \sum_{i\in v(Q_\lambda)} w_i = 2\sum_i w_i|\Lambda_i^\circ|.$$

The lemma now follows from the bounds on the cost of uncharged services and the cost of charged services. □

Let OPT_0 denote the cost of the optimal solution due to joint ordering cost and OPT_i be its cost due to item i ordering cost.

Lemma A.4. *The set of triggering requests of primary services are disjoint. Thus,* $w_0|\Lambda_1| \le \mathrm{OPT}_0$

Proof. Since each primary service occurs at the deadline of its triggering request, and by definition, its triggering request is not leftover from any previous service, we get that the set of triggering requests of primary services are pairwise-disjoint. Thus, any feasible solution has to have at least $|\Lambda_1|$ services. Since each service costs at least w_0, we get that $w_0|\Lambda_1| \le \mathrm{OPT}_0$. □

Definition A.5 (i-triggering requests)**.** *For each service $\lambda \in \Lambda_i^\circ$, for each item i except the let q_λ^i be the earliest-deadline pending request that caused λ to include item i. Let $Q_i := \{q_\lambda^i\}_{\lambda\in\Lambda_i^\circ}$. We say that the requests in Q_i are* i-triggering requests*.*

Lemma A.6. *If at most D i-triggering requests overlap, then $w_i|\Lambda_i^\circ| \le D\,\mathrm{OPT}_i$.*

Proof. Since at most D i-triggering requests overlap, the set of i-triggering requests Q_i can be partitioned into at most D disjoint subsets and so there exists a disjoint subset of Q_i with at least $|Q_i|/D$ requests. Thus, any feasible solution needs to have at least $|\Lambda_i^\circ|/D$ services that contain item i and so $w_i|\Lambda_i^\circ| \le D\,\mathrm{OPT}_i$. □

A.2 Analysis of Charged Services

Definition A.7 (Premature requests)**.** *Let q be a request served by a charged service λ and q' be its leftover triggering request. We say that q is* premature *if $d_q > d_{q'}$.*

Definition A.8 (Pre-empted requests)**.** *Let λ be a charged service and q' be its leftover triggering request. The request q' is said to be* pre-empted *if there exists a request q served by λ such that $d_q > d'_q$.*

A.2.1 Request Inversions

Even though item inversions are a error metric than request inversions (as discussed in Section 1), we get a stronger bound as a function of request inversions, and so if the number of request inversions is not too much larger than the number of item inversions than an analysis based on request inversions it better. We now give such a bound and analysis.

Lemma A.9. *If $D \geq 3$ i-triggering requests overlap, then $\#\textsc{ReqInv} \geq \frac{(D-2)(D-1)}{2}$.*

Proof. Suppose $q_1, \ldots, q_D \in Q_i$ overlap and are served at time $t_1 < t_2 < \ldots < t_D$, respectively. For $1 \leq j \leq D$, let λ_j be the charged service that served q_j. Since λ_j is charged, it has a leftover request q'_j whose deadline triggered the next service λ'_j after λ_j. For $1 \leq j \leq D$, we write $a_j := a_{q_j}$, $d_j := d_{q_j}$, $a'_j := a_{q'_j}$ and $d'_j := d_{q'_j}$. Recall that, by definition, λ'_j was triggered at time d'_j and there is no other service between λ_j and λ'_j. Thus, we have

$$t_1 < d'_1 \leq t_2 < d'_2 \leq \ldots < t_{D-1} < d'_{D-1} \leq t_D < d'_D.$$

We first show that $d_j > d'_k$ for every $j, k \in [1, D-2]$. Consider request q_D. Since it was served at t_D, its deadline $d_D \geq t_D > t_{D-1}$, so its arrival time $a_D > t_{D-1}$, as otherwise it would have been served by λ_{D-1}. Since q_j overlaps with q_D, we have $d_j \geq a_D > t_{D-1} > d'_{D-2} \geq d'_k$.

Next, for $j \in [1, D-2]$, we re-index as follows: let $q'_{(j)}$ be the leftover triggering request with the j-th earliest predicted deadline, $\lambda_{(j)}$ be the charged service it is leftover from, and $q_{(j)}$ be the i-triggering request served by $\lambda_{(j)}$. Since Folklore-Greedy chose to add $q_{(j)}$ to $\lambda_{(j)}$ instead of $q'_{(j)}$, we get that $\tilde{d}_{(j)} < \tilde{d}'_{(j)}$. Thus, for $j \leq k \leq D-2$, $\tilde{d}_{(j)} \leq \tilde{d}'_{(k)}$ and so $q_{(j)}$ is inverted with $q'_{(k)}$. Therefore, we conclude that the total number of request inversions is at least $\sum_{j=1}^{D-2}(D-2-j+1) = \frac{(D-2)(D-1)}{2}$. □

Theorem A.10 (Request Inversions Bound)**.** *The competitive ratio of Folklore-Greedy is $O(\sqrt{\#\textsc{ReqInv}})$.*

Proof. Combining Lemmas A.3, A.4, A.6 and A.9, we get that the cost of Folklore-Greedy is at most

$$3w_0|\Lambda_1| + \sum_i w_i|\Lambda_i^\circ| \leq 3\,\mathrm{OPT}_0 + O(\sqrt{\#\textsc{ReqInv}}) \sum_i \mathrm{OPT}_i \leq O(\sqrt{\#\textsc{ReqInv}})\,\mathrm{OPT}\,. \qquad \square$$

A.2.2 Item Inversions

We now give our main upper bound on Folklore-Greedy: a linear bound in the number of item inversions. Let $\hat{Q}_i \subseteq Q_i$ be the subset of premature i-triggering requests and $\hat{\Lambda}_i$ be the services they were served in. The proof of the following lemma is exactly the same as for Lemma A.6.

Lemma A.11. *For every item i, no three requests from $Q_i \setminus \hat{Q}_i$ can overlap.*

Let $\mathrm{inv}(i)$ be the set of items that are inverted with i. Let $\mathrm{disj}(i)$ be the maximum number of disjoint requests of item i.

Lemma A.12. $|\hat{\Lambda}_i| \leq \sum_{j \in \mathrm{inv}(i)} \mathrm{disj}(j)$.

Proof. Let $\hat{\Lambda}_{ij}$ be the subset of services $\lambda \in \hat{\Lambda}_i$ whose leftover triggering request is on item j. By definition, $\hat{\Lambda}_{ij} \neq \emptyset$ only if $j \in \mathrm{inv}(i)$. Since the leftover triggering requests on the same item must be disjoint, we get that $|\hat{\Lambda}_{ij}| \leq \mathrm{disj}(j)$. Thus, we get $|\hat{\Lambda}_i| = \sum_j |\hat{\Lambda}_{ij}| \leq \sum_{j \in \mathrm{inv}(i)} \mathrm{disj}(j)$. □

Theorem A.13 (Item Inversions Bound)**.** *The competitive ratio of* Folklore-Greedy *is* $O(\eta)$.

Proof. By Lemmas A.3 and A.4, A.6 and A.12, we get that the cost of the greedy algorithm is at most

$$3w_0|\Lambda_1| + \sum_i w_i|\Lambda_i^\circ| \le 3\,\mathrm{OPT}_0 + \sum_i w_i|\Lambda_i^\circ|.$$

We have that

$$\begin{aligned}
\sum_i w_i|\Lambda_i^\circ| &= \sum_i w_i(|\hat{\Lambda}_i| + |\Lambda_i^\circ \setminus \hat{\Lambda}_i|) \\
&\le \sum_i w_i \sum_{j\in \mathrm{inv}(i)} \mathrm{disj}(j) + 2\,\mathrm{OPT}_i && \text{(Lemma A.12)} \\
&\le \eta \cdot \max_j \mathrm{disj}(j) + 2\sum_i \mathrm{OPT}_i \\
&\le \eta \cdot \mathrm{OPT}_0 + 2\sum_i \mathrm{OPT}_i\,.
\end{aligned}$$

Thus, the cost of Folklore-Greedy is at most $O(\eta)\,\mathrm{OPT}$ as claimed. □