LEARNING-AUGMENTED ALGORITHMS FOR TSP AND VEHICLE ROUTING PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider a canonical vehicle-routing problem in which a sequence of point-to-point requests arrives online and must be inserted into the schedule to minimize the total travel time required to serve them. This captures classical problems such as the online Traveling Salesperson Problem (TSP) and the online Stacker Crane Problem. Our main result is a learning-augmented algorithm with a competitive ratio of 3.5 when the input requests are drawn from (possibly non-identical) distributions, given just a *single sample* from each distribution as offline predictions. This is in sharp contrast to the best-known logarithmic bounds without offline predictions. We further extend this result in two directions: first, by showing that the algorithm can be made *robust* to adversarial noise in the samples, and second, by considering a fleet of $m > 1$ vehicles rather than a single vehicle. Finally, we complement our theoretical results with empirical evaluations.

1 INTRODUCTION

The classical theory of online algorithms studies decision making under uncertainty by requiring performance guarantees that hold for every possible input sequence. Although this worst-case perspective provides strong robustness, it can be overly pessimistic in settings where machine-learned predictions using historical data provide useful information about future inputs. Motivated by this observation, a rapidly growing literature on *learning-augmented algorithms*, also called *algorithms with predictions*, seeks to design algorithms that use machine-learned predictions to improve their performance while retaining meaningful guarantees when those predictions are inaccurate (see surveys by Mitzenmacher & Vassilvitskii (2020; 2022)). A central goal is to obtain algorithms that are simultaneously *consistent*, achieving substantially better guarantees than the worst-case when the predictions are accurate, and *robust*, degrading gracefully as prediction quality deteriorates to worst-case guarantees for arbitrary predictions.

In this paper, we develop learning-augmented algorithms for the canonical vehicle routing problem of designing the most efficient route to serve a set of point-to-point requests. Vehicle routing is a fundamental component of many modern applications including ride-sharing platforms, delivery services, and automated warehouse systems. In many such applications, transportation requests arrive over time and must be incorporated into an evolving schedule before future requests are known. Historical data are often plentiful in these settings, making vehicle routing a natural domain to investigate whether learned information can overcome the limitations of worst-case online algorithms.

We consider a sequence of point-to-point transportation requests, where each request specifies a source and a destination. The requests arrive online and must be inserted incrementally into the existing schedule, without changing the relative order of previously scheduled requests. The objective is to minimize the total travel time, including both the distances traveled while serving requests and the distances traveled between consecutive requests. We refer to this problem as the *Online Stacker Crane problem* (Frederickson et al., 1978). It includes the classical Online Traveling Salesperson Problem (TSP) as a special case: when the source and destination of every request coincide, serving the requests reduces to visiting a sequence of points arriving online (Rosenkrantz et al., 1977).

When all requests are known in advance, standard techniques for metric TSP yield constant-factor approximations (Williamson & Shmoys, 2011; Karlin et al., 2024). In the online setting, however, the best-known worst-case guarantee grows logarithmically with the number of requests (Rosenkrantz et al., 1977; Imase & Waxman, 1991; Azar, 1994). In sharp contrast, we show

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

that a remarkably small amount of learned information suffices to overcome this barrier. Suppose the i-th request is independently drawn from an arbitrary distribution $\mathcal{P}_i$, where the distributions need not be identical. (Note that the worst-case online setting can be realized by defining each distribution to be deterministically the corresponding worst-case input.) We show that an offline prediction comprising just a *single sample* from each distribution suffices to dramatically improve the competitive ratio to 3.5. Furthermore, we show that our algorithm can be made robust to noise in the samples, smoothly degrading from constant-competitiveness for perfect samples to logarithmic-competitiveness for arbitrary inputs. Finally, we extend these results to the case of multiple vehicles.

1.1 Our Contributions

Our main contributions are as follows:

- **A constant competitive ratio from a single sample.** For the single-vehicle Online Stacker Crane problem, we give an algorithm with an expected competitive ratio of 3.5 using only one offline sample from each request distribution. This sharply improves on the best-known logarithmic guarantee without learned information. Notably, we do not require an explicit description of the request distributions and allow them to be arbitrary and non-identical.
- **Extension to multiple vehicles.** We extend our framework to a fleet of $m > 1$ vehicles, where each vehicle originates from and returns to a designated depot (possibly shared with other vehicles), and obtain an expected competitive ratio of 4 for any value of m. Thus, a single sample from each distribution continues to yield a constant-factor guarantee even when requests must be assigned across multiple vehicle schedules.
- **Robustness to corrupted samples.** The preceding guarantees assume that every offline sample is reliable. We then consider a robust model in which samples can be noisy or corrupted. We give an algorithm that achieves an expected competitive ratio of $\mathcal{O}(\log k)$, where k is the number of sampling errors, thereby smoothly interpolating between $O(1)$-consistency for perfect samples and $O(\log n)$-robustness for arbitrary samples.
- **Robust algorithm for multiple vehicles.** The robust result also extends from the single-vehicle setting to multiple vehicles. We provide a robust algorithm that runs in polynomial time for any fleet size m, achieving an expected competitive ratio of $\mathcal{O}(\log k)$.
- **Empirical evaluation.** Finally, we complement our theoretical results with empirical evaluation on synthetically generated instances. Our experiments investigate the benefits provided by offline samples and the effect of sample corruption.

Outline. First, we present our 3.5-competitive algorithm for the single-vehicle Online Stacker Crane problem in Section 3. Next, in Section 4, we introduce our robust algorithm for the setting where a fraction of the samples are corrupted. The formal proofs of these algorithms are deferred to the appendix. In Section 5, we present an empirical evaluation to assess the practical performance of our algorithms. Finally, we generalize our framework to the multi-vehicle scenario in Appendix C.

1.2 Related Work

Online and Incremental Vehicle Routing. The Stacker Crane Problem (SCP) and the Traveling Salesperson Problem (TSP) are fundamental vehicle-routing problems. Offline, SCP admits a 1.8-approximation (Frederickson et al., 1978), while the best-known bound for TSP is slightly better than 1.5 (Karlin et al., 2024). We instead consider the setting in which requests arrive sequentially and must be irrevocably inserted into an evolving tour. Without learning augmentation, the greedy algorithm is $\mathcal{O}(\log n)$-competitive (Rosenkrantz et al., 1977; Imase & Waxman, 1991), and this logarithmic dependence is unavoidable for arbitrary insertion strategies (Azar, 1994). More specifically, Azar (1994) proved lower bound of $\Omega(\log n / \log\log n)$. Recent work has also studied stricter variants in which the available insertion space is limited (Azar et al., 2026). Since TSP is the special case of SCP in which every request has identical source and destination, all lower bounds for incremental TSP apply immediately to incremental SCP.

Stochastic and Sample-Augmented Algorithms. Stochastic vehicle routing has a long history; see, e.g., Gendreau et al. (1996). Particularly relevant is the Probabilistic TSP (Jaillet, 1988; Bert-

simas, 1992), in which a fixed collection of potential locations is known, each location is independently active with a specified probability, and an *a priori* tour is constructed before the active set is revealed. Our setting differs in that request locations are themselves drawn from arbitrary, non-identical distributions and are not known before arrival.

Our single-sample model is also related to online optimization with limited distributional information. Single-sample prophet inequalities use one independent sample from each value distribution (Azar et al., 2014; Rubinstein et al., 2020). A distinct "online algorithms with a sample" model reveals a random subset of the eventual input before the remaining elements arrive; this has been studied for problems including Steiner tree, facility location, load balancing, matching, and covering (Argue et al., 2022; Hebbar et al., 2026). The latter work also considers adversarial corruption of the observed sample. Most closely related to our distributional model, recent work on online metric matching obtains constant guarantees in Euclidean metrics from one sample per request distribution under additional smoothness assumptions (Li et al., 2026). Our results give constant guarantees for incremental TSP and SCP in arbitrary metrics without regularity assumptions on the distributions.

Learning-Augmented Graph and Routing Algorithms. Our work belongs to the broader literature on learning-augmented algorithms, which seeks algorithms that combine *consistency* under accurate predictions, *robustness* under arbitrary errors, and *smoothness* as prediction quality deteriorates (Lykouris & Vassilvitskii, 2021; Purohit et al., 2018). For online graph and metric problems, prior work has developed learning-augmented algorithms for metrical task systems (Antoniadis et al., 2023), Steiner tree (Xu & Moseley, 2022), and other network design problems (Azar et al., 2022; Bernardini et al., 2022). Learning augmentation has also been applied to some versions of online TSP: existing work considers general metrics (Hu et al., 2022), the real line (Gouleakis et al., 2023), and general metrics with efficient implementations for rings, trees, and flowers (Bampis et al., 2023). Predictions have additionally been studied for online TSP with time windows (Chawla & Christou, 2024). These works predict properties of the realized requests, such as their locations or release times, whereas our predictions are independent samples from the request distributions.

2 PRELIMINARIES

Metric and requests. Let $(\mathcal{V}, d)$ be a symmetric metric space, where $d(u, v)$ denotes the distance between locations $u, v \in \mathcal{V}$. We write $\Delta = \max_{u,v\in\mathcal{V}} d(u, v)$ for the diameter of the metric space.

The input consists of n transportation requests $R = \{R_1, \ldots, R_n\}$. Each request $R_i = (s_i, t_i)$ specifies a source $s_i \in \mathcal{V}$ and a destination $t_i \in \mathcal{V}$. To serve R_i, a vehicle must travel from some location ℓ to s_i, incurring deadheading cost $d(\ell, s_i)$ (ℓ may be the same as s_i), and then from s_i to t_i, incurring service cost $d(s_i, t_i)$. We further use the shorthands $R_{<i} = \{R_1, \ldots, R_{i-1}\}$ and $R_{\leq i} = \{R_1, \ldots, R_i\}$ to refer to the respective prefixes of the request sequence, and $V(R)$ for the set of source and destination locations of the requests in R.

A feasible single-vehicle solution is a cyclic ordering π of the requests. Its cost is

$$c(\pi) = \sum_{j=1}^{n} d\big(s_{\pi(j)}, t_{\pi(j)}\big) + \sum_{j=1}^{n} d\big(t_{\pi(j)}, s_{\pi(j+1)}\big),$$

where $\pi(n + 1) = \pi(1)$. The first term is the total mandatory service cost. The second is the total deadheading cost incurred in traveling from the destination of one request to the source of the next. We use $\mathrm{OPT}(R)$ to denote the minimum cost of a feasible offline solution for R.

If $s_i = t_i$ for every i, then all service costs are zero and the problem reduces to finding a tour through the points $s_1, \ldots, s_n$. Thus, metric TSP is a special case of the Stacker Crane problem. We use $\mathrm{OPT}_{TSP}(V)$ to denote the optimal offline TSP tour for a TSP instance over vertices V.

Online Problem. The requests arrive online in the order $R_1, \ldots, R_n$. Upon the arrival of R_i, the algorithm must insert it into its current cyclic ordering. This insertion is irrevocable: the relative order of previously arrived requests cannot subsequently be changed. After all requests have arrived, the cost incurred by the algorithm is the cost of the resulting ordering.

Single-sample model. For each $i \in [n]$, there is an arbitrary distribution $\mathcal{P}_i$ over $\mathcal{V} \times \mathcal{V}$. The online request R_i is drawn independently from $\mathcal{P}_i$. Before any online requests arrive, the algorithm receives an independent sample $R'_i = (s'_i, t'_i) \sim \mathcal{P}_i$. The samples and online requests are mutually independent. The algorithm is given the samples but not explicit descriptions of the distributions. The distributions $\mathcal{P}_1, \ldots, \mathcal{P}_n$ need not be identical and may be entirely unrelated. In particular, our model imposes no stationarity or shared-distribution assumption. It also permits point distributions and therefore includes arbitrary deterministic request sequences.

For an online algorithm $\mathcal{A}$, let $\mathcal{A}(R, R')$ denote its cost on the online requests R given the sample set $R' = \{R'_1, \ldots, R'_n\}$. We say that $\mathcal{A}$ is c-competitive in expectation if

$$\mathbb{E}[\mathcal{A}(R, R')] \leq c \cdot \mathbb{E}[\mathrm{OPT}(R)]\,,$$

where the expectation is over the online requests, the offline samples, and any internal randomness of the algorithm.

Robust Setting. In the robust setting, an adversary replaces at most k samples in R' with arbitrary requests. The index set of the corrupted samples is chosen obliviously, meaning it is strictly independent of both the offline samples and the online requests. However, the adversary can choose the new modified values adaptively after observing the entire set of samples. The identities of the corrupted samples are not revealed to the algorithm. Throughout the paper, we write $p = k/n$ for the fraction of corrupted samples, and we refer to requests whose samples are corrupted as *corrupted requests*.

An algorithm is *consistent* if it achieves a constant competitive ratio when $k = 0$, and *robust* if it retains a nontrivial guarantee for arbitrary corruptions. Our robust algorithm achieves an expected competitive ratio of $\mathcal{O}(\log k)$, which smoothly interpolates between a constant bound when the samples are exact and the worst-case bound of $\mathcal{O}(\log n)$ as the number of corruptions increases.

Multiple vehicles. In the multi-vehicle setting, the algorithm maintains m schedules, one for each vehicle. Each vehicle is associated with a specific, fixed depot location (the depots of different vehicles may be distinct). A valid route for a vehicle must begin at its designated depot, serve its assigned requests, and return to its depot. Each arriving request must be assigned to exactly one vehicle and inserted irrevocably into that vehicle's current schedule without altering the relative order of previously scheduled requests. The total cost is the sum of the costs of the m resulting closed routes. If a vehicle serves no requests, it remains at its depot and incurs a cost of zero.

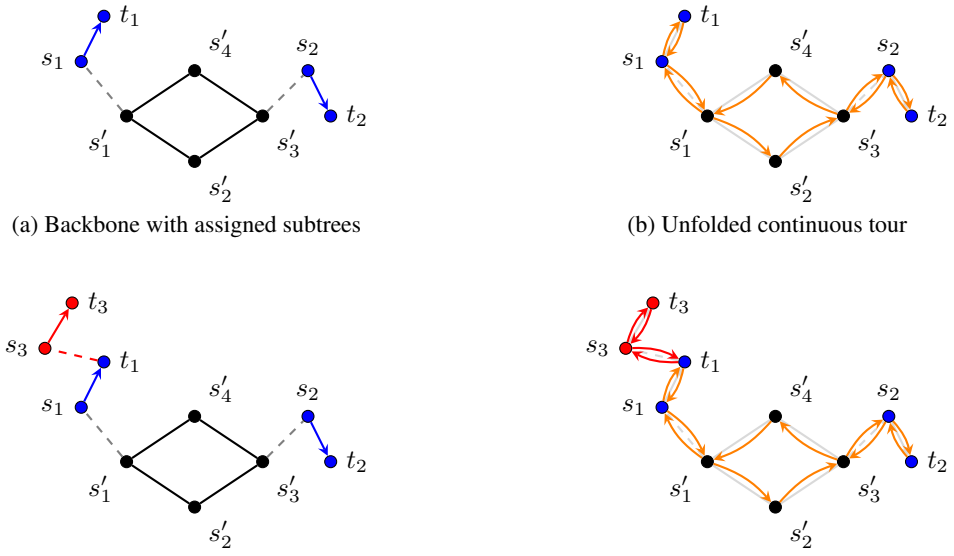
3 Routing with Reliable Samples

In this section, we study the single-vehicle Online Stacker Crane problem when the samples are reliable and drawn from their corresponding distributions. We provide a 3.5-approximate algorithm by precomputing a backbone tour that guides the vehicle's overall path. As online requests arrive, we connect them to the active routing structure to form the final tour, as detailed in Algorithm 1.

To construct the backbone, we compute a 1.5-approximate TSP tour over the n sampled pick-up vertices. When online requests arrive, they are deterministically assigned to their nearest vertex among the sampled pick-ups and the online requests that have already arrived.

Algorithm 1 Offline-Sampling with Star Excursions

Input: Sequence of samples $R'_1, \ldots, R'_n$ where $R'_i = (s'_i, t'_i)$
Output: Eulerian connected graph M which maintains the traversal path of the vehicle
Phase 1: Offline Precomputation
$V_B \leftarrow \bigcup_{i=1}^{n} \{s'_i\}$
$B \leftarrow$ OFFLINE_TSP(V_B) ▷ Generate a backbone tour on sampled pick-ups
Phase 2: Online Assignment & Execution
$M \leftarrow B$
for each arriving online request $R_i = (s_i, t_i)$ **do**
    $v_i \leftarrow \arg\min_{v \in V_B \cup V(R_{<i})} \min(d(v, s_i), d(v, t_i))$
    $M \leftarrow M \cup \{(v_i, s_i), (s_i, t_i), (t_i, v_i)\}$
end for



(a) Backbone with assigned subtrees

(b) Unfolded continuous tour

(c) New request anchored to t_1

(d) Unfolded tour with new request

Figure 1: Visual representation of the offline-sampling backbone and dynamic star-excursions.

Conceptually, the initial backbone acts as a core route. Because online requests can anchor to previously arrived requests as well as to the original backbone, the assignments dynamically form a nested, tree-like structure. During the online execution, the vehicle traverses this evolving structure in a Depth-First Search (DFS) manner. Whenever the vehicle reaches a vertex that has pending requests anchored to it, it pauses its current path to recursively service them. Specifically, to service an assigned request, the vehicle departs from the current node, travels empty to the pick-up location, completes the mandatory trip to the drop-off location, and then returns empty to the anchor node. Appending these closed triangular trips forms a fully connected Eulerian graph covering all requests. When multiple nested branches originate from a single node, the algorithm prioritizes requests based on their arrival time, servicing older requests and their respective sub-branches entirely before moving to newer ones. This chronological prioritization guarantees that the relative ordering of previously scheduled requests remains unaltered when a new request is inserted. Figure 1 illustrates how this tour is constructed and updated after a new request arrives.

Analysis of Algorithm 1. Let $\mathrm{ALG} = c(M)$ be the total cost of the algorithm's final routing sequence. This cost is partitioned into the cost of the backbone tour $c(B)$ and the cost of the back-and-forth excursions to serve the assigned requests. Let w_i be the additional cost incurred to serve request $R_i = (s_i, t_i)$, i.e., $w_i = d(v_i, s_i) + d(s_i, t_i) + d(t_i, v_i)$, where v_i is the closest vertex in our structure to which R_i is connected. Based on Lines 3 and 6, we have:

$$\mathrm{ALG} = c(B) + \sum_{i=1}^{n} w_i.$$

We formally bound the expected value of each component independently. The proofs of these bounds are deferred to Appendix A. We first bound the cost of our backbone.

Lemma 1. *The expected cost of the pre-computed offline backbone tour is bounded by*

$$\mathbb{E}\left[c(B)\right] \leq 1.5\mathbb{E}\left[c(\mathrm{OPT}(R))\right].$$

Then, we bound the additional cost of serving requests.

Lemma 2. *The expected total distance of all star excursions is bounded by*

$$\mathbb{E}\left[\sum_{i=1}^{n} w_i\right] \leq 2\mathbb{E}\left[c(\mathrm{OPT}(R))\right].$$

Combining Lemmas 1 and 2 proves the approximation guarantee of Algorithm 1.

Theorem 1. *For the Online Stacker Crane problem minimizing total cost under independent arrivals from arbitrary distributions, there exists an algorithm achieving an expected competitive ratio of* 3.5 *using a single sample from each distribution.*

4 Robustness to Corrupted Samples

In practical environments, assuming perfectly reliable samples from perfectly accurate distributions is often unrealistic. Data pipelines are subject to noise, and historical logs used to generate the distribution models may contain adversarial entries or outdated parameters. We generalize our framework to address the scenario where a $p = k/n$ fraction of our drawn samples R' are corrupted.

In our formal model, as defined in Section 2, the adversary obliviously selects a set of at most k indices to corrupt, strictly independent of the realized sample values and the sequence of real online arrivals. The remaining uncorrupted samples successfully mirror the true underlying distributions of the online arrivals. In line with standard criteria for learning-augmented algorithms, we design an $\mathcal{O}(\log(pn))$-competitive algorithm that provides strong theoretical guarantees across three key dimensions: consistency, robustness, and smoothness.

4.1 Robust Algorithm via Budget-Doubling

To satisfy consistency, we try to isolate and filter out the corrupted samples. Similar to the previous section, we aim to find a backbone that contains the pick-up locations of all uncorrupted samples while excluding the corrupted ones, as they can excessively increase the backbone cost. However, because we do not know which samples are corrupted in advance, we aim to construct a backbone with almost the same cost as the ideal uncorrupted backbone; we refer to this optimal cost as T^*. Since T^* is also unknown a priori, we employ a budget-doubling guessing framework coupled with the 2-approximation algorithm for the Rooted Metric k-TSP problem by Garg (2005):

Definition 1 (ROOTED_KTSP)**.** *Given a metric graph $G = (\mathcal{V}, d)$, a specified root vertex $r \in \mathcal{V}$, and a target integer k, the Rooted Metric k-TSP asks for a minimum-weight tour that visits at least k vertices in G and contains the root r.*

To define a positive initial budget, we observe and process the arriving online requests until we encounter a prefix whose optimal offline routing cost is strictly greater than zero. Because these initial requests are actual arrivals drawn from the sequence of distributions, they act as guaranteed uncorrupted samples. We compute the baseline cost to serve this prefix and assign this positive cost as our initial budget. We also set the pick-up location s_1 of the very first request as our permanent root for running ROOTED_KTSP. Anchoring backbones of all phases to a single fixed vertex ensures that the routing structures we obtain across all phases remain connected through that shared vertex.

Let budget T be a guess for T^* at this level. To construct the backbone B_T for this level, we run the ROOTED_KTSP subroutine for all target sizes $j \in [n]$ and select the largest target j whose returned tour has length at most $2T$. This provides a useful bicriteria implication: if there exists an appropriate rooted j-tour of cost at most T, our selection covers at least j vertices within length $2T$.

During the online phase, we serve each arriving request by connecting it to the nearest vertex in the current backbone or to a previously arrived request. Crucially, if the online excursion cost in the current phase exceeds the budget limit of $2T$, we halt, double the budget, and initiate the next phase. This guarantees that the total actual incurred cost of any aborted phase is structurally capped. By hitting the limit, we effectively double the budget and search for a larger backbone. The complete pseudocode is provided in Algorithm 2 in Appendix B.

4.2 Analysis of the Robust Algorithm

Let ALG denote the total cost incurred by the algorithm. First, we condition on the sets of online requests R and offline samples R' being fixed. Under this assumption, we bound the maximum budget reached by the algorithm (Lemma 7), and then sum the budgets across all phases to bound the total deterministic cost (Lemma 8). Finally, taking the expectation over the realizations of R' and R yields our main theoretical guarantee (Theorem 2). The proofs of these lemmas are deferred to Appendix B. We begin by defining key concepts for fixed sets of requests R and samples R'.

Definition 2 (Ideal Anchor). *Let $U \subseteq R$ be the set of $(1-p)n$ uncorrupted online requests, and let $U'_P \subseteq V_B$ be the corresponding full set of $(1-p)n$ uncorrupted sampled pick-ups. For each $R_i \in U$, we establish its "ideal anchor" $q_i \in U'_P \setminus \{s'_i\}$ as the absolute nearest uncorrupted sampled pick-up to its drop-off location t_i, i.e., $q_i = \arg\min_{q \in U'_P \setminus \{s'_i\}} d(q, t_i)$. Let $d_i = d(q_i, t_i)$ be the distance between the drop-off of R_i and its ideal anchor.*

Next, we partition the requests into four categories based on whether their distributions were corrupted, whether their anchor is in backbone, and whether an earlier request shares the same anchor:

Definition 3. *For a fixed backbone B, we categorize the requests as follows:*

- $\mathcal{A}_B = R \setminus U$*: Corrupted online requests.*
- $\mathcal{B}_B = \{R_i \in U \mid q_i \notin B \wedge (\forall j < i : R_j \notin U \vee q_j \neq q_i)\}$*: Uncorrupted requests whose anchors are not in the backbone, and for which there is no prior uncorrupted request R_j $(j < i)$ sharing the same anchor.*
- $\mathcal{C}_B = \{R_i \in U \mid q_i \notin B \wedge (\exists j < i : R_j \in U \wedge q_j = q_i)\}$*: Uncorrupted requests whose anchors are not in the backbone, and for which there exists a prior uncorrupted request R_j $(j < i)$ sharing the same anchor.*
- $\mathcal{D}_B = \{R_i \in U \mid q_i \in B\}$*: Uncorrupted requests whose anchors are in the backbone.*

It is easy to see that $\mathcal{A}_B, \mathcal{B}_B, \mathcal{C}_B$, and $\mathcal{D}_B$ are disjoint subsets of R such that their union is R.

Next, we define a cost function representing the expense of serving a request via a specific backbone.

Definition 4. *For a backbone B, we define $w_B(i)$ as the additional cost incurred to serve request R_i, i.e., $w_B(i) = d(v_i, s_i) + d(s_i, t_i) + d(t_i, v_i)$, where v_i is the closest vertex in $V(B) \cup V(R_{<i})$. Additionally, for a subset of requests $X \subseteq R$, we define $w_B(X) = \sum_{R_i \in X} w_B(i)$ as the sum of the additional costs for all requests in X.*

Using these definitions, we upper-bound the assignment cost of categories $\mathcal{A}_B$ and $\mathcal{B}_B$ in the worst-case scenario where the backbone yields no cost savings. In this case, requests connect exclusively to previously arrived requests. To establish this bound, we apply Lemma 9 (adapted from Rosenkrantz et al. (1977) and deferred to Appendix B) along with the fact that $|\mathcal{A}_B|, |\mathcal{B}_B| \leq pn$.

Lemma 3. *For a backbone B with size at least $|B| \geq (1-p)n$, we have*

$$w_B(\mathcal{A}_B \cup \mathcal{B}_B) \leq \mathcal{O}(\log(pn))c(\mathrm{OPT}(R)).$$

Next, we bound the service cost for requests in $\mathcal{D}_B$. Because the ideal anchors for requests in $\mathcal{D}_B$ are inside the backbone, we can bound the insertion cost using the metric triangle inequality by directly analyzing their distance to these anchors, similar to the proof of Lemma 2 in the previous section.

Lemma 4. *For a backbone B, we have*

$$w_B(\mathcal{D}_B) \leq \sum_{R_i \in \mathcal{D}_B} 2\left(d_i + d(s_i, t_i)\right)$$

Finally, we bound the service cost for requests in $\mathcal{C}_B$. While the ideal anchors of these requests are absent from the backbone, we bound their routing cost via a chaining argument. We recursively consider the connection distances through prior uncorrupted requests that share the same ideal anchor.

Lemma 5. *For a backbone B, we have*

$$w_B(\mathcal{C}_B) \leq \sum_{R_i \in \mathcal{B}_B \cup \mathcal{C}_B} 4\left(d_i + d(s_i, t_i)\right)$$

Now that we have bounded costs across all categories, we now determine the maximum total cost for any valid backbone containing at least $(1-p)n$ vertices.

Definition 5 (Additional Cost Bound W_{max}). *We define*

$$W_{max} = \max_{B:|B| \geq (1-p)n} \{w_B(R)\} = \max_{B:|B| \geq (1-p)n} \left\{ \sum_{i=1}^{n} w_B(i) \right\}$$

as a universal upper bound on the additional excursion cost incurred by the online routing strategy for any valid backbone generated with a size of at least $(1-p)n$.

Since categories $\mathcal{A}_B$, $\mathcal{B}_B$, $\mathcal{C}_B$, and $\mathcal{D}_B$ partition R, we obtain a universal upper bound on W_{max} by directly summing the individual components established in Lemmas 3, 4, and 5.

Lemma 6. *For any backbone B with $|B| \geq (1-p)n$, we have*

$$W_{max} \leq \mathcal{O}(\log(pn))c(\mathrm{OPT}(R)) + \sum_{R_i \in U} 4(d_i + d(s_i, t_i))$$

Next, let T^* be the actual optimal cost to cover the $(1-p)n$ uncorrupted sampled pick-up locations. We can bound the budget of the final phase by showing that once the budget exceeds T^*, any computed backbone naturally caps the online excursion cost by W_{max}.

Lemma 7 (Deterministic Halting)**.** *The budget-doubling loop is deterministically guaranteed to halt at a final phase budget T_{final} satisfying $T_{final} \leq 2T^* + W_{max}$.*

Summing budgets across all executed phases yields the following bound on the total incurred cost.

Lemma 8 (Total Actual Cost)**.** *The total actual cost incurred by the algorithm across all phases (including prematurely aborted ones) is deterministically bounded by* $\mathrm{ALG} \leq 16T^* + 8W_{max}$.

Finally, taking the expectation of this cost over all realizations of the offline samples and online requests establishes our main competitive ratio guarantee.

Theorem 2. *For the Online Stacker Crane problem minimizing total cost under independent arrivals from arbitrary distributions, with one independent sample from each distribution, where a p fraction of the samples are corrupted, there exists a robust, consistent, and smooth algorithm achieving an expected competitive ratio of $\mathcal{O}(\log(pn))$.*

5 Experiments

In this section, we empirically evaluate the performance of our proposed learning-augmented algorithms against the greedy online routing and offline optimum algorithm.

Experimental Setup. To properly analyze the algorithms in an adversarial environment, we generate request sequences using the instance introduced by Azar (1994) with $n = 85$, which establishes the $\Omega(\log n / \log\log n)$ lower bound for arbitrary insertion strategies for the online TSP. This adversarial construction actively exploits the localized decision-making of greedy heuristics, making it a rigorous benchmark for evaluating our learning-augmented framework.

We compare four algorithmic approaches:

- **Offline Match & Route (Offline Baseline):** An offline algorithm that computes a 1.8-approximate Stacker Crane solution with full knowledge of all requests in advance. This serves as our empirical proxy for the offline optimal routing cost.
- **Greedy Algorithm (Online Baseline):** The classical nearest-insertion heuristic. At each step, it inserts the newly arrived request into the existing tour at the location that minimizes the immediate increase in total cost. It guarantees an $\mathcal{O}(\log n)$ approximation but makes no use of predictive distributions.
- **Star Excursions (Algorithm 1):** Our baseline sampling-based online routing strategy, which assumes the predicted distributions are perfectly reliable.
- **Robust Budget-Doubling Routing (Algorithm 2):** Our robust framework that employs a budget-doubling technique and a bicriteria k-TSP subroutine to dynamically filter out corrupted spatial outliers.

To simulate the known distribution model, we define the true distribution for each request's vertices as follows: for any vertex v (source or destination) of an online request, its sampled counterpart remains at v with probability 0.5. With the remaining probability 0.5, it is placed at a different vertex u in the metric space, chosen randomly with probability proportional to the inverse of its distance to v (i.e., proportional to $1/d(v,u)$). This ensures the uncorrupted samples are spatially correlated with the true requests while capturing natural variance.

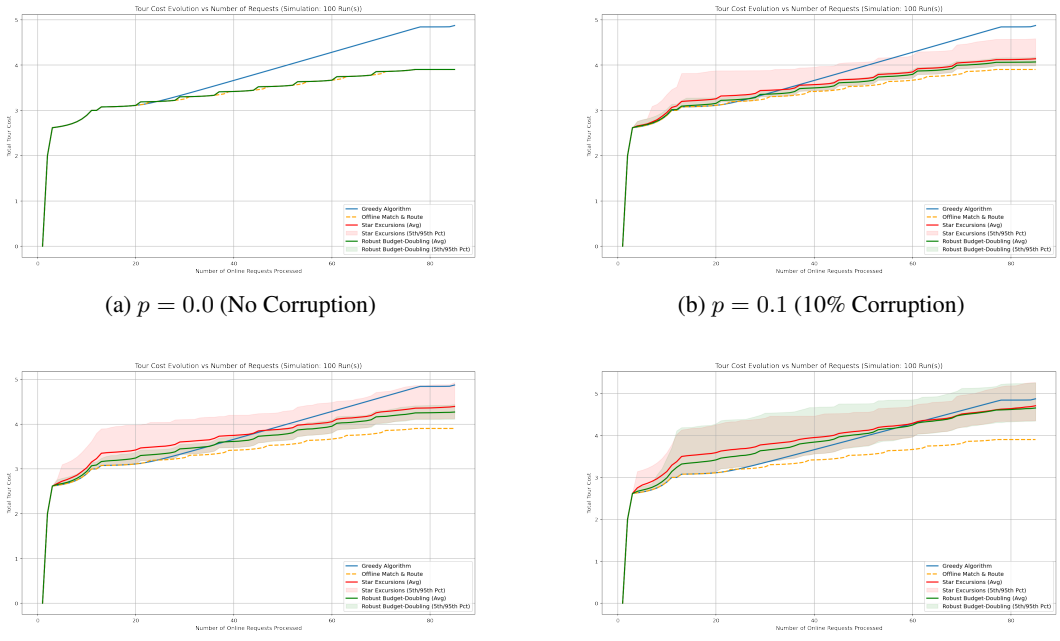
To evaluate our algorithms under varying degrees of reliability, we explicitly test them using a randomized instantiation of our oblivious corruption model. We evaluate the algorithms across four distinct corruption fractions: $p \in \{0.0, 0.1, 0.2, 0.5\}$. For each setting, we run our two proposed algorithms 100 times, drawing a different set of samples each time. In each run, the adversary obliviously selects exactly k sample indices uniformly at random. The sample values at these chosen indices are then corrupted by replacing their spatial coordinates with uniformly random locations. To capture the variance in performance, we plot the average cost across these 100 runs, along with shaded regions denoting 95% confidence intervals.

Results and Discussion. Figure 2 depicts the total routing cost increase as online requests arrive.

When the predictive distributions are completely reliable ($p = 0.0$, Figure 2a), both of our proposed algorithms perform exceptionally well. Their cost trajectories remain nearly indistinguishable from the offline optimal baseline. Conversely, the standard Greedy Algorithm quickly falls victim to the adversarial input, demonstrating a steep increase in routing cost. This visually confirms our constant-factor approximation guarantees when distributions are uncorrupted.

As the corruption fraction p increases (Figures 2b, 2c, and 2d), sample quality degrades, leading to worse overall solution quality that eventually approaches the performance of the Greedy baseline. However, in these noisy environments, the performance gap between our two proposed algorithms becomes evident. Our first algorithm, Star-Excursions, degrades relatively quickly as p grows: because it lacks a filtering mechanism and treats all samples as genuine, it is forced to construct an excessively long backbone to cover corrupted sample pick-ups.

In contrast, our Robust Budget-Doubling Routing algorithm handles corrupted samples much better. By dynamically scaling its budget and anchoring to a robust structure, it tries to isolate corrupted samples and prevents them from ruining the global topology. Consequently, its performance degrades much more gracefully, maintaining a consistent advantage over the Greedy baseline and reliably outperforming the Star-Excursions algorithm. This empirical behavior demonstrates the consistency, robustness, and smoothness properties guaranteed by our framework.



(a) $p = 0.0$ (No Corruption)

(b) $p = 0.1$ (10% Corruption)

(c) $p = 0.2$ (20% Corruption)

(d) $p = 0.5$ (50% Corruption)

Figure 2: Total tour cost evolution versus the number of online requests processed on Azar's adversarial instance across varying corruption fractions (p).

AI USE STATEMENT

In this work, we used generative AI tools for identifying relevant literature, drafting parts of the appendix, editing the paper to improve readability, and creating or editing experiment code. In particular, the theoretical ideas for the multi-vehicle section were entirely generated by the authors and then explained to the AI to produce a draft, which was subsequently edited by the authors. Furthermore, the data generation process was completely designed and decided by the authors, with AI assistance strictly limited to writing the test generation code and the code for running experiments. All code and text generated by AI were reviewed by at least two authors. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in finding the proofs, or proposing hypotheses, and the rest of the required disclosure tasks are not applicable to this work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM transactions on algorithms*, 19(2):1–34, 2023.

C. J. Argue, Alan M. Frieze, Anupam Gupta, and Christopher Seiler. Learning from a sample in online algorithms. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Pablo D. Azar, Robert Kleinberg, and S. Matthew Weinberg. Prophet inequalities with limited information. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1358–1377. SIAM, 2014. doi: 10.1137/1.9781611973402.100.

Yossi Azar. Lower bounds for insertion methods for tsp. *Combinatorics, Probability and Computing*, 3(3):285–292, 1994.

Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In Joseph (Seffi) Naor and Niv Buchbinder (eds.), *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pp. 35–66. SIAM, 2022. doi: 10.1137/1.9781611977073.3. URL https://doi.org/10.1137/1.9781611977073.3.

Yossi Azar, Debmalya Panigrahi, and Or Vardi. Online metric TSP: Beyond the $\sqrt{n}$ barrier, 2026.

Evripidis Bampis, Bruno Escoffier, Themis Gouleakis, Niklas Hahn, Kostas Lakis, Golnoosh Shahkarami, and Michalis Xefteris. Learning-Augmented Online TSP on Rings, Trees, Flowers and (Almost) Everywhere Else. In *Leibniz International Proceedings in Informatics (LIPIcs)*, volume 274 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 12:1–12:17, Amsterdam, Netherlands, September 2023. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. doi: 10.4230/LIPIcs.ESA.2023.12. URL https://hal.sorbonne-universite.fr/hal-04208875.

Giulia Bernardini, Alexander Lindermayr, Alberto Marchetti-Spaccamela, Nicole Megow, Leen Stougie, and Michelle Sweering. A universal error measure for input predictions applied to online graph problems. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/15212bd2265c4a3ab0dbc1b1982c1b69-Abstract-Conference.html.

Dimitris J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40 (3):574–585, 1992.

Shuchi Chawla and Dimitris Christou. Online time-windows TSP with predictions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024)*, volume 317 of *Leibniz International Proceedings in Informatics*

(LIPIcs), pp. 2:1–2:21. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. doi: 10.4230/LIPIcs.APPROX/RANDOM.2024.2.

Greg N Frederickson, Matthew S Hecht, and Chul E Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In Harold N. Gabow and Ronald Fagin (eds.), *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pp. 396–402. ACM, 2005. doi: 10.1145/1060590.1060650. URL https://doi.org/10.1145/1060590.1060650.

Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996. doi: 10.1016/0377-2217(95)00050-X.

Themistoklis Gouleakis, Konstantinos Lakis, and Golnoosh Shahkarami. Learning-augmented algorithms for online TSP on the line. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11989–11996, 2023.

Anish Hebbar, Ravi Kumar, Roie Levin, Joseph Seffi Naor, and Debmalya Panigrahi. Online algorithms with a sample: Tight bounds and adversarial robustness, 2026.

Hsiao-Yu Hu, Hao-Ting Wei, Meng-Hsi Li, Kai-Min Chung, and Chung-Shou Liao. Online tsp with predictions. *arXiv preprint arXiv:2206.15364*, 2022.

Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

Patrick Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936, 1988.

Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. *Oper. Res.*, 72(6):2543–2594, 2024. doi: 10.1287/OPRE.2022.2338. URL https://doi.org/10.1287/opre.2022.2338.

Yingxi Li, Ellen Vitercik, and Mingwei Yang. Smoothed analysis of online metric matching with a single sample: Beyond metric distortion. In Shubhangi Saraf (ed.), *17th Innovations in Theoretical Computer Science Conference, ITCS 2026, Bocconi University, Milan, Italy, January 27-30, 2026*, volume 362 of *LIPIcs*, pp. 94:1–94:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. doi: 10.4230/LIPICS.ITCS.2026.94. URL https://doi.org/10.4230/LIPIcs.ITCS.2026.94.

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

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden (ed.), *Beyond the Worst-Case Analysis of Algorithms*, pp. 646–662. Cambridge University Press, 2020. doi: 10.1017/9781108637435.037. URL https://doi.org/10.1017/9781108637435.037.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Commun. ACM*, 65 (7):33–35, 2022. doi: 10.1145/3528087. URL https://doi.org/10.1145/3528087.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. *Advances in Neural Information Processing Systems*, 31, 2018.

Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581, 1977.

Aviad Rubinstein, Jack Z. Wang, and S. Matthew Weinberg. Optimal single-choice prophet inequalities from samples. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 60:1–60:10. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPIcs.ITCS.2020.60.

David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. ISBN 9780521195270. doi: 10.1017/CBO9780511921735.

Chenyang Xu and Benjamin Moseley. Learning-augmented algorithms for online steiner tree. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8744–8752, 2022. doi: 10.1609/aaai.v36i8.20854.

A PROOFS OF ALGORITHM 1

In this section, we prove the lemmas from Section 3, establishing 3.5-approximation factor of Algorithm 1.

Proof of Lemma 1.

$$\begin{aligned}\mathbb{E}\left[c(B)\right] &\leq 1.5\mathbb{E}\left[c(\text{OPT}_{TSP}(V_B))\right] && \text{(Given Karlin et al. (2024))}\\ &\leq 1.5\mathbb{E}\left[c(\text{OPT}(R'))\right]\\ &= 1.5\mathbb{E}\left[c(\text{OPT}(R))\right],\end{aligned}$$

where the second inequality follows from the fact that the optimal Stacker Crane solution on R' will cover all pick-up vertices in V_B, and the last equality stems from the fact that the samples and the online requests are drawn from the exact same sequence of independent distributions. □

Proof of Lemma 2. Let $V_{B_{-i}} = V_B \setminus \{s'_i\}$ denote the pick-up vertices of the samples excluding the i-th sampled pair R'_i. Because $V_{B_{-i}} \subset V_B$, we can upper-bound the assignment minimization by restricting the search to $V_{B_{-i}}$:

$$\begin{aligned}\mathbb{E}\left[w_i\right] &= \mathbb{E}\left[d(v_i, s_i) + d(s_i, t_i) + d(t_i, v_i)\right]\\ &\leq \mathbb{E}\left[2\min(d(v_i, s_i), d(v_i, t_i)) + 2d(s_i, t_i)\right] && \text{(Metric Triangle Inequality)}\\ &= 2\mathbb{E}\left[\min_{v \in V_B \cup V(R_{<i})} \min(d(v, s_i), d(v, t_i)) + d(s_i, t_i)\right] && \text{(Line 5)}\\ &\leq 2\mathbb{E}\left[\min_{s'_j \in V_{B_{-i}}} \min(d(s'_j, s_i), d(s'_j, t_i)) + d(s_i, t_i)\right] && (V_{B_{-i}} \subset V_B \cup V(R_{<i}))\\ &= 2\mathbb{E}\left[\min_{s'_j \in V_{B_{-i}}} \min(d(s'_j, s'_i), d(s'_j, t'_i)) + d(s'_i, t'_i)\right],\end{aligned}$$

where the last equality follows from the fact that the online request R_i and the sample R'_i are drawn from the exact same distribution $\mathcal{P}_i$ and are completely independent of the remaining sampled pick-up vertices $V_{B_{-i}}$.

Consider the optimal routing sequence $\text{OPT}(R')$ traversing the sampled pairs. In this optimal routing, the pair (s'_i, t'_i) is serviced, incurring distance $d(s'_i, t'_i)$, and is then immediately followed by a deadhead trip from t'_i to the pick-up node of the next pair in the sequence, which we denote $s'_{next(i)}$. Because $s'_{next(i)}$ is guaranteed to be a valid vertex in $V_{B_{-i}}$, the distance to the nearest vertex in $V_{B_{-i}}$ is at most $d(t'_i, s'_{next(i)})$. Therefore, for each pair:

$$\min_{s'_j \in V_{B_{-i}}} \min(d(s'_j, s'_i), d(s'_j, t'_i)) \leq \min(d(s'_{next(i)}, s'_i), d(s'_{next(i)}, t'_i)) \leq d(t'_i, s'_{next(i)}) \tag{1}$$

Summing this upper bound across all n sampled pairs yields the total service cost of $c(\text{OPT}(R'))$ plus the total deadheading cost required to connect the pairs:

$$\sum_{i=1}^{n} \left(d(s'_i, t'_i) + d(t'_i, s'_{next(i)})\right) = c(\text{OPT}(R')) \tag{2}$$

By combining the above equations, we get:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{n} w_i\right] &\leq 2\mathbb{E}\left[\sum_{i=1}^{n} \left(\min_{s'_j \in V_{B_{-i}}} \min(d(s'_j, s'_i), d(s'_j, t'_i)) + d(s'_i, t'_i)\right)\right]\\ &\leq 2\mathbb{E}\left[\sum_{i=1}^{n} \left(d(t'_i, s'_{next(i)}) + d(s'_i, t'_i)\right)\right] && \text{(Equation 1)}\\ &= 2\mathbb{E}\left[c(\text{OPT}(R'))\right] && \text{(Equation 2)}\\ &= 2\mathbb{E}\left[c(\text{OPT}(R))\right],\end{aligned}$$

where the last equality holds because the online requests R and the samples R' are identically distributed. □

Proof of Theorem 1. We substitute the bounds established in Lemma 1 and Lemma 2 directly into the total expected cost equation of the algorithm:

$$\begin{aligned}\mathbb{E}[\mathrm{ALG}] &= \mathbb{E}[c(B)] + \mathbb{E}\left[\sum_{i=1}^{n} w_i\right] \\ &\leq 1.5\mathbb{E}[c(\mathrm{OPT}(R))] + 2 \cdot \mathbb{E}[c(\mathrm{OPT}(R))] \qquad \text{(Lemmas 1 and 2)} \\ &= 3.5\mathbb{E}[c(\mathrm{OPT}(R))]\end{aligned}$$

Thus, the proposed online assignment and excursion strategy guarantees a 3.5 expected competitive ratio against the offline optimum. □

B Omitted Details for the Robust Algorithm

This section contains the full pseudocode for the robust algorithm and the omitted proofs for the lemmas establishing its approximation guarantee.

Algorithm 2 Robust Budget-Doubling Unified Routing

Input: Sequence of samples $R'_1, \ldots, R'_n$ where $R'_i = (s'_i, t'_i)$
Output: Eulerian connected graph M which maintains the traversal path of the vehicle
Phase 1: Offline Pre-computation
Observe the first online request $R_1 = (s_1, t_1)$
$R'_1 \leftarrow R_1$
$V_B \leftarrow \bigcup_{i=1}^{n} \{s'_i\}$
for $j = 1$ to n **do** ▷ Search over different sizes of the backbone
  $O_j \leftarrow \textsc{Rooted_kTSP}(V_B, s_1, j)$ ▷ Find a 2-approximate tour covering j pick-ups
end for
Observe requests up to index ℓ until their optimal routing cost C is strictly positive
$T \leftarrow C$
$i \leftarrow \ell + 1$
$M \leftarrow \emptyset$
Phase 2: Online Assignment & Budget Doubling
while $i \leq n$ **do**
  $B_T \leftarrow \emptyset$
  for $j = 1$ to n **do** ▷ Search over different sizes of the backbone
    if $c(O_j) \leq 2T$ **then**
      $B_T \leftarrow O_j$
    end if
  end for
  $M \leftarrow M \cup B_T$
  $W_T \leftarrow 0$
  while $i \leq n$ **do**
    Consider the current online request $R_i = (s_i, t_i)$
    $v_i \leftarrow \arg\min_{v \in V(B_T) \cup V(R_{<i})} \min(d(v, s_i), d(v, t_i))$
    $w_i \leftarrow d(v_i, s_i) + d(s_i, t_i) + d(t_i, v_i)$
    if $W_T + w_i > 2T$ **then**
      $T \leftarrow 2T$ ▷ Threshold exceeded; double the budget guess
      break ▷ Halt phase; retry handling R_i with the doubled budget in the next phase
    end if
    $M \leftarrow M \cup \{(v_i, s_i), (s_i, t_i), (t_i, v_i)\}$
    $W_T \leftarrow W_T + w_i$
    $i \leftarrow i + 1$
  end while
end while

Proofs of Algorithm 2. In this section, we provide the deferred proofs for the lemmas introduced in Section 4.2 to establish the robustness and exact bounding of Algorithm 2.

Lemma 9 (Inspired by Rosenkrantz et al., 1977)**.** *Let $O \subseteq R$ with $|O| \geq 2$ be a subset of online requests. The total excursion cost W_O incurred by servicing O is bounded by:*

$$W_O = \mathcal{O}(\log(|O|))c(\text{OPT}(R)).$$

Proof. Let $\text{OPT}(O)$ be the optimal solution for satisfying all requests in O. Let C_O be the cycle derived from $\text{OPT}(O)$ wherein the pick-up and drop-off of each request are contracted into a single vertex. We can create a matching on the vertices of O and match each vertex to one of its adjacent vertices in C_O such that at most one vertex remains unmatched.

If the vertices corresponding to requests R_i and R_j are matched, one must arrive before the other. Assume without loss of generality that $i > j$, meaning R_j arrives before R_i. When R_i arrives, it will connect to a vertex v_i that minimizes $\min_{v \in V(B) \cup V(R_{<i})}(d(v, s_i), d(v, t_i))$. Since $j < i$, the vertices of R_j are in $V(R_{<i})$. Combining this with the metric triangle inequality, and independent of the backbone B, we have $w_B(i) \leq 2\left(d(s_i, t_i) + \min(d(s_j, s_i), d(s_j, t_i), d(t_j, s_i), d(t_j, t_i))\right)$, where the latter term evaluates the distance between the corresponding vertices in C_O. Therefore, for each matched pair, the cost we need to pay for one of them is at most their distance in C_O. Since vertices are matched exclusively to their adjacent counterparts, the total cost of this matching is bounded by $c(C_O)$.

Furthermore, since we formed C_O by contracting the pick-up and drop-off of each request, we have $c(C_O) + \sum_{R_i \in O} d(s_i, t_i) = c(\text{OPT}(O)) \leq c(\text{OPT}(R))$. This means that for $\lfloor \frac{|O|}{2} \rfloor$ of the vertices in O, the sum of their w_B values is bounded by $2c(\text{OPT}(R))$. If we recursively apply this analysis on the remaining requests of O, after $\lceil \log_2(|O|) \rceil$ rounds, we will have bounded the total sum of w_B for all requests in O by $\mathcal{O}(\log(|O|))c(\text{OPT}(R))$. □

Proof of Lemma 3. If we set $O = \mathcal{A}_B \cup \mathcal{B}_B \cup \{R_1\}$, using Lemma 9, we obtain

$$w_B(\mathcal{A}_B \cup \mathcal{B}_B) \leq \mathcal{O}(\log(|\mathcal{A}_B| + |\mathcal{B}_B| + 1))c(\text{OPT}(R)).$$

Given Definition 3, and since there are pn corrupted requests, we know $|\mathcal{A}_B| = pn$.

Moreover, since $|B| \geq (1-p)n$, the number of distinct anchors not in the backbone B is at most pn. From Definition 3, we know that for each anchor, only the first request assigned to that anchor belongs to $\mathcal{B}_B$. Therefore, $|\mathcal{B}_B| \leq pn$.

Combining the above inequalities completes the proof. Note that we included R_1 in O to ensure that the first request in $\mathcal{A}_B \cup \mathcal{B}_B$ has a preceding request to connect to, allowing us to accurately bound its connection cost. □

Proof of Lemma 4. We can prove this by noting that the anchors of requests in $\mathcal{D}_B$ belong to the backbone B. In the worst case, each request is connected directly to its anchor.

$$\begin{aligned}
w_B(\mathcal{D}_B) &= \sum_{R_i \in \mathcal{D}_B} (d(v_i, s_i) + d(s_i, t_i) + d(t_i, v_i)) \\
&\leq \sum_{R_i \in \mathcal{D}_B} 2(\min(d(v_i, s_i), d(v_i, t_i)) + d(s_i, t_i)) && \text{(Metric Triangle Inequality)} \\
&= \sum_{R_i \in \mathcal{D}_B} 2\left(\min_{v \in V(B) \cup V(R_{<i})} \min(d(v, s_i), d(v, t_i)) + d(s_i, t_i)\right) && \text{(Line 22)} \\
&\leq \sum_{R_i \in \mathcal{D}_B} 2(d(q_i, t_i) + d(s_i, t_i)) && (q_i \in V(B) \cup V(R_{<i})) \\
&= \sum_{R_i \in \mathcal{D}_B} 2(d_i + d(s_i, t_i)). && \text{(Definition 2)}
\end{aligned}$$

□

Proof of Lemma 5. We upper-bound the routing cost for a request $R_i \in \mathcal{C}_B$ using a chaining sequence. For request i, the ideal anchor q_i is not part of the backbone B. However, there exists a prior uncorrupted request $R_{prev(i)}$ ($prev(i) < i$) sharing the same anchor, meaning $q_{prev(i)} = q_i$.

$$
\begin{aligned}
w_B(\mathcal{C}_B) &= \sum_{R_i \in \mathcal{C}_B} (d(v_i, s_i) + d(s_i, t_i) + d(t_i, v_i)) \\
&\leq \sum_{R_i \in \mathcal{C}_B} 2(\min(d(v_i, s_i), d(v_i, t_i)) + d(s_i, t_i)) && \text{(Metric Triangle Inequality)} \\
&= \sum_{R_i \in \mathcal{C}_B} 2\left(\min_{v \in V(B) \cup V(R_{<i})} \min(d(v, s_i), d(v, t_i)) + d(s_i, t_i)\right) && \text{(Line 22)} \\
&\leq \sum_{R_i \in \mathcal{C}_B} 2(d(t_{prev(i)}, t_i) + d(s_i, t_i)) && (t_{prev(i)} \in V(R_{<i})) \\
&\leq \sum_{R_i \in \mathcal{C}_B} 2(d(t_{prev(i)}, q_i) + d(q_i, t_i) + d(s_i, t_i)) && \text{(Metric Triangle Inequality)} \\
&= \sum_{R_i \in \mathcal{C}_B} 2d_{prev(i)} + \sum_{R_i \in \mathcal{C}_B} 2(d_i + d(s_i, t_i)) && (q_{prev(i)} = q_i) \\
&\leq \sum_{R_i \in \mathcal{B}_B \cup \mathcal{C}_B} 2d_i + \sum_{R_i \in \mathcal{C}_B} 2(d_i + d(s_i, t_i))
\end{aligned}
$$

The last inequality holds because for each request R_j, there is at most one unique request R_i such that $prev(i) = j$. Furthermore, we utilize the fact that $R_{prev(i)} \in \mathcal{B}_B \cup \mathcal{C}_B$ since, given $R_i \in \mathcal{C}_B$, because $q_{prev(i)} = q_i \notin B$. □

Proof of Lemma 6. To prove this lemma, we use the fact that $\mathcal{A}_B$, $\mathcal{B}_B$, $\mathcal{C}_B$, and $\mathcal{D}_B$ are disjoint and their collective union is R. By combining Lemmas 4 and 5, we get:

$$
w_B(\mathcal{C}_B \cup \mathcal{D}_B) \leq \sum_{R_i \in \mathcal{B}_B \cup \mathcal{C}_B} 4\,(d_i + d(s_i, t_i)) + \sum_{R_i \in \mathcal{D}_B} 2\,(d_i + d(s_i, t_i)) \leq \sum_{R_i \in U} 4\,(d_i + d(s_i, t_i))
$$

Combining this directly with Lemma 3, we obtain:

$$
w_B(R) = w_B(\mathcal{A}_B \cup \mathcal{B}_B) + w_B(\mathcal{C}_B \cup \mathcal{D}_B) \leq \mathcal{O}(\log(pn))c(\text{OPT}(R)) + \sum_{R_i \in U} 4\,(d_i + d(s_i, t_i))
$$

Since the right-hand side is inherently independent of the backbone B, this completes the proof. □

Lemma 10. *During any phase where the active budget satisfies $T \geq T^*$, the selected robust backbone B_T successfully captures at least $(1-p)n$ sampled pick-ups.*

Proof. In any phase where $T \geq T^*$, the guessing loop in Line 13 explicitly evaluates the target size $j = (1-p)n$. The Rooted k-TSP algorithm guarantees a tour visiting at least j vertices with a length of at most $2T^*$ Garg (2005). Therefore, we have $c(O_j) \leq 2T^* \leq 2T$, which means this tour satisfies the length filter in Line 14. Consequently, B_T captures at least $(1-p)n$ sampled pick-up vertices at a cost of at most $2T$. □

Proof of Lemma 7. For any phase where $T \geq T^*$, Lemma 10 structurally guarantees that our backbone has at least $(1-p)n$ vertices. Combined with Definition 5, this implies that $W_T \leq W_{max}$, even if we connect all requests to the backbone in this phase and ignore the budget doubling limit.

In the phase prior to the final phase (which ran on a budget of $T_{final}/2$), the algorithm must have triggered the limit in Line 24 and subsequently doubled the budget. Thus, we know that $W_{T_{final}/2} + w_i > 2(T_{final}/2) = T_{final}$, where R_i is the exact request that triggered the budget doubling. Now we have two cases:

- $T_{final}/2 < T^*$.

- $T_{final}/2 \geq T^*$. In this scenario, Lemma 10 guarantees that our backbone features at least $(1-p)n$ vertices. By Definition 5, the total bounded routing cost of any backbone meeting this size is at most W_{max}. Because every additional excursion cost w_j is non-negative, the partial sum representing the cost incurred so far ($W_{T_{final}/2}+w_i$) can never exceed the total cost bound (W_{max}). Combining this inequality ($W_{T_{final}/2} + w_i \leq W_{max}$) directly with the aforementioned $W_{T_{final}/2} + w_i > T_{final}$ yields $T_{final} < W_{max}$.

Since the above scenarios cover all possibilities, we can conclusively bound T_{final} by $T_{final} \leq \max(2T^*, W_{max}) \leq 2T^* + W_{max}$. □

Proof of Lemma 8. Because the budget iteratively doubles, the total budget spent across all aborted guesses naturally forms a geometric series culminating in the successful phase budget T_{final}. The total expenditure is bounded by:

$$\text{ALG} \leq \sum_{\text{phases}} c(M_{\text{phase}}) = \sum_{\text{phases}} (c(B_{T_{\text{phase}}}) + W_{T_{\text{phase}}}) \leq \sum_{\text{phases}} 4T_{\text{phase}} \leq 2(4T_{final}) = 8T_{final}$$

Substituting the deterministic bound for T_{final} established in Lemma 7, the total actual cost is at most $8(2T^* + W_{max}) = 16T^* + 8W_{max}$. □

Proof of Theorem 2. We apply the above lemmas to bound the expected value of ALG over any statistical realization of the samples and online requests:

$$\begin{aligned}
\mathbb{E}[\text{ALG}] &\leq 16\mathbb{E}[T^*] + 8\mathbb{E}[W_{max}] && \text{(Lemma 8)} \\
&\leq 16\mathbb{E}[T^*] + 8\mathbb{E}\left[\mathcal{O}(\log(pn))c(\text{OPT}(R)) + \sum_{R_i \in U} 4(d_i + d(s_i, t_i))\right] && \text{(Lemma 6)} \\
&= \mathcal{O}(\log(pn))\mathbb{E}[c(\text{OPT}(R))] + 32\mathbb{E}\left[\sum_{R_i \in U} (d_i + d(s_i, t_i))\right] \\
& && (\mathbb{E}[T^*] \leq \mathbb{E}[c(\text{OPT}(R))])
\end{aligned}$$

Showing that $\mathbb{E}\left[\sum_{R_i \in U}(d_i + d(s_i, t_i))\right] \leq \mathbb{E}[c(\text{OPT}(R))]$ completes the proof.

Because the ideal anchor q_i evaluates its distance against the *full* set of uncorrupted sampled pick-ups U'_P, it is independent of the k-TSP solver's specific algorithmic choices. This structural property completely eliminates any risk of selection bias. By invoking the i.i.d. property, substituting t_i for its identically distributed independent counterpart t'_i yields:

$$\begin{aligned}
\mathbb{E}[d_i] &= \mathbb{E}\left[\min_{q \in U'_P \setminus \{s'_i\}} d(t_i, q)\right] \\
&= \mathbb{E}\left[\min_{q \in U'_P \setminus \{s'_i\}} d(t'_i, q)\right] \\
&\leq \mathbb{E}\left[d(t'_i, s'_{next(i)})\right]
\end{aligned}$$

where $s'_{next(i)}$ is defined as the pick-up node of the next pair in the optimal offline routing sequence traversing the uncorrupted sampled pairs. This establishes:

$$\mathbb{E}\left[\sum_{R_i \in U} (d_i + d(s_i, t_i))\right] \leq \mathbb{E}\left[\sum_{R_i \in U} \left(d(t'_i, s'_{next(i)}) + d(s'_i, t'_i)\right)\right]$$

By combining the expected deadhead distance with the expected service distances $d(s'_i, t'_i)$, the total cost matches that of the optimal offline Stacker Crane tour $\text{OPT}(U')$ on the uncorrupted sample set $U' \subseteq R'$:

$$\mathbb{E}\left[\sum_{R_i \in U} \left(d(s'_i, t'_i) + d(t'_i, s'_{next(i)})\right)\right] \leq \mathbb{E}[c(\text{OPT}(U'))] \leq \mathbb{E}[c(\text{OPT}(R))]$$

□

C Extension to Multiple Vehicles

In this section, we extend our algorithmic framework to accommodate a multi-vehicle setting. Instead of assigning all online requests to a single vehicle, we operate a fleet of m vehicles. As formalized in Section 2, each vehicle originates from and must return to a designated, fixed depot location (the depots of different vehicles may be distinct). When an online request arrives, we must dynamically assign it to one of the m vehicles without altering prior assignments or changing the relative chronological execution order of previously scheduled requests for each vehicle.

In this multi-vehicle setting, our structural backbone naturally generalizes to consist of exactly m individual tours instead of a single tour, with each tour rooted at its respective depot. Arriving online requests are globally assigned to their nearest vertex across the entire m-tour forest, and the respective vehicle associated with that tour executes the localized star-like excursion. Because every vehicle must return to its designated depot, every active vehicle is structurally guaranteed to form a closed cyclic route. This inherently resolves edge cases involving single-request assignments without requiring additional artificial assumptions.

C.1 Routing with Reliable Samples

For our first result (where the samples are fully reliable), instead of running an offline TSP algorithm to find a single tour on the sampled pick-ups V_B, we compute a Multi-Vehicle TSP to find m tours that collectively cover all sample pick-ups and connect them to the m respective depots.

To find such a structure efficiently, we conceptually contract all m depots $\{o_1, \ldots, o_m\}$ into a single super-vertex v_{depot}. We then compute a standard Minimum Spanning Tree (MST) spanning v_{depot} and the sampled pick-ups V_B. When we un-contract the super-vertex v_{depot} back into the original distinct depots, the tree naturally decomposes into a multi-vehicle Minimum Spanning Forest (MSF) consisting of exactly m connected components, where each component is rooted at exactly one depot (some depots may have no assigned vertices). By unfolding these trees (duplicating every edge), we obtain a 2-approximate solution for the offline Multi-Vehicle TSP problem. By substituting the single tour with this m-tour backbone, our algorithm yields a 4-approximation for the multi-vehicle Online Stacker Crane problem.

Lemma 11. *For a fleet of m vehicles operating under independent arrivals from arbitrary distributions, with one independent sample from each distribution, the offline-sampling algorithm achieves an expected competitive ratio of 4.*

Proof. Let B denote the backbone consisting of m tours obtained by unfolding the optimal m-component MSF on the sampled pick-ups V_B and their respective depots. The optimal offline multi-vehicle Stacker Crane solution $\mathrm{OPT}(R')$ inherently partitions the samples into at most m continuous traversal paths that originate from and return to the respective depots and collectively cover all pick-up vertices. Therefore, the cost of the optimal m-component MSF is at most $c(\mathrm{OPT}(R'))$. Because unfolding the trees doubles the edge weights, the expected cost of our backbone is bounded by:

$$\begin{aligned}\mathbb{E}\left[c(B)\right] &\leq 2 \cdot \mathbb{E}\left[c(\mathrm{OPT}(R'))\right] \\ &= 2 \cdot \mathbb{E}\left[c(\mathrm{OPT}(R))\right],\end{aligned}$$

where the equality follows from the fact that the samples and the online requests are drawn from the exact same sequence of independent distributions, directly mirroring the property established in Lemma 1.

For the localized excursions, the assignment logic remains identical: each online request R_i is anchored to its closest available node in the combined backbone. Because every assigned route is anchored to a fixed depot, the node $s'_{next(i)}$ (the pick-up node of the next pair assigned to the *same vehicle* in the optimal multi-vehicle sequence for samples, or the vehicle's depot if R_i is its final request) is well-defined and exists within the remaining pool of assigned nodes (or the depots). Therefore, the spatial distance mapping to the nearest sampled counterpart is bounded exactly as in the single-vehicle case (see Equation 1), completely preserving the expected total one-way distance bound of all star excursions established in Lemma 2:

$$\mathbb{E}\left[\sum_{i=1}^{n} w_i\right] \leq 2 \cdot \mathbb{E}\left[c(\mathrm{OPT}(R))\right].$$

Summing the expected cost of the m-tour backbone and the expected cost of all localized excursions yields a total expected routing cost bounded by $4\mathbb{E}\left[c(\mathrm{OPT}(R))\right]$, establishing the 4-approximation. □

C.2 Robustness to Corrupted Samples

In the corrupted samples setting, we aim to find m tours, rooted at their respective fixed depots, with a total maximum cost of the given budget that cover the most sample pick-ups. This can be done via a direct reduction to a rooted $k-$MST approximation subroutine.

Lemma 12. *For a fleet of m vehicles operating under independent arrivals from arbitrary distributions, with one independent sample from each distribution, where a p fraction of the samples are corrupted, there exists a robust algorithm achieving an expected competitive ratio of $\mathcal{O}(\log(pn))$.*

Proof. Let $D = \{o_1, \ldots, o_m\}$. Construct a weighted graph with a root r and one vertex per sample index, with edge costs

$$c(i,j) = d(s'_i, s'_j), \qquad c(r,i) = \min_{o \in D} d(o, s'_i).$$

Record a minimizing depot for each root edge, without taking the metric closure.

For each target coverage $j \in \{1, \ldots, n\}$, apply Garg's rooted k-MST with target $j+1$ vertices, counting the root. Also, allow $j = 0$, represented by the root alone. Given a budget guess T, select the largest j whose returned tree has cost at most $2T$. Replace each selected root edge by its recorded depot edge and include unused depots as isolated vertices. Doubling the resulting forest produces vehicle backbones of total cost at most $4T$.

If a feasible depot-routing solution covers j sample indices with cost at most T, contracting its depots and removing cycles gives a feasible rooted tree of cost at most T. The approximation therefore returns a candidate covering at least j sample indices with cost at most $2T$. Therefore, our selection covers at least j sample indices with total backbone cost at most $4T$, establishing the required bicriteria guarantee.

Initialize each vehicle's route at its depot. For each arriving request, define

$$b_i = \min_{o \in D} \big[d(o, s_i) + d(s_i, t_i) + d(t_i, o) \big].$$

Before initialization, serve requests with $b_i = 0$ immediately at a minimizing depot. At the first request with $b_i > 0$, set $T_0 = T = b_i$ and process that request through the budget-doubling procedure. If no such request occurs, return the resulting zero-cost routes.

In each phase, use Algorithm 2's excursion rule with candidate anchors

$$D \cup V(B_T) \cup V(R_{<i}).$$

Insert each new backbone tour into its vehicle's route at its depot. Assign each new request to the vehicle owning its chosen anchor occurrence, preserving all previous assignments and relative orders. Before accepting an excursion that would make the phase's excursion cost exceed $2T$, double T, construct a new backbone, and retry the same request. Each phase costs at most $4T + 2T = 6T$, so the total cost is at most $12T_{\mathrm{final}}$, where T_{final} is the budget used in the final phase.

Use the notation of Section 4.2, with ideal anchors chosen from $D \cup \{s'_j : R_j \in U,\ j \neq i\}$. Let T^* be the minimum depot-rooted forest cost covering the uncorrupted sampled pickups. The swapping argument of Lemma 2, charging each sampled request to its successor or return depot, gives the first bound below. The forest bound of Lemma 11 and monotonicity under deleting requests give the second:

$$\mathbb{E}\left[\sum_{R_i \in U} (d_i + d(s_i, t_i)) \right] \leq \mathbb{E}[c(\mathrm{OPT}(R))], \qquad \mathbb{E}[T^*] \leq \mathbb{E}[c(\mathrm{OPT}(R))].$$

For any backbone covering at least $(1-p)n$ sample indices, $|A_B|, |B_B| \leq pn$. For this analysis, identify the depots as one root and use the induced shortest-path metric. The optimal routes concatenate into a tour of cost at most $c(\mathrm{OPT}(R))$. For each pickup of a request in $A_B \cup B_B$, consider

its distance to the root or an earlier pickup from the same set. The pickups with the ℓ largest such distances, together with the root, are pairwise separated by at least the ℓ-th largest distance. Shortcutting the tour through these $\ell + 1$ points bounds that distance by $c(\mathrm{OPT}(R))/(\ell + 1)$. Summing these bounds and adding service costs yields

$$w_B(A_B \cup B_B) \le O(\log(pn))\, c(\mathrm{OPT}(R)).$$

Applying Lemmas 4–5, with $\mathrm{prev}(i)$ chosen as the immediately preceding uncorrupted request sharing q_i, therefore gives

$$W_{\max} \le O(\log(pn))\, c(\mathrm{OPT}(R)) + 4 \sum_{R_i \in U} (d_i + d(s_i, t_i)).$$

Whenever $T \ge T^*$, the backbone covers at least $(1 - p)n$ sample indices. The doubling argument, including the case with no doubling, gives

$$T_{\mathrm{final}} \le \max\{T_0, 2T^*, W_{\max}\}.$$

Since $T_0 \le c(\mathrm{OPT}(R))$ by shortcutting to the initializing request, the phase-cost bound implies

$$\mathbb{E}[\mathrm{ALG}] \le 12\, \mathbb{E}[T_0 + 2T^* + W_{\max}] \le O(\log(pn))\, \mathbb{E}[c(\mathrm{OPT}(R))].$$

□