Learning-Augmented Algorithms for Online TSP on the Line

Themistoklis Gouleakis[1], **Konstantinos Lakis**[2], **Golnoosh Shahkarami**[3]

[1]National University of Singapore
[2]ETH Zürich
[3]Max Planck Institute for Informatics, Universität des Saarlandes
tgoule@nus.edu.sg, klakis@student.ethz.ch, gshahkar@mpi-inf.mpg.de

Abstract

We study the online Traveling Salesman Problem (TSP) on the line augmented with machine-learned predictions. In the classical problem, there is a stream of requests released over time along the real line. The goal is to minimize the makespan of the algorithm. We distinguish between the *open* variant and the *closed* one, in which we additionally require the algorithm to return to the origin after serving all requests. The state of the art is a 1.64-competitive algorithm and a 2.04-competitive algorithm for the closed and open variants, respectively. In both cases, a tight lower bound is known.

In both variants, our primary prediction model involves predicted *positions* of the requests. We introduce algorithms that (i) obtain a tight 1.5 competitive ratio for the closed variant and a 1.66 competitive ratio for the open variant in the case of perfect predictions, (ii) are robust against unbounded prediction error, and (iii) are smooth, i.e., their performance degrades gracefully as the prediction error increases.

Moreover, we further investigate the learning-augmented setting in the *open* variant by additionally considering a prediction for the last request served by the optimal offline algorithm. Our algorithm for this enhanced setting obtains a 1.33 competitive ratio with perfect predictions while also being smooth and robust, beating the lower bound of 1.44 we show for our original prediction setting for the open variant. Also, we provide a lower bound of 1.25 for this enhanced setting.

Introduction

The Traveling Salesman Problem (TSP) is one of the most fundamental and widely studied problems in computer science, both in its offline version (Lawler 1985), where the input is known in advance, and the online version (Ausiello et al. 2001) where it arrives sequentially. In this paper, we consider the online Traveling Salesman Problem (TSP) on the real line. This version of the problem arises in real-world scenarios such as one dimensional delivery/collection tasks. Such tasks include the operation of elevator systems, robotic screwing/welding, parcel collection from massive storage facilities and cargo collection along shorelines (Ascheuer et al. 1999; Psaraftis et al. 1990). An illustrative example, which was also described in (Chen et al. 2019), is the following. Consider a robot that is used in a row of storage shelves in an intelligent warehouse of a large shipping company. This robot is tasked with moving left and right on the aisle to collect the items ordered by customers. However, these orders arrive online, meaning that the release time and location of the item are only revealed at the time of order. The goal is to route the robot such that it collects and returns all the items as soon as possible, minimizing the makespan as it is called. This practically interesting task is indeed captured by the theoretical framework of online TSP on the line. The offline version of the problem can be solved in quadratic time (Bjelde et al. 2021), therefore the difficulty lies in the late availability of the input, such as the locations of the requests. However, the great availability of data as well as the improved computer processing power and machine learning algorithms can make it possible for predictions to be made on these locations (e.g combining information from historical data, events that may affect demand of stored items, etc). In a line of work that started a few years ago (Lykouris and Vassilvtiskii 2018) and sparked a huge interest (Purohit, Svitkina, and Kumar 2018; Antoniadis et al. 2020a; Gollapudi and Panigrahi 2019; Wang and Li 2020; Angelopoulos et al. 2020; Wei 2020; Rohatgi 2020), it has been demonstrated that such prior knowledge about the input of an online algorithm has the potential to achieve improved performance (i.e competitive ratio) compared to known algorithms (or even lower bounds) that do not use (resp. assume the absence of) any kind of prediction. Therefore, it is natural to consider ways to utilize this information in this problem using a so-called learning-augmented approach.

The input to our online algorithm consists of a set of requests, each associated with a position on the real line as well as a release time. An algorithm for this problem faces the task of controlling an agent that starts at the origin and can move with at most unit speed. The agent may serve a request at any time after it is released. The algorithm's objective is to minimize the makespan, which is the total time spent by the agent before serving all requests. We have two different variants of the problem, depending on whether the agent is required to return to the origin after serving all the requests or not. This requirement exists in the *closed* variant, while it does not in the *open* variant. The makespan in the closed variant is the time it takes the agent to serve the requests *and* return to the origin.

We quantify the performance of an online algorithm by its

competitive ratio, i.e., the maximum ratio of the algorithm's *cost* to that of an optimal *offline* algorithm OPT, over all possible inputs. We say that an algorithm with a competitive ratio of c is c-competitive. Under this scope, the online TSP on the line has been extensively studied and there have been decisive results regarding lower and upper bounds on the competitive ratio for both variants of the problem. Namely, a tight bound of ≈ 1.64 was given for the closed variant, while the corresponding value for the open variant was proven to be ≈ 2.04 (Ausiello et al. 2001; Bjelde et al. 2021). However, no previous work exists for the learning-augmented setting of the problem, and neither does for the learning-augmented setting of any other restriction of the online TSP on general metric spaces. We address this shortcoming in this work.

Our Setup

First of all, to define our prediction model and algorithms, it is necessary to know the number of requests n[1]. This setting shows up in various real world scenarios. For example, in the case of item collection from a horizontal/vertical storage facility, the capacity of the receiving vehicle, which awaits the successful collection of all items in order to deliver them to customers, dictates the number of items to be collected. We note that since n is known, we can assume that each prediction corresponds to a specific request determined by a given labeling, which is shared by both sets (requests and predictions). Under this assumption, we define the $LOCATIONS$ prediction model. In this model, the predictions are estimates for the positions of the requests. The error η increases along with the maximum distance of a predicted location to the actual location of the identically labeled request and is normalized by the length of the smallest interval containing the entire movement of the optimal algorithm. We also define an enhanced prediction model for the open variant named $LOCATIONS + FINAL$ (LF in short) that additionally specifies a request which is predicted to be served last by OPT. In this model, we additionally consider the error metric δ, which increases with the distance of the predicted request to the request actually served last by OPT. We also normalize δ in the same way as η. These models and their respective errors are defined formally in the Preliminaries section.

Properties of Learning-Augmented Algorithms. In the following we formalize the consistency, robustness and smoothness properties. We say that an algorithm is:

1) α-*consistent*, if it is α-*competitive* with no prediction error.

2) β-*robust*, if it is β-*competitive* with any prediction error.

3) γ-*smooth* for a continuous function $\gamma(err)$, if it is $\gamma(err)$-*competitive*, where err is the prediction error. Note that err could potentially be a tuple of different errors.

In general, if c is the best competitive ratio achievable without predictions, it is desirable to have $\alpha < c$, $\beta \leq k \cdot c$ for some constant k and also the function γ should increase from α to β along with the error err. We note that c, α, β and the outputs of γ may be functions of the input and not constant in that regard.

Our Contributions

Throughout this paper, we give upper and lower bounds for our three different settings (closed variant-$LOCATIONS$, open variant-$LOCATIONS$, and open variant-$LOCATIONS + FINAL$). These settings are deterministic, i.e. the algorithms do not have access to random numbers. We do not consider any randomized settings in this work. The lower bounds refer to the case of perfect predictions and are established via different $attack$ strategies. That is, we describe the actions of an adversary ADV, who can control only the release times of the requests and has the goal of *maximizing* the competitive ratio of any algorithm ALG. We emphasize that ADV is given the power to observe ALG's actions and act accordingly. In more detail, ADV does not need to specify the release times in advance, but can release a request at time t, taking the actions of ALG until time t into account. This is, in fact, the most powerful kind of adversary. The upper bounds are established via our algorithms and are defined for every value of the error(s). Recall that η and δ refer to the two types of error we consider. Our algorithms and attack strategies are intuitively described in their respective sections. We now present the main ideas and our results.

Closed Variant under *LOCATIONS*. We will start by intuitively describing our algorithm for this setting and then continue with our lower bound. We design the algorithm $FARFIRST$. The main idea is that we first focus entirely on serving the requests on the side with the furthest extreme, switching to the other side when all such requests are served. When serving the requests on one side, we prioritize them by order of decreasing amplitude. The intuition is that we have the least possible amount of leftover work for our second departure from the origin, which limits the ways in which an adversary may attack us. We obtain the theorem below. More details are given in the "Closed Variant" section.

Theorem 1 *$FARFIRST$ is $\min\{f(\eta), 3\}$-competitive, where $f(\eta)$ is the following function.*

$$f(\eta) = \frac{3(1+\eta)}{2}$$

We emphasize that for $\eta = 0$, this competitive ratio remarkably matches our lower bound of 1.5, making $FARFIRST$ optimal.

Our lower bound for this setting is accomplished via an attack strategy that is analogous to a cunning magician's trick. Suppose that the magician keeps a coin inside one of their hands. They then ask a pedestrian to make a guess for which hand contains the coin. If the pedestrian succeeds, they get to keep the coin. However, the magician can always make it so that the pedestrian fails, for example by having a coin up

[1]Since this (slightly) modifies the original problem definition, the previous competitive ratio lower bound results for the classical problem do not necessarily hold for our setup even without predictions. We show in the full version of the paper that the lower bound of 1.64 still holds for the closed variant and that a tight lower bound of 2 holds for the open variant.

each of their sleeves and producing the one not chosen by the pedestrian. One can draw an analogy from this trick to our attack strategy, which is described in the "Closed Variant" section in more detail.

Theorem 2 *For any $\epsilon > 0$, no algorithm can be $(1.5 - \epsilon)$-competitive for closed online TSP on the line under the $LOCATIONS$ prediction model.*

Open Variant under *LOCATIONS*. The algorithm we present for this setting is named $NEARFIRST$. This algorithm first serves the requests on the side opposite to the one $FARFIRST$ would choose. Another divergence from $FARFIRST$ that should be noted is that for the side focused on second, $NEARFIRST$ prioritizes requests that are predicted to be *closer* to the origin, since there is no requirement to return to it, thus avoiding unnecessary backtracking. More details about the algorithm and a proof sketch of the following theorem are given in the relevant section further in the paper, in the section "Open Variant".

Theorem 3 *$NEARFIRST$ is $min\{f(\eta), 3\}$-competitive, for the following function $f(\eta)$.*

$$f(\eta) = \begin{cases} 1 + \frac{2(1+\eta)}{3-2\eta}, & for \quad \eta < \frac{2}{3} \\ 3, & for \quad \eta \geq \frac{2}{3} \end{cases}$$

As in the previous setting, we utilize the "magician's trick" in order to design a similar attack strategy. We describe exactly how this is done in the corresponding section for the open variant under LOCATIONS. This leads to the establishment of a lower bound, as stated below.

Theorem 4 *For any $\epsilon > 0$, no algorithm can be $\left(1.\overline{44} - \epsilon\right)$-competitive for open online TSP on the line under the $LOCATIONS$ prediction model.*

Open Variant under *LOCATIONS+FINAL*. Our algorithmic approach to this setting is again similar to the one implemented in $NEARFIRST$. The difference is that instead of choosing the side with the near extreme first, we choose the side whose extreme is further away from the predicted endpoint of OPT. We name this algorithm $PIVOT$, to emphasize that the prediction for the last request acts as a pivot for the algorithm to decide the first side it will serve. A theorem about $PIVOT$ is presented below, for which a proof sketch has been given in the corresponding section.

Theorem 5 *$PIVOT$ is $min\{f(\eta, \delta), 3\}$-competitive, for the function $f(\eta, \delta)$ below.*

$$f(\eta, \delta) = \begin{cases} 1 + \frac{1+2(\delta+3\eta)}{3-2(\delta+2\eta)}, & 3 - 2(\delta + 2\eta) > 0 \\ 3, & 3 - 2(\delta + 2\eta) \leq 0 \end{cases}$$

For this setting, we reuse the attack strategy initially designed for the closed variant. The only difference is that we add another request at the origin with a release time of 4. We explain how we derive the following theorem in the corresponding section.

Theorem 6 *For any $\epsilon > 0$, no algorithm can be $(1.25 - \epsilon)$-competitive for open online TSP on the line under the LF prediction model.*

Setting	L.B.	U.B.	Best known
Closed	1.64	1.64	1.64
Closed - LOC.	1.5	1.5	$min\left\{\frac{3(1+\eta)}{2}, 3\right\}$
Open	2	2	2
Open - LOC.	1.44	1.66	$min\left\{1 + \frac{2(1+\eta)}{3-2\eta}, 3\right\}$
Open - L.F.	1.25	1.33	$min\left\{1 + \frac{1+2(\delta+3\eta)}{3-2(\delta+2\eta)}, 3\right\}$

Table 1: Summary of results. L.B. and U.B. stand for lower and upper bound respectively. LOC. stands for LOCATIONS and L.F. stands for LOCATIONS+FINAL.

We briefly summarize our results in Table 1. Note that the lower and upper bound entries correspond to the no error case. We strongly emphasize that these results are for the case where the number of requests n is known. All missing proofs can be found in the full version of the paper along with an assessment of our algorithms using synthetic data.

Related Work

Online TSP. The online TSP for a general class of metric spaces has been studied in (Ausiello et al. 2001), where the authors show lower bounds of 2 for the open variant and 1.64 for the closed variant. These bounds are actually shown on the real line. Additionally, a 2.5-competitive algorithm and a 2-competitive algorithm are given for the general open and closed variants respectively. A stronger lower bound of 2.04 was shown for the open variant in (Bjelde et al. 2021), where both bounds are also matched in the real line. For the restriction of the closed online TSP to the non-negative part of the real line, (Blom et al. 2001) give a tight 1.5-competitive algorithm. By imposing a fairness restriction on the adversary, they also obtain a 1.28-competitive algorithm. In (Jaillet and Wagner 2006), the authors introduce the "online TSP with disclosure dates", where each request may also be communicated to the algorithm before it is released. The authors show improvements to the competitive ratio of various previous algorithms as a function of the difference between disclosure and release dates.

Learning-Augmented Algorithms. Learning-augmented algorithms have received significant attention since the seminal work of (Lykouris and Vassilvtiskii 2018), where they investigated the online caching problem with predictions. Based on that model, (Purohit, Svitkina, and Kumar 2018) proposed algorithms for the ski-rental problem as well as non-clairvoyant scheduling. Subsequently, (Gollapudi and Panigrahi 2019), (Wang and Li 2020), and (Angelopoulos et al. 2020) improved the initial ski-rental problem. The latter also proposed algorithms with predictions for the list update and bin packing problem and demonstrated how to show lower bounds for algorithms with predictions. Several works, including (Rohatgi 2020), (Antoniadis et al. 2020a), and (Wei 2020), improved the initial results regarding the caching problem.

The scheduling problems with machine-learned advice have been extensively studied in the literature. In (Moseley et al. 2020), the makespan minimization problem with restricted assignments was considered, while (Mitzenmacher 2020) used predicted job processing times in different scheduling scenarios. The works of (Bamas et al. 2020) and (Antoniadis, Ganje, and Shahkarami 2021) focused on the online speed scaling problem using predictions for workloads and release times/deadlines, respectively.

There is literature on classical data structures. Examples include the indexing problem, (Kraska et al. 2018), bloom filters, (Mitzenmacher 2018). Further learning-augmented approaches on online selection and matching problems (Antoniadis et al. 2020b; Dütting et al. 2021) and a more general framework of online primal-dual algorithms (Bamas, Maggiori, and Svensson 2020) also emerged, and there is a survey (Mitzenmacher and Vassilvitskii 2020).

Independent Work. Compared to the problem considered in this paper, a more general one, the online metric TSP, as well as a more restricted version in the half-line, have been studied in (Bernardini et al. 2022) under a different setting, concurrently to our work. We note that only the closed variant is considered in (Bernardini et al. 2022). Since the prediction model is different (predictions for the positions as well as release times of the requests are given) and a different error definition is used, the results are incomparable.

Preliminaries

The Problem Definition. In the online TSP on the line, an algorithm controls an agent that can move on the real line with at most unit speed. We have a set $Q = \{q_1, \ldots, q_n\}$ of n requests. We emphasize that for this problem definition, the algorithm receives the value n as input. Each request q has an associated position and release time. To simplify notation, whenever a numerical value is expected from a request q (for a calculation, finding the minimum of a set etc.) the term q will refer to the *position* of the request. Whenever we need the release time of a request, we shall use $rel(q)$. Additionally, the algorithm receives as input a set $P = \{p_1, \ldots, p_n\}$ of predictions regarding the positions of the requests. That is, each p_i attempts to approximate q_i. We assume without loss of generality that Q always contains a request q_0 at the origin with release time 0 and that P contains a perfect prediction $p_0 = 0$ for this request[2].

We use t to quantify time. To describe the position of the agent of an algorithm ALG at time $t \geq 0$, we use $pos_{ALG}(t)$. We may omit this subscript when ALG is clear from context. We can assume without loss of generality that $pos(0) = 0$. The speed limitation of the agent is given formally via $|pos(t') - pos(t)| \leq |t' - t|$, $\forall t, t' \geq 0$. A request q is considered served at time t if $\exists\, t' : \; pos(t') = q,\; rel(q) \leq t' \leq t$, i.e., the agent has moved to the request no earlier than it is released. We will say that a request q is *outstanding* at time t, if ALG has not served it by time t, even if $rel(q) > t$, i.e. q has not been released yet. Let t_{serve} denote the first point in time when all requests have been served by the agent. Also, let $|ALG|$ denote the makespan of an algorithm ALG, for either of the two variants. Then, for the open variant $|ALG| = t_{serve}$ while for the closed one $|ALG| = min\{t : pos(t) = 0,\; t \geq t_{serve}\}$. For any sensible algorithm, this is equivalent to $t_{serve} + |pos(t_{serve})|$, since the algorithm knows the number of requests and will immediately return to the origin after serving the last one. The objective is to minimize the value $|ALG|$, utilizing the predictions.

[2]This can be seen to be without loss of generality by considering a "handler" algorithm ALG_0 which adds this request/prediction pair to *any* input and copies the actions of any of our algorithms ALG for the modified input. We observe that $|OPT|$ is unchanged and $|ALG_0| = |ALG|$.

Notation. We define $L = min(Q)$ and $R = max(Q)$. Recall that Q contains a request at the origin and thus $L \leq 0$ and $R \geq 0$. We refer to each of these requests as an *extreme* request. If $|L| > |R|$, we define $Far = L, Near = R$. Otherwise, $Far = R, Near = L$. That is, Far is the request with the largest distance from the origin out of all requests. Then, $Near$ is simply the other extreme. We will also refer to the value $|q|$ as q's *amplitude*. We will say that a prediction p is (un)released/outstanding/served if the associated request q is (un)released/outstanding/served.

The *LOCATIONS* Prediction Model. We now introduce the $LOCATIONS$ prediction model. Let $q_1, ..., q_n$ be a labeling of the requests in Q. The predictions consist of the values $p_1, ..., p_n$, where each p_i attempts to predict the position of q_i.

Error Definition for the *LOCATIONS* Prediction Model. To give an intuition for the metric we will introduce, let us first describe what it means for a prediction to be bad. In any well-posed definition, the further p_i is from q_i, the worse it should be graded. However, we must also take into account the "scale" of the problem, meaning the length of the interval $[L, R]$ that must be traveled by any algorithm, including OPT. The larger this interval, the more lenient our penalty for p_i should be. Therefore, we define the error as

$$\eta[Q, P] = \frac{max_i\{|q_i - p_i|\}}{|L| + |R|}.$$

Additionally, we define $M = \eta \cdot (|L| + |R|)$.

An Important Lemma for the *LOCATIONS* Prediction Model. We now present a lemma about this prediction model that will be used widely in our proofs and that contains intuitive value.

Lemma 1 *Let $L_P = min(P), R_P = max(P)$. Then, $|L_P| \geq |R_P|$ implies $|L| \geq |R| - 2M$, and $|R_P| \geq |L_P|$ implies $|R| \geq |L| - 2M$.*

Enhanced Prediction Model for the Open Variant. Motivated by the performance of our algorithm under the $LOCATIONS$ prediction model, we enhance it with a prediction f' which attempts to guess the label f of a request on which OPT may finish. We name this new model LF (short for $LOCATIONS + FINAL$). The error η is unchanged. We also introduce a new error metric δ. Let $q_{f'}$ be the request associated with the prediction $p_{f'}$. We then choose q_f

to be a request on which OPT may finish that minimizes the distance to $q_{f'}$. We then define the new error as

$$\delta[Q, q_f, q_{f'}] = \frac{|q_{f'} - q_f|}{|L| + |R|}.$$

Similarly to before, we define $\Delta = \delta \cdot (|L| + |R|)$.

Justification of the $LOCATIONS + FINAL$ Model. This prediction model might in first glance seem arbitrary, but we argue that it is a natural one and that a machine learning model for it can be trained in practice. As we have previously mentioned, the offline version of the TSP on the line admits a quadratic time solution. Therefore, the optimal solutions of past instances can easily be calculated in retrospect. From these solutions we can extract the final request served by OPT. We can then use this in our training data along with any other information deemed relevant.

Closed Variant

In this section, we consider the closed variant under the $LOCATIONS$ prediction model. We provide the $FARFIRST$ algorithm, which obtains a competitive ratio of 1.5 with perfect predictions and is also smooth and robust. Additionally, we give an attack strategy that implies a lower bound of 1.5 for the competitive ratio of any algorithm in this setting, making $FARFIRST$ optimal. The formal proofs are deferred to the full version of the paper.

The $FARFIRST$ Algorithm. Before giving the algorithm, we define the $FARFIRST$ ordering on the predictions of an input. For simplicity, we assume that the furthest prediction from the origin is positive. Let $r_1, \ldots, r_a$ be the positive predictions in descending order of amplitude and $l_1, \ldots, l_b$ be the negative predictions ordered in the same way. The $FARFIRST$ ordering is $r_1, \ldots, r_a, l_1, \ldots, l_b$. Any predictions on the origin are placed in the end. Ties are broken via an arbitrary label ordering.

We present the algorithm through an update function used whenever a request is released. This update function returns the plan of moves to be executed until the next release of a request. Note that $ext(side, set)$ returns the extreme element of the input set in the side specified, where $side = true$ means the right side. Also, the $\oplus$ symbol is used to join moves one after another. When all the moves are executed, the agent waits for the next release. This only happens when waiting on a prediction.

In order to give some further intuition on $FARFIRST$, we first give the definition of a *phase*.

Definition 1 *A phase of an algorithm ALG is a time interval $[t_s, t_e]$ such that $pos_{ALG}(t_s) = 0$, $pos_{ALG}(t_e) = 0$ and $pos_{ALG}(t') \neq 0$, $\forall\, t' \in (t_s, t_e)$. That is, ALG starts and ends a phase at the origin and does not cross the origin at any other time during the phase.*

In the following, when we refer to the *far* side, we mean the side with the furthest *prediction* from the origin. The *near* side is the one opposite to that. We see that $FARFIRST$ works in at most three phases. The first phase ends when all predictions on the far side have been released and the agent has managed to return to the origin with no released and outstanding request on the far side. During this phase, any request on the far side is served as long as $FARFIRST$ does not move closer to the origin than the far side's extreme unreleased prediction. Note that some *surprise* requests may appear, i.e., far side requests that were predicted to lie on the near side. These requests are also served in this phase. The second phase lasts while at least one prediction is unreleased. During this phase, the agent serves any request released on the near side, using the predictions as guidance, similarly to the first phase. Requests released on the far side are ignored during this phase. Note that no surprises can occur here, since all far side predictions were released during the first phase. A third phase may exist if some requests were released on the far side during the second phase. These requests' amplitudes are bounded by M, since they were predicted to be positioned on the near side. This simple algorithm is consistent, smooth and robust, as implied by the following theorem.

Algorithm 1: $FARFIRST$ update function.

Input : Current position pos, set O of unserved released requests, first unreleased prediction p in $FARFIRST$ ordering or 0 if none exist, the side $farSide$ with the furthest prediction from the origin.
Output: A series of (unit speed) moves to carry out until the next request is released.

$posSide \leftarrow (pos > 0)$;
$pSide \leftarrow (p > 0)$;
if $pos = 0$ **then** $posSide \leftarrow farSide$;
if $p = 0$ **then** $pSide \leftarrow \overline{posSide}$;
return $move(ext(posSide, O \cup \{pos\})) \oplus move(ext(pSide, O \cup \{p\})) \oplus move(p)$;

Theorem 1 *$FARFIRST$ is $\min\{f(\eta), 3\}$-competitive, where $f(\eta)$ is the following function.*

$$f(\eta) = \frac{3(1+\eta)}{2}$$

We now give a proof sketch that covers the intuition behind our formal proof. The 3-robustness is seen using an absolute worst case scenario in which $FARFIRST$ is $|OPT|$ units away from the origin at time $|OPT|$ (due to the unit speed limitation), and all the requests to serve are on the opposite side. For the consistency and smoothness, we note that $|OPT| \geq 2(|Near| + |Far|)$. It is therefore sufficient to prove that $|FARFIRST| - |OPT| \leq |Near| + |Far| + 3\eta \cdot (|Near| + |Far|) = |Near| + |Far| + 3M$.

We refer to the left hand side as the *delay* of $FARFIRST$. We now see why this bound holds intuitively. We first describe a worst case scenario. In this scenario, OPT first serves the near side completely, and then does the same for the far side, without stopping. Let t_e denote the end time of the first phase. We see that $t_e \leq |OPT| + M$, because $FARFIRST$ follows the fastest possible route serving the requests on the far side, except for a possible delay of M attributable to a misleading prediction. Note that

in this worst case, all requests on the near side must have been released by t_e. Therefore, $FARFIRST$ accumulates an extra delay of at most 2 times the maximum amplitude of these requests. By Lemma 1, this value is at most $|Near|+|Far|+2M$. There are also other possibilities than this worst case, but they also can incur a delay of at most $|Near|+|Far|+3M$, because $|OPT|$ and $|FARFIRST|$ both increase when such cases occur.

A 1.5-Attack. We now describe an attack strategy that imposes a lower bound of 1.5 on the competitive ratio of any algorithm ALG. For the sake of exposition, we assume that there is a request on every real number in the interval $[-1, 1]$. This is approximated by a limiting process in the formal proof. These requests are released in two phases. The first phase lasts while $pos_{ALG}(t)$ has not exited the interval $[L_U(t), R_U(t)]$, where $L_U(t), R_U(t)$ are the leftmost and rightmost unreleased requests respectively at time t. During this phase, any request with distance d from the origin is released at time $2 - d$. Note that OPT could start serving requests in either side immediately and without stopping during the first phase. This phase ends when $pos_{ALG}(t)$ first exits the aforementioned interval. Assuming without loss of generality that the interval is exited from the left side (which corresponds to choosing the left hand in the magician analogy given in a previous section), the unreleased requests on the left side have their release time delayed to $4 - d$ while the requests on the right side are released as in the first phase (which corresponds to the magician producing the coin in the right hand). Note that OPT can finish by $t = 4$ by moving to 1, then to -1 and then back to the origin with full speed. At the start of the second phase, ALG can either wait for the delayed requests on the side it chose or travel some extra distance to first serve the other side. It turns out that $|ALG|$ can be shown to be arbitrarily close to 6 via the limiting process we mentioned, yielding the theorem below.

Theorem 2 *For any $\epsilon > 0$, no algorithm can be $(1.5 - \epsilon)$-competitive for closed online TSP on the line under the $LOCATIONS$ prediction model.*

Open Variant

In this section, we consider the open variant. We have two prediction models for this variant. The first one is the $LOCATIONS$ prediction model and the second is the enhanced $LOCATIONS + FINAL$ model (LF in short). For both settings, we give algorithms and lower bounds.

The *LOCATIONS* Prediction Model

Under the $LOCATIONS$ prediction model, we design the $NEARFIRST$ algorithm, which achieves a competitive ratio of $1.\overline{66}$ with perfect predictions and is also smooth and robust. We complement this result with a lower bound of $1.\overline{44}$ using a similar attack strategy to the one used for the closed variant.

The $NEARFIRST$ Algorithm. As we mentioned in the introduction, $NEARFIRST$ is similar to $FARFIRST$ and actually slightly simpler. In essence, $NEARFIRST$ simply picks a direction in which it will serve the requests. Then, it just serves the requests either from left to right or from right to left, using the predictions as guidance. The pseudocode for $NEARFIRST$ is given below. Recall that $move(x) \oplus move(y)$ is used to indicate a move to x followed by a move to y. We present the following theorem regarding

Algorithm 2: $NEARFIRST$ update function.

Input : Current position pos, set O of unserved released requests, set P of predictions.
Output: A series of (unit speed) moves to carry out until the next request is released.

$P' \leftarrow$ the unreleased predictions in P;
if P' *is empty* **then**
  if $pos < \frac{max(O)+min(O)}{2}$ **then return**
    $move(min(O)) \oplus move(max(O))$;
  else return $move(max(O)) \oplus move(min(O))$;
end
if $|min(P)| < |max(P)|$ **then return**
  $move(min(P' \cup O)) \oplus move(min(P'))$;
else return
  $move(max(P' \cup O)) \oplus move(max(P'))$;

the competitive ratio of $NEARFIRST$.

Theorem 3 *$NEARFIRST$ is $\min\{f(\eta), 3\}$-competitive, for the following function $f(\eta)$.*

$$f(\eta) = \begin{cases} 1 + \frac{2(1+\eta)}{3-2\eta}, & for \quad \eta < \frac{2}{3} \\ 3, & for \quad \eta \geq \frac{2}{3} \end{cases}$$

We now give an intuitive proof sketch for this theorem. As in the case of $FARFIRST$, the 3-robustness holds because at time $|OPT|$, $NEARFIRST$ has "leftover work" of at most $2|OPT|$ time units (to return to the origin and then copy OPT). For the consistency/smoothness, we draw our attention to the request q_f served last by OPT. For the following, we assume that $NEARFIRST$ serves the requests left to right. Let $d = |q_f - R|$. We will show that the delay of $NEARFIRST$ is bounded by $M + d$. Let t_{q_f} be the time when $NEARFIRST$ has served all requests to the left of q_f, including q_f. It turns out that $t_{q_f} \leq |OPT| + M$, because $NEARFIRST$ serves this subset of requests as fast as possible, except for a possible delay of M due to a misleading prediction. Then, in this worst case, $NEARFIRST$ accumulates an extra delay of at most d, proving our claim.

Finally, we bound OPT from below as a function of d. We see that OPT can either serve the requests L, R, q_f in the order L, R, q_f or in the order R, L, q_f. The worst case is the latter, where we see that $|OPT| \geq 2|R| + |L| + (|L| + |R| - d) = 3|R| + 2|L| - d$. Since $d \leq |L| + |R|$, we obtain

$$\frac{|NEARFIRST|}{|OPT|} = 1 + \frac{|NEARFIRST| - |OPT|}{|OPT|} \leq 1 + \frac{M + |L| + |R|}{2|R| + |L|}.$$

Because $NEARFIRST$ considers L the near extreme due to the predictions, by Lemma 1 we find that $|R| \geq \frac{1-2\eta}{2}(|L| + |R|)$, which in turn proves our bound.

A $1.\overline{44}$-Attack. The logic of our attack is the same as that used for the attack described in the section for the closed variant. There are two technical differences. The first phase here ends when $pos_{ALG}(t)$ first exits the interval $[3L_U(t)+2, 3R_U(t)-2]$, where $L_U(t), R_U(t)$ are the leftmost and rightmost unreleased request respectively at time t. The other difference lies in the release times of the second phase. We again delay the release times of the requests in the side chosen by ALG, i.e. the side from which the interval was exited. But now, each request with distance d from the origin has its release time delayed to $2+d$ instead of $4-d$. Note that OPT can finish by $t=3$ by first going to the side not chosen by ALG. However, $|ALG|$ can be seen to be arbitrarily close to $4+\frac{1}{3}$, yielding the theorem below.

Theorem 4 *For any $\epsilon > 0$, no algorithm can be $\left(1.\overline{44}-\epsilon\right)$-competitive for open online TSP on the line under the $LOCATIONS$ prediction model.*

The *LOCATIONS+FINAL* Prediction Model

In our final setting we consider the open variant under the LF prediction model. We give the $PIVOT$ algorithm, which is $1.\overline{33}$-competitive with perfect predictions and is also smooth and robust. We also reuse the attack strategy described for the closed variant to achieve a lower bound of 1.25.

The $PIVOT$ Algorithm. The final algorithm we present works in the same way as $NEARFIRST$, except for the order in which it focuses on the two sides of the origin. Instead of heading to the near extreme first, $PIVOT$ prioritizes the side whose extreme is further away from the predicted endpoint of OPT, which is provided by the LF prediction model. The pseudocode for $PIVOT$ is given below. Note that $P_{f'}$ refers to the element in P with label f'.

Algorithm 3: $PIVOT$ update function.

Input : Current position pos, set O of unserved released requests, set P of predictions, label f' of OPT's predicted endpoint.
Output: A series of (unit speed) moves to carry out until the next request is released.

$P' \leftarrow$ the unreleased predictions in P;
if P' *is empty* **then**
  if $pos < \frac{max(O)+min(O)}{2}$ **then return** $move(min(O)) \oplus move(max(O))$;
  else return $move(max(O)) \oplus move(min(O))$;
end
if $P_{f'} > \frac{max(P)+min(P)}{2}$ **then return** $move(min(P' \cup O)) \oplus move(min(P'))$;
else return $move(max(P' \cup O)) \oplus move(max(P'))$;

As for the previous algorithms, we show a theorem that pertains to $PIVOT$'s competitive ratio for different values of the η and δ errors.

Theorem 5 *$PIVOT$ is $min\{f(\eta,\delta), 3\}$-competitive, for the function $f(\eta,\delta)$ below.*

$$f(\eta,\delta) = \begin{cases} 1+\frac{1+2(\delta+3\eta)}{3-2(\delta+2\eta)}, & 3-2(\delta+2\eta) > 0 \\ 3, & 3-2(\delta+2\eta) \leq 0 \end{cases}$$

We provide a proof sketch of this theorem. The proof is very similar to the one used for $NEARFIRST$'s competitive ratio. In fact, the robustness is shown in exactly the same way. For the consistency/smoothness, the delay is bounded by $M+d$ in the same way, where d is the distance of the last request q_f served by OPT to the extreme served second by $PIVOT$. The same lower bounds for $|OPT|$ hold as well. We additionally bound d as a function of the error-dependent values Δ and M. When there is no error, we can bound d to be at most $\frac{|L|+|R|}{2}$ instead of $|L|+|R|$, which gives a better competitive ratio than that of $NEARFIRST$. An important distinction is that we do not make use of Lemma 1, since the algorithm does not consider the values $|L|$ and $|R|$.

A 1.25-Attack. The attack strategy we employ in the current setting is almost the same as the one used for the closed variant. The only difference is that a special request is placed at the origin with a release time of 4. This request is also the last request served by OPT (and thus the optimal solution of the open variant also works for the closed variant) and ALG is informed of this by the LF prediction model. The idea of the proof is that if ALG were to finish before $t=5$, then another algorithm ALG' could solve the closed variant of this input in less than 6 time units, contradicting our first lower bound of 1.5 for the closed variant. This attack strategy implies the following theorem.

Theorem 6 *For any $\epsilon > 0$, no algorithm can be $(1.25-\epsilon)$-competitive for open online TSP on the line under the LF prediction model.*

Conclusion

We have examined the online TSP on the line and provided lower bounds as well as algorithms for three different learning-augmented settings. An immediate extension of our results would be to bridge the gap between the lower and upper bounds we have shown for the open variant. Also, it would be interesting to establish error-dependent lower bounds and/or optimal consistency-robustness tradeoffs. Moreover, an improvement would be to remove the assumption of knowing the number of requests n. A technique that could perhaps allow an algorithm to achieve that is to periodically make sure that the algorithm terminates in case no new requests appear. Another interesting direction is for more general versions of online TSP to be investigated, like the case of trees. Finally, we believe that the combination of learning-augmented techniques along with randomization would lead to much better results, and therefore suggest this direction as future work.

References

Angelopoulos, S.; Dürr, C.; Jin, S.; Kamali, S.; and Renault, M. P. 2020. Online Computation with Untrusted Advice. In *ITCS*.

Antoniadis, A.; Coester, C.; Eliás, M.; Polak, A.; and Simon, B. 2020a. Online metric algorithms with untrusted predictions. In *ICML*.

Antoniadis, A.; Ganje, P. J.; and Shahkarami, G. 2021. A Novel Prediction Setup for Online Speed-Scaling. *CoRR*.

Antoniadis, A.; Gouleakis, T.; Kleer, P.; and Kolev, P. 2020b. Secretary and Online Matching Problems with Machine Learned Advice. In *NeurIPS*.

Ascheuer, N.; Grötschel, M.; Krumke, S. O.; and Rambau, J. 1999. Combinatorial Online Optimization. In *Operations Research Proceedings*.

Ausiello, G.; Feuerstein, E.; Leonardi, S.; Stougie, L.; and Talamo, M. 2001. Algorithms for the On-Line Travelling Salesman. *Algorithmica*.

Bamas, É.; Maggiori, A.; Rohwedder, L.; and Svensson, O. 2020. Learning Augmented Energy Minimization via Speed Scaling. In *NeurIPS*.

Bamas, É.; Maggiori, A.; and Svensson, O. 2020. The Primal-Dual method for Learning Augmented Algorithms. In *NeurIPS*.

Bernardini, G.; Lindermayr, A.; Marchetti-Spaccamela, A.; Megow, N.; Stougie, L.; and Sweering, M. 2022. A Universal Error Measure for Input Predictions Applied to Online Graph Problems. *CoRR*.

Bjelde, A.; Hackfeld, J.; Disser, Y.; Hansknecht, C.; Lipmann, M.; Meißner, J.; SchlÖter, M.; Schewior, K.; and Stougie, L. 2021. Tight Bounds for Online TSP on the Line. *ACM Trans. Algorithms*.

Blom, M.; Krumke, S. O.; De Paepe, W. E.; and Stougie, L. 2001. The Online TSP Against Fair Adversaries. *INFORMS journal on computing*.

Chen, P.-C.; Demaine, E. D.; Liao, C.-S.; and Wei, H.-T. 2019. Waiting is not easy but worth it: the online TSP on the line revisited. *CoRR*.

Dütting, P.; Lattanzi, S.; Leme, R. P.; and Vassilvitskii, S. 2021. Secretaries with Advice. In *EC*.

Gollapudi, S.; and Panigrahi, D. 2019. Online Algorithms for Rent-Or-Buy with Expert Advice. In *ICML*.

Jaillet, P.; and Wagner, M. R. 2006. Online Routing Problems: Value of Advanced Information as Improved Competitive Ratios. *Transportation Science*.

Kraska, T.; Beutel, A.; Chi, E. H.; Dean, J.; and Polyzotis, N. 2018. The Case for Learned Index Structures. In *SIGMOD*.

Lawler, E. 1985. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons.

Lykouris, T.; and Vassilvtiskii, S. 2018. Competitive caching with machine learned advice. In *ICML*.

Mitzenmacher, M. 2018. A Model for Learned Bloom Filters and Optimizing by Sandwiching. In *NeurIPS*.

Mitzenmacher, M. 2020. Scheduling with Predictions and the Price of Misprediction. In *ITCS*.

Mitzenmacher, M.; and Vassilvitskii, S. 2020. Algorithms with Predictions. In *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press.

Moseley, B.; Vassilvitskii, S.; Lattanzi, S.; and Lavastida, T. 2020. Online Scheduling via Learned Weights. In *SODA*.

Psaraftis, H. N.; Solomon, M. M.; Magnanti, T. L.; and Kim, T.-U. 1990. Routing and Scheduling on a Shoreline with Release Times. *Management Science*.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving Online Algorithms via ML Predictions. In *NeurIPS*.

Rohatgi, D. 2020. Near-optimal bounds for online caching with machine learned advice. In *SODA*.

Wang, S.; and Li, J. 2020. Online Algorithms for Multi-shop Ski Rental with Machine Learned Predictions. In *AAMAS*.

Wei, A. 2020. Better and Simpler Learning-Augmented Online Caching. In *APPROX/RANDOM*.