Online Minimum Spanning Trees with Weight Predictions*

Magnus Berg[0000−0001−8637−7113], Joan Boyar[0000−0002−0725−8341], Lene M. Favrholdt[0000−0003−3054−2997], and Kim S. Larsen[0000−0003−0560−3794]

University of Southern Denmark
{magbp,joan,lenem,kslarsen}@imada.sdu.dk
https://imada.sdu.dk/u/{magbp,joan,lenem,kslarsen}

Abstract. We consider the minimum spanning tree problem with predictions, using the weight-arrival model, i.e., the graph is given, together with predictions for the weights of all edges. Then the actual weights arrive one at a time and an irrevocable decision must be made regarding whether or not the edge should be included into the spanning tree. In order to assess the quality of our algorithms, we define an appropriate error measure and analyze the performance of the algorithms as a function of the error. We prove that, according to competitive analysis, the simplest algorithm, Follow-the-Predictions, is optimal. However, intuitively, one should be able to do better, and we present a greedy variant of Follow-the-Predictions. In analyzing that algorithm, we believe we present the first random order analysis of a non-trivial online algorithm with predictions, by which we obtain an algorithmic separation. This may be useful for distinguishing between algorithms for other problems when Follow-the-Predictions is optimal according to competitive analysis.

Keywords: Online Algorithms · Predictions · Random Order Analysis · Minimum Spanning Tree.

1 Introduction

The *Minimum Spanning Tree* (MST) problem is one of the classical graph algorithms problems, where one must select edges from a weighted graph such that these constitute a spanning tree of minimal weight. We consider an online version of this problem in the relatively new context of predictions, a direction that emerged following the successes of machine learning that has provided more accessible and reliable predictions.

In the area of online algorithms, we consider problems, many of which have offline counterparts, where input is presented to an algorithm in a piece-wise

* Supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B and in part by the Innovation Fund Denmark, grant 9142-00001B, Digital Research Centre Denmark, project P40: Online Algorithms with Predictions.

fashion (often referred to as *requests*), and irrevocable decisions must be made when each item is presented. The quality of an online algorithm is often assessed using competitive analysis, which essentially focuses on the worst-case ratio of the cost of the online algorithm to the cost of an optimal, offline algorithm, Opt.

When considering graph problems, various models, inspired by different application scenarios, exist. In the vertex-arrival model, the requests are the vertices of the graph, arriving together with the subset of its incident edges that connect to vertices that have already arrived. In the edge-arrival model, requests are the edges, identified by their two endpoints. For weighted graphs, there is also the *weight-arrival model*, where the graph is known, and the weights arrive online. In the vertex-arrival and edge-arrival models, there is only one possible online algorithm, the one that accepts every edge that does not create a cycle, since otherwise the algorithm's output might not span the entire graph. Even in the weight-arrival model, no deterministic algorithm for online MST can be competitive [15]. This makes the problem hard, but interesting in the context of advice or predictions.

Partially in an attempt to measure how much information about the future is needed for various online problems, online algorithms with advice were introduced [11, 9, 7, 4]. In the model used most often, it is an information-theoretical game of how few oracle-produced bits in total are needed to obtain a particular competitive ratio or optimality. Obviously, the connection here is that oracle-based advice can be considered infallible predictions. The MST problem has been considered by Bianchi et al. in this model [3]. They obtain results for various arrival models and restricted graph classes, including the weight-arrival model, but with only two different weights allowed.

The seminal paper by Lykouris and Vassilvitskii [16], introducing machine-learned advice, which is now more often referred to as predictions, has inspired rapidly growing [1] efforts in the area [17]. In this context, ideally we want algorithms to use the predictions and perform optimally when predictions are correct (referred to as *consistency*), perform as well as a good online algorithm when predictions are all wrong (*robustness*), and degrade gracefully from one to the other as the predictions become increasingly erroneous (*smoothness*). The ideal situation described above can of course often not be reached, so one proves upper and lower bounds, as is customary in the field. Discussing smoothness requires a definition of error. This is problem-dependent and requires some thought. We want to distinguish between good and bad algorithms, and defining error measures that exaggerate or underestimate the importance of errors leads to unreliable results.

For the online MST problem with predictions, there are natural error measures. We arrive at an error measure, defined as the sum of differences between the predicted and actual values of the $n-1$ edges (the number of edges in a spanning tree) with the largest discrepancies; a measure with desirable properties.

We focus on the MST problem with predictions in the weight-arrival model. Our first somewhat surprising result is that with this error measure (or any of some reasonable alternatives), competitive analysis [18, 13] cannot distinguish between different, correct algorithms. This means that the most naïve algorithm, Follow-the-Predictions (FtP), is optimal under this measure, with a competitive ratio of $1 + 2\varepsilon$, where ε is the error, normalized by the value of Opt. Of course, this also means that the perhaps more reasonable algorithm, we call Greedy Follow-the-Predictions (GFtP), that switches to another edge when a revealed actual weight matches or does better than the predicted weight of an edge it could replace, is indistinguishable from FtP under competitive analysis.

In online algorithms, there are other performance measures one can turn to when competitive analysis is insufficient, as discussed in [8, 6, 5]. One of the most well accepted is Random Order Analysis [14], also called the Random Order Model; a chapter in [10] discusses some results. Note that the problem from [10] of finding a maximum forest is not very similar to our problem, since the forest is not required to be spanning. The random order analysis technique reduces the power of the adversary, compared to competitive analysis. In competitive analysis, the adversary chooses the requests and the order in which they a presented, while in random order analysis, the adversary chooses the requests, but those requests are presented to the algorithm uniformly at random. Using random order analysis, we establish a separation between FtP and GFtP. We believe this is the first time random order analysis has been applied in the context of predictions.

Omitted details and proofs can be found in the full paper [2].

2 Preliminaries

Given an online algorithm Alg for an online minimization problem Π, and an instance I of Π, we let Alg$[I]$ denote Alg's solution on instance I, and Alg(I) denote the cost of Alg$[I]$. Then, the *competitive ratio* of Alg is

$$\text{CR}_{\text{ALG}} = \inf\{c \mid \exists b \colon \forall I \colon \ \text{ALG}(I) \leqslant c\,\text{OPT}(I) + b\}.$$

When online algorithms have access to a predictor, a further parameter is introduced into the problem, namely the accuracy of that predictor. Throughout this paper, we let η be the error measure that computes the quality of the predictions, and we let $\varepsilon = \frac{\eta}{\text{OPT}}$ be the normalized error measure. Our error measure is defined later (see Definition 1).

Given an online algorithm with predictions, Alg, we express the competitive ratio of Alg as a function of ε, and evaluate it based on the three criteria: *consistency*, *robustness*, and *smoothness*. Following [16], we define consistency as Alg's competitive ratio, when the prediction error is 0. Alg is *α-consistent* if there exists a constant, α, such that $\text{CR}_{\text{ALG}}(0) = \alpha$.

As ε grows, the competitive ratio of Alg will decay as a function of ε. For a function, β, we say that Alg is *β-smooth*, if $\text{CR}_{\text{ALG}}(\varepsilon) \leqslant \beta(\varepsilon)$, for all $\varepsilon \geqslant 0$.

An algorithm, $\textsc{Alg}$, is said to be *γ-robust*, if there exists some constant γ such that $\textsc{cr}_{\textsc{Alg}}(\varepsilon) \leqslant \gamma$, for all $\varepsilon \geqslant 0$. Since no online algorithm for the WMST problem can be competitive, the robustness of any deterministic online algorithm with predictions cannot be worse than the competitive ratio of any online algorithm.

2.1 Random Order Analysis

Given an online algorithm, $\textsc{Alg}$, for a problem, Π, and an instance of Π with request sequence $I = \langle i_1, i_2, \dots, i_n \rangle$, a permutation σ of I is chosen uniformly at random, and $\sigma(I)$ is presented to $\textsc{Alg}$. The *random order ratio* of $\textsc{Alg}$ is defined as

$$\textsc{ror}_{\textsc{Alg}} = \inf\{c \mid \exists b \colon \forall I \colon \mathbb{E}_\sigma[\textsc{Alg}(\sigma(I))] \leqslant c\,\textsc{Opt}(I) + b\},$$

As with the competitive ratio, we express the random order ratio of algorithms with predictions as a function of ε.

2.2 Weight-Arrival MST Problem

The offline MST problem is a thoroughly studied problem, for which efficient optimal algorithms are known, for example Kruskal's and Prim's Algorithms. Given a graph $G = (V, E, w)$, the task is to find a spanning tree T for G that minimizes the objective function $c(T) = \sum_{e \in E(T)} w(e)$. For the MST problem in the weight-arrival model (WMST), online algorithms are initially provided with the underlying graph $G = (V, E)$, and then the weights of the edges in G arrive online. At the time the true weight of an edge e arrives, the online algorithm has to irrevocably accept or reject e for its final tree. We focus on the WMST problem where we assume that an online algorithm has access to predicted weights for all edges in G before the online computation is initiated.

2.3 Notation and Nomenclature

We use the notation $\mathbb{R}^+$ and $\mathbb{Z}^+$ to denote the positive real numbers and the positive integers, respectively. Graphs, in the following, are weighted, simple, connected and undirected, with weights in $\mathbb{R}^+$. Given a graph G, $n = |V(G)|$ and $m = |E(G)|$. For any clarification on graph theory, we refer to [19]. Further, we define a *WMST-instance* to be a triple $(G, \hat{w}, w)$ consisting of a graph G, and two maps $\hat{w} \colon E(G) \to \mathbb{R}^+$ and $w \colon E(G) \to \mathbb{R}^+$, defining for each edge $e \in E(G)$, a predicted weight $\hat{w}(e)$ and a true weight $w(e)$. Given a graph G and a tree $T \subset G$, when writing T, we implicitly refer to $E(T)$.

Given an algorithm with predictions, $\textsc{Alg}$, for the WMST problem, and a WMST-instance $(G, \hat{w}, w)$, we let $\textsc{Alg}[\hat{w}(E(G)), w(E(G))]$ denote the tree that $\textsc{Alg}$ outputs. When G is clear from the context, we write $\textsc{Alg}[\hat{w}, w]$ and let $\textsc{Alg}(\hat{w}, w)$ denote the cost of $\textsc{Alg}[\hat{w}, w]$. We let $\textsc{Opt}[w]$ be an optimal MST of G, and $\textsc{Opt}[\hat{w}]$ be an optimal MST of G with respect to $\hat{w}$.

2.4 Error Measure

We use the following measure, denoted by η, selected due to its desirable properties and its ability to distinguish between algorithms under random order analysis. In the full paper [2], we show that intuitive alternatives have flaws.

Definition 1. *Let $(G, \hat{w}, w)$ be any WMST-instance, $e_1, e_2, \ldots, e_m$ be any ordering of $E(G)$, and $\{p_i\}_i$ be the sequence where $p_i := |w(e_i) - \hat{w}(e_i)|$. Further, let $\{p_{i_j}\}_j$ be the sequence $\{p_i\}_i$, sorted such that $p_{i_1} \geqslant p_{i_2} \geqslant \cdots \geqslant p_{i_m}$. The* error*, η, is given by $\eta(\hat{w}, w) = \sum_{j=1}^{n-1} p_{i_j}$. When $(G, \hat{w}, w)$ is clear from the context, we write η for $\eta(\hat{w}, w)$. The* normalized error *is $\varepsilon = \frac{\eta}{\textsc{Opt}}$.*

Note that $n-1$ is the number of edges in a spanning tree. Thus, the risk of unreasonably large prediction errors for dense graphs as with other possible error measures has been eliminated (see the full paper [2]). This measure also satisfies the monotonicity and Lipschitzness properties from [12].

3 Optimal Algorithms under Competitive Analysis

We prove that our two algorithms FtP and GFtP, defined in Algorithms 1 and 2, respectively, are 1-consistent and $(1 + 2\varepsilon)$-smooth algorithms and that this is best possible. First, we focus on the simplest algorithm, called Follow-the-Predictions (FtP), defined in Algorithm 1.

Algorithm 1 FtP

```
1: Input: A WMST-instance $(G, \hat{w}, w)$
2: Let $T$ be a MST of $G$ w.r.t. $\hat{w}$
3: while receiving inputs $(w(e_i), e_i)$ do
4:     if $e_i \in T$ then
5:         Accept $e_i$                          ▷ Add $e_i$ to the solution
```

Theorem 1. $\textsc{cr}_{\textsc{FtP}}(\varepsilon) \leqslant 1 + 2\varepsilon$.

We also present a non-trivial algorithm, called Greedy-FtP (GFtP) that starts by producing the tree that FtP outputs. Whenever the true weight of an edge, e, that is not contained in GFtP's current tree is revealed, the algorithm checks whether e can replace an edge in its current tree. It does so by comparing the predicted weights of a subset of edges in its current tree by the newly revealed true weight. We formalize the strategy of GFtP in Algorithm 2.

Theorem 2. $\textsc{cr}_{\textsc{GFtP}}(\varepsilon) \leqslant 1 + 2\varepsilon$.

Algorithm 2 GFtP

```
1: Input: A WMST-instance $(G, \hat{w}, w)$
2: Let $T$ be a MST of $G$ w.r.t. $\hat{w}$
3: $S = \emptyset$                                          ▷ $S$ contains the seen edges
4: while receiving inputs $(w(e_i), e_i)$ do
5:     $S = S \cup \{e_i\}$
6:     if $e_i \in T$ then
7:         Accept $e_i$                                     ▷ Add $e_i$ to the solution
8:     else                                                 ▷ $e_i \notin T$
9:         $C$ is the cycle $e_i$ introduces in $T$
10:        $C' = C \setminus S$
11:        if $C' \neq \emptyset$ then
12:            $e_{\max} = \arg\max_{e_j \in C'}\{\hat{w}(e_j)\}$
13:            if $w(e_i) \leqslant \hat{w}(e_{\max})$ then
14:                $T = (T \setminus \{e_{\max}\}) \cup \{e_i\}$        ▷ Update $T$
15:                Accept $e_i$                             ▷ Add $e_i$ to the solution
```

Theorem 3. *For any $\varepsilon \in [0, 1/2)$, any deterministic online algorithm,* Alg*, for the WMST problem with weight predictions has* $\text{CR}_{\text{Alg}}(\varepsilon) \geqslant 1 + 2\varepsilon$.

Proof. We only sketch the proof. The adversary uses the following graph, where $k = \left\lceil \frac{5}{2-r} \right\rceil$ and $\ell = k^2$.

- $V = \{v_1, v_2, \ldots, v_{2k}\} \cup \{z_j \mid 1 \leqslant j \leqslant \ell\}$,
- $E = I \cup \bigcup_{j=1}^{\ell} E_j$, where
  - $I = \bigcup_{i=1}^{2k-1}\{(v_i, v_{i+1})\}$ and
  - $E_j = \{(z_j, v_i) \mid 1 \leqslant i \leqslant 2k\}$, for all $j = 1, 2, \ldots, \ell$,
- $\hat{w}_{k,\ell}(e) = w_{k,\ell}(e) = 1$, for all $e \in I$,
- $\hat{w}_{k,\ell}((z_j, v_i)) = k + i - 1$, for all $(z_j, v_i) \in E_j$, and
- For $1 \leq j \leq \ell$, if Alg accepts an edge (z_j, v_i), $1 \leq i \leq 2k-1$, then $w_{k,\ell}((z_j, v_{i+1})) = i$. Otherwise, $w_{k,\ell}((z_j, v_{2k})) = 4k$.

To see that the result holds for any $\varepsilon < 1/2$, note that adding some real number to all predicted and true weights changes Opt, but Alg − Opt as well as η remain unchanged. □

In the full paper, we prove that, for FtP and GFtP, the lower bound of $1 + 2\varepsilon$ holds for any $\varepsilon \in \mathbb{R}^+$. Thus, we obtain:

Corollary 1. $\text{CR}_{\text{FtP}}(\varepsilon) = \text{CR}_{\text{GFtP}}(\varepsilon) = 1 + 2\varepsilon$*, and this is optimal among deterministic algorithms.*

4 Separation by Random Order Analysis

We show that GFtP has a better random order ratio than FtP, separating the two algorithms. Throughout, we set $T_{\text{FtP}} = \text{Opt}[\hat{w}]$, $T_{\text{Opt}} = \text{Opt}[w]$, and $T_{\text{GFtP}} = \text{GFtP}[\hat{w}, w]$. Further, we denote by T the tree that GFtP makes online changes to. Note that initially $T = T_{\text{FtP}}$, and after GFtP has processed the full input sequence, $T = T_{\text{GFtP}}$. Finally, we denote by S the collection of *seen* edges in $E(G)$, i.e., those edges whose true weight has been revealed.

Theorem 4. $\text{ROR}_{\text{FtP}}(\varepsilon) = 1 + 2\varepsilon$.

Proof. Since FtP does not make online changes to T_{FtP}, the competitive analysis of FtP translates directly to a random order analysis of FtP. Hence, the result follows from Corollary 1. □

We start with the following lower bound on GFtP.

Theorem 5. $\text{ROR}_{\text{GFtP}}(\varepsilon) \geqslant 1 + \varepsilon$.

We now turn to proving an upper bound of $1 + (1 + \ln(2))\varepsilon \approx 1 + 1.69\,\varepsilon$ on the random order ratio of GFtP (Theorem 6). To this end, we apply the following lemmas.

Lemma 1. *Let G be a graph, and let T_1 and T_2 be two spanning trees of G. Then, for any edge $e_1 \in T_1 \setminus T_2$, there exists an edge $e_2 \in T_2 \setminus T_1$ such that e_2 introduces a cycle into T_1 that contains e_1, and e_1 introduces a cycle into T_2 that contains e_2.*

Lemma 2. *Let $e \in T_{\text{FtP}} \setminus S$. If, at any point, an edge e' introduces a cycle in T that contains e, then $\hat{w}(e) \leqslant \hat{w}(e')$.*

Lemma 3. *For all integers $n \geqslant 2$, we have that*

$$\frac{1}{n-1} \sum_{i=0}^{n-2} \left(1 + \frac{n-1}{2n-2-i}\right) \leqslant 1 + \ln(2)\,.$$

Lemma 4. *Suppose that* GFtP *has just rejected $e' \notin T$. Then, at any future point, any unseen edge e that is contained in the cycle that e' introduces into T, at that point, satisfies that $\hat{w}(e) < w(e')$.*

Theorem 6. $\text{ROR}_{\text{GFtP}}(\varepsilon) \leqslant 1 + (1 + \ln(2))\varepsilon$.

Proof. Given a WMST-instance $(G, \hat{w}, w)$, we let $T_{\text{GFtP},\sigma}$ denote the output tree that GFtP constructs when run on $(G, \hat{w}, w)$, where the order in which the weights arrive has been permuted according to a uniformly randomly chosen permutation σ of $\{1, 2, \ldots, m\}$. Further, we denote by $\text{GFtP}(\hat{w}, w, \sigma)$ the cost of $T_{\text{GFtP},\sigma}$.

The idea towards a random order ratio upper bound for GFtP is to prove the existence of a subset $\text{E}_{\text{Blame}} \subset T_{\text{Opt}} \cup T_{\text{GFtP},\sigma}$ such that

$$\mathbb{E}_\sigma[\text{GFtP}(\hat{w}, w, \sigma)] - \text{Opt}(w) \leqslant \sum_{e \in \text{E}_{\text{Blame}}} |\hat{w}(e) - w(e)| \,.$$

and

$$\mathbb{E}_\sigma[|\text{E}_{\text{Blame}}|] \leqslant (n-1)(1 + \ln(2)) \,.$$

More specifically, we define a function $f \colon T_{\text{GFtP},\sigma} \to T_{\text{Opt}}$ and prove that f is bijective, implying that

$$\mathbb{E}_\sigma[\text{GFtP}(\hat{w}, w, \sigma)] - \text{Opt}(w) = \sum_{e \in T_{\text{GFtP},\sigma}} (w(e) - w(f(e))) \,,$$

and then, for each $e \in T_{\text{GFtP},\sigma}$, argue that $w(e) - w(f(e))$ is upper bounded by the prediction error of either e or $f(e)$, or the sum of the two. Then, we show that the expected number of edges for which the upper bound is the error of both e and $f(e)$ is upper bounded by $(n-1)\ln(2)$. We also show that the edges whose errors are used as upper bounds are all distinct.

Recall that throughout the execution of GFtP, its current tree is called T. For the remainder of this proof, we denote by T' a spanning tree of G that is initially set to T_{Opt}. We use T' to keep track of which edges in T_{Opt} have been associated with an edge in $T_{\text{GFtP},\sigma}$ under f. Any time GFtP accepts an edge e, we associate e with an edge $e' \in T'$ under f. We consider two cases:

(a) If $e \in T'$, we set $f(e) = e$ and leave T' unchanged.

(b) If $e \notin T'$, then Lemma 1 implies that there exists an edge $e' \in T' \setminus T$ such that e' introduces a cycle into T that contains e, and e introduces a cycle into T' that contains e'. We select such an edge e', set $f(e) = e'$, and replace e' by e in T'.

We repeat this process every time GFtP accepts an edge. This, however, requires T' to remain a spanning tree at all times. To see that T' remains a spanning tree, we note that in case (a), T' remains unchanged and is therefore still a spanning tree. In case (b), we replace e' with e in T'. Since e introduces a cycle into T' that contains e', it follows that T' remains acyclic after the replacement, and so T' is still a spanning tree.

Towards bijectivity of *f*: In case (a), $f(e) = e$, and so $e \in (T \cap T' \cap S)$. Since $e \notin ((T \cap T') \setminus S) \cup (T' \setminus T)$, we never map to e again later. In case (b), $f(e) = e'$, and after replacing e' by e in T', we find that $e \in (T \cap T' \cap S)$, and $e' \in \overline{T \cup T'}$. Hence, as neither e nor e' is contained in $(T \cap T' \setminus S) \cup (T' \setminus T)$, we never map to either again later. Hence f is injective, and since $|T_{\text{GFtP},\sigma}| = |T_{\text{OPT}}|$, f is bijective.

We now describe how the set E_{BLAME} is constructed. For each edge e accepted by GFtP, we do the following, based on the situation at the time when $w(e)$ has been revealed, but e has not yet been handled by GFtP.

- If $e \in T'$, no edge is added to E_{BLAME}.
- If $e \in \overline{T \cup T'}$, we add $f(e)$ to E_{BLAME}.
- If $e \in T \setminus T'$, then
  - if $w(f(e))$ arrives before $w(e)$, we add e to E_{BLAME},
  - otherwise, we add both e and $f(e)$ to E_{BLAME}.

For each edge e accepted by GFtP, let E_e be the set of edges added to E_{BLAME} because of e by the above scheme. By the following case analysis, we prove that

$$w(e) - w(f(e)) \leq \sum_{e' \in E_e} |\hat{w}(e') - w(e')| .$$

Whenever an edge, e_{next}, has its weight revealed, we consider the following cases.

Case $\boldsymbol{e_{\text{next}} \in T \cap T'}$: GFtP accepts e_{next}. By (a), $f(e_{\text{next}}) = e_{\text{next}}$, implying that $w(e_{\text{next}}) - w(f(e_{\text{next}})) = 0$.

Case $\boldsymbol{e_{\text{next}} \in \overline{T \cup T'}}$: If GFtP accepts $e_{\text{next}} \in \overline{T \cup T'}$, it does so due to e_{next} replacing some edge $e \in T$ that is contained in the cycle that e_{next} introduces into T. We let e_{OPT} denote $f(e_{\text{next}})$ and argue that

$$w(e_{\text{next}}) - w(e_{\text{OPT}}) \leqslant |\hat{w}(e_{\text{OPT}}) - w(e_{\text{OPT}})| .$$

Note that since GFtP swapped out e for e_{next}, we have that $w(e_{\text{next}}) \leqslant \hat{w}(e)$. Further, we can argue that $\hat{w}(e) \leqslant \hat{w}(e_{\text{OPT}})$. Indeed, if $e = e_{\text{OPT}}$, this is trivial. If $e \neq e_{\text{OPT}}$, then, since e_{OPT} introduces a cycle that contains e_{next}, it follows that before swapping out e for e_{next}, e_{OPT} would introduce a cycle into T containing e, and so $\hat{w}(e) \leqslant \hat{w}(e_{\text{OPT}})$, by Lemma 2. Hence,

$$\begin{aligned} w(e_{\text{next}}) - w(e_{\text{OPT}}) &\leqslant \hat{w}(e) - w(e_{\text{OPT}}) \\ &\leqslant \hat{w}(e_{\text{OPT}}) - w(e_{\text{OPT}}) \\ &\leqslant |\hat{w}(e_{\text{OPT}}) - w(e_{\text{OPT}})| . \end{aligned}$$

Case $\boldsymbol{e_{\text{next}} \in T' \setminus T}$: In this case, e_{next} introduces a cycle C in T. Denote by e an edge in $C \setminus S$ for which $e = \arg\max_{e_i \in C \setminus S}\{\hat{w}(e_i)\}$. We split the remaining analysis of this case into two subcases.

Subcase (accept): If $w(e_{\text{next}}) \leqslant \hat{w}(e)$, then GFtP accepts e_{next} and removes e from its tree. Then, by (a), $f(e_{\text{next}}) = e_{\text{next}}$ and so $w(e_{\text{next}}) - w(f(e_{\text{next}})) = 0$.

Subcase (reject): If $\hat{w}(e) < w(e_{\text{next}})$, then GFtP rejects e_{next}. Since each edge in T' will, at some point, be associated with an edge in $T_{\text{GFtP},\sigma}$, by the bijectivity of f, it follows that GFtP will later accept some edge that will be associated with e_{next} under f. Denote this edge by e_{future}, such that $f(e_{\text{future}}) = e_{\text{next}}$.

Note that GFtP can accept e_{future} either due to a swap, or because $w(e_{\text{future}})$ was revealed while contained in T. In the latter case, at the time where GFtP accepts e_{future}, we find that e_{future} is contained in the cycle that e_{next} introduces into T, and so, by Lemma 4, $\hat{w}(e_{\text{future}}) < w(e_{\text{next}})$, implying that $w(e_{\text{future}}) - w(e_{\text{next}}) < w(e_{\text{future}}) - \hat{w}(e_{\text{future}})$.

Case $e_{\text{next}} \in T \setminus T'$: In this case, GFtP accepts e_{next}. By (b), there exists an edge $e_{\text{Opt}} \in T' \setminus T$ such that $f(e_{\text{next}}) = e_{\text{Opt}}$. Since T remains unchanged when $w(e_{\text{next}})$ is revealed, it follows that e_{Opt} would introduce a cycle in T containing e_{next} before $w(e_{\text{next}})$ was revealed. Hence, by Lemma 2, we find that $\hat{w}(e_{\text{next}}) \leqslant \hat{w}(e_{\text{Opt}})$, and so

$$\begin{aligned} w(e_{\text{next}}) - w(e_{\text{Opt}}) &= w(e_{\text{next}}) - \hat{w}(e_{\text{next}}) + \hat{w}(e_{\text{next}}) - w(e_{\text{Opt}}) \\ &\leqslant w(e_{\text{next}}) - \hat{w}(e_{\text{next}}) + \hat{w}(e_{\text{Opt}}) - w(e_{\text{Opt}}) \\ &\leqslant |w(e_{\text{next}}) - \hat{w}(e_{\text{next}})| + |\hat{w}(e_{\text{Opt}}) - w(e_{\text{Opt}})| \,. \end{aligned}$$

If $w(f(e_{\text{next}}))$ is revealed before $w(e_{\text{next}})$, we obtain a stronger upper bound of $|w(e_{\text{next}}) - \hat{w}(e_{\text{next}})|$, as shown in Case $e_{\text{next}} \in T' \setminus T$ Subcase (reject).

This ends the case analysis.

Next, we show that all edges in E_{Blame} are distinct. To this end, let e be an edge that GFtP has just accepted. By construction of E_{Blame}, we either add e, $f(e)$, or both to E_{Blame}. After e has been accepted, $e \in T \cap T'$, and can therefore never be hit under f. Hence, e will not be added to E_{Blame} again later. Similarly, if $f(e) = e$, $f(e)$ will not be added to E_{Blame} again later. On the other hand, if $f(e) \neq e$, then after replacing $f(e)$ with e in T', $f(e) \in \overline{T \cup T'}$, and $f(e)$ may therefore be accepted later due to a swap. In this case, by construction of E_{Blame}, $f(f(e)) \neq f(e)$ is added to E_{Blame}, and so we do not add $f(e)$ twice.

Now, all that remains is to show that $\mathbb{E}_\sigma[|\text{E}_{\text{Blame}}|] \leqslant (n-1)(1+\ln(2))$.

For the remainder of this proof, let i be the random variable that counts the number of edges that have either been accepted by GFtP (now in $T' \cap T \cap S$), or belong to $T' \setminus T$ and have been rejected (now in $(T' \setminus T) \cap S$). Thus, $i = |T' \cap T \cap S| + |(T' \setminus T) \cap S| = |T' \cap S|$. One may observe that i remains unchanged when $e_{\text{next}} \in \overline{T \cup T'}$ and GFtP rejects e_{next}. In all other of the above cases, i is incremented.

We prove the following invariants.

Invariant (i): Any edge in $T \setminus T'$ is unseen.

Proof of (i): Initially, all edges are unseen. Whenever the weight of an edge $e \in T \setminus T'$ is revealed, we replace $f(e)$ with e in T', so now $e \in T \cap T'$. Hence, after replacing $f(e)$ with e in T', all edges in $T \setminus T'$ are again unseen.

Invariant (ii): For any $0 \leqslant i \leqslant n-2$, the probability that the next edge is contained in $T \setminus T'$, denoted p_i, satisfies

$$p_i = \frac{|(T \setminus T') \setminus S|}{|E(G) \setminus S|} \leqslant \frac{n-1}{2n-2-i},$$

Proof of (ii): We let j count the number of seen edges. At any point in time, $i \leq j$. Now, observe that

$$p_i = \frac{|(T \setminus T') \setminus S|}{|E(G) \setminus S|} \leqslant \frac{|(T \setminus T') \setminus S|}{|(T \cup T') \setminus S|}$$

From Invariant (i), it follows that $|(T \setminus T') \setminus S| = n-1-a_j-x_j$, where a_j is the number of edges that have been accepted after j edges have had their weights revealed, i.e., the number of edges in $T \cap T' \cap S$, and x_j is the number of edges in $T \cap T' \setminus S$. Then,

$$p_i \leqslant \frac{n-1-x_j-a_j}{|(T \cup T') \setminus S|}.$$

Now,

$$|(T \cup T') \setminus S| = |T \cup T'| - |(T \cup T') \cap S|.$$

For any $0 \leqslant i \leqslant n-2$,

$$\begin{aligned} & |T \cup T'| = |T| + |T'| - |T \cap T'| = 2n-2-x_j-a_j, \\ \text{and} \quad & |(T \cup T') \cap S| = |(T \setminus T') \cap S| + |T' \cap S| = |T' \cap S| = i. \end{aligned}$$

Here the second to last equality follows from Invariant (i), and the last equality follows from the definition of i. Hence,

$$p_i \leqslant \frac{n-1-x_j-a_j}{2n-2-i-x_j-a_j}.$$

Using that $a_j + x_j \geqslant 0$ and $i \leqslant n-1$, it follows that

$$p_i \leqslant \frac{n-1}{2n-2-i}.$$

The only time we add two edges to E_{BLAME} is when $e_{\text{next}} \in T \setminus T'$ and $f(e_{\text{next}}) \notin (T' \setminus T) \cap S$. For each $i = 0, 1, \ldots, n-2$, the probability that e_{next} is in $T \setminus T'$ is p_i. In any other case, we add at most one edge to E_{BLAME}.

Note that i is the size of the set of edges, e, for which either $w(e)$ has been revealed or $w(f(e))$ but not $w(e)$ has been revealed. If the weight of $f(e)$ but not that of e has been revealed, then $w(e) - w(f(e))$ is accounted for in Case $e_{\text{next}} \in T' \setminus T$, Subcase (reject), where $e_{\text{next}} = f(e)$. Therefore, when $i = n-1$, all edges in $T_{\text{GFTP},\sigma}$ have been accounted for, and so we can compute the size of E_{BLAME}. We obtain

$$\mathbb{E}_\sigma[|\text{E}_{\text{BLAME}}|] \leqslant \sum_{i=0}^{n-2}(1+p_i) \leqslant \sum_{i=0}^{n-2}\left(1+\frac{n-1}{2n-2-i}\right) \leqslant (n-1)(1+\ln(2)),$$

where the last inequality follows from Lemma 3.

Since the prediction error of any edge is only used to upper bound incurred cost once, and since the average of the $n-1$ largest prediction errors upper bound the average prediction error of any set of $(n-1)(1+\ln(2))$ edges, it follows that

$$\mathbb{E}_\sigma[\text{GFTP}(\hat{w}, w, \sigma) - \text{OPT}(w)] \leqslant (1+\ln(2))\eta,$$

so

$$\frac{\mathbb{E}_\sigma[\text{GFTP}(\hat{w}, w, \sigma)]}{\text{OPT}(w)} \leqslant 1 + (1+\ln(2))\varepsilon,$$

and, hence, $\text{ROR}_{\text{GFTP}}(\varepsilon) \leqslant 1 + (1+\ln(2))\varepsilon$. □

5 Open Problems

An obvious open problem is to determine the exact random order ratio of GFTP, in the range $1+\varepsilon$ to $1+\ln(2)\varepsilon$.

GFTP can be seen as an improvement of FTP, and we are interested in what we believe could be a further improvement: In addition to accepting some edges that are not in the chosen minimum spanning tree based on predictions, also reject *some* that *are* in that tree, if the actual weight is higher than the predicted. The obvious approach gives an algorithm with a worse competitive ratio than FTP's, but restricting which edges the algorithm can accept after such a rejection gives rise to another optimal algorithm under competitive analysis. It would be interesting to apply random order analysis to such an algorithm as well.

More generically, it would be interesting to apply random order analysis to other online problems with predictions, as well as to consider error measures similar to ours for other problems.

References

1. Algorithms with Predictions. `https://algorithms-with-predictions.github.io/`, accessed: 2023-02-14
2. Berg, M., Boyar, J., Favrholdt, L.M., Larsen, K.S.: Online minimum spanning trees with weight predictions (2023), arXiv:2302.12029
3. Bianchi, M.P., Böckenhauer, H., Brülisauer, T., Komm, D., Palano, B.: Online minimum spanning tree with advice. International Journal of Foundations of Computer Science **29**(4), 505–527 (2018)
4. Boyar, J., Favrholdt, L.M., Kudahl, C., Larsen, K.S., Mikkelsen, J.W.: Online Algorithms with Advice: A Survey. ACM Computing Surveys **50**(2), 1–34 (2017), article No. 19
5. Boyar, J., Favrholdt, L.M., Larsen, K.S.: Relative Worst-Order Analysis: A Survey. ACM Computing Surveys **54**(1), 1–21 (2020), article No. 8
6. Boyar, J., Irani, S., Larsen, K.S.: A Comparison of Performance Measures for Online Algorithms. Algorithmica **72**(4), 969–994 (2015)
7. Dobrev, S., Královič, R., Pardubská, D.: Measuring the problem-relevant information in input. RAIRO - Theoretical Informatics and Applications **43**(3), 585–613 (2009)
8. Dorrigiv, R., López-Ortiz, A.: A survey of performance measures for on-line algorithms. SIGACT News **36**(3), 67–81 (2005)
9. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theoretical Computer Science **412**(24), 2642–2656 (2011)
10. Gupta, A., Singla, S.: Random-order models. In: Roughgarden, T. (ed.) Beyond the Worst-Case Analysis of Algorithms, pp. 234–258. Columbia University, New York (2020)
11. Hromkovič, J., Královič, R., Královič, R.: Information complexity of online problems. In: 35th International Symposium on Mathematical Foundations of Computer Science (MFCS). LNCS, vol. 6281, pp. 24–36. Springer (2010)
12. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Non-clairvoyant scheduling with predictions. In: 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). p. 285–294. ACM (2021)
13. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. Algorithmica **3**, 77–119 (1988)
14. Kenyon, C.: Best-fit bin-packing with random order. In: 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 359–364. SIAM (1996)
15. Komm, D.: An Introduction to Online Computation: Determinism, Randomization, Advice. Springer (2016)
16. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. Journal of the ACM **68**(4) (2021)
17. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with predictions. Communications of the ACM **65**(7), 33–35 (2022)
18. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**(2), 202–208 (1985)
19. West, D.B.: Introduction to Graph Theory. Featured Titles for Graph Theory, Prentice Hall, 2nd edn. (2001)