Online Minimum Spanning Trees with Weight Predictions

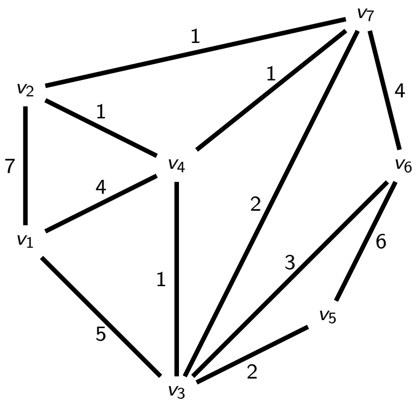
Magnus Berg, Joan Boyar, Lene M. Favrholdt and Kim S. Larsen

WADS, Concordia University

July 31, 2023

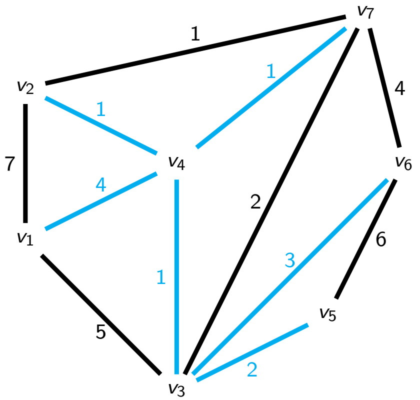
The Minimum Spanning Tree Problem

An instance: weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}^+$.
Objective: Construct spanning tree of minimum cost.



The Minimum Spanning Tree Problem

An instance: weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}^+$.
Objective: Construct spanning tree of minimum cost.

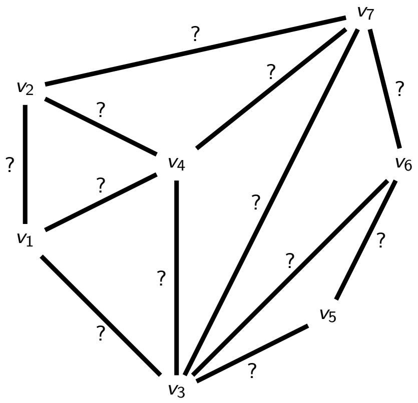


In this case, we have that $\text{OPT}(G) = 12$.

Online MST Problem - Weight Arrival Model (WMST)

An instance:

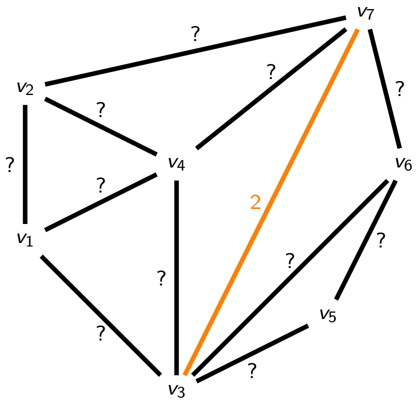
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

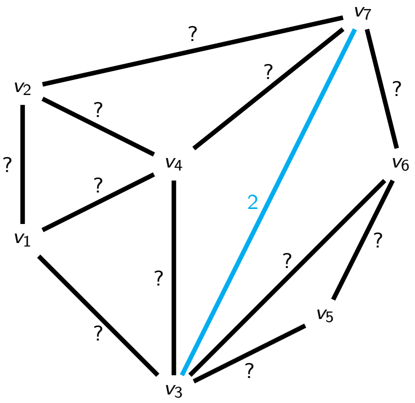
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

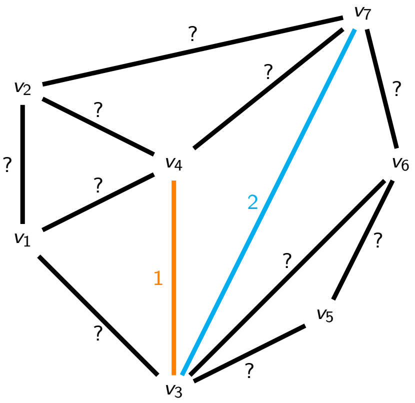
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

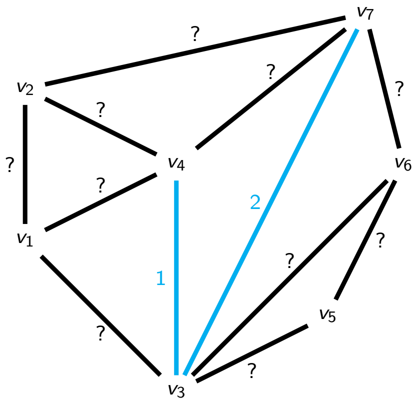
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

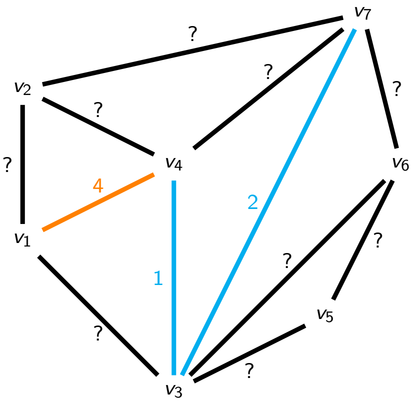
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

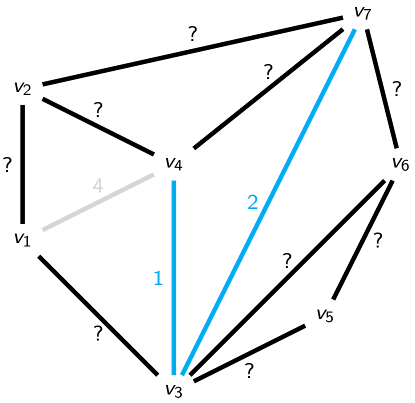
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

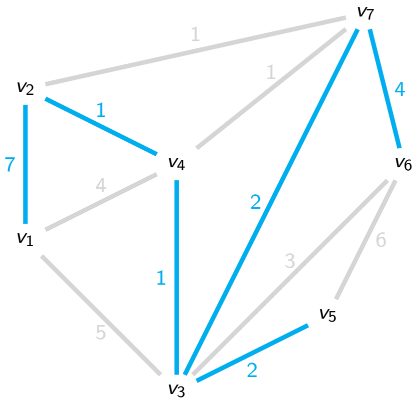
- $G_u = (V, E)$
- Sequence of $(w(e), e)$



Online MST Problem - Weight Arrival Model (WMST)

An instance:

- $G_u = (V, E)$
- Sequence of $(w(e), e)$



In this case $\text{ALG}(G) = 17$, whereas $\text{OPT}(G) = 12$.

Comparing Online Algorithms

Given an online algorithm, ALG, for an online minimization problem, we typically measure the quality of ALG by its *competitive ratio*:

Comparing Online Algorithms

Given an online algorithm, $\textsc{Alg}$, for an online minimization problem, we typically measure the quality of $\textsc{Alg}$ by its *competitive ratio*:

Definition

An online algorithm, $\textsc{Alg}$, for a minimization problem Π is said to be *c-competitive* if there exists a constant b such that for all instances I of Π:

$$\textsc{Alg}(I) \leqslant c \cdot \textsc{Opt}(I) + b.$$

The *competitive ratio* of $\textsc{Alg}$ is then

$$\textsc{cr}_{\textsc{Alg}} = \inf\{c \mid \textsc{Alg} \text{ is } c\text{-competitive}\}.$$

$\textsc{Alg}$ is *competitive* if there exists a c so that $\textsc{Alg}$ is c-competitive.

Comparing Online Algorithms

Given an online algorithm, $\textsc{Alg}$, for an online minimization problem, we typically measure the quality of $\textsc{Alg}$ by its *competitive ratio*:

Definition

An online algorithm, $\textsc{Alg}$, for a minimization problem Π is said to be *c-competitive* if there exists a constant b such that for all instances I of Π:

$$\textsc{Alg}(I) \leqslant c \cdot \textsc{Opt}(I) + b.$$

The *competitive ratio* of $\textsc{Alg}$ is then

$$\textsc{cr}_{\textsc{Alg}} = \inf\{c \mid \textsc{Alg} \text{ is } c\text{-competitive}\}.$$

$\textsc{Alg}$ is *competitive* if there exists a c so that $\textsc{Alg}$ is c-competitive.

For the WMST problem, no online algorithm is competitive.

Online Algorithms with Predictions

- Competitive analysis: Optimize for worst case.
- Machine Learning: Optimize for common cases.
- Question is: can we combine the best of both worlds?

In recent years, a lot of work has been done on *Online Algorithms with Predictions*.

The idea is to assume that the online algorithms can access some predictions providing (unreliable) information about the instance.

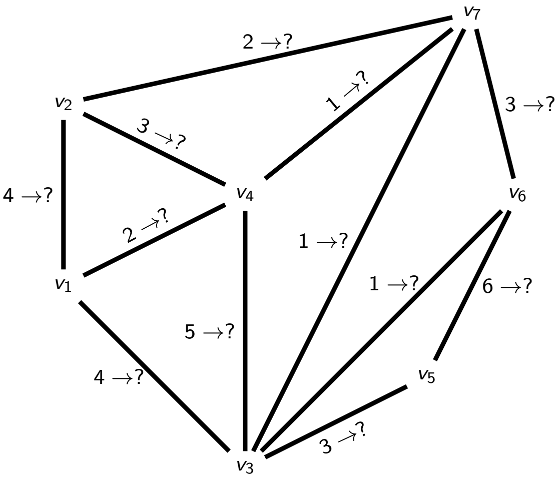
Formally, these predictions need not be Machine Learned.

Our playground

An instance:

$G = (V, E, \hat{w})$, where $\hat{w}\colon E \to \mathbb{R}^+$ is a prediction for the edge weights.

Sequence of $(w(e), e)$.



Details of this framework

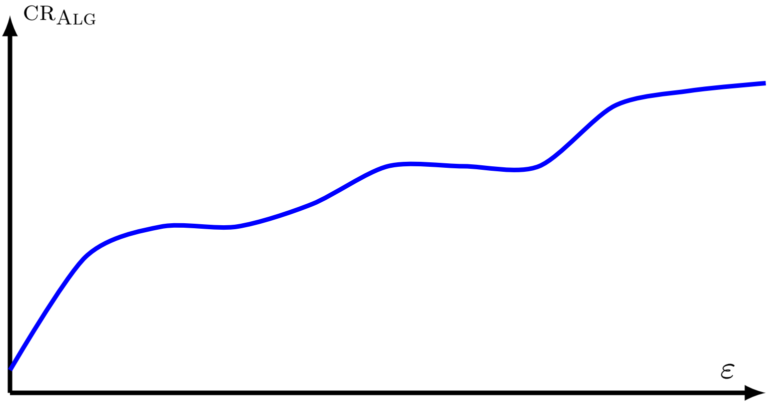
- Error η: sum of the $n-1$ greatest prediction errors, where $n = |V(G)|$.
- $\varepsilon = \frac{\eta}{\textsc{Opt}}$.

Details of this framework

- Error η: sum of the $n-1$ greatest prediction errors, where $n = |V(G)|$.
- $\varepsilon = \frac{\eta}{\textsc{Opt}}$.

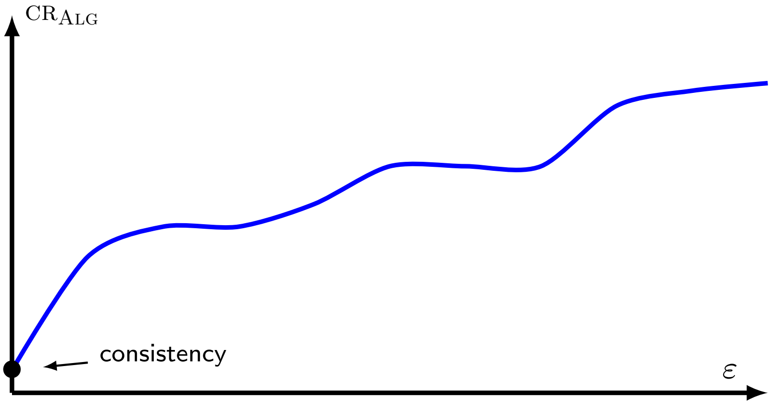
Details of this framework

- Error η: sum of the $n-1$ greatest prediction errors, where $n = |V(G)|$.
- $\varepsilon = \frac{\eta}{\textsc{Opt}}$.



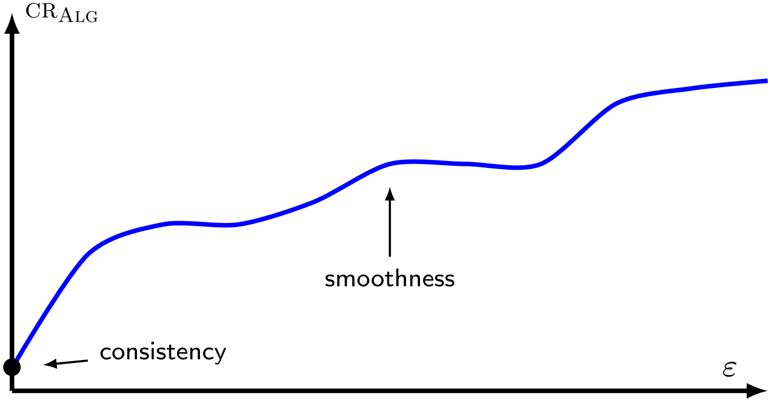
Details of this framework

- Error η: sum of the $n - 1$ greatest prediction errors, where $n = |V(G)|$.
- $\varepsilon = \frac{\eta}{\textsc{Opt}}$.



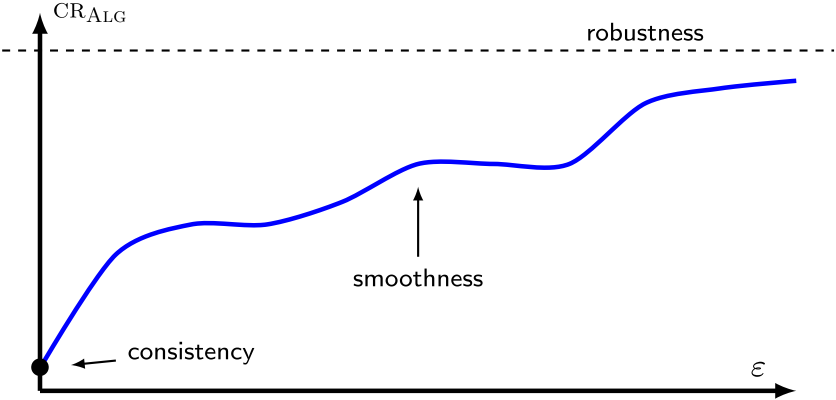
Details of this framework

- Error η: sum of the $n-1$ greatest prediction errors, where $n = |V(G)|$.
- $\varepsilon = \frac{\eta}{\textsc{Opt}}$.



Details of this framework

- Error η: sum of the $n-1$ greatest prediction errors, where $n = |V(G)|$.
- $\varepsilon = \frac{\eta}{\textsc{Opt}}$.



Follow-the-predictions

Algorithm 1 FTP

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

Greedy-FTP

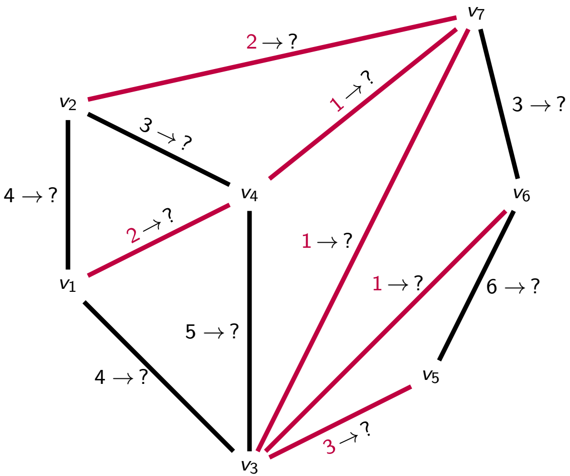
Algorithm 2 GFTP

Input: A WMST-instance $(G, \hat{w})$
Let T_{G} be a MST of G w.r.t. $\hat{w}$
$U = E(G)$ ▷ U contains the *unseen* edges
while receiving inputs $(w(e_i), e_i)$ **do**
  $U = U \setminus \{e_i\}$
  if $e_i \in T_{\mathrm{G}}$ **then**
    Accept e_i ▷ Add e_i to the solution
  else if $e_i \notin T_{\mathrm{G}}$ **then**
    C is the cycle e_i introduces in T_{G}
    $C_U = U \cap C$
    if $C_U \neq \emptyset$ **then**
      $e_{\max} = \arg\max_{e_j \in C_U} \{\hat{w}(e_j)\}$
      if $w(e_i) \leqslant \hat{w}(e_{\max})$ **then**
        $T_{\mathrm{G}} = (T_{\mathrm{G}} \setminus \{e_{\max}\}) \cup \{e_i\}$ ▷ Update T_{G}
        Accept e_i ▷ Add e_i to the solution

Running GFTP on our example graph.

Color codes:

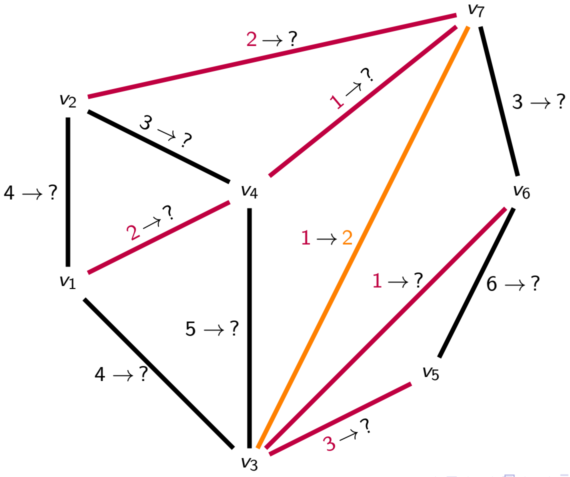
- *T*
- Just revealed
- Accepted by GFTP
- Rejected by GFTP



Running GFtP on our example graph.

Color codes:

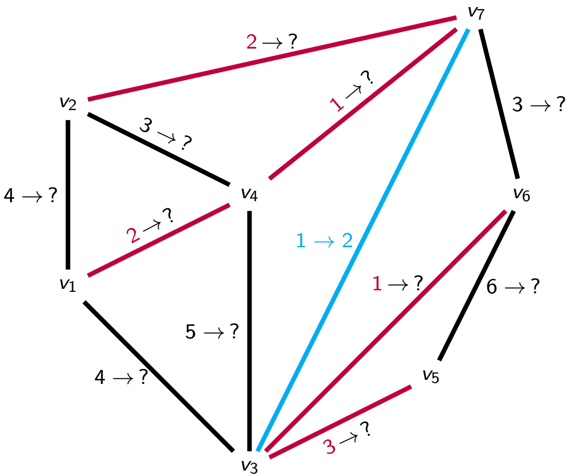
- *T*
- Just revealed
- Accepted by GFtP
- Rejected by GFtP



Running GFtP on our example graph.

Color codes:

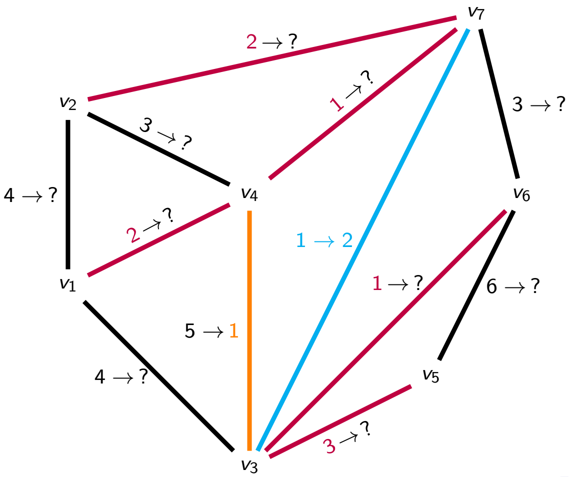
- *T*
- Just revealed
- Accepted by GFtP
- Rejected by GFtP



Running GFTP on our example graph.

Color codes:

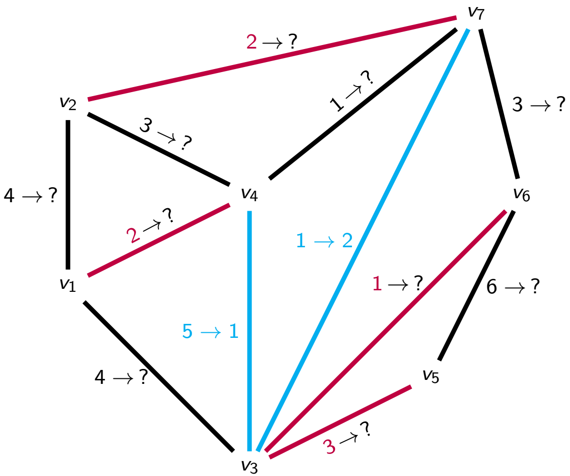
- *T*
- Just revealed
- Accepted by GFTP
- Rejected by GFTP



Running GFtP on our example graph.

Color codes:

- T
- Just revealed
- Accepted by GFtP
- Rejected by GFtP



Cannot distinguish using competitive analysis

Theorem

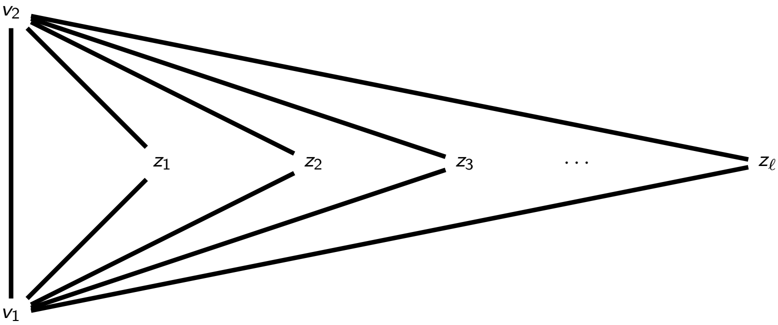
$\mathrm{CR}_{\mathrm{FtP}}(\varepsilon) = 1 + 2\varepsilon$

Theorem

$\mathrm{CR}_{\mathrm{GFtP}}(\varepsilon) = 1 + 2\varepsilon$

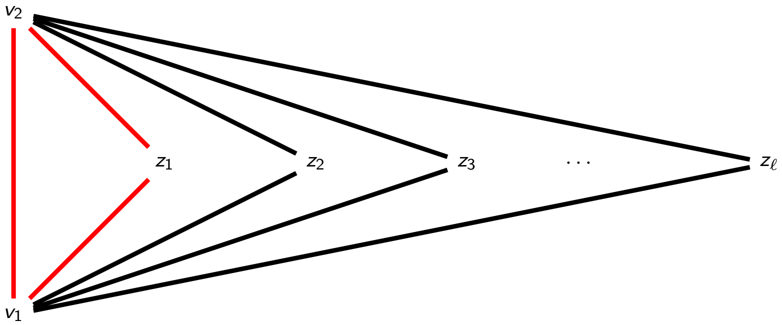
Intuition

Basic graph:

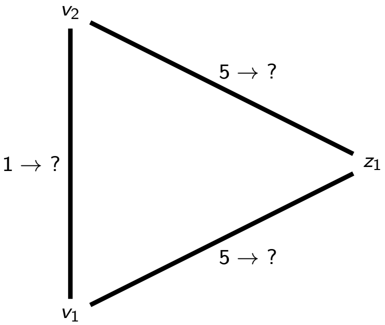


Intuition

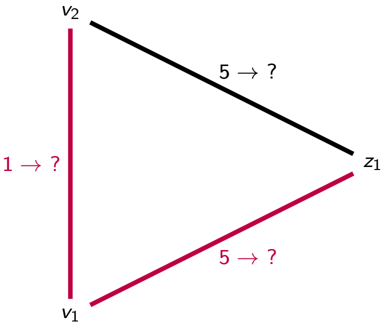
Basic graph:



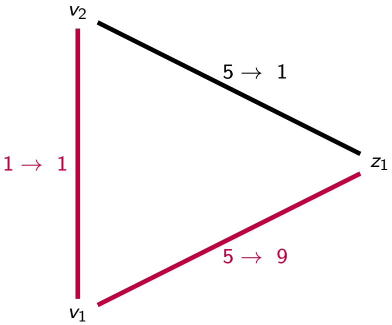
Focussing on a single triangle



Focussing on a single triangle



Focussing on a single triangle



$\text{FTP}(G) = 9 \cdot \ell + 1$

$\text{OPT}(G) = \ell + 1$

$\eta = 4 \cdot (\ell + 1)$, and so $\varepsilon = 4$.

No algorithm can do better

Theorem

For all $\varepsilon < \frac{1}{2}$, and all online algorithm with predictions $\textsc{Alg}$, $\textsc{cr}_{\textsc{Alg}}(\varepsilon) \geqslant 1 + 2\varepsilon$.

Random Order Analysis

A weakening of the adversary:

Definition

Let ALG be an online algorithm for a minimization problem Π, and let $I = \langle r_1, r_2, \ldots, r_m \rangle$ be an instance of Π. Then, a permutation σ of $\{1, 2, \ldots, m\}$ is chosen uniformly at random, and $\sigma(I)$ is presented to ALG. The random order ratio of ALG is

$$\text{ROR}_{\text{ALG}} = \inf\{c \mid \exists b : \forall I : \mathbb{E}_\sigma[\text{ALG}(\sigma(I))] \leqslant c\text{OPT}(\sigma(I)) + b\}$$

As competitive ratio, we describe the random order ratio of ALG as a function of ε.

This is the first time random order analysis has been used in online algorithms with predictions.

Separation by Random Order Analysis

This analysis separates FTP and GFTP:

Theorem

$\text{ROR}_{\text{FTP}}(\varepsilon) = 1 + 2\varepsilon$.

Theorem

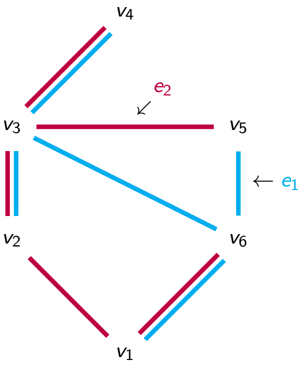
$\text{ROR}_{\text{GFTP}}(\varepsilon) \leqslant 1 + (1 + \ln(2))\varepsilon$.

The idea behind this separation...

Random Order Analysis

Lemma

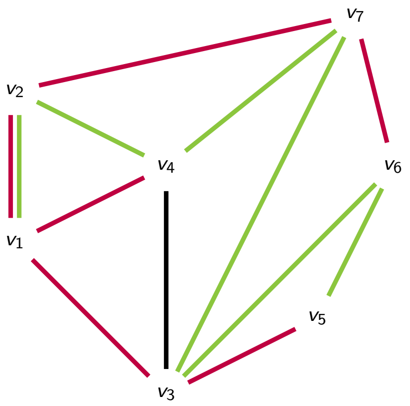
Let G be a graph, and let T_1 and T_2 be two spanning trees of G. Then, for any edge $e_1 \in T_1 \setminus T_2$, there exists an edge $e_2 \in T_2 \setminus T_1$ such that e_2 introduces a cycle into T_1 that contains e_1, and e_1 introduces a cycle into T_2 that contains e_2.



How we use this Lemma

Dominate the “random variable” $\textsc{GFtP}(G, \hat{w}) - \textsc{Opt}(G)$ online.

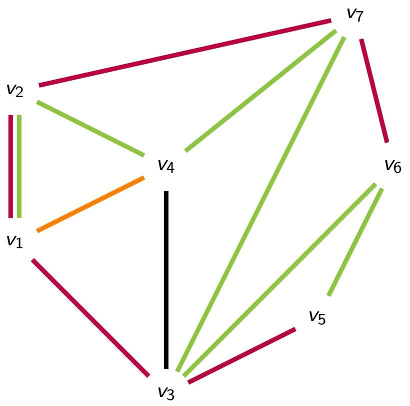
Orange - edge whose weight has most recently been revealed



How we use this Lemma

Dominate the "random variable" $\text{GFTP}(G, \hat{w}) - \text{OPT}(G)$ online.

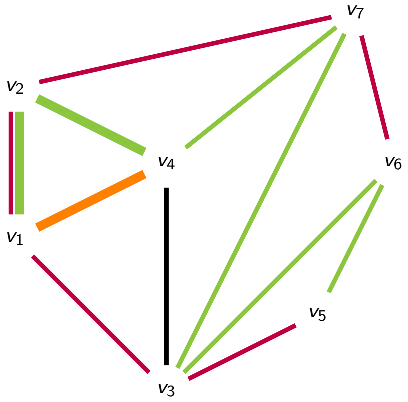
Orange - edge whose weight has most recently been revealed



How we use this Lemma

Dominate the "random variable" GFtP$(G, \hat{w}) -$ Opt(G) online.

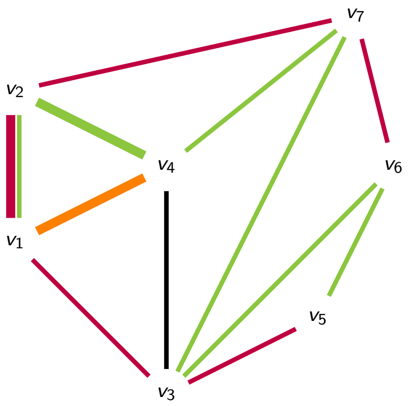
Orange - edge whose weight has most recently been revealed



How we use this Lemma

Dominate the "random variable" GFTP$(G, \hat{w}) - $ OPT(G) online.

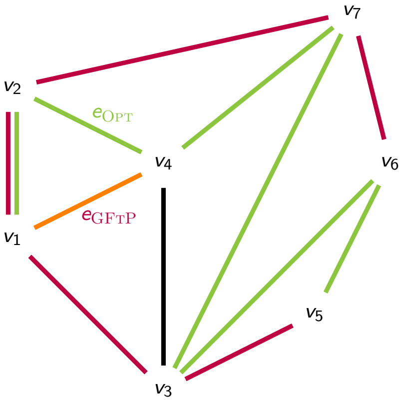
Orange - edge whose weight has most recently been revealed



How we use this Lemma

Dominate the "random variable" $\text{GFTP}(G, \hat{w}) - \text{OPT}(G)$ online.

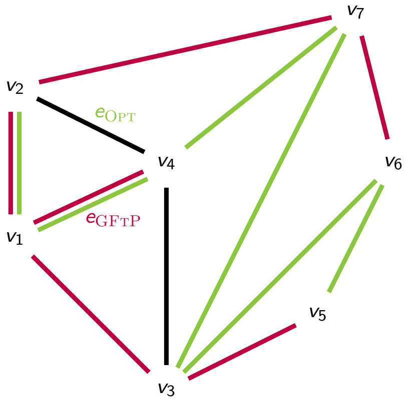
Orange - edge whose weight has most recently been revealed



How we use this Lemma

Dominate the "random variable" $\textsc{GFtP}(G, \hat{w}) - \textsc{Opt}(G)$ online.

Orange - edge whose weight has most recently been revealed

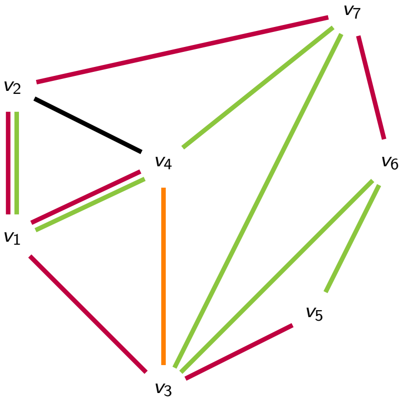


- $w(e_{\textsc{GFtP}}) - w(e_{\textsc{Opt}}) \leqslant |\hat{w}(e_{\textsc{GFtP}}) - w(e_{\textsc{GFtP}})| + |\hat{w}(e_{\textsc{Opt}}) - w(e_{\textsc{Opt}})|$

How we use this Lemma

Dominate the “random variable” $\mathrm{GFTP}(G, \hat{w}) - \mathrm{OPT}(G)$ online.

Orange - edge whose weight has most recently been revealed

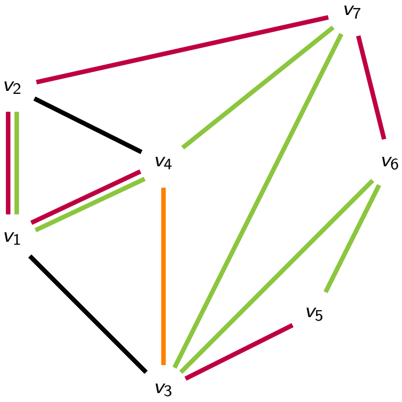


- $w(e_{\mathrm{GFTP}}) - w(e_{\mathrm{OPT}}) \leqslant |\hat{w}(e_{\mathrm{GFTP}}) - w(e_{\mathrm{GFTP}})| + |\hat{w}(e_{\mathrm{OPT}}) - w(e_{\mathrm{OPT}})|$

How we use this Lemma

Dominate the "random variable" $\text{GFTP}(G, \hat{w}) - \text{OPT}(G)$ online.

Orange - edge whose weight has most recently been revealed

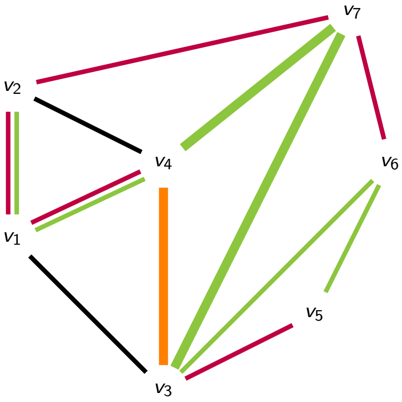


- $w(e_{\text{GFTP}}) - w(e_{\text{OPT}}) \leqslant |\hat{w}(e_{\text{GFTP}}) - w(e_{\text{GFTP}})| + |\hat{w}(e_{\text{OPT}}) - w(e_{\text{OPT}})|$

How we use this Lemma

Dominate the "random variable" $\text{GFTP}(G, \hat{w}) - \text{OPT}(G)$ online.

Orange - edge whose weight has most recently been revealed

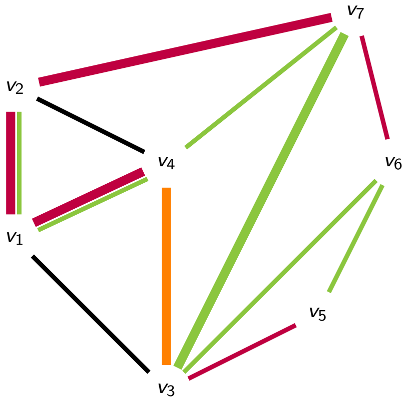


- $w(e_{\text{GFTP}}) - w(e_{\text{OPT}}) \leqslant |\hat{w}(e_{\text{GFTP}}) - w(e_{\text{GFTP}})| + |\hat{w}(e_{\text{OPT}}) - w(e_{\text{OPT}})|$

How we use this Lemma

Dominate the "random variable" $\text{GFtP}(G, \hat{w}) - \text{Opt}(G)$ online.

Orange - edge whose weight has most recently been revealed

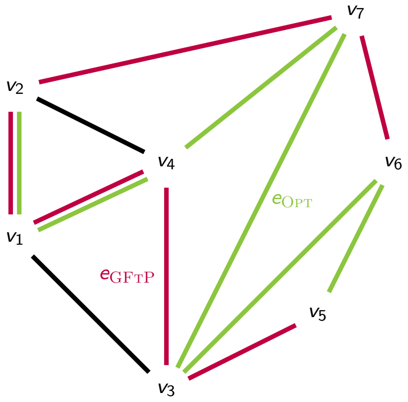


- $w(e_{\text{GFtP}}) - w(e_{\text{Opt}}) \leqslant |\hat{w}(e_{\text{GFtP}}) - w(e_{\text{GFtP}})| + |\hat{w}(e_{\text{Opt}}) - w(e_{\text{Opt}})|$

How we use this Lemma

Dominate the "random variable" $\textsc{GFtP}(G, \hat{w}) - \textsc{Opt}(G)$ online.

Orange - edge whose weight has most recently been revealed

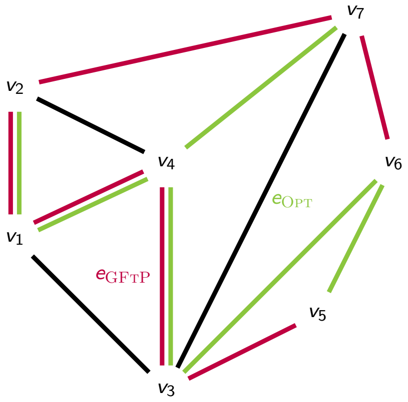


- $w(e_{\textsc{GFtP}}) - w(e_{\textsc{Opt}}) \leqslant |\hat{w}(e_{\textsc{GFtP}}) - w(e_{\textsc{GFtP}})| + |\hat{w}(e_{\textsc{Opt}}) - w(e_{\textsc{Opt}})|$

How we use this Lemma

Dominate the "random variable" $\text{GFtP}(G, \hat{w}) - \text{Opt}(G)$ online.

Orange - edge whose weight has most recently been revealed

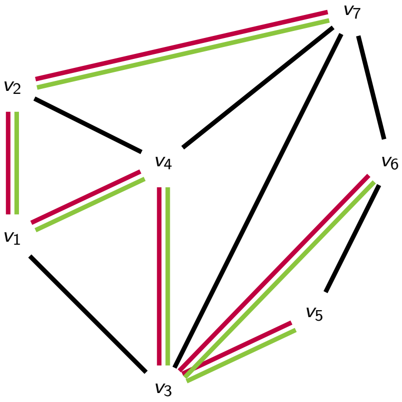


- $w(e_{\text{GFtP}}) - w(e_{\text{Opt}}) \leqslant |\hat{w}(e_{\text{GFtP}}) - w(e_{\text{GFtP}})| + |\hat{w}(e_{\text{Opt}}) - w(e_{\text{Opt}})|$

How we use this Lemma

Dominate the "random variable" $\text{GFtP}(G, \hat{w}) - \text{Opt}(G)$ online.

Orange - edge whose weight has most recently been revealed



- $w(e_{\text{GFtP}}) - w(e_{\text{Opt}}) \leqslant |\hat{w}(e_{\text{GFtP}}) - w(e_{\text{GFtP}})| + |\hat{w}(e_{\text{Opt}}) - w(e_{\text{Opt}})|$
- $\mathbb{E}[\#\text{dominating edges}] \leqslant (1 + \ln(2))(n-1)$, and all distinct.

Summary:

- FTP is best in the eyes of competitive analysis.
- We obtain a separation between FTP and GFTP using random order analysis.

Summary:

- FTP is best in the eyes of competitive analysis.
- We obtain a separation between FTP and GFTP using random order analysis.

Open problems:

- Tight random order ratio for GFTP. Somewhere in $[1+\varepsilon, 1+(1+\ln(2))\varepsilon]$.
- Apply random order analysis to other online problems with predictions.
- Create a less asymmetric algorithm - an algorithm which does not always accept edges in the expected tree.

Summary:

- FTP is best in the eyes of competitive analysis.
- We obtain a separation between FTP and GFTP using random order analysis.

Open problems:

- Tight random order ratio for GFTP. Somewhere in $[1+\varepsilon, 1+(1+\ln(2))\varepsilon]$.
- Apply random order analysis to other online problems with predictions.
- Create a less asymmetric algorithm - an algorithm which does not always accept edges in the expected tree.

Thank you for your attention!