Online Interval Scheduling with Predictions

Joan Boyar[1], **Lene M. Favrholdt**[1], **Shahin Kamali**[2] **and Kim S. Larsen**[1]

July 31, 2023

1. University of Southern Denmark, Odense, Denmark
2. York University, Toronto, Canada

The Algorithms and Data Structures Symposium (WADS)

Outline

- **Interval Scheduling** and **Disjoint Path Allocation** problems.

Outline

- **Interval Scheduling** and **Disjoint Path Allocation** problems.
- Online Algorithms with Predictions

Outline

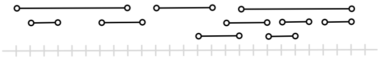
- **Interval Scheduling** and **Disjoint Path Allocation** problems.
- Online Algorithms with Predictions
- Main results:
  - Disjoint Path Allocation problem
  - Interval Scheduling: competitive results, consistency/robustness tradeoffs, and experimental results

Outline

- **Interval Scheduling** and **Disjoint Path Allocation** problems.
- Online Algorithms with Predictions
- Main results:
  - Disjoint Path Allocation problem
  - Interval Scheduling: competitive results, consistency/robustness tradeoffs, and experimental results
- Implied results

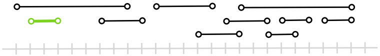
Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



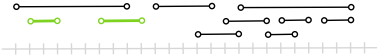
Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



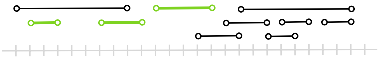
Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



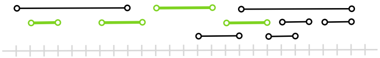
Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



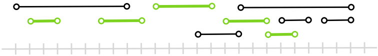
Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



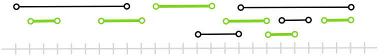
Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



Interval Scheduling Problem

- Given a set of intervals, select a subset of non-overlapping intervals with maximum cardinality.
- In the offline setting, a simple greedy algorithm solves the problem optimally.



Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.

Online Interval Scheduling

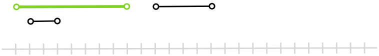
- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.

Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.

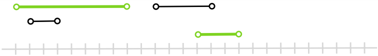
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



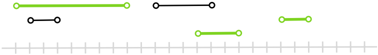
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



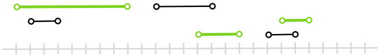
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



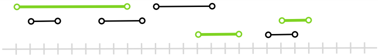
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



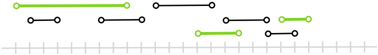
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



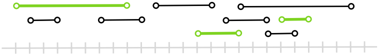
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



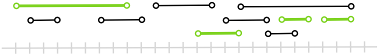
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



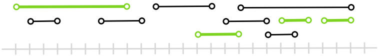
Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.

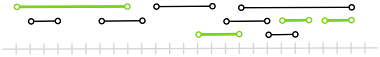


- An algorithm has a **competitive ratio** of r iff

$$\forall I : \textsc{Alg}(I) \geq r\textsc{Opt}(I) - o(\textsc{Opt}(I))$$

Online Interval Scheduling

- In the online setting, intervals appear one by one in any order, and an irrevocable decision must be made to accept to reject each item before the next ones appear.



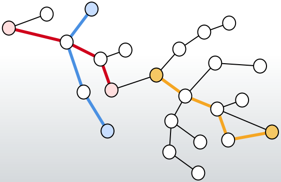
- An algorithm has a **competitive ratio** of r iff

$$\forall I : \textsc{Alg}(I) \geq r\textsc{Opt}(I) - o(\textsc{Opt}(I))$$

- The best competitive ratio is $\Theta(1/m)$ and $\Theta(1/\log m)$ for deterministic and online algorithms, respectively, where m is the maximum interval length [Awerbuch et al., SODA'94, Lipton & Tomkins, SODA'94].

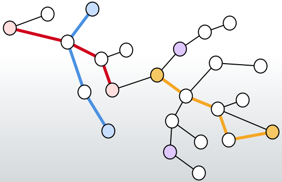
Disjoing Path Allocation Problem

- Instead of intervals, the input is formed by pairs of vertices in a given graph.
  - The goal is to accept a maximum number of pairs with disjoint paths between them.



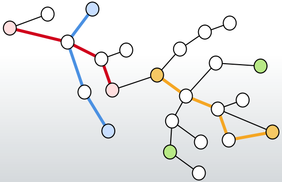
Disjoing Path Allocation Problem

- Instead of intervals, the input is formed by pairs of vertices in a given graph.
  - The goal is to accept a maximum number of pairs with disjoint paths between them.



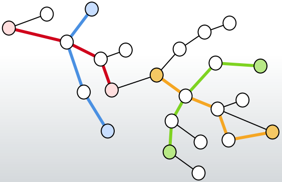
Disjoing Path Allocation Problem

- Instead of intervals, the input is formed by pairs of vertices in a given graph.
  - The goal is to accept a maximum number of pairs with disjoint paths between them.



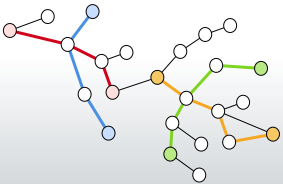
Disjoing Path Allocation Problem

- Instead of intervals, the input is formed by pairs of vertices in a given graph.
  - The goal is to accept a maximum number of pairs with disjoint paths between them.



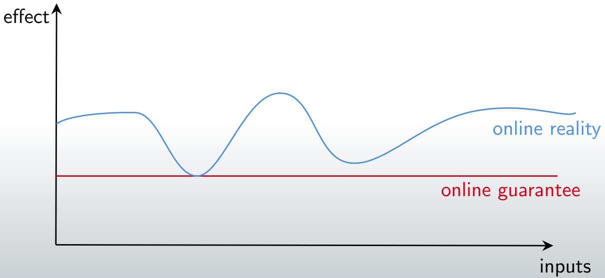
Disjoing Path Allocation Problem

- Instead of intervals, the input is formed by pairs of vertices in a given graph.
  - The goal is to accept a maximum number of pairs with disjoint paths between them.
  - The problem is NP-hard for general graphs (even SP-graphs) [Even and Etai, 1976] and polynomial-time solvable for trees and outerplanar graphs [Garg, Vazirani, and Yannakakis, 1977], Wagner, 1995].



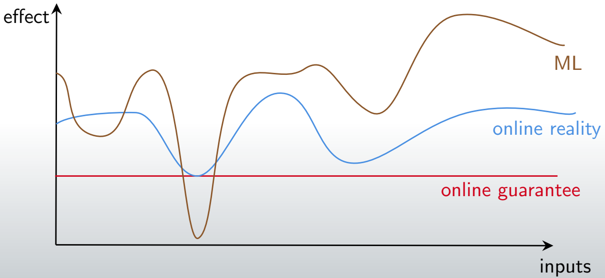
Online Algorithms with Prediction

- **Online ALgorithms:** give worst-case guarantees but do not provide any insight into typical (average-case) performance.



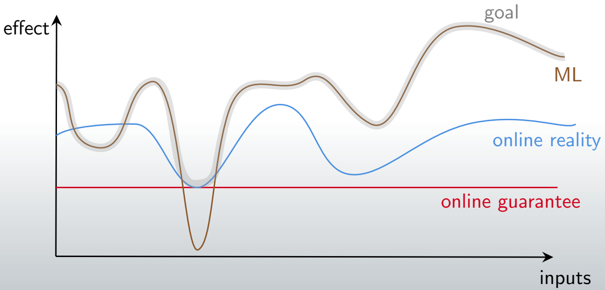
Online Algorithms with Prediction

- **Online ALgorithms:** give worst-case guarantees but do not provide any insight into typical (average-case) performance.
- **Machine Learning:** works well on typical inputs but can go terribly wrong on unusual (worst-case) inputs.



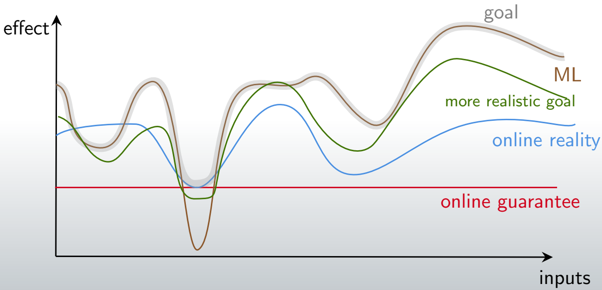
Online Algorithms with Prediction

- **Online ALgorithms:** give worst-case guarantees but do not provide any insight into typical (average-case) performance.
- **Machine Learning:** works well on typical inputs but can go terribly wrong on unusual (worst-case) inputs.
- **Online Algorithms with Prediction:** get the best of two worlds via potentially erroneous **prediction** about the input.



Online Algorithms with Prediction

- **Online ALgorithms:** give worst-case guarantees but do not provide any insight into typical (average-case) performance.
- **Machine Learning:** works well on typical inputs but can go terribly wrong on unusual (worst-case) inputs.
- **Online Algorithms with Prediction:** get the best of two worlds via potentially erroneous **prediction** about the input.

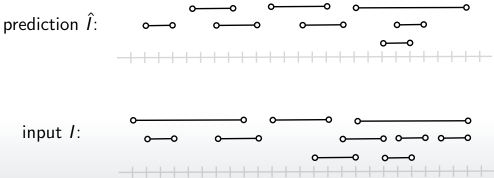


Online Algorithms with Prediction

- What prediction should be?
- How to measure error?
- Algorithm Design and Analysis

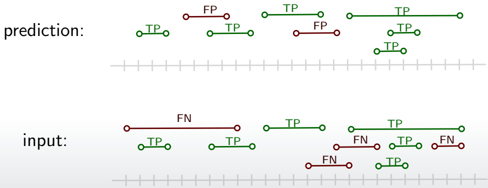
Interval Scheduling with predictions

- We consider predictions that concern membership in the input sequence.



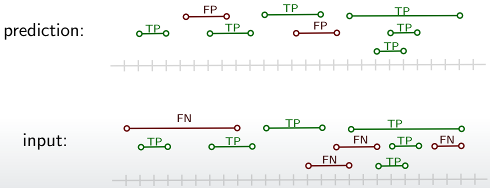
Interval Scheduling with predictions

- We consider predictions that concern membership in the input sequence.



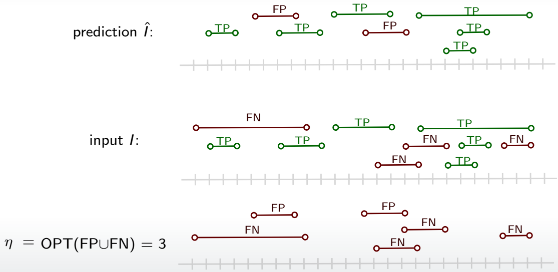
Interval Scheduling with predictions

- We consider predictions that concern membership in the input sequence.
- Statistical predictions such as average input length are unlikely to help.



The Error Measure

- We use $\eta = Opt(FP \cup FN)$



The Error Measure

- We use $\eta = \textsc{Opt}(\mathit{FP} \cup \mathit{FN})$
- Desirable Properties:

The Error Measure

- We use $\eta = \textsc{Opt}(\mathit{FP} \cup \mathit{FN})$
- Desirable Properties:
  - **Monotonicity**: eliminating false negatives/positives must not increase the error.

The Error Measure

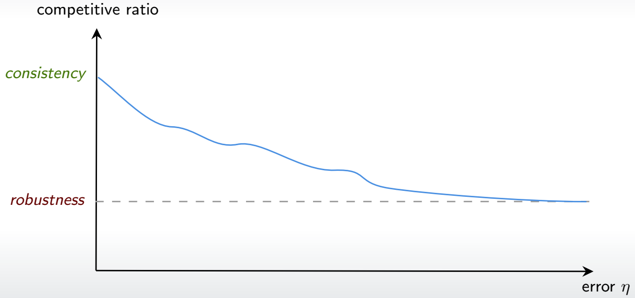
- We use $\eta = \textsc{Opt}(\mathit{FP} \cup \mathit{FN})$
- Desirable Properties:
  - **Monotonicity**: eliminating false negatives/positives must not increase the error.
  - **Lipschitz:** the error is not "too small".
  - **Completeness:** the error is not "too large".

The Error Measure

- We use $\eta = \textsc{Opt}(FP \cup FN)$
- Desirable Properties:
  - **Monotonicity**: eliminating false negatives/positives must not increase the error.
  - **Lipschitz:** the error is not "too small".
  - **Completeness:** the error is not "too large".
- We define **normalized error** $\gamma(\hat{I}, I) = \frac{\eta(\hat{I}, I)}{\textsc{Opt}(\hat{I}, I)}$

Algorithmic Goal

- Design an algorithm that is **consistent**, **robust**, and **smooth**.



Follow the Prediction

- The most obvious algorithm to try (first) just follows the prediction:

Algorithm Trust

From $\hat{I}$, compute an optimal solution, I^*
for all requests $r \in I$:
  if $r \in I^*$:
    accept
  else:
    reject

Follow the Prediction

- A positive result for general graphs:

Theorem

On any graph, $\textsc{Trust}(\hat{I}, I) \geq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

Proof.

$$\begin{aligned}
\textsc{Trust}(\hat{I}, I) &\geq \textsc{Opt}(\hat{I}) - \textsc{Opt}(FP) \\
&\geq \textsc{Opt}(I) - \textsc{Opt}(FN) - \textsc{Opt}(FP) \\
&\geq \textsc{Opt}(I) - 2\textsc{Opt}(FP \cup FN) \\
&= \textsc{Opt}(I) - 2\eta(\hat{I}, I) \\
&= (1 - 2\gamma(\hat{I}, I))\textsc{Opt}(I)
\end{aligned}$$

Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$

Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.

Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\text{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$

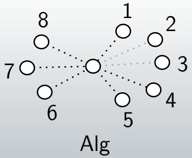
Follow the Prediction

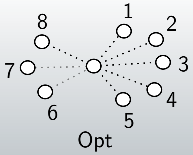
- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$





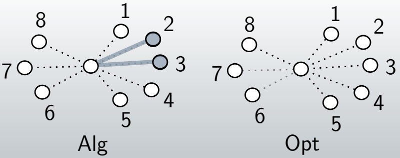
I = (2,3)

Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

> **Theorem**
>
> *On a star graphs S_8,* $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



$\mathrm{I} = (2,3)$

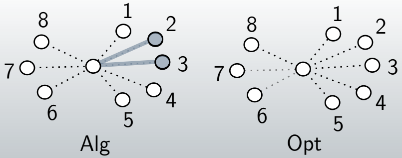
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



I = (2,3) , (6,7)

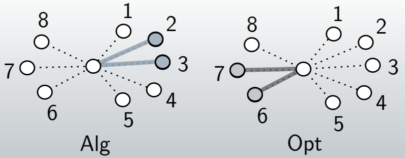
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



I = (2,3) , (6,7)

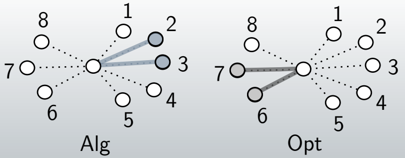
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



$\mathrm{I} = (2{,}3)\ ,\ (6{,}7)\ \ ,\ (1{,}2)$

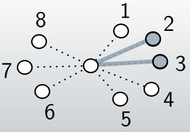
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

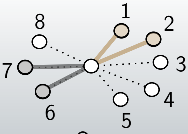
Theorem

On a star graphs S_8, $\text{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \text{Opt}(I)$

- Showing, equivalently, $\text{Alg}(\hat{I}, I) \leq \text{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of Opt is 2 more than the profit of Alg.
- One case of the proof: $\hat{I} = \{(1,2), (2,3), (3,4), (4,5), (6,7), (7,8)\}$



Alg



Opt

I = (2,3) , (6,7) , (1,2)

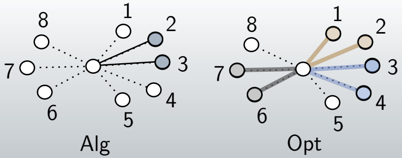
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



I = (2,3) , (6,7) , (1,2) , (3,4)

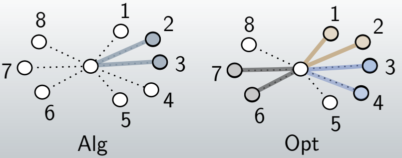
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



I = (2,3) , (6,7) , (1,2) , (3,4) , (7,8)

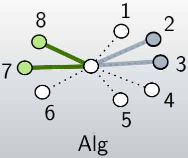
Follow the Prediction

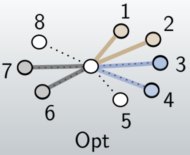
- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$





I = (2,3) , (6,7) , (1,2) , (3,4) , (7,8)

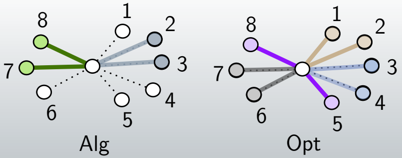
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2), (2,3), (3,4), (4,5), (6,7), (7,8)\}$



I = (2,3) , (6,7) , (1,2) , (3,4) , (7,8) , (5,8)

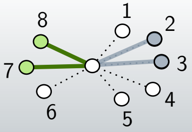
Follow the Prediction

- This is the best possible for the disjoint path allocation problem:

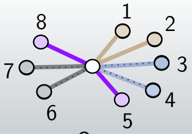
Theorem

On a star graphs S_8, $\textsc{Alg}(\hat{I}, I) \leq (1 - 2\gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Showing, equivalently, $\textsc{Alg}(\hat{I}, I) \leq \textsc{Opt}(\mathrm{I}) - 2\eta(\hat{I}, I)$
- Exhibit an input (and prediction) s.t., the prediction error is 1, and the profit of $\textsc{Opt}$ is 2 more than the profit of $\textsc{Alg}$.
- One case of the proof: $\hat{I} = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$



Alg



Opt

I = (2,3) , (6,7) , (1,2) , (3,4) , (7,8) , (5,8)

Opt(FN∪FP) = Opt({(4,5),(5,8)}) = 1

Conclusion for Disjoint Path Allocation

- As long as a graph class contains S_8, $\textsc{Trust}$ (follow-the-prediction) is the best possible.

Conclusion for Disjoint Path Allocation

- As long as a graph class contains S_8, TRUST (follow-the-prediction) is the best possible.
- This motivates us to focus on interval graphs (interval scheduling).

TrustGreedy Algorithm

- An improved algorithm for interval scheduling:

Algorithm TrustGreedy

From $\hat{I}$, compute a left-most optimal solution, I^*

for all requests $r \in I$:

if r does not overlap an accepted request **and**
(is in I^* **or**
does not overlap any I^*-requests **or**
overlaps exactly one I^*-request ending no earlier than r)

accept r

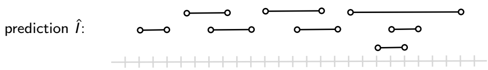
update I^* if necessary

else:

reject r

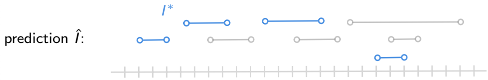
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



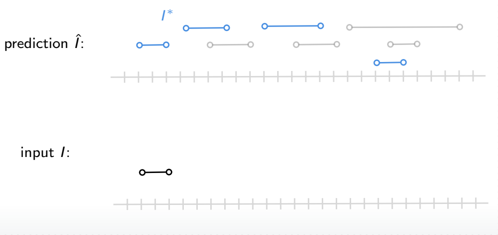
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



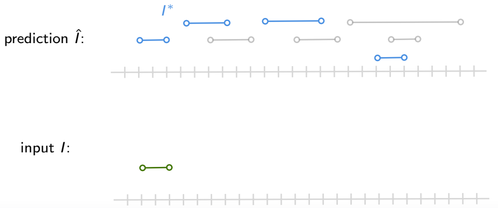
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



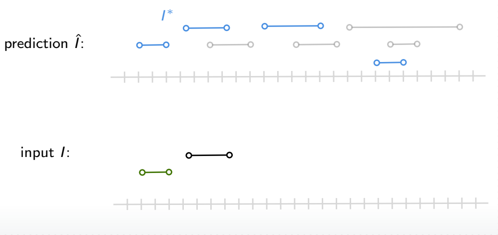
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



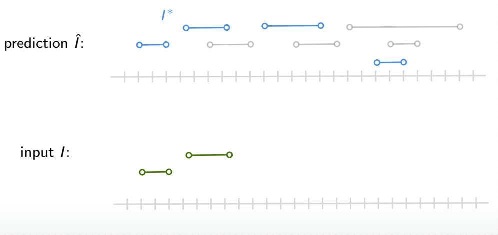
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



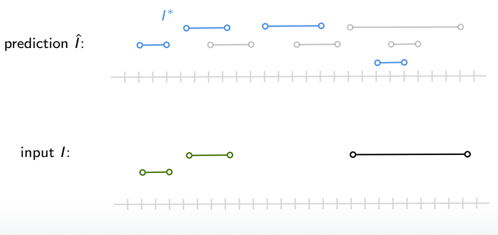
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



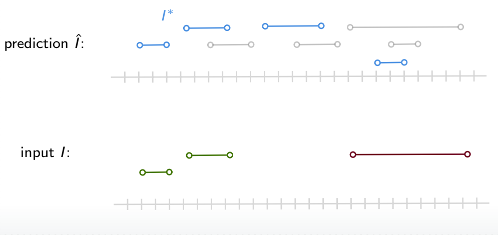
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



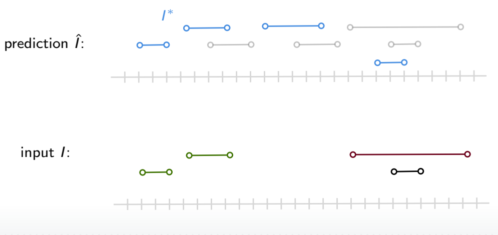
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



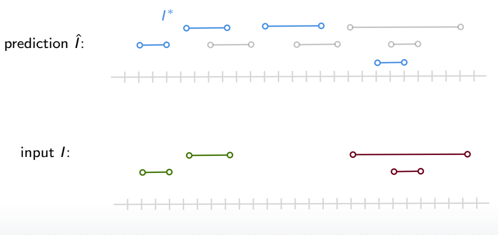
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



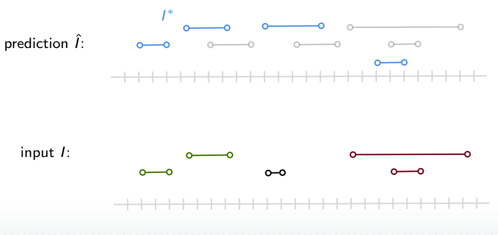
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:

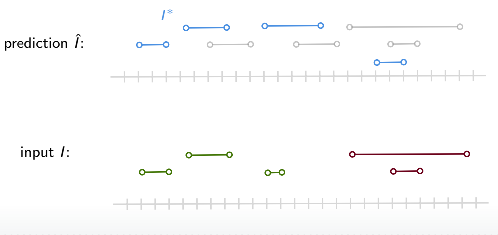


TrustGreedy Algorithm

- An improved algorithm for interval scheduling:

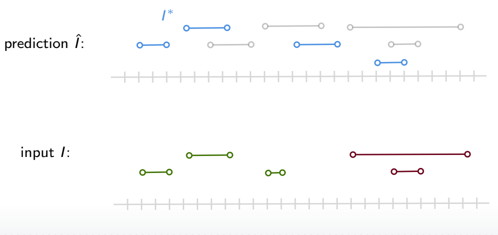


- An improved algorithm for interval scheduling:



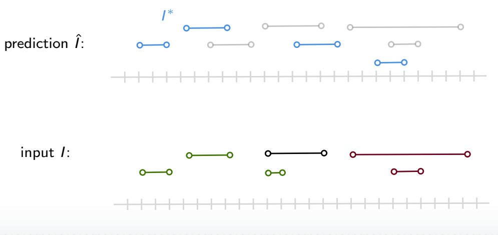
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



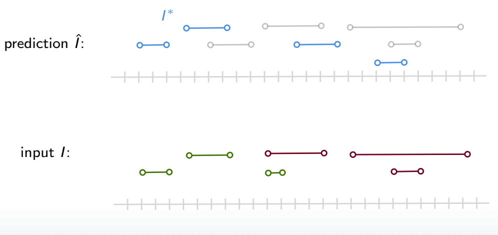
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



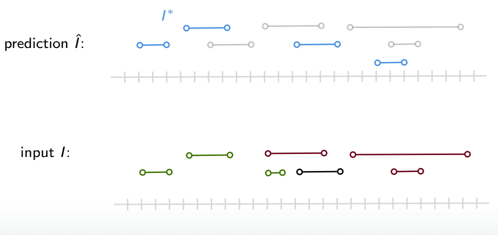
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



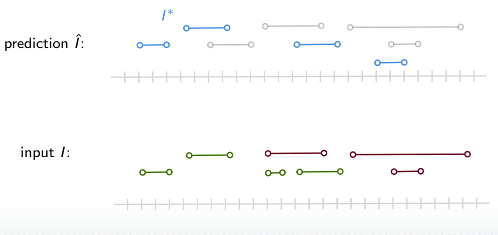
TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



TrustGreedy Algorithm

- An improved algorithm for interval scheduling:



TrustGreedy Algorithm

Theorem

For any prdeiction $\hat{I}$ and input sequenec I, $\textsc{TrustGreedy}(\hat{I}, I) \geq (1 - \gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- An improvement over the competitive ratio $1 - 2\gamma(\hat{I}, I)$ of $\textsc{Trust}$.

TrustGreedy Algorithm Optimality

Theorem

For any deterministic algorithm $\textsc{Alg}$*, there are input sequences and predictions* I *and* $\hat{I}$*, so* $\textsc{Alg}(\hat{I}, I) \leq (1 - \gamma(\hat{I}, I))\ \textsc{Opt}(I)$

TrustGreedy Algorithm Optimality

Theorem

For any deterministic algorithm $\textsc{Alg}$*, there are input sequences and predictions* I *and* $\hat{I}$*, so* $\textsc{Alg}(\hat{I}, I) \leq (1 - \gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Let $\hat{I} = \{(0,2),(0,1)\}$, and I start with $(0,2)$.

TrustGreedy Algorithm Optimality

Theorem

For any deterministic algorithm $\textsc{Alg}$*, there are input sequences and predictions* I *and* $\hat{I}$*, so* $\textsc{Alg}(\hat{I}, I) \leq (1 - \gamma(\hat{I}, I))\ \textsc{Opt}(I)$

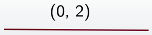
- Let $\hat{I} = \{(0,2),(0,1)\}$, and I start with $(0,2)$.
  - If $\textsc{Alg}$ rejects $(0,2)$, $\textsc{Opt}$ accepts it and input ends, so $(0,1) \in FP$, and $\eta = 1$.

TrustGreedy Algorithm Optimality

Theorem

For any deterministic algorithm $\textsc{Alg}$*, there are input sequences and predictions* I *and* $\hat{I}$*, so* $\textsc{Alg}(\hat{I}, I) \leq (1 - \gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Let $\hat{I} = \{(0,2), (0,1)\}$, and I start with $(0,2)$.
  - If $\textsc{Alg}$ rejects $(0,2)$, $\textsc{Opt}$ accepts it and input ends, so $(0,1) \in FP$, and $\eta = 1$.
  - If $\textsc{Alg}$ accepts $(0,2)$, then I continues with $(0,1)$ and $(1,2)$ that $\textsc{Opt}$ accepts, so $(1,2) \in FN$ and $\eta = 1$.



Alg $= 0$ Opt $= 1$

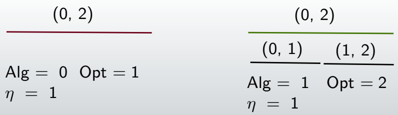
$\eta = 1$

TrustGreedy Algorithm Optimality

Theorem

For any deterministic algorithm $\textsc{Alg}$*, there are input sequences and predictions* I *and* $\hat{I}$*, so* $\textsc{Alg}(\hat{I}, I) \leq (1 - \gamma(\hat{I}, I))\ \textsc{Opt}(I)$

- Let $\hat{I} = \{(0,2),(0,1)\}$, and I start with $(0,2)$.
  - If $\textsc{Alg}$ rejects $(0,2)$, $\textsc{Opt}$ accepts it and input ends, so $(0,1) \in FP$, and $\eta = 1$.
  - If $\textsc{Alg}$ accepts $(0,2)$, then I continues with $(0,1)$ and $(1,2)$ that $\textsc{Opt}$ accepts, so $(1,2) \in FN$ and $\eta = 1$.



Consistency/Robustness Tradeoffs

- **Consistency** refers to the competitive ratio when the predictions are correct, and **robusteness** is the competitive ratio when predictions are adversarial.

Consistency/Robustness Tradeoffs

- **Consistency** refers to the competitive ratio when the predictions are correct, and **robusteness** is the competitive ratio when predictions are adversarial.
- Starting with a negative result:

Theorem

If a (possibly randomized) algorithm $\textsc{Alg}$ *is both* α*-consistent, then its robustness is at most* $\beta = \frac{2(1-\alpha)}{\lfloor \log m \rfloor - 1}$.

Consistency/Robustness Tradeoffs

- For a positive result, we define $\textsc{Robust-Trust}(\alpha)$ as follows:

Algorithm RobustTrust (α)

Draw probablity p uniformly at random
if $p < \alpha$:
    apply algorithm TrustGreedy
else
    apply algorithm Classify-and-Randomly-Select

Consistency/Robustness Tradeoffs

- For a positive result, we define $\textsc{Robust-Trust}(\alpha)$ as follows:

Algorithm RobustTrust (α)

Draw probablity p uniformly at random
if $p < \alpha$:
  apply algorithm TrustGreedy
else
  apply algorithm Classify-and-Randomly-Select

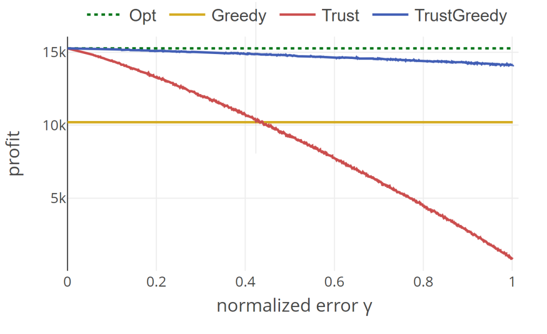
Theorem

$\textsc{Robust-Trust}(\alpha)$ *has consistency at least* α *and robustness at least* $\frac{1-\alpha}{\lceil \log m \rceil}$

- Robust-Trust(α) asymptotically Pareto optimal.

Experimental Result

- Consider Trust, TrustGreedy, Greedy, and Opt on real-world scheduling data on parallel machines [Chapin et al. IPPS/SPDP, 1999]



Implied Results

- The negative result on star graphs implies a negative result for **matching** in general graphs (even if restricted to planar graphs).

Implied Results

- The negative result on star graphs implies a negative result for **matching** in general graphs (even if restricted to planar graphs).
- The negative results on matching implies a negative result for **indepenedent set** in general graphs

Summary

- For disjoint path allocation problem, $\textsc{Trust}$ has a competitive ratio of $1 - 2\gamma(\hat{I}, I)$, which is optimal.
- For interval scheduling, $\textsc{TrustGreedy}$ has a competitive ratio of $1 - \gamma(\hat{I}, I)$, which is optimal.
- For consistency/robustness tradeoff, $\textsc{RobustTrust}(\alpha)$ is α-consistent and $(1 - \alpha)/\lceil \log m \rceil$-robust, which is asymptotically Pareto-optimal.